PLUMED

2.9.0

Generated by Doxygen 1.8.17

Chapter 1

Introduction

PLUMED is a plugin that works with a large number of molecular dynamics codes (Codes interfaced with PLUMED). It can be used to analyze features of the dynamics on-the-fly or to perform a wide variety of free energy methods. PLUMED can also work as a Command Line Tools to perform analysis on trajectories saved in most of the existing formats. If PLUMED is useful for your work please read and cite [1], if you are interested in the PLUMED 1 original publication please read and cite [2] .

To follow the development of PLUMED 2, you can look at the detailed Change Log .

To install PLUMED, see this page: Installation , while in Getting Started you can find a brief introduction on how to write your first PLUMED input file.

Tutorials are available to introduce basic as well as more advanced features of PLUMED.

1.1 About this manual

This manual has been compiled from PLUMED version **2.9.0** (git version: **e5b5370**). Manual built on GitHub Actions on ref refs/heads/v2.9.

This is the user manual - if you want to modify PLUMED or to understand how it works internally, have a look at the developer manual .

1.2 Codes interfaced with PLUMED

PLUMED can be incorporated into an MD code and used to analyze or bias a molecular dynamics run on the fly. Some MD code could already include calls to the PLUMED library and be PLUMED-ready in its original distribution. As far as we know, the following MD codes can be used with PLUMED out of the box:

- Amber, pmemd module, since version 20.
- AmberTools, sander module, since version 15.
- CP2K, since Feb 2015.
- ESPResSo, in a version that has been patched with PLUMED can be found here.
- PINY-MD, in its plumed branch.

- IPHIGENIE.
- AceMD, see this link.
- OpenMM, using the openmm-plumed plugin.
- DL_POLY4.
- VNL-ATK, see this link.
- ABIN.
- i-pi.
- LAMMPS since Nov 2018.
- Yaff, since Jul 2019.
- DFTB+, since release 20.1.
- Metalwalls
- ASE
- GPUMD

Please refer to the documentation of the MD code to know how to use it with the latest PLUMED release. If you maintain another MD code that is PLUMED-ready let us know and we will add it to this list.

Additionally, we provide patching procedures for the following codes:

- gromacs-2020-7
- gromacs-2021-7
- gromacs-2022-5
- gromacs-2023
- namd-2-12
- namd-2-13
- namd-2-14
- qespresso-5-0-2
- qespresso-6-2
- qespresso-7-0
- qespresso-7-2

Alternatively, one can use PLUMED as a Command Line Tools for post processing the results from molecular dynamics or enhanced sampling calculations. Notice that PLUMED can be used as an analysis tool also from the following packages:

- PLUMED-GUI is a VMD plugin that computes PLUMED collective variables.
- HTMD can use PLUMED collective variables for analysis.
- OpenPathSampling, using the PLUMED Wrapper for OpenPathSampling.

Chapter 2

Change Log

Here you can find a history of changes across different PLUMED versions. The future releases are expected to follow more or less the pace of the old release. This means:

- Approximately once per year, after summer, a new release (2.X). These releases typically group together all the features that were contributed during the year.
- Approximately every three month, we announce a patch (e.g. 2.2.X). This typically contains bug fixes, and could occasionally contain a new feature.

A few months before each new release we provide a beta release. We typically maintain release branches until the fifth patch release (2.X.5), which should come out approximately 15 month after the original release (2.X). After that, branches are not supported anymore.

Notice that occasionally we publish patches on the mailing list. These patches are always included in the following release, but we encourage users that want to be up to date to follow the mailing list.

Below you can find change logs for all the published releases. We mostly add new features without breaking existing ones. However, some of the changes lead to incompatible behavior. In the Change Log we try to give as much visibility as possible to these changes to avoid surprises.

We also log changes that are relevant if you are developing the code. These change lists are however not complete, and if you want to put your hands in the code and maintain your own collective variables we suggest you to follow the development on github.

2.1 Version 2.0

2.1.0.1 Version 2.0.0 (September 27, 2013)

Version 2.0 is a complete rewrite, so there is no way to write a complete set of difference with respect to plumed 1.3. Here is a possibly incomplete summary of the difference:

- The input is simpler, more flexible, and more error proof. Many checks are now performed and in this way common errors are avoided.
- The units are now the same for all MD codes. If you want to use a different unit than the default you set it in the input file.
- The analysis tools are now much more flexible. As an example of this it is now possible to write different collective variables with different frequencies.
- Many complex collective variables are considerably faster than they were in plumed1. In particular, all variables based on RMSD distances.
- Centers of mass can be used as if they were atoms. Hence, unlike plumed 1.3, you can use center of mass positions in ALL collective variables.
- The virial contribution is now computed and passed to the MD code. Plumed can thus now be used to perform biased NPT simulations.
- Variables can be dumped on different files, and are computed only when this is necessary.
- PLUMED is now compiled as a separate library. This simplifies the patching procedure, but might require some extra work to configure PLUMED properly. Since PLUMED can be loaded as a shared library, it is possible to setup everything such that PLUMED and MD codes can be updated independently from each other.

In addition, it is now much easier to contribute new functionality to the code because:

- There is a much simpler interface between plumed and the base MD codes. This makes it much easier to add plumed to a new MD code. Hopefully, in the future, interfaces with MD codes will be maintained by the developers of the MD codes independently from PLUMED developers. This will allow more MD codes to be compatible with PLUMED.
- There is C++ object oriented programming and full compatibility with the C++ standard library
- A modular structure.
- New collective variables and methods can be released independently.
- There is an extensive developer documentation.
- User documentation is provided together inside the implementation files.

Caveats:

- PLUMED 2 input file (plumed.dat) has a syntax which is not compatible with PLUMED 1. Transition should be easy, but cannot be done just using the new version with the old input file.
- PLUMED 2 is written in C++, thus requires a C++ compiler
- PLUMED 2 may not include all the features that were available in PLUMED 1.

A tutorial explaining how to move from PLUMED 1 to PLUMED 2 is available (see Moving from PLUMED 1 to PLUMED 2).

2.1.0.2 Version 2.0.1 (Nov 14, 2013)

For users:

- Fixed a bug in HISTOGRAM with REWEIGHT_BIAS. Reweighting was only done when also temperature-reweighting was enabled.
- Fixed a bug that was sometime crashing code with domain decomposition and non-dense simulation boxes (e.g. implicit solvent).
- Performance improvements for GYRATION.
- Flush all files every 10000 steps by default, without need to use FLUSH
- Errors when writing input for switchingfunction are now properly recognized.
- Added message when simplemd is used on a non-existing file.
- Fixed `plumed mklib` such that it deletes the target shared library in case of compilation error.
- Several small fixes in documentation and log file.

For developers:

- Added possibility to setup replica exchange from MD codes in Fortran (commands "GREX setMPIF↩ Intercomm" and "GREX setMPIFIntracomm").
- cmd("setStopFlag") should now be called after PLUMED initialization.
- Several small fixes in documentation.

2.1.0.3 Version 2.0.2 (Feb 11, 2014)

For users:

- Fixed bug with METAD with INTERVAL and replica exchange, including bias exchange. Now the bias is correctly computed outside the boundaries. Notice that this is different from what was done in PLUMED 1.3. Also notice that INTERVAL now works correctly with grids and splines.
- Fixed bug with READ and periodic variables.
- Fixed bug with HISTOGRAM (option USE_ALL_DATA was not working properly).
- Gromacs patch updated to 4.6.5.
- Gromacs patch for 4.6 has been modified to allow for better load balancing when using GPUs.
- Added option 'plumed info –long-version' and 'plumed info –git-version'.
- Added full reference (page/number) to published paper in doc and log.
- Fixed a bug in file backups (only affecting Windows version - thanks to T. Giorgino).
- Added possibility to search in the documentation.
- Several small fixes in documentation and log file.

For developers:

- Fixed Makefile dependencies in some auxiliary files in src/lib (∗cmake and ∗inc).
- Changed way modules are linked in src/. E.g. src/colvar/tools/ is not anymore a symlink to src/colvar but a real directory. (Notice that this introduces a regression: when using plumed as an external library some include files could not work - this only applies when plumed is installed; also notice that this is fixed in 2.0.3)
- Patch for gromacs 4.6 now also include original code so as to simplify its modification.
- Added option 'plumed patch –save-originals'.
- Fixed regtest regtest/secondarystructure/rt32 to avoid problems with NUMERICAL_DERIVATIVES.
- Removed include graphs in the documentation (too large).
- Several small fixes in documentation.

2.1.0.4 Version 2.0.3 (June 30, 2014)

For users:

- Now compiles on Blue Gene Q with IBM compilers.
- Fixed bug in CENTER where default WEIGHTS were missing.
- Fixed broken CONTACTMAP with SUM
- Fixed DUMPATOMS with gro file and more than 100k atoms.
- Added CMDIST in CONTACTMAP to emulate plumed1 CMAP.
- Several small fixes in documentation and log file.

For developers:

- Fixed cmd("getBias") to retrieve bias. It was not working with single precision codes and it was not converting units properly.
- Fixed a regression in 2.0.2 concerning include files from installed plumed (see commit 562d5ea9dfc3).
- Small fix in tools/Random.cpp that allows Random objects to be declared as static.
- Small fix in user-doc compilation, so that if plumed is not found the sourceme.sh file is sourced
- Fixed non-ANSI syntax in a few points and a non-important memory leakage.
- Split cltools/Driver.cpp to make parallel compilation faster.

2.1.0.5 Version 2.0.4 (September 15, 2014)

For users:

- Fixed a bug in BIASVALUE that could produce wrong acceptance with replica exchange simulations.
- Fixed a few innocuous memory leaks.
- Fixed reader for xyz files, that now correctly detects missing columns. Also a related regtest has been changed.
- Several small fixes in documentation and log file.

For developers:

- Renamed Value.cpp to BiasValue.cpp

2.1.0.6 Version 2.0.5 (December 15, 2014)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

For users:

- Fixed a bug in replica exchange with different Hamiltonians (either lambda-dynamics or plumed XX-hrex branch) possibly occurring when using charge or mass dependent variables.
- Fixed a bug in analysis (e.g. HISTOGRAM) leading to wrong accumulation of statistics when running a replica exchange simulation.
- Fixed a bug in the calculation of derivatives in histograms. This should be harmless since people usually only consider the value in histograms and not the derivatives.
- Fixed an issue in Makefile that could results in problems when patching an MD code with –shared option (pointed out by Abhi Acharya). This fixes a regression introduced in 2.0.2.
- Small fixes in documentation.

For developers:

- Added warning when performing regtests using an instance of plumed from a different directory

2.2 Version 2.1

2.2.0.1 Version 2.1.0 (September 15, 2014)

Version 2.1 contains several improvements with respect to 2.0. Users currently working with 2.0 should have a look at the section "Changes leading to incompatible behavior" below and might need tiny adjustments in their input files. In 2.1 we restored more features of 1.3 that were missing in 2.0, so users still working with 1.↩ 3 could opt for an upgrade. A tutorial explaining how to move from PLUMED 1 to PLUMED 2 is available (see Moving from PLUMED 1 to PLUMED 2).

Below you find a list of all the changes with respect to version 2.0. Notice that version 2.1 includes already all the fixes in branch 2.0 up to 2.0.4.

Changes from version 2.0 which are relevant for users:

- Changes leading to incompatible behavior:
  - COORDINATION now skips pairs of one atom with itself.
  - Labels of quantities calculated by BIASVALUE have changed from *label*.bias.*argname* to *label.argname*_bias, which is more consistent with steered MD
  - Labels of quantities calculated by ABMD have change from *label*.min_*argname* to *label.argname*_min, which is more consistent with steered MD
  - Labels of quantities calculated by PIECEWISE have change from *label.argnumber* to *label.argname*↩ _pfunc, which is more consistent with steered MD
  - For multicolvars components calculated with LESS_THAN and MORE_THAN keywords are now labelled lessthan and morethan. This change is necessary as the underscore character now has a special usage in component names.

  - In CONTACTMAP components are now labelled *label*.contact-n.
  - The command SPHERE has been replaced by UWALLS.

- New configuration system based on autoconf (use ./configure from root directory). Optional packages are detected at compile time and correctly enabled or disabled. An internal version of LAPACK and BLAS will be used if these libraries are not installed.

- New actions:

  - SPRINT topological collective variables.
  - CH3SHIFTS collective variable.
  - POSITION collective variable.
  - FIT_TO_TEMPLATE.
  - COMMITTOR analysis.
  - LOCAL_AVERAGE.
  - NLINKS.
  - DIHCOR.
  - NOE.
  - RDC.
  - CLASSICAL_MDS.
  - XDISTANCES.
  - YDISTANCES.
  - ZDISTANCES.
  - DUMPMULTICOLVAR.
  - Crystallization module, including Q3, LOCAL_Q3, Q4, Q6, LOCAL_Q4, LOCAL_Q6, MOLECULES, SIMPLECUBIC, TETRAHEDRAL and FCCUBIC.
  - ENSEMBLE to perform Replica-Averaging on any collective variable.

- New features for existing actions:

  - METAD : WALKERS_MPI flag (multiple walkers in a mpi-based multi-replica framework), ACCELE↩ RATION flag (calculate on the fly the Metadynamics acceleration factor), TAU option (alternative way to set Gaussian height in well-tempered metadynamics), GRID_SPACING (alternative to GRID_BIN to set grid spacing). Notice that now one can also omit GRID_BIN and GRID_SPACING when using fixed size Gaussian, and the grid spacing will be automatically set.
  - DISTANCE : added SCALED_COMPONENTS
  - COORDINATION : if a single group is provided, it avoids permuted atom indexes and runs at twice the speed.
  - DUMPATOMS : PRECISION option to set number of digits in output file.
  - GROUP : NDX_FILE and NDX_GROUP options to import atom lists from ndx (gromacs) files.
  - In many multicolvars, MIN and MAX options can be used.
  - HISTOGRAM : GRID_SPACING (alternative to GRID_BIN to set grid spacing), FREE-ENERGY flags in addition to standard probability density, additional option for KERNEL=DISCRETE to accumulate standard histograms.
  - sum_hills : added options –spacing (alternative to –bin to set grid spacing) and –setmintozero to translate the minimum of the output files to zero.
  - CONTACTMAP : parallelized and added weights.

- New features in MD patches (require re-patch):

  - New patch for Gromacs 5.0
  - Gromacs 4.6.X patch updated to 4.6.7

  - Gromacs 4.6.7 supports COMMITTOR analysis; can be now be used to perform energy minimization; now passes temperature to PLUMED (this allows temperature to be omitted in some actions, namely METAD and analysis actions).

  Notice that if you use runtime binding it is not compulsory to re-patch, and that all combinations should work correctly (new/old PLUMED with re-patched/non-re-patched MD code).

- Other new features:
  - driver can now read trajectories in many formats using VMD molfile plugin (requires VMD plugins to be compiled and installed). In case VMD plugins are not installed, the configuration system falls back to an internal version which implements a minimal list of plugins (gromacs and dcd) (kindly provided by T. Giorgino).
  - switchingfunction : added STRETCH flag.
  - Negative strides in atom ranges (e.g. ATOMS=10-1:-3 is expanded to ATOMS=10,7,4,1).
  - COORDINATION and DHENERGY with NLIST now work correctly in replica exchange simulations.
  - Multicolvars with neighbor lists now work correctly in replica exchange simulations.
  - Improved multicolvar neighbor lists.
- Optimization:
  - Root-mean-square deviations with align weights different from displace weights are now considerably faster. This will affect RMSD calculations plus other variables based on RMSD.
  - WHOLEMOLECULES is slightly faster.
  - COORDINATION is slightly faster when NN and MM are even and D_0=0.
  - Atom scattering with domain decomposition is slightly faster.
  - Link cells are now exploited in some multicolvars.
  - Derivatives are not calculated unless they are specifically required, because for instance you are adding a bias.
- Documentation:
  - All tutorial material from the recent plumed meeting in Belfast is now in the manual
  - Improvements to documentation, including lists of quantities that are output by each action that can be referenced
  - Manual has been re-organized following suggestions received at the plumed meeting.
  - An experimental PDF version of the manual is now provided (a link can be found in the documentation homepage).

Changes from version 2.0 which are relevant for developers:

- Added regtests for plumed as a library (e.g. basic/rt-make-0). plumed command has an additional flag (–is-installed) to probe if running from a compilation directory or from a fully installed copy (this is needed for regtests to work properly).
- Improved class Communicator. Many operations can now be done directly on Vectors, Tensors, std::vector and PLMD::Matrix.
- Modified class RMSD.
- Patches for GPL codes (Quantum Espresso and Gromacs) now also include original code so as to simplify their modification.
- Fixed dependencies among actions such that it is now possible (and reliable) to use MPI calls inside Action↩ ::prepare()
- colvar/CoordinationBase.cpp has been changed to make it faster. If you devised a class which inherits from here, consider that CoordinationBase::pairing now needs *squared* distance instead of distance

- It is possible to run "make install" from sub-directories (e.g. from src/colvar)
- There is a small script which disables/enables all optional modules (make mod-light/mod-heavy/mod-reset)
- Added "-q" option to plumed patch
- You can now create new metrics to measure distances from a reference configurations. If you do so such metrics can then be used in paths straightforwardly
- You can now use multicolvars in tandem with manyrestraints in order to add a large numbers of restraints.
- Can now do multicolvar like things in which each colvar is a vector rather than a scalar.
- Updated script that generated header files so that they properly show years. Notice that the script should new be run from within a git repository

This list is likely incomplete, if you are developing in PLUMED you are encouraged to follow changes on github.

2.2.0.2 Version 2.1.1 (December 15, 2014)

This release includes all the fixes available in branch 2.0 until 2.0.5.

For users:

- New patch for AMBER 14 (sander module only). This patch should be compatible with any PLUMED 2 version (including 2.0). It includes most PLUMED features with the notable exception of multi-replica framework.
- Changed definition in arbitrary phase of eigenvectors. This will change the result of some analysis method where the phase does matter (e.g. CLASSICAL_MDS) and make some regression test better reproducible.
- Fixed a portability issue in BG/P where gettimeofday is not implemented. Notice that this fix implies that one should execute again ./configure to have plumed timing working correctly.
- CS2Backbone: fixed a bug that resulted in only a fraction of the chemical shifts being printed with WRITE_CS and parallel simulations (requires to get the last almost updated from SVN)
- NOE: fixed a bug in the replica-averaging
- Fixed a linking issue with ALMOST, where bz2 was always used to link ALMOST to PLUMED even if it is not compulsory to build ALMOST.
- Fixed a wrong include in the GMX5 patch.
- FUNCPATHMSD can now be used together with CONTACTMAP to define pathways in contact-map space
- Configuration is more verbose, a warning is given if a default option cannot be enabled and an error is given if an option explicitly enabled cannot be enabled.
- Compilation is less verbose (use "make VERBOSE=1" to have old behavior)
- Small fixes in documentation.

For developers:

- Tests are now performed at every single push on travis-ci.org
- Manual is built and pushed to the online server from travis-ci.org (see developer doc)
- Fixes in developer doc.

2.2.0.3 Version 2.1.2 (Mar 16, 2015)

For users:

- Added two new short tutorials to the manual (Cambridge tutorial and Munster tutorial).
- Fixed a severe bug on DRMSD - cutoff values were ignored by PLUMED. Notice that this bug was introduced in 2.1.0, so that it should not affect the 2.0.x series.
- Fixed a bug affecting LAMMPS patch used with a single processor. Notice that the fix is inside PLUMED, thus it does not necessarily requires re-patching.
- Sander patch now works with multiple replica (no replica exchange yet). It also contains some fix from J. Swails.
- GMX5 patch was not working for bias-exchange like cases
- Patching system now checks for the availability of shared/static/runtime version of plumed before patching
- Configure now check better if compiler flag are accepted by the compiler. This makes configure on bluegene more robust.
- Sourceme.sh now sets proper library path in linux also.

2.2.0.4 Version 2.1.3 (June 30, 2015)

For users:

- Fixed bug in ENSEMBLE derivatives when more than 1 argument was provided
- Fixed bug in GHOST : virial is now computed correctly.
- Fixed a serious bug in virial communicated from plumed to gromacs, for both gromacs versions 4.6 and 5.↩ 0. See #132. This fix requires gromacs to be re-patched and could be very important if you run biased simulations in the NPT ensemble.
- Fixed a bug in the virial computed with FIT_TO_TEMPLATE when the reference pdb had center non located at the origin.
- Fixed a bug in the the forces computed with FIT_TO_TEMPLATE when used in combination with COM, CENTER, or GHOST
- Fixed a bug that could lead plumed to be stuck with domain decomposition in some extreme case (one domain with all atoms, other domains empty).
- Fixed a bug when COMBINE or MATHEVAL are used with PERIODIC keyword. Now when PERIODIC keyword is used the result of the calculation is brought within the periodicity domain. See #139.
- Fixed a bug related to RANDOM_EXCHANGES followed by INCLUDE
- Fixed bug in derivatives of histogram bead with triangular kernels
- Updated gromacs patch 4.5.5 to 4.5.7
- Updated internal molfile plugins to VMD 1.9.2.
- Included crd and crdbox formats to internal molfile.
- Added –natoms to driver . This is required to read coordinate files with VMD plugins when number of atoms is not present (e.g. amber crd files)
- Added the checks in the driver to detect cases where molinfo does not provide box information (e.g. pdb).

- Added support for readdir_r when available, which makes opening files thread safe.
- CFLAGS now include -fPIC by default
- Added a warning when using METAD without grids with a large number of hills.
- Fixes in user documentation.

For developers:

- Allow external VMD plugins to be detected with –has-external-molfile. This is required to enable some regtest with amber files.
- Added –dump-full-virial to driver
- Allow definition of variables where some of the components have derivatives and some haven't (#131).
- Improved travis tests with more debug options.
- Improved some regtest to check out-of-diagonal virial components
- Improved make cppcheck options.
- Fixes in developer documentation.

2.2.0.5 Version 2.1.4 (Oct 13, 2015)

For users:

- Fixed NAMD patch. Masses and charges were not passed correctly, thus resulting in wrong COM or CENTER with MASS. This fix required re-patching NAMD. Notice that this bug was present also in v2.0 but in a different form. More information here (#162), including a workaround that allows masses to be fixed without re-patching.
- When installing with PLUMED_LIBSUFFIX an underscore is used as separator instead of a dash. E.g. `make install PLUMED_LIBSUFFIX=2.1` will result in an executable named `plumed_v2.1`. This fix a potential problem (see Installation).
- Fixed erroneously reported message about MPI at the end of ./configure.
- Changed warning message about undocumented components.
- PLUMED now says in the log file if it was compiled from a dirty git repository.
- Fixed a problem leading to rare random crashes when using METAD with WALKERS_MPI and multiple processors per replica.
- Small change in numerical accuracy of lattice reduction. Should be more robust when running with highly optimizing compilers.
- Fixed a bug in normalization of kernel functions. This affects HISTOGRAM If these actions were used with previous versions of the code care should be taken when analyzing the results.
- Fixed a bug in derivatives of kernel functions with non-diagonal covariance matrices. This affects the derivatives output by sum_hills

2.2.0.6 Version 2.1.5 (Jan 18, 2016)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

For users:

- PLUMED now reports an error when using HISTOGRAM with FREE-ENERGY without USE_ALL_DATA. See #175
- Fixed a bug in configure together with –enable-almost. The check for lbz2 library was not working properly.

2.3 Version 2.2

2.3.0.1 Version 2.2 (Oct 13, 2015)

Version 2.2 contains several improvements with respect to 2.1. Users currently working with 2.1 should have a look at the section "Changes leading to incompatible behavior" below and might need tiny adjustments in their input files. In 2.2 we restored more features of 1.3 that were missing in 2.1, so users still working with 1.↩ 3 could opt for an upgrade. A tutorial explaining how to move from PLUMED 1 to PLUMED 2 is available (see Moving from PLUMED 1 to PLUMED 2).

Below you find a list of all the changes with respect to version 2.1. Notice that version 2.2 includes already all the fixes in branch 2.1 up to 2.1.4 indicated in Version 2.1 .

Changes from version 2.1 which are relevant for users:

- Changes leading to incompatible behavior:
  - Labels of quantities calculates by SPRINT have changed from *label*.coord_*num* to *label*.coord-*num*
  - METAD with WALKERS_MPI now writes a single hills file, without suffixes
  - removed the ./configure.sh script of v2.0.x, now plumed can only be configured using autotools (./configure)
  - COM, CENTER, and GYRATION now automatically make molecules whole. In case you do not want them to do it, use NOPBC flag, which recovers plumed 2.1 behavior
  - Some MD code could now automatically trigger restart (e.g. gromacs when starting from cpt files). This can be overwritten using RESTART NO.
  - Replica suffixes are now added by PLUMED *before* extension (e.g. use plumed.0.dat instead of plumed.dat.0)
  - When using switchingfunction the STRETCH keyword is now implicit. NOSTRETCH is available to enforce the old behavior.
- Module activation can now be controlled during configure with `--enable-modules` option.
- Almost complete refactoring of installation procedure. Now DESTDIR and other standard autoconf directories (e.g. bindir) are completely supported. Additionally, everything should work properly also when directory names include spaces (#157). Finally, compiler is not invoked on install unless path are explicitly changed (#107).
- Related to installation refactoring, upon install a previously installed PLUMED is not removed. This is to avoid data loss if prefix variable is not properly set

- Several changes have been made in the Makefile.conf that makes it not compatible with those packaged with plumed 2.0/2.1. Please use ./configure to generate a new configuration file.
- Added partial OpenMP parallelization, see OpenMP
- Added multiple time step integration for bias potentials, see Multiple time stepping
- Link cells are now used in all multicolvars that involve switchingfunction. The link cell cutoff is set equal to 2.$*$ $d_{\max}$. Where $d_{\max}$ is the (user-specified) point at which the switching function goes to zero. Users should always set this parameter when using a switching function in order to achieve optimal performance.
- DHENERGY option is no longer possible within DISTANCES. You can still calculate the DHENERGY colvar by using DHENERGY
- Reweighting in the manner described in [3] is now possible using a combination of the METAD and HISTOGRAM actions. The relevant keywords in METAD are REWEIGHTING_NGRID and REWEIGHT↩ ING_NHILLS. The $c(t)$ and the appropriate weight to apply to the configurations are given by the values labeled rct and rbias.
- News in configure and install:
  - ./configure now allows external BLAS to be used with internal LAPACK. This is done automatically if only BLAS are available, and can be enforced with –disable-external-lapack.
  - ./configure supports –program-prefix, –program-suffix, and –program-transform-name.
  - make install supports DESTDIR and prefix.
  - Environment variables PLUMED_LIBSUFFIX and PLUMED_PREFIX are deprecated and will be removed in a later version.
- New actions
  - DUMPMASSCHARGE to dump a file with mass and charges during MD.
  - EFFECTIVE_ENERGY_DRIFT to check that plumed forces are not screwing the MD integrator.
  - EXTENDED_LAGRANGIAN : in combination with METAD it implements metadynamics with Extended Lagrangian; standalone it implements TAMD/dAFED.
  - DFSCLUSTERING calculate the size of clusters
  - DUMPMULTICOLVAR print out a multicolvar
  - MFILTER_LESS filter multicolvar by the value of the colvar
  - MFILTER_MORE
  - MFILTER_BETWEEN
  - PCARMSD PCA collective variables using OPTIMAL rmsd measure
  - PCAVARS PCA collective variables using any one of the measures in reference
  - GRADIENT can be used to calculate the gradient of a quantity. Used to drive nucleation
  - CAVITY
  - PUCKERING implemented for 5-membered rings (thanks to Alejandro Gil-Ley).
  - WRAPAROUND to fix periodic boundary conditions.
- New features for existing actions:
  - Keywords UPDATE_FROM and UPDATE_UNTIL to limit update step in a defined time window, available only for actions where it would be useful.
  - Keyword UNNORMALIZED for HISTOGRAM.
  - Possibility to use Tiwary-Parrinello reweighting for METAD
  - Keywords for GROUP (REMOVE, SORT, UNIQUE) to allow more flexible editing of groups.
  - DUMPATOMS now supports dumping xtc and trr files (requires xdrfile library).
  - driver can now read xtc and trr files also with xdrfile library.

  - driver accepts a –mc flag to read charges and masses from a file produced during molecular dynamics with DUMPMASSCHARGE
  - Possibility to enable or disable RESTART on a per action basis, available only for actions where it would be useful.
  - MOLINFO now supports many more special names for rna and dna (thanks to Alejandro Gil-Ley).
  - VMEAN and VSUM allow one to calculate the sum of a set of vectors calculated by VectorMultiColvar. Note these can also be used in tandem with AROUND or MFILTER_MORE to calculate the average vector within a particular part of the cell or the average vector among those that have a magnitude greater than some tolerance
  - New way of calculating the minimum value in multicolvars (ALT_MIN). This is less susceptible to overflow for certain values of β.
  - New keywords for calculating the LOWEST and HIGHEST colvar calculated by a multicolvar
  - Added components to DIPOLE (#160).

- Other changes:
  - File reader now supports dos newlines as well as files with no endline at the end.

For developers:

- In order to be able to use openMP parallelism within multicolvar, secondarystructure, manyrestraints and crystallisation we had to make some substantial changes to the code that underlies these routines that is contained within vesselbase. In particular we needed to get rid of the derivatives and buffer private variables in the class ActionWithVessel. As a consequence the derivatives calculated in the various performTask methods are stored in an object of type MultiValue. Within multicolvar this is contained within an object of type Atom↩ ValuePack, which stores information on the atom indices. If you have implemented a new multicolvar it should be relatively straightforward to translate them so they can exploit this new version of the code. Look at what has been done to the other multicolvars in there for guidance. Sorry for any inconvenience caused.
- Changed the logic of several PLUMED ifdef macros so as to make them consistent. Now every feature based on external libraries is identified by a __PLUMED_HAS_$*$ macro.

2.3.0.2 Version 2.2.1 (Jan 18, 2016)

For users:

- PBMETAD implement the new Parallel Bias Metadynamics flavor of the Metadynamics sampling method.
- PLUMED now reports an error when using HISTOGRAM with UNNORMALIZED without USE_ALL_DATA. See #175
- Fixed a bug in configure together with –enable-almost. The check for lbz2 library was not working properly.
- Fixed a bug in install procedure that was introducing an error in linking with CP2K.
- Fixed a bug that sometimes was preventing the printing of a useful error message.

For developers:

- Vector and Tensor now support direct output with $<<$.
- Added some missing matmul operation Vector and Tensor.
- ./configure is automatically relaunched when changing ./configure or Makefile.conf. This makes it more robust to switch between branches.

2.3.0.3 Version 2.2.2 (Apr 13, 2016)

For users:

- MOLINFO for RNA accepts more residue names, see #180.
- added two mpi barries (one was missing in PBMetaD for multiple walkers) to help synchronized initialisation
- Fixed a bug in internal stopwatches that was making DEBUG logRequestedAtoms not working
- Some multicolvars (including BRIDGE, ANGLES, and INPLANEDISTANCES) now crashes if one asks for too many atoms, see #185.
- Optimisations (activation of the dependencies, secondary structures, DRMSD)
- Fixed a performance regression with RMSD=OPTIMAL-FAST
- Fixed a bug in the normalization of kernel functions (relevant for HISTOGRAM).
- Fixed a regression introduced in v2.2 that was making METAD with non-MPI multiple walkers crash if reading frequently. See #190
- Updated patch for gromacs 5.x. Patches for gromacs 5.0 and 5.1 have been fixed so as to allow patching in runtime mode.
- Possibility to control manual generation (including pdf) from ./configure. Pdf manual is now off by default. Notice that on travis CI it is still generated.

For developers:

- Fixed a bug in the interpretation of cmd strings. Namely, an erroneous string was not triggering an error. This is harmless for MD codes properly patched, but could have introduced problems in MD codes with typoes in cmd strings.
- ./configure is not automatically relaunched anymore when doing `make clean`.

2.3.0.4 Version 2.2.3 (Jun 30, 2016)

For users:

- Updated patches for gromacs 5.1.x and 5.0.x to fix a problem when plumed was trying to write to an already closed gromacs log file.
- When looking for a value outside the GRID now the error include the name of the responsible collective variable
- Numerical check in LatticeReduction made less picky. This should solve some of the internal errors reported by `LatticeReduction.cpp` when using aggressive compilers.
- Files are now flushed at the correct step. Before this fix, they were flushed at the step before the requested one (e.g. with FLUSH STRIDE=100 at step 99, 199, etc).
- In METAD, INTERVAL with periodic variables now report an error.
- LOAD now works also when plumed is installed with a suffix.
- Added `--md-root` option to `plumed patch` which allows it to be run from a directory different from the one where the md code is located.
- Wham script in Munster tutorial tutorial now writes weights in scientific notation.

For developers:

- `./configure` checks if dependencies can be generated. If not, they are disabled.
- Added –disable-dependency-tracking to ./configure
- Added a make target `all_plus_doc` that builds both code and docs.
- Added possibility to set a default location for plumed library in runtime binding. If the plumed wrapped is compiled with `-D__PLUMED_DEFAULT_KERNEL=/path/libplumedKernel.so`, then if the env var PLUMED_KERNEL is undefined or empty PLUMED will look in the path at compile time.
- Tentative port files are now available at `this link`. They can be used to install PLUMED using MacPorts.

2.3.0.5 Version 2.2.4 (Dec 12, 2016)

For users:

- Fix a bug in PBMETAD when biasing periodic and not periodic collective variables at the same time
- GSL library is now treated by `./configure` in the same way as other libraries, that is `-lgsl -lgslcblas` are only added if necessary.
- Fix a bug in METAD when using INTERVAL and ADAPTIVE gaussians at the same time
- Updated gromacs patch for 5.1.x to 5.1.4
- Fix a performance regression in the calculate loop where derivatives and forces were set to zero even if an action was not active, this is relevant for postprocessing and for the on-the-fly analysis
- Torsion calculation has been made slightly faster and improved so as to provide correct derivatives even for special angles (e.g. +pi/2 and -pi/2).

For developers:

- Macports portile is now tested on travis at every plumed push.

2.3.0.6 Version 2.2.5 (Mar 31, 2017)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

For users:

- Fixed a problem with large step numbers in driver (see `#209`).
- Fixed a problem leading to crashes when using switching functions without cutoff with some compiler (see `#210`).
- Fixed a bug when using FIT_TO_TEMPLATE and domain decomposition (see `#214`).
- Added an automatic flush of HILLS files when using METAD with file-based multiple walkers.
- Root dir is logged to allow easier debugging of problems.

2.4 Version 2.3

2.4.0.1 Version 2.3 (Dec 12, 2016)

Version 2.3 contains several improvements with respect to 2.2. Users currently working with 2.2 should have a look at the section "Changes leading to incompatible behavior" below and might need tiny adjustments in their input files.

Below you find a list of all the changes with respect to version 2.2. Notice that version 2.3 includes already all the fixes in branch 2.2 up to 2.2.3 indicated in Version 2.2 .

Changes from version 2.2 which are relevant for users:

- Changes leading to incompatible behavior:
  - COMMITTOR can now be used to define multiple basins, but the syntax has been changed
  - Syntax for SPRINT and DFSCLUSTERING has changed. We have separated the Actions that calculate the contact matrix from these actions. These actions thus now take a contact matrix as input. This means that we these actions can be used with contact matrices that measures whether or not a pair of atoms are hydrogen bonded. For more details on this see Exploiting contact matrices. For clustering the output can now be passed to the actions CLUSTER_PROPERTIES, CLUSTER_DIAMETER, CLUSTER_NATOMS, OUTPUT_CLUSTER and CLUSTER_DISTRIBUTION. These provide various different kinds of information about the connected components found by clustering
  - In driver masses and charges are set by default to NaN. This makes it less likely to do mistakes trying to compute centers of mass or electrostatic-dependent variables when masses or charges were not set. To compute these variables from the driver you are now forced to use `--pdb` or `--mc`.
  - In rational switching functions, by default MM is twice NN. This is valid both in switchingfunction with expanded syntax and when specifying MM on e.g. COORDINATION
  - Patch script `plumed patch` now patches by default with `--shared`. This should make the procedure more robust (see #186).
  - Faster GYRATION but new default behavior is not mass weighted
  - When using HISTOGRAM you now output the accumulated grid using DUMPGRID or DUMPCUBE to get the free energy you use the method CONVERT_TO_FES. These changes allow one to use grids calculated within PLUMED in a work flow of tasks similarly to the way that you can currently use Values.
  - The way that reweighting is performed is now different. There are three separate actions REWEIGHT_BIAS, REWEIGHT_TEMP and REWEIGHT_METAD. These actions calculate the quantities that were calculated using the keywords REWEIGHT_BIAS and REWEIGHT_TEMP that used to appear in the old HISTOGRAM method. Now those these methods can be used in any methods that calculate ensemble averages for example HISTOGRAM and AVERAGE
  - Manual is now build with locally compiled plumed
  - Removed CH3SHIFT
  - CS2BACKBONE is now native in PLUMED removing the need to link ALMOST, small syntax differences
  - CS2BACKBONE, NOE, RDC, removed the keyword ENSEMBLE: now ensemble averages can only be calculated using ENSEMBLE
  - RDC, syntax changes
  - It is not possible anymore to select modules using `modulename.on` and `modulename.off` files. Use `./configure --enable-modules` instead.
  - Removed IMD modules. In case someone is interested in restoring it, please contact the PLUMED developers.
- New actions:
  - FIXEDATOM
  - HBOND_MATRIX

  - CLUSTER_PROPERTIES
  - CLUSTER_DIAMETER
  - CLUSTER_NATOMS
  - OUTPUT_CLUSTER
  - CLUSTER_DISTRIBUTION
  - ROWSUMS
  - COLUMNSUMS
  - UPDATE_IF
  - DUMPGRID
  - DUMPCUBE
  - CONVERT_TO_FES
  - INTERPOLATE_GRID
  - FIND_CONTOUR
  - FIND_SPHERICAL_CONTOUR
  - FIND_CONTOUR_SURFACE
  - AVERAGE
  - REWEIGHT_BIAS
  - REWEIGHT_TEMP
  - REWEIGHT_METAD
  - PCA
  - PRE
  - STATS
  - METAINFERENCE
  - LOCALENSEMBLE
  - FRET
  - RESET_CELL
  - JCOUPLING
  - ERMSD

- New features in MD patches (require re-patch):
  - Patch for amber 14 now passes charges with appropriate units (fixes #165). Notice that the patch is still backward compatible with older PLUMED version, but the charges will only be passed when using PLUMED 2.3 or later.
  - Patch for GROMACS 5.1 incorporates Hamiltonian replica exchange, see Using Hamiltonian replica exchange with GROM
  - Gromacs 2016, 5.1.x, 5.0.x, flush the plumed output files upon checkpointing
  - Added patch for Gromacs 2016.1
  - gromacs 5.1.x patch updated to 5.1.4
  - Removed the patch for Gromacs 4.6.x
  - LAMMPS patch updated to support multiple walkers and report plumed bias to LAMMPS (thanks to Pablo Piaggi).

- New features for existing actions:
  - The SPECIES and SPECIESA keyword in MultiColvars can now take a multicolvar as input. This allows one to calculate quantities such as the Q4 parameters for those atoms that have a coordination number greater than x.
  - Added MATHEVAL type in switchingfunction
  - Added Q type native contacts in switchingfunction (thanks to Jan Domanski).

  - COMMITTOR can now be used to define multiple basins
  - The number of atoms admitted in BRIDGE has been significantly increased, see #185.
  - driver now allows –trajectory-stride to be set to zero when reading with –ixtc/–itrr. In this case, step number is read from the trajectory file.
  - METAD and PBMETAD can now be restarted from a GRID
  - Added keywords TARGET and DAMPFACTOR in METAD
  - When using METAD with file-based multiple walkers and parallel jobs (i.e. mpirun) extra suffix is not added (thanks to Marco De La Pierre).
  - ENSEMBLE added keywords for weighted averages, and calculation of higher momenta
  - MOLINFO now allows single atoms to be picked by name.
  - FIT_TO_TEMPLATE now supports optimal alignment.
  - CONSTANT added the possibility of storing more values as components with or without derivatives
  - PUCKERING now supports 6 membered rings.
  - Extended checkpoint infrastructure, now METAD and PBMETAD will write GRIDS also on checkpoint step (only the GROMACS patch is currently using the checkpointing interface)

- Other features:

  - Added a plumed-config command line tool. Can be used to inspect configuration also when cross compiling.
  - Added a `--mpi` option to `plumed`, symmetric to `--no-mpi`. Currently, it has no effect (MPI is initialized by default when available).
  - PLUMED now generate a VIM syntax file, see Using VIM syntax file
  - The backward cycle is now parallelized in MPI/OpenMP in case many collective variables are used.
  - GSL library is now searched by default during `./configure`.
  - Tutorials have been (partially) updated to reflect some of the changes in the syntax
  - Parser now reports errors when passing numbers that cannot be parsed instead of silently replacing their default value. See #104.
  - More and more documentation

- Bug fixes:

- Fixed a bug in PBMETAD that was preventing the writing of GRIDS if a hill was not added in that same step

For developers:

- IMPORTANT: BIAS can now be BIASED as well, this changes can lead to some incompatibility: now the "bias" component is always defined automatically by the constructor of Bias as a componentWithDerivatives, derivatives are automatically obtained by forces. The main change is that you don't have to define the bias component anymore in your constructor and that you can use setBias(value) to set the value of the bias component in calculate.

- Added new strings for plumed cmd: setMDMassUnits, setMDChargeUnits, readInputLine, performCalcNo↩ Update, update and doCheckPoint.

- Easier to add actions with multiple arguments

- New functions to access local quantities in domain decomposition

- Active modules to enable regtests are chosen using `plumed config`.

- A script is available to check if source code complies plumed standard. Notice that this script is run together with cppcheck on travis-ci.

- Cppcheck on travis-ci has been updated to 1.75. Several small issues triggering errors on 1.75 were fixed (e.g. structures passed by value are now passed by const ref) and false positives marked as such.

- Added coverage scan.

2.4.0.2 Version 2.3.1 (Mar 31, 2017)

- Fix to FIT_TO_TEMPLATE as in 2.2.5. Notice that in 2.3.0 also the case with TYPE=OPTIMAL was affected. This is fixed now.
- small change in CS2BACKBONE to symmetrize the ring current contribution with respect to ring rotations (also faster)
- fixed `plumed-config` that was not working.
- log file points to the `config.txt` files to allow users to check which features were available in that compiled version.
- `make clean` in root dir now also cleans `vim` sub-directory.
- Updated gromacs patch to version 2016.3

For developers:

- Cppcheck on travis-ci has been updated to 1.77.
- Doxygen on travis-ci has been updated to 1.8.13

2.4.0.3 Version 2.3.2 (Jun 12, 2017)

See branch `v2.3` on git repository.

- Resolved problem with nan in SMAC with SPECIESA and SPECIESB involving molecules that are the same
- PDB reader is now able to read files with dos newlines (see `#223`).
- Fixed bug in CS2BACKBONE (v2.3.1) related to ring currents of HIS and TRP
- Fixed bug in if condition in PCAVARS so that you can run with only one eigenvector defined in input
- Fixed bug with timers in sum_hills `#194`.
- Fixed bug when using MOVINGRESTRAINT with periodic variables such as TORSION `#225`.
- Fixed bug in HBOND_MATRIX that used to appear when you used DONORS and ACCEPTORS with same numbers of atoms
- Fixed bug in DISTANCES that appears when using BETWEEN and link cells.
- Prevented users from causing segfaults by storing derivatives without LOWMEM flag. In these cases PLUMED crashes with meaningful errors.
- Fixed bug in HISTOGRAM that causes NaNs when using KERNEL=DISCRETE option
- Fixed a bug in the parser related to braces, see `#229`
- Fixed a bug that appeared when using Q3, Q4 and Q6 with LOWEST or HIGHEST flag
- Fixed a bug that appears when you use MFILTER_LESS as input to COORDINATIONNUMBER with SPECIESA and SPECIESB flags
- Fixed a bug that was making flushing when gromacs checkpoints not functional (thanks to Summer Snow).

- Fixed a bug affecting EXTENDED_LAGRANGIAN and METAD with ADAPT=DIFF when using an argument with periodicity (min,max) such that min is different from -max. This does not affect normal TORSION, but would affect PUCKERING component phi with 6-membered rings. In addition, it would affect any variable that is created by the user with a periodicity domain not symmetric around zero. See #235 (thanks to Summer Snow for reporting this bug).
- Fixed numerical issue leading to simulations stuck (LatticeReduction problem) with intel compiler and large simulation cells.
- Fixed a bug affecting LOCAL_AVERAGE and outputting all multicolvars calculated by Q6 with DUMPMULTICOLVAR
- `plumed info --user-doc` and `plumed info --developer-doc` now fall back to online manual when local doc is not installed, see #240.

For developers:

- IMPORTANT: we started to enforce code formatting using astyle. Check the developer documentation to learn how to take care of not-yet-formatted branches.
- plumedcheck validation has been made stricter. All the checks are now described in the developer manual.
- New flag `--disable-libsearch` for `configure`, allowing an easier control of linked libraries when installing PLUMED with a package manager such as MacPorts.
- Added `--disable-static-patch` to `./configure` to disable tests related to static patching. It can be used when static patching is not needed to make sure a wrong c++ library is not linked by mistake.
- Using `install_name_tool` to fix the name of the installed library on OSX. Allows linking the PLUMED shared library without explicitly setting `DYLD_LIBRARY_PATH`.
- Added environment variable `PLUMED_ASYNC_SHARE` to enforce synchronous/asynchronous atom sharing (mostly for debug purpose).
- On travis-ci, using ccache to speedup builds.
- On travis-ci, added a regtest using Docker with gcc6 and MPI.
- On travis-ci, docs for unofficial or unsupported branches are set not to be indexed by search engines (see #239)
- Cppcheck on travis-ci has been updated to 1.79.

2.4.0.4 Version 2.3.3 (Oct 3, 2017)

For users:

- Fixed a bug in switchingfunction MATHEVAL, leading to inconsistent results when using OpenMP with multiple threads (see #249).
- FIT_TO_TEMPLATE now reports when it is used with a reference file with zero weights.
- Fixed logging of UNITS (thanks to Omar Valsson).
- Fixed a possible bug with EFFECTIVE_ENERGY_DRIFT and domain decomposition with a domain containing zero atoms.

For developers:

- Fixed a bug in `./configure --disable-libsearch` when searching for molfile plugins.
- Cppcheck on travis-ci has been updated to 1.80.
- Configure script now has a list of better alternatives to find a working `ld -r -o` tool to merge object files. This solves linking issues on some peculiar systems (see #291, thanks to Massimiliano Culpo).
- Using `install_name_tool` also on non-installed libraries. This makes it possible to link them and later find them without explicitly setting `DYLD_LIBRARY_PATH`. This should also make the `DYLD_LIBRARY_PATH` irrelevant. Notice that `DYLD_LIBRARY_PATH` is not well behaved in OSX El Capitan.

2.4.0.5 Version 2.3.4 (Dec 15, 2017)

For users:

- GROMACS patch updated to gromacs-2016.4. This patch was also fixed in order to properly work with ENERGY (see #316) and to implement `-hrex` option (see #197).
- Patch for GROMACS 5.1.4 updated to fix an error with ENERGY (see #316).
- Solved a bug in ERMSD leading to incorrect results when using non-default length units (e.g. with `UNITS LENGTH=A`).

For developers:

- Regtest script also reports when exitcode different from zero is returned.
- Patch script reports errors returning a nonzero exit code.
- cppcheck update to 1.81
- Solved small bug in stored PLUMED_ROOT directory as obtained from statically patched MD codes. Namely, the compilation directory was stored rather than the installation one.

2.4.0.6 Version 2.3.5 (Mar 2, 2018)

For users:

- Fixed `plumed partial_tempering` to agree with GROMACS conventions for the choice of dihedral angles (see #337). Should be irrelevant for the vast majority of cases.
- Fixed small bug in regexp parser - the part outside the parentheses was just ignored.

For developers:

- Doxygen on travis-ci has been updated to 1.8.14.
- Embedded astyle updated to 3.1.
- `make clean` now correctly removes the `src/lib/plumed` executable.

2.4.0.7 Version 2.3.6 (Jul 2, 2018)

For users:

- Fixed a problem leading to NaN derivatives of switchingfunction `Q` when distance between two atoms is large.
- GROMACS patch updated to gromacs-2016.5.
- `./configure` crashes if prefix is set to present working directory (notice that this choice was already leading to issues).
- DUMPATOMS reports an error when trying to write xtc/xdr files without the xdrfile library installed.
- Fixed a bug appearing when using PATH or GPROPERTYMAP with virtual atoms without simultaneously using the same atoms in a different action.
- Fixed incorrect format of the pdb file written by PCA (see #363).
- Fixed behavior of natural units. When an MD code asks for natural units, it is not necessary to also set units within PLUMED using UNITS (see #364).

For developers:

- Fixed small issue in debug options of driver (see #245).
- `plumed patch -e` now accepts a name closely matching the patch name (e.g. `plumed patch -e gromacs2016.5` will try to patch even if the stored patch is for `gromacs-2016.4`). This simplifies managing Portfiles. Nothing changes when picking the patch from the interactive menu.
- Install newer ccache on travis-ci, build faster.
- Small fix in provided env modules (`PLUMED_VIMPATH` is set also when shared libraries are disabled).

2.4.0.8 Version 2.3.7 (Oct 5, 2018)

For users:

- Fixed flag DETAILED_TIMERS in DEBUG (flag was ignored and detailed timers always written).
- Small fix in DUMPMASSCHARGE (atoms are now correctly requested only at first step).

2.4.0.9 Version 2.3.8 (Dec 19, 2018)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

For users:

- Fixed some openMP regression (some related to the whole codes and some specifics for Coordination and Multicolvar), this were compiler dependent so not all users may have experienced them
- Fixed an issue with CS2BACKBONE when more than 2 chains were used
- Fixed memory leak in RDC.
- Fixed segmentation fault with more than two CVs in reweighting METAD (see #399, thanks to Fiskissimo).

For developers:

- Small fix in LDFLAGS when enabling coverage.
- Fixed order of flags in tests for static linking done by configure (see #407).
- Fixed the way paths are hard-coded so as to facilitate conda packaging (see #416).

*/

2.5 Version 2.4

2.5.0.1 Version 2.4 (Dec 15, 2017)

Version 2.4 contains several improvements with respect to 2.3. Users currently working with 2.3 should have a look at the section "Changes leading to incompatible behavior" below and might need tiny adjustments in their input files. Notice that version 2.4 includes already all the fixes in branch 2.3 up to 2.3.3 indicated in Version 2.3 .

Changes from version 2.3 which are relevant for users:

- Changes leading to incompatible behavior:
  - A c++11 compliant compiler is required (see #212). This should mean:
    - gcc 4.8
    - clang 3.3
    - intel 15 Since the number of c++11 features that we use is limited, older compilers might work as well.
  - The meaning of `BIASFACTOR=1` in METAD has been modified and can now be used to indicate unbiased simulations. Non-well-tempered metadynamics is BIASFACTOR=-1, which is the new default value. Notice that this has an implication on the bias factor written in the HILLS file when doing non-well-tempered metadynamics.
  - Due to a change in COMMITTOR, the format of its output file has been slightly changed.
  - HISTOGRAM : When using weights default is now to output histogram divided by number of frames from which data was taken. In addition the UNORMALIZED flag has been replaced with the keyword `NORMALIZATION`, which can be set equal to true, false or ndata.
  - All switching functions are now stretched by default, also when using the "simple syntax" (e.g. `COORDINATION NN=6`). Switching functions were already stretched by default when using the advanced syntax (e.g. `COORDINATION SWITCH={}`) since version 2.2. Notice that this will introduce small numerical differences in the computed switching functions.
- New modules:
  - A new PLUMED-ISDB module have been included, this module includes a number of CVs to calculate experimental data with the internal ability to also calculate a METAINFERENCE score.
    - New actions include:
      - EMMI
      - SAXS
      - RESCALE, SELECT, SELECTOR
    - Updated actions include:
      - CS2BACKBONE
      - FRET
      - JCOUPLING
      - METAINFERENCE
      - NOE
      - PRE
      - RDC, PCS
      - PBMETAD
  - A new EDS module have been included, contributed by Glen Hocky and Andrew White. This module implements the following methods:
    - EDS
  - A new DRR module have been included, contributed by Haochuan Chen and Haohao Fu. This module implements the following methods:

    - DRR
    - drr_tool
  - A new VES module have been included, contributed by Omar Valsson. This module implements the following methods:
    - BF_CHEBYSHEV
    - BF_COMBINED
    - BF_COSINE
    - BF_CUSTOM
    - BF_FOURIER
    - BF_LEGENDRE
    - BF_POWERS
    - BF_SINE
    - OPT_AVERAGED_SGD
    - OPT_DUMMY
    - TD_CHI
    - TD_CHISQUARED
    - TD_CUSTOM
    - TD_EXPONENTIAL
    - TD_EXPONENTIALLY_MODIFIED_GAUSSIAN
    - TD_GAUSSIAN
    - TD_GENERALIZED_EXTREME_VALUE
    - TD_GENERALIZED_NORMAL
    - TD_GRID
    - TD_LINEAR_COMBINATION
    - TD_PRODUCT_COMBINATION
    - TD_PRODUCT_DISTRIBUTION
    - TD_UNIFORM
    - TD_VONMISES
    - TD_WELLTEMPERED
    - VES_LINEAR_EXPANSION
    - VES_OUTPUT_BASISFUNCTIONS
    - VES_OUTPUT_FES
    - VES_OUTPUT_TARGET_DISTRIBUTION
    - ves_md_linearexpansion
- New collective variables:
  - DIMER (thanks to Marco Nava).
  - EEFSOLV : EEF1 implicit solvent solvation energy
  - ADAPTIVE_PATH : Adaptive path variables using the method from [4]
- New actions:
  - INENVELOPE
  - TOPOLOGY_MATRIX
  - BOND_DIRECTIONS
  - DUMPGRAPH
  - GRID_TO_XYZ
  - INTEGRATE_GRID
  - LWALLS
  - MAXENT

  - MCOLV_COMBINE
  - MCOLV_PRODUCT
  - POLYMER_ANGLES
  - XANGLES , YANGLES , ZANGLES
  - XYTORSIONS , XZTORSIONS , YXTORSIONS , YZTORSIONS , ZXTORSIONS , and ZYTORSIONS

- New command line tools:
  - pesmd : Tool for performing Langevin dynamics on an energy landscape that is specified using a PLUMED input file
  - pathtools

- Other changes:
  - Sharing coordinates and applying force is now faster (in some cases these can result in much better scaling of the performances in parallel).
  - COMMITTOR : new flag to use committor to keep track of the visited basins without stopping the simulation
  - PBMETAD : multiple walkers using files (thanks to Marco De La Pierre).
  - PBMETAD : adaptive Gaussian kernels
  - PBMETAD : default names for `GRID` and `FILE` (useful with many collective variables)
  - METAD : BIASFACTOR=1 is allowed and performs unbiased sampling. HILLS file can be used to recover free energy also in this case.
  - METAD : a RECT option is available that allows setting an array of bias factors, one for each replica.
  - METAD : added options to perform Transition Tempered Metadynamics (thanks to James Dama)
  - PATHMSD and PROPERTYMAP now support alignment to a close structure (thanks to Jana Pazurikova)
  - PDB files with more than 100k atoms can now be read using `hybrid 36` format, see #226.
  - Added lepton support. Set env var `export PLUMED_USE_LEPTON=yes` to activate lepton as a matheval replacement in MATHEVAL, CUSTOM, and MATHEVAL switching function. Notice that in v2.5 matheval support will be dropped and all these keywords will use lepton. See #244.
  - When parsing constants, PLUMED uses lepton library. This allows to pass arguments such as `HEIGHT=exp(0.5)` (see Parsing constants).
  - CUSTOM function has been added as an alias to MATHEVAL .
  - Trajectories read in driver also support the usual replica convention, that is if trajectory with replica suffix is not found the driver will look for a trajectory without the replica suffix.
  - A new syntax (`@replicas:`) can be used to specify different arguments for different replicas (see Special replica syntax).
  - Internal molfile implementation has been updated to VMD 1.9.3.
  - Examples in the documentation now have syntax highlighting and links to the documentation of used actions.
  - COORDINATIONNUMBER : Added option to have pairwise distance moments of coordination number in the multicolvar module
  - GROMACS patch updated to gromacs-2016.4
  - Implemented HREX for gromacs-2016.4.
  - Added patch for Quantum ESPRESSO 6.2 (thanks to Ralf Meyer).
  - Fixed a bug in LOCAL_AVERAGE which appears when you use `SPECIESA` and `SPECIESB` keywords instead of just `SPECIES`
  - Added possibility to pass `--kt` from driver.

Changes from version 2.3 which are relevant for developers:

- A few fixes has been made to improve exception safety. Although we still cannot declare PLUMED totally exception safe (there are still many non-safe pointers around), this made it possible to add a regtest that actually tests erroneous cmd strings and erroneous inputs.
- Due to the required c++11 support, travis-ci test on Ubuntu Precise has been removed.
- `gettimeofdate` and `gettime` have been replaced with portable `chrono` classes introduced in c++11.
- C++ exceptions are enabled by default.
- A large number of loops have been changed to use the `auto` keyword in order to improve code readability.
- Stack trace is not written upon error anymore, unless environment variable `PLUMED_STACK_TRACE` is set at runtime.
- Fixed a potential bug using single precision system BLAS on a mac (notice that currently plumed only uses double precision, so it is harmless).
- Added `--enable-rpath` option for autoconf (off by default).
- Files related to changelog are now stored as `.md` files. This makes it possible to navigate them from github.
- `configure.ac` has been simplified and improved in order to more easily probe C++ libraries.
- added `plumed_custom_skip` function to regtests in order to skip specific tests based on specific conditions (e.g. OS).
- environment variable `LDSO` has been renamed to `LDSHARED`, which is standard in the python community.
- a `libplumedWrapper.a` library is installed as well, that is used in `--runtime` patching.
- pkgconfig files are installed.
- `plumed config makefile_conf` can be used to retrieve `Makefile.conf` file a posteriori.
- Store `MPIEXEC` variable at configure time and use it later for running regtests. Notice that in case `MPIEXEC` is not specified regtests will be run using the command stored in env var `PLUMED_MPIRUN` or, if this is also not defined, using `mpirun`.
- Added canonical Makefile targets `check` and `installcheck`. Notice that `check` runs checks with non-installed plumed whereas `installcheck` uses the installed one, including its correct program name if it was personalized (e.g. with suffixes). Notice that this modifies the previously available `check` target.

2.5.0.2 Version 2.4.1 (Mar 2, 2018)

For users:

- Fixed an important bug affecting RMSD calculations with compilers supporting OpenMP 4 (e.g.: intel compiler). Notice that this bug might potentially affect not only RMSD variable, but also PATHMSD variables using RMSD, FIT_TO_TEMPLATE, PCAVARS, and possibly other variables based on RMSD calculations and optimal alignments (see #343). Results might depend on the exact architecture and on how aggressive is the compiler. The bug is a consequence of some erroneous SIMD directives introduced in 2.4.0, so it does not affect PLUMED 2.3.x.
- Resolved a problem with CS2BACKBONE and glycine atom names.
- Module VES: Fixed a bug with basis functions that have a constant function different from 1 (e.g. scaled version of the Legendre basis functions, BF_LEGENDRE) that was causing a time-dependent shift in the bias potential.

- Module VES: In optimizers (OPT_AVERAGED_SGD and OPT_DUMMY) the output of quantities related to the instantaneous gradients are now off by default as these quantities are generally not useful for normal users, their output can instead by re-enabled by using the MONITOR_INSTANTANEOUS_GRADIENT keyword. Also added an keyword MONITOR_AVERAGE_GRADIENT that allows to monitor the averaged gradient and output quantities related to it.
- RMSD variable and other collective variables using reference PDB files now crash when zero weights are passed (see #247).
- Using COM with driver without passing masses now triggers an error instead of reporting NaNs (see #251).

For developers:

- plumed patch -p command can be used twice without triggering an error. This will allow e.g. building again on MacPorts in cases where the build was interrupted. Notice that this only works for patches without special after/before patch/revert functions.

2.5.0.3 Version 2.4.2 (Jul 2, 2018)

For users:

- All fixes done in version 2.3.6. Notice that #363 in version 2.4 also applies to pathtools.
- Additional residue names (without the prefix D) are now supported by MOLINFO for DNA. See #367.
- Solved an important bug appearing in NAMD interface. Notice that the bug was a regression introduced in 2.4.0. As consequence, versions <= 2.3 and versions >=2.4.2 are expected to work correctly. See #254.
- GROMACS patch for gromacs-2018.1.
- Using VIM syntax file now highlights __FILL__ strings.
- METAD and PBMETAD give a warning when one restarts a simulation and the old hills file is not found. See #366.

For developers:

- LDSHARED is now correctly taken into account when launching ./configure.
- Fixed installation with --disable-shared.
- Cppcheck upgraded to 1.84.

2.5.0.4 Version 2.4.3 (Oct 5, 2018)

For users:

- All fixes done in version 2.3.7.
- Module VES: Fixed a bug in TD_GRID for 2D grids where the grid spacing is not the same for both dimensions.
- GROMACS patch for gromacs-2018.3.

2.5.0.5 Version 2.4.4 (Dec 19, 2018)

For users:

- Fixed some performances regression issue with OpenMP
- Updated NAMD patches to version 2.12 and 2.13. Old patches have been removed.
- GROMACS patch for gromacs-2018.4.
- Fixed a thread safety issue using forces on HISTOGRAM
- Fixed error message suggesting wrong actions (see #421).

For developers:

- All fixed done in version 2.3.8
- Cppcheck updated to 1.85

2.5.0.6 Version 2.4.5 (Apr 1, 2019)

For users:

- Fixed an inconsistency in parsing of braces. It is now possible to pass individual options including spaces (e.g. with `FILE={/path with space/file}`). Notice that this invalidates syntax such as `ATOMS={1}{2}{3}{4}`. See more at #434.
- Fixed simplemd so as to call "runFinalJobs" at the end of the simulation.
- GROMACS patch for gromacs-2016.6.
- GROMACS patch for gromacs-2018.6.
- Added aliases for some actions/options containing dashes (`-`) in their name. This will improve backward compatibility when these actions/options will be removed (see #449).

2.5.0.7 Version 2.4.6 (Jul 19, 2019)

For users:

- Fixed a bug in COORDINATIONNUMBER where derivatives were wrong when using `R_POWER` $>$ 2, thanks to `@MoleOrbitalHybridAnalyst` for spotting and fixing
- Fixed a bug in library search, possibly affecting linked blas/lapack on OSX (see #476).
- Fixed a bug in METAD with `TARGET` and `GRID_SPARSE` (see #467).

2.5.0.8 Version 2.4.7 (Jan 27, 2020)

For users:

- Fixed a bug with CONVERT_TO_FES and periodic variables, see #441 (backported from v2.5.3).
- More robust backup for output files when running over multiple processes
- Fixed a regression in the performances of GEOMETRY based flexible hills in METAD and PBMETAD
- Fixed #538.
- Fixed potential issue with VMD plugins from 1.9.4 (#545, thanks to Lixin Sun).
- Module VES: Fixed an off-by-one bug in the output of target distribution averages. The bug only affects the output and does not affect results. The bug also affected the output of coefficients when using a bias cutoff.
- Module VES: Made sure that all relevant output files are written out at the final step when shutting down the simulation. This solves issues reported by @PabloPiaggi with restarting when there is a mismatch been the output of files and the number of MD steps.

2.5.0.9 Version 2.4.8 (Jul 8, 2020)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

For users:

- Take into account UNITS when using MD codes that feeds one line at a time to PLUMED (e.g., OpenMM). See #582.
- Fix PDB parser for non justified atom numbers. See #592.
- Fix in INPLANEDISTANCES. See #595.

For developers:

- Tests and doc building moved from Travis-CI to GitHub Actions (see #634).

2.6 Version 2.5

2.6.0.1 Version 2.5 (Dec 19, 2018)

This page contains changes that will end up in 2.5

Changes from version 2.4 which are relevant for users:

- Changes leading to incompatible behavior:
  - RMSD, MULTI-RMSD, PATHMSD, PROPERTYMAP, PCAVARS, PCARMSD, FIT_TO_TEMPLATE, DIPOLE, ALPHARMSD, ANTIBETARMSD, and PARABETARMSD now automatically make molecules whole. In case you do not want them to do it, use NOPBC flag,

  - There is some subtle change in the installation layout (see below). There should be no visible effect, however it is now compulsory to set correctly the `LD_LIBRARY_PATH` variable for the linux executable to work correctly. The procedure has been tested well on OSX and Linux, but could give problems on other platform. Please report possible problems on the mailing list.
  - driver now stops correctly when using COMMITTOR. If you want to continue the analysis, use the `NOSTOP` flag in COMMITTOR.
  - METAD the calculation of the reweighting factor is now activated by CALC_RCT instead of REWEIG↩ HTING_NGRID and REWEIGHTING_NHILLS, the frequency of update can be set by RCT_USTRIDE, the default value is 1 and should be OK for most of the cases
  - Fixed sign in Cartesian components of PUCKERING with 6 membered rings (thanks to Carol Simoes and Javi Iglesias).

- New actions:
  - COLLECT_FRAMES
  - EUCLIDEAN_DISSIMILARITIES
  - HBPAMM_MATRIX
  - HBPAMM_SH
  - LANDMARK_SELECT_FPS
  - LANDMARK_SELECT_RANDOM
  - LANDMARK_SELECT_STAGED
  - LANDMARK_SELECT_STRIDE
  - OUTPUT_ANALYSIS_DATA_TO_COLVAR
  - OUTPUT_ANALYSIS_DATA_TO_PDB
  - OUTPUT_PCA_PROJECTION
  - PAMM
  - PLUMED
  - PRINT_DISSIMILARITY_MATRIX
  - PROJECT_ALL_ANALYSIS_DATA
  - READ_DISSIMILARITY_MATRIX
  - RESELECT_LANDMARKS
  - REWEIGHT_WHAM
  - SKETCHMAP_CONJGRAD
  - SKETCHMAP_POINTWISE
  - SKETCHMAP_READ
  - SKETCHMAP_SMACOF
  - SKETCH_MAP
  - SMACOF_MDS
  - WHAM_HISTOGRAM
  - WHAM_WEIGHTS

- New command line tools:
  - completion (used to generate command line completion scripts).
  - pdbrenumber (see #371).

- New modules:
  - A new PIV module has been included, contributed by Silvio Pipolo and Fabio Pietrucci. This module implements the following collective variable:
    * PIV

  - A new LOGMFD module has been included, contributed by Tetsuya Morishita. This module implements the following bias:
    * LOGMFD

- Changes in the ISDB module
  - CS2BACKBONE is now mpi parallelized in particular with DOSCORE and CAMSHIFT
  - SAXS has an additional implementation based on Bessel functions that can be faster for large systems (new keyword BESSEL)
  - SAXS keyword SCEXP has been renamed into SCALEINT
  - SAXS includes the MARTINI bead structure factors for Proteins and Nucleic Acids
  - SAXS includes a GPU implementation based on ArrayFire (need to be linked at compile time) that can be activated with GPU
  - METAINFERENCE and all related methods has a new keyword REGRES_ZERO to scale data using a linear scale fit
  - CALIBER new bias to perform Maximum Caliber replica-averaged restrained simulations
- Changes in the eABF/DRR module (contributed by Haochuan Chen and Haohao Fu):
  - DRR now supports the extended generalized ABF(egABF) method.
  - DRR accepts different GRID options for CVs and extended variables.
  - The MAXFACTOR option is added in DRR to control the factor of biasing force.
  - drr_tool can calculate the divergence of gradients now. (Maybe useful for future pABF)
  - Fixed conflicts of output files in multiple replicas.
- Changes in the EDS module:
  - EDS implements Levenberg-Marquardt optimization in addition to previous gradient descent.
  - EDS no longer automatically increases prefactor for bias parameter updates. This results in more stable optimization for the cases tested.
  - EDS now has a larger default RANGE parameter to go with these other changes.
- Other changes:
  - METAD there is a new FLYING_GAUSSIAN keyword to activate the flying gaussian methods by Spiwok (contributed by Spiwok and Hozzova)
  - EXTERNAL can now SCALE the input grid. This allows for more flexibility without modifying the grid file.
  - ALPHABETA can now combine dihedral angles with different coefficients
  - INCLUDE can now be used also before setup actions.
  - CENTER can now be computed using trigonometric functions (PHASES) to simplify its calculation with periodic boundary conditions.
  - Libmatheval is not used anymore. MATHEVAL (and CUSTOM) are still available but employ an internal implementation of the lepton library. Functions available in libmatheval and absent in the original lepton library have been added so as to have backward compatibility. `atan2(y,x)` function has also been added. Notice that MATHEVAL (and CUSTOM) switching functions using the lepton library have been further optimized with respect to PLUMED 2.4. Finally, notice that it is possible to use asmjit to optimize performance (see Making lepton library faster).
  - Implemented bash autocompletion, see Using bash autocompletion.
  - MOLINFO now allows selecting atoms from chains with a numeric ID (see `#320`).
  - Removed the patch for GMX 5.1.4
  - LAMMPS patch has been finally removed. Notice that LAMMPS has native support for PLUMED now.
  - AMBER patch has been finally removed. Notice that AMBER (sander module) has native support for PLUMED starting from version 15.

  - RMSD calculation has been optimized. This should positively affect the performances of CVs where many RMSD values are computed on small groups of atoms, such as secondary structure variables.
  - In METAD, when using a bias factor equal to one (no bias) the `rct` component is set to zero rather than to one.
  - New shortcuts are available for selecting atoms: `@allatoms` and `@mdatoms` (see Specifying Atoms).
  - When using MOLINFO, also the following shortcuts are available for selecting atoms: `@nucleic`, `@protein`, `@water`, `@ions`, `@hydrogens`, `@nonhydrogens`.
  - When using MOLINFO, individual atoms can be chosen also from water molecules (e.g. `@OW-100`).
  - Additional switching function COSINUS contributed by Michael King
  - added API to set the number of used openMP threads from the linked code, updated gromacs 2018.3 patch to use it

Changes from version 2.4 which are relevant for developers:

- Code has been cleanup up replacing a number of pointers with `std::unique_ptr`. All `delete` statements in the core parts of the code have been eliminated.
- Exceptions cannot be disabled (`--disable-cxx-exceptions` option has been removed from `./configure`).
- Every exception thrown in PLUMED now also writes its message on PLUMED log.
- Runtime loader in `Plumed.c` now works also when linked without `-rdynamic` (that is, its names are not exported). Notice that all the combinations are expected to work, that is: `Plumed.c` from <=2.4 or >=2.5 combined with libplumedKernel from <=2.4 or >=2.5. In order to achieve this the following changes are implemented:
  - libplumedKernel does not depend anymore on `Plumed.c`. This allows loading it even in cases where names in the loader are not visible. The relevant function needed to be compatible with `Plumed.c` <=2.4 are found using `dlsym`.
  - `Plumed.c` does not need anymore libplumedKernel to register itself, but rather searches the relevant functions using `dlsym`. In addition, if it is not able to load `libplumedKernel` since the latter is <=2.4 and needs `Plumed.c` to be visible, it just uses as a fallback `libplumed`, which should load properly.
- In addition to the capability mentioned above, the MD-code interface has been significantly improved and allows for:
  - Translation of exception (allowing to mix PLUMED and an MD-code linked against a different C++ library).
  - Possibility to choose the path to the PLUMED kernel while instantiating a Plumed object. See the developer documentation for more information.
- The installation layout of shared libraries has been modified. In particular, both `libplumed.so` and `plumed` links to `libplumedKernel.so`. This reduces considerably the size of the installed package. In addition, it allows using two-level namespace on OSX. Notice that this implies that on Linux one should always set the `LD_LIBRARY_PATH` flag to have a working executable.
- A smaller number of header files is installed. In particular, all the files that were historically generated in subdirectories (such as 'plumed/core/tools/Vector.h', just including `plumed/tools/Vector.h`) are not installed and the related include statements are fixed. This makes the installed package smaller.
- List of preferred compilers (used when `CXX` or `CC` are not set) has been changed. On OSX, `./configure` will try `clang++/clang` as first choices.
- Added `--enable-static-archive` to `./configure` to build a `libplumed.a` static library (yes by default).

- Stop setting `DYLD_LIBRARY_PATH` in `sourceme.sh` and in modulefile. Notice that as of PLUMED v2.3.3 it should not be needed.
- Coverage scan is not anymore contained in developer manual. It can be found in a separate repository `github.com/coverage-branch` (see #348). In addition, coverage for third-party libraries included in PLUMED is reported as well.
- It is not possible anymore to use `make install prefix=/path`. Prefix can only be changed during `./configure` (see #332).
- Exception class has been rewritten to allow more extensive messages. Now also function name is shown.
- On linux, library is linked with `-Bsymbolic`.
- When launching `plumed`, flags `--no-mpi` and `--mpi` can appear multiple times. The last appearance is the effective one.
- Internal BLAS and LAPACK libraries updated to gromacs 2018.
- Choosing `./configure --prefix=$PWD` does not lead anymore to deletion of all header files.
- A copy of `plumed-runtime` is installed in `prefix/lib/plumed` and can be used for testing.
- Absolute/relative soname/install_name can be configured on linux/OSX. This feature is only for testing, the default choice is the typical one used on the respective operating system.
- On OSX, `plumed` and `libplumed.dylib` will find `libplumedKernel.dylib` using `@loader_path`.
- Using CXX compiler to link the main program.
- plumed can be compiled with ArrayFire to enable for gpu code. SAXS collective variable is available as part of the isdb module to provide an example of a gpu implementation for a CV

2.6.0.2 Version 2.5.1 (Apr 1, 2019)

For users:

- in SAXS the keyword ADDEXP is removed. Furthemore, SAXS intensities are automatically normalised for I(0)=1, in case experimental data are provided, the intensity is rescaled with the intensity of the lowest q provided. As a consequence SCALEINT is only needed for additional adjustments.
- gromacs patch updated to gromacs 2018.5
- Fixed a bug in gromacs patch that was resulting in incorrect number of threads (0) set when not explicitly using `-ntomp` on the command line or setting `OMP_NUM_THREADS` (see #446). To apply this fix you need to re-patch gromacs. Notice that setting the number of threads to zero might lead to inconsistent results when using secondary structure variables or other multicolvars.
- Fixed PLUMED so that when zero threads are selected from gromacs (see previous fix) the number of used threads is set to 1. This fix allows to use a GROMACS executable patched with PLUMED 2.5.0 and linked at runtime with PLUMED 2.5.1 without introducing errors. However, re-patching is preferred since it selectes the correct number of threads.
- Python wrappers:
  - Fixed building of python interface on MacOS Mojave (see #445, thanks to Omar Valsson).
  - Numpy is not required anymore at build time (though it is required at runtime for our tests).
  - Raw python arrays can be passed as an alternative to Numpy ndarrays.

2.6.0.3 Version 2.5.2 (Jul 19, 2019)

For users:

- New shortcuts are available for selecting protein atoms: `@chi2-#`, `@chi3-#`,`@chi4-#` and `@chi5-#`
- Fixed performance of CUSTOM when having zero derivatives with respect to some arguments.
- New –parse-only option in driver to check the validity of a plumed input file
- New patch for GROMACS 2019.2
- Module VES: Fixed performance of BF_CUSTOM for basis functions with linear terms (e.g. having zero derivatives).
- Python wrappers:
  - Python module is now always named `plumed` irrespectively of program prefix and suffix. Notice that python module is installed inside the `lib/program_name` directory and thus it is not necessary to use `program_name` in order to install multiple modules side by side.
  - Python module can be compiled without compiling PLUMED first.
  - `Plumed` object can be explicitly finalized using `finalize()`. Can be used to make sure all files are closed, but it is not necessary if the `Plumed` object gets correctly collected by Python.
  - `Plumed` object can be used in context managers (e.g. `with plumed.Plumed() as p:`).
- Precompiled binaries are available on Anaconda cloud on the `conda-forge channel`.

2.6.0.4 Version 2.5.3 (Oct 11, 2019)

For users:

- Fixed a bug with CONVERT_TO_FES and periodic variables, see #441
- Fixed a bug with FOURIER_TRANSFORM
- Updated patch for GROMACS 2019.4
- Updated patch for GROMACS 2018.8
- Python module:
  - Fixed building with clang-8.
  - Set `language_level` for cython to the actually used language level.
  - Force using cython when compiling from source. Still using the pre-generated cpp file when installing from PyPI, to avoid cython dependency.
  - Using python 2 to create the cpp file uploaded on PyPI (this will change to python 3 in 2.6, see #502).
- Module VES: Fixed a bug in updating of bias potential in VES_LINEAR_EXPANSION that is present for certain integrators that call the calculation of the bias multiple times (see here) and replica exchange.

2.6.0.5 Version 2.5.4 (Jan 27, 2020)

For users:

- Includes all fixes up to 2.4.7

2.6.0.6 Version 2.5.5 (Jul 8, 2020)

For users:

- Includes all fixes up to 2.4.8

For developers:

- Small fix to avoid unique global symbols (see #549)

2.6.0.7 Version 2.5.6 (Oct 26, 2020)

For users:

- Report an error when using all weights set to zero in reference PDB files. Same as #247 fixed in 2.4, but now the check is done in more cases.
- Fixed overflow in reweighting factor when using non well-tempered metadynamics (thanks to Michele Invernizzi, see #599).
- Fixed READ with EVERY when reading a trajectory file (see #619).

For developers:

- Fixed a warning in `wrapper/Plumed.h` appearing with recent clang versions.

2.6.0.8 Version 2.5.7 (Apr 16, 2021)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

For users:

- Fixed handling of periodic variables in ves_md_linearexpansion (see #649).
- Small fix that might affect performance (backport of a fix needed for master branch, see #680).

2.7 Version 2.6

2.7.0.1 Version 2.6 (Jan 27, 2020)

Changes from version 2.5 which are relevant for users:

- Changes leading to incompatible behavior:
  - PLUMED input file parsing is now case insensitive that is that all directives can be written using uppercase characters (compatible with former versions) as well as lowercase characters (not compatible) internally PLUMED still uses uppercase definitions
  - `plumed partial_tempering` now uses `gawk` instead of `awk`. You might need to install `gawk` for it to work correctly.
- Other changes:
  - Asmjit is now embedded into PLUMED. In order to enable it, it is sufficient to configure with `--enable-asmjit`. See this page.
  - Fixed grids so as to decrease memory footprint of derivatives (see #465).
  - Added option `--idlp4` to driver to read DLPOLY4 HISTORY files (see #478, thanks to Alin Marin Elena).
  - Added atom selectors using mdtraj/MDAnalysis/VMD syntax, see MOLINFO and #448.
  - EEFSOLV is now faster in scalar and also mpi/openmp parallel
  - New shortcuts are available for selecting protein atoms: `@sidechain-#`, `@back-#`
  - VIM syntax highlight is now case insensitive. Notice that autocompletion still only works with upper case commands.
- New contributed modules:
  - A new Maze module by Jakub Rydzewski
    - MAZE_LOSS
    - MAZE_MEMETIC_SAMPLING
    - MAZE_RANDOM_ACCELERATION_MD
    - MAZE_RANDOM_WALK
    - MAZE_SIMULATED_ANNEALING
    - MAZE_STEERED_MD
    - MAZE_OPTIMIZER_BIAS
  - A new ANN module by Wei Chen and Andrew Ferguson
    - ANN
- New patches:
  - added support for AMBER PMEMD 18 (contributed by Viktor Drobot, see #486).
- Changes in the VES module
  - new VES_DELTA_F bias.
  - ves_md_linearexpansion now outputs one-dimensional free energy projections of the potential energy landscape.
- Changes in the DRR module
  - The MAXFACTOR option now is tunable for each CV in multidimensional cases.
  - Output .zcount file (the same as .czar.count) for compatibility with newer abf_integrate.
  - The citation of DRR module has been updated.

- Changes in the ISDB module
  - in METAINFERENCE we removed the MC_STRIDE keyword
  - in METAINFERENCE the bias value (metainference score) now includes the Jeffrey's prior (values are different, but forces are equal)
  - components were previously named using _ but now they abide to the standard is -
  - removed ADDEXP keywords for JCOUPLING NOE PRE RDC
  - METAINFERENCE performs more check on the input and restart files to ensure a consistent setup
  - SAXS is slightly faster and scales better, removed BESSEL options
- Python module:
  - Removed compatibility with Python 2.
  - Added capability to read and write pandas dataset from PLUMED files (see #496).

Changes from version 2.5 which are relevant for developers:

- Components documentation is now enforced
- `readdir_r` is deprecated and is thus not used by default (can be enabled with `./configure --enable-readdir-r`).

2.7.0.2 Version 2.6.1 (Jul 8, 2020)

For users:

- Includes all fixes up to 2.5.5
- New patches:
  - added gromacs 2019.6
  - added gromacs 2020.2 (experimental)
- Fixed handling of truncated octahedron box in Amber (see #584). Notice that the fix is for the PMEMD patch to be used with Amber 18. Amber 20 has been fixed upstream, both in PMEMD and Sander code.

For developers:

- Small fix to avoid unique global symbols (see #549)

2.7.0.3 Version 2.6.2 (Oct 26, 2020)

For users:

- Includes all fixes up to 2.5.6
- Updated patches:
  - added gromacs 2020.4 (experimental: it does not yet support modular simulator)

2.7.0.4 Version 2.6.3 (Apr 16, 2021)

For users:

- Includes all fixes up to 2.5.7

2.7.0.5 Version 2.6.4 (Jul 27, 2021)

For users:

- Fixed `plumed partial_tempering` so as to correctly process `[ pairs ]` sections. The incorrect script was leading to unscaled 14 interactions with Glycam force field. (reported by Isabell Grothaus).

For developers:

- Added integer macros `PLUMED_VERSION_MAJOR` `PLUMED_VERSION_MINOR` and `PLUMED_VERSION_PATCH` to `config/version.h`. Can be used to write LOAD -able source code portable across multiple versions.
- Fix for compilation with GCC 11 (reported by Axel Kohlmeyer, see #693).

2.7.0.6 Version 2.6.5 (Dec 1, 2021)

For users:

- Fixed configure problem on XL compiler (see #731).
- Fixed a bug in METAINFERENCE where the score was not properly updated upon multiple MC moves in the same MD step

For developers:

- Fixed several regtests decreasing their numeric precision.

2.7.0.7 Version 2.6.6 (Feb 22, 2022)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

For users:

- Fixed some incorrectly formatted output

For developers:

- Several fixes to improve portability on Debian and FreeBSD

2.8 Version 2.7

2.8.0.1 Version 2.7.0 (Dec 23, 2020)

Changes from version 2.6 which are relevant for users:

- Changes leading to differences with previous versions
  - The definition of the omega angle has been modified to adhere to the IUPAC standard (i.e. with the previous amino acid)
- New contributed modules:
  - A new Funnel module by Stefano Raniolo and Vittorio Limongelli
    - FUNNEL_PS
    - FUNNEL
  - A new Infinite Switch Simulated Tempering in Force module by Glen Hocky
    - FISST
  - A new OPES module by Michele Invernizzi
    - OPES_METAD
- New actions:
  - ENVIRONMENTSIMILARITY from Pablo Piaggi
  - PROJECTION_ON_AXIS from
  - FUNCPATHGENERAL from
- Other improvements:
  - MOLINFO action can now be used multiple times. Every action doing a search will use the latest appearance. See #134.
  - Neighbor lists are now OpenMP and MPI parallel so improving the scalability of all actions employing them
  - It is now possible to pass pdb files with all weights set to zero. Instead of reporting an error, PLUMED will now assume they are all equal to 1/n, where n is the number of atoms (see #608).
  - All the examples in the manual are now displayed with contextual help and regularly tested for correctness.
  - A tool to build PLUMED input directly within a python script has been added (see #611 and documentation for class `plumed.InputBuilder()`).
  - Python function `plumed.read_as_pandas()` now also accepts an argument `index_col`.
  - Lepton arithmetics can be used also when reading integers (e.g., `METAD PACE=2*5`, see #614).
- GROMACS:
  - When using `-hrex` flag, the neighbor lists are update automatically at every exchange step. This relaxes the requirements on the choice of `-replex` stride (see #579, thanks to Chang Junhan).
- Changes in the DRR module
  - Support multi-time stepping. Now the STRIDE keyword should work with DRR correctly.
  - Support reflecting boundary conditions, which should be a better solution to the boundary effect of eABF in non-periodic cases. You can use REFLECTINGWALL to enable it.
  - Stop the simulation when the temperature is not passed from the MD engine to PLUMED. In this case, users should set the temperature by the TEMP keyword.
- Changes in the ISDB module

  - There is a new option for OPTSIGMAMEAN, SEM_MAX that allows to automatically determine an optimal value for SIGMA_MAX

- Changes in the VES module

  - Small changes to TD_MULTICANONICAL and TD_MULTITHERMAL_MULTIBARIC. Bug fix concerning the calculation of the logarithm of the target distribution. Added the keyword EPSILON to avoid dealing with regions of zero target probability.

For developers:

- small fix in `Plumed.h` too avoid unique global symbols (see #549)
- Added `cmd("readInputLines")` to allow reading input from a buffer with comments and continuation lines (see #571).
- fixed error when the install prefix contained unicode characters

2.8.0.2 Version 2.7.1 (Apr 16, 2021)

- Includes all fixes up to 2.6.3
- In python interface, fixed usage of python arrays to allow compatibility with PyPy.
- New/updated patches:
  - updated patch for gromacs-2020.5
  - new patch for gromacs-2021
    - this should work with multiple-time stepping (plumed forces are integrated with the smallest time step, plumed can internally implement a multiple-time step if needed).
    - Modular simulator is still not supported
    - hrex, lambda cv and replica-exchange are not yet tested

2.8.0.3 Version 2.7.2 (Jul 27, 2021)

- Includes all fixes up to 2.6.4
- Fixed a bug in the `-hrex` implementation for GROMACS 2020 and 2021 (see #691, thanks to Chang Junhan).
- Changes in the OPES module
  - the CALC_WORK option now outputs the accumulated work, as in METAD, instead of the work done in the last bias update

2.8.0.4 Version 2.7.3 (Dec 1, 2021)

- Includes all fixes up to 2.6.5
- GROMACS patches now take a note of the used PLUMED version in the GROMACS log (see #737)
- GROMACS 2021 patch renamed to 2021.4 for consistency.

2.8.0.5 Version 2.7.4 (Feb 22, 2022)

- Includes all fixes up to 2.6.6

2.8.0.6 Version 2.7.5 (Oct 21, 2022)

- Minor fixes in error reporting.
- Fix in building python package with MacPorts and MacOS 11.
- Fixed overflows when using `plumed sum_hills --idw`, see #823.
- Renamed version file to `VERSION.txt` to avoid issues with some MacOS compilers.
- Fixed periodicity bug in psemd .
- Fixed an issue with timestep roundoff apparent in Windows build.
- Fixed an issue with METAINFERENCE noisetype OUTLIERS/MOUTLIERS when not using replicas, thanks @hmcezar #847
- Fixed #833.
- Fixed #841.
- Fixed an incorrect `const` conversion in `wrapper/Plumed.h`.

2.8.0.7 Version 2.7.6 (Mar 13, 2023)

Warning

This branch is not maintained. Users are invited to upgrade to a newer version

- Fixed a bug in PATHTOOLS where the distance was not squared and the suggested lambda was incorrect, thanks @rebelot #894
- Fixed a bug in SAXS cv using recent versions of arrayfire
- Fixed checks on the number of used CVs in TD_MULTICANONICAL and TD_MULTITHERMAL_MULTIBARIC
- Fixed PIV with `VOLUME`, #883.
- Fixed generation of documentation with MPI examples.
- plumed patch properly detects code patched with `--include` option (available as of plumed 2.9, might become the default)

2.9 Version 2.8

2.9.0.1 Version 2.8 (Feb 22, 2022)

Changes from version 2.7 which are relevant for users:

- Changes leading to differences with previous versions
  - in METAD and PBMETAD, Gaussians are now stretched rather than truncated, making the energy a continuous function of the collective variable. See #420.
  - sum_hills is now aware of stretched Gaussians. This change also fixes a minor bug in the set of grid points where Gaussians were different from zero that is still present up to version 2.7.
  - it is possible to restart from a HILLS file produced with PLUMED $<$ 2.8, but Gaussians will be reinterpreted as stretched and a warning will be written in the log file. This might lead to small numerical changes in bias potentials.
  - in METAD if possible the root walker in WALKERS_MPI will set the folder from which reading the GR↩ ID/HILLS file upon restart
  - in METAD work is not calculated by default anymore, if needed it can be obtained using CALC_WORK
  - in METAD an error will be thrown if, when restarting from FILE, the file is not found
  - the parser is more strict. Specifically, the explicitly crashes when a string cannot be parsed correctly. This was true only in a limited number of cases until v2.7 and might lead to errors when reading incorrectly formatted files. See #717.
- New actions:
  - GHBFIX to compute generalized hydrogen-bond fixes
- New contributed module:
  - A new SASA module by Andrea Arsiccio
    - SASA_HASEL
    - SASA_LCPO
  - A new S2 contact model module by Omar Valsson
    - S2CM
- Fixed patches:
  - A bug in using GROMACS with expanded ensemble in combination with PLUMED has been fixed (version 2020.6 and 2021.4, see #793). Notice that this fix requires PLUMED 2.8, so it won't be backward compatible.
- Other improvements
  - in METAD a new keyword NLIST has been added to use a neighbor list for bias evaluation, this should be faster than grids with many CVs
  - in METAD there are more checks that a restart of WALKERS_MPI is working consistently among walkers
  - in driver there is a flag `--restart` that can be used to enforce restart (similar to using RESTART in the PLUMED input file).
  - Added configure option `--enable-cxx`. Can be used to select C++14 with `--enable-cxx=14`. Required to compile against libraries whose header files need C++14.
- Changes in the OPES module
  - new action OPES_EXPANDED
  - various new actions of type EXPANSION_CV to be used with OPES_EXPANDED

  - new action OPES_METAD_EXPLORE
  - new option EXTRA_BIAS in OPES_METAD, to sample custom target distributions
  - new option EXCLUDED_REGION in OPES_METAD, to define a region where no kernels are deposited

- Changes in the VES module
  - New localized basis functions: Wavelets (BF_WAVELETS), Gaussians (BF_GAUSSIANS), and cubic splines (BF_CUBIC_B_SPLINES). In particular, symmetric wavelets (symlets) have shown the best performance and are recommended of the localized basis functions. Furthermore, symlets have been shown to perform better than delocalized Chebyshev and Legendre polynomials.
  - New optimizer based on Adam (OPT_ADAM). Still experimental, and restarting with it does not work yet.
  - New optimizer based on classical Robbins Monro stochastic gradient descent (OPT_ROBBINS_MONRO_SGD). Only included for reference and not recommended for usage in simulations.
  - Fixed a bug in VES_LINEAR_EXPANSION for multidimensional bias potential if one (or more) of the CVs is outside the range of the bias potential. Previously, there was a force acting on the CVs if this happened. Now, there is no biasing force acting on the CVs if one (or more) of the CVs is outside the bias potential range.

- Changes in the DRR module
  - Added a new option MERGEHISTORYFILES to output a single history file instead of many .drrstate files.

- For developers:
  - The C++ interface now performs type checking (see `https://github.com/plumed/plumed2/pull/653`). This should require no change for MD codes that were calling PLUMED with correct arguments. Checks could be disabled at runtime with `export PLUMED_TYPESAFE_IGNORE=yes`.
  - Two new Fortran modules have been added. One of them provides explicit interfaces for the already available wrappers. With no change in calling code, just by including this module, one could perform runtime type/shape checking. In addition, a novel object oriented Fortran interface has been designed which allow to better manipulate PLUMED instances from Fortran. Both interfaces were written with a significant help from Balint Aradi.
  - The C interface (`plumed_cmd`) also performs type checking and allows overload-like syntax to pass additional size and shape information. This is obtained redefining `plumed_cmd` to a macro that calls the C++ interface, when using a C++ compiler, or using C11 `_Generic`, if the C compiler supports it. This feature is not supported if used a pre-C11 C compiler (pre-C++11 C++ compilers are ok instead).
  - `xxd` replaced by a awk script. This removed the build dependence on vim.
  - Lepton has been updated with OpenMM 7.6.0
  - Asmjit is now enabled by default on supported architectures.
  - Xdrfile library is now embedded and always available.
  - `--enable-rpath` now also includes the path where `libplumedKernel.so` is installed (see `#767`).

2.9.0.2 Version 2.8.1 (Oct 21, 2022)

- Includes all fixes up to 2.7.5
- It is now possible to pass a mpi4py communicator from the Python interface. This is backported from master, see `#818` (thanks to Henrique Musseli Cezar).
- Fix in `--enable-rpath` (see `#807`).
- Updated gromacs patches
- Fixed gromacs patches (2020 and 2021) to solve `#829`.
- Fixed a few incorrect `const` conversions in `wrapper/Plumed.h`.

2.9.0.3 Version 2.8.2 (Mar 13, 2023)

- Includes all fixes up to 2.7.6
- Fixed a regression introduced in v2.8.0 which would make multi-thread simulations, with a separate Plumed object in each thread, crash randomly
- Fixed a bug in SAXS cv using recent versions of arrayfire
- Fixed a bug in the GROMACS 2022 patch when atoms reordering happens also without domain decompisition: needs patch to be reapplied
- Updated GROMACS patches to warn about the joint use of update gpu and plumed: needs patch to be reapplied
- GROMACS patches for v2021 and v2022 have been updated to use -rerun with -plumed again: needs patch to be reapplied
- Fixed a few cases where plumed was aborting rather than throwing an exception
- Fixed `wrapper/Plumed.h` so that more compilers are covered (see #897).

2.9.0.4 Version 2.8.3 (May 25, 2023)

- Fixed a numerical instability in OPES_EXPANDED that could cause `-inf` to appear in the DELTAFS file when biasing large systems
- Small fixes in the test suite to make sure `plumed` is always invoked with `--no-mpi` when testing for features. This avoids problems that were appearing when testing with some specific versions of OpenMPI.

2.9.0.5 Version 2.8.4

- Added the possibility to disable `RTLD_DEEPBIND` (see #952).
- Fixed a bug in `switchingfunction` mode `Q` thanks to @nm3787, (see #951)
- Improved error reporting in CUSTOM switching function: an error is thrown if one uses `x` and `x2` arguments simultaneously (reported by Olivier Languin-Cattoen).
- Fixed bug in diagonalization of fixed-sized matrices that could lead to segmentation faults in the following case: RMSD calculations with `TYPE=OPTIMAL`, same weights used for alignment and displacement, either the running frame or the reference frame is invariant for rotation (i.e., atoms are placed along a straight line). This was happening in `benchmark tests in version 2.10`, and is very unlikely (but possible) in real simulations. In addition, the diagMatSym function on fixed size tensors was incorrectly modifying its argument matrix. By inspection of the places this was used (RMSD and Gyration), this should have no effect since a temporary matrix is built, diagonalized, and discarded in all those cases, but it has been fixed nonetheless.
- Removed a number of incorrect dependencies between modules
- Addressed numerical instabilities in the calculation of the derivative of the SwitchingFunction RATIONAL without simplification (where `NN!=2*MM`) around `d=R_0`.
- Fixed an error in checking array shapes in the interface. Arrays with shape (N,4) should not be accepted for positions or forces.
- Small fix in Python, where we do not assume anymore that strings are null terminated.

2.10 Version 2.9

2.10.0.1 Version 2.9 (May 25, 2023)

Changes from version 2.8 which are relevant for users:

- Changes leading to differences with previous versions:
  - Number of bins in METAD, PBMETAD and sum_hills is now computed correctly when setting a spacing that divide exactly the range. See #868.
  - SAXS in the ISDB module does not have the SCALEINT keyword anymore and the default normalisation of the intensity is to set the intensity at q=0 to 1.
- New contributed modules:
  - A new pytorch module by Luigi Bonati
    - PYTORCH_MODEL
  - A new membranefusion model by Ary Lautaro Di Bartolo
    - MEMFUSIONP
    - FUSIONPORENUCLEATIONP
    - FUSIONPOREEXPANSIONP
- Other improvements:
  - PBMETAD now supports partitioned families bias thanks to @lemmoi @pfaendtner
  - Construction of atom lists have been optimized (see #811).
  - SimpleMD has been parallelized with OpenMP. Keyword maxneighbor has been removed (not needed anymore). In addition, SimpleMD now supports custom values for epsilon and sigma
  - CENTER and COM have new options SET_CHARGE and SET_MASS to assign them ad hoc charge and mass values
  - A tool to compute time-averaged bias potentials has been added in the python module (see help(plumed.hills_time_average)).
- New in LogMFD module:
  - TEMPPD keyword has been newly introduced, which allows for manually setting the temperature of the Boltzmann factor in the Jarzynski weight in LogPD.
  - The output format has been modified to present the CV data in a more consistent way.
  - The algorithm for evaluating the mean force at the initial MFD step has been modified to handle a continued MD trajectory that begins with non-zero timestep number.
- New in ISDB module:
  - the SAXS CV now includes a new very efficient and accurate hybrid SAXS (hySAXS) algorithm that can be activated using the keyword ONEBEAD.
  - a new SANS CV to calculate small-angles neutron scattering has been added, including both the ATOMISTIC and hySAXS ONEBEAD approaches.
- New in DRR module:
  - The module now writes the .zgrad file for inspecting and debugging the $\boldsymbol{\xi}$-averaged spring forces.
- New Patches:
  - Patch for GROMACS 2023 (preliminary, in particular for replica-exchange, expanded ensemble, hrex features).

  - Patch for QEspresso 7.0 has been added.
  - Patch for GROMACS 2019 has been removed.

- Changes relevant for developers:
  - Nested exception can be passed to calling codes using C/C++/Fortran/Python interfaces #879.
  - Lepton has been updated with OpenMM 7.7.0
  - All explicit destructors that could be removed were removed, including in contributed modules. This might guarantee basic exception safety. Notice that this is not enforced, so that new code might violate this.
  - Improvements towards thread-safety:
    * Added thread-safe reference counter to wrapper.
    * Added locks for thread-unsafe molfile plugins.
  - Plumed patch now accepts the `--include` option. Might become the default in a future version.
  - Python (cython) wrappers now only use plain C instead of C++. Plumed exceptions are mapped to python exceptions.
  - Step number is now stored as a `long long int`. Might facilitate Windows compatibility.

2.10.0.2 Version 2.9.1 (to be released)

- Includes all fixes up to 2.8.4
- Plumed now fetches the massses correctly from QEspresso 7.0
- Fixed a size check in python interface when passing native arrays.

Chapter 3

Installation

In this page you can learn how to configure, compile, and install PLUMED. For those of you who are impatient, the following might do the job:

```
> ./configure --prefix=/usr/local
> make -j 4
> make doc # this is optional and requires proper doxygen version installed
> make install
```

Notice that `make install` is not strictly necessary as plumed can be used from the compilation directory. This is very useful so as to quickly test the implementation of new features. However, we strongly recommend to perform a full install.

Once the above is completed the `plumed` executable should be in your execution path and you will be able to use PLUMED to analyze existing trajectories or play with the Lennard-Jones code that is included. However, because PLUMED is mostly used to bias on the fly simulations performed with serious molecular dynamics packages, you can find instructions about how to patch your favorite MD code so that it can be combined with PLUMED below. Again, if you are impatient, something like this will do the job:

```
> cd /md/root/dir
> plumed patch -p
```

Then compile your MD code. For some MD codes these instructions are insufficient. It is thus recommended that you read the instructions at the end of this page. Notice that MD codes could in principle be "PLUMED ready" in their official distribution. If your favorite MD code is available "PLUMED ready" you will have to compile PLUMED first, then (optionally) install it, then check the MD codes' manual to discover how to link it.

3.1 Supported compilers

As of PLUMED 2.4, we require a compiler that supports C++11. The following compilers (or later versions) should be sufficient:

- gcc 4.8.1
- clang 3.3
- intel 15

Notice that the `./configure` script verifies that your compiler supports C++11. Some compilers do not declare full support, but implement anyway a number of C++11 features sufficient to compile PLUMED (this is the case for instance of intel 15 compiler). In case you see a warning about C++11 support during `./configure` please make sure that PLUMED compiles correctly and, if possible, execute the regtests (using `make regtest`). Notice that we regularly test a number of compilers on travis-ci, and at least those compilers are guaranteed to be able to compile PLUMED correctly.

3.2 Configuring PLUMED

The `./configure` command just generates a Makefile.conf file and a sourceme.sh file. In PLUMED 2.0 these files were prepared and stored in the directory configurations/. The new ones generated by ./configure are similar to the old ones but are not completely compatible. In particular, some of the -D options have been changed in version 2.2, and several new variables so as to specify the installation directories have been added. For this reason, you now should run `./configure` again. Anyway, it should be easy to enforce a similar setup with autoconf by passing the proper arguments on the command line. If you have problems on your architecture, please report them to the mailing list.

Useful command line options for ./configure can be found by typing

```
> ./configure --help
```

PLUMED is made up of modules. Some of them are on by default, some others aren't. Since version 2.3, the activation of modules should be made during configuration using the `--enable-modules` option (see List of modules).

Notice that some of the methods within PLUMED depend on external libraries which are looked for by configure. You can typically avoid looking for a library using the "disable" syntax, e.g.

```
> ./configure --disable-mpi --disable-gsl
```

Notice that when MPI search is enabled (by default) compilers such as "mpic++" and "mpicxx" are searched for first. On the other hand, if MPI search is disabled ("./configure --disable-mpi") non-mpi compilers are searched for. Notice that only a few of the possible compiler name are searched. Thus, compilers such as "g++-mp-4.8" should be explicitly requested with the CXX option.

You can better control which compiler is used by setting the variables CXX and CC. E.g., to use Intel compilers use the following command:

```
> ./configure CXX=icpc CC=icc
```

Notice that we are using icpc in this example, which is not an MPI compiler as a result MPI will not be enabled. Also consider that this is different with respect to what some other configure script does in that variables such as MPICXX are completely ignored here. In case you work on a machine where CXX is set to a serial compiler and MPICXX to a MPI compiler, to compile with MPI you should use

```
> ./configure CXX="$MPICXX"
```

Warning

> This procedure could be somehow confusing since many other programs behave in a different way. The flag `--enable-mpi` is perfectly valid but is not needed here. Autoconf will check if a code containing MPI calls can be compiled, and if so it will enable it. `--disable-mpi` could be used if you are using a compiler that supports MPI but you don't want PLUMED to be compiled with MPI support. Thus the correct way to enable MPI is to pass to ./configure the name of a C++ compiler that implements MPI using the CXX option. In this way, MPI library is treated similarly to all the other libraries that PLUMED tries to link by default.

To tune the compilation options you can use the CXXFLAGS variable:

```
> ./configure CXXFLAGS=-O3
```

If you are implementing new functionality and want to build with debug flags in place so as to do some checking you can use

```
> ./configure --enable-debug
```

This will perform some extra check during execution (possibly slowing down PLUMED) and write full symbol tables in the executable (making the final executable much larger).

The main goal of the automatic configure is to find the libraries. When they are stored in unconventional places it is thus sensible to tell autoconf where to look! To do this there are some environment variable that can be used to instruct the linker which directories it should search for libraries inside. These variables are compiler dependent, but could have been set by the system administrator so that libraries are found without any extra flag. Our suggested procedure is to first try to configure without any additional flags and to then check the log so as to see whether or not the libraries were properly detected.

If a library is not found during configuration, you can try to use options to modify the search path. For example if your gsl libraries is in /opt/local (this is where MacPorts put it) and configure is not able to find it you can try

```
> ./configure LDFLAGS=-L/opt/local/lib CPPFLAGS=-I/opt/local/include
```

Notice that PLUMED will first try to link a routine from say gsl without any additional flag, and then in case of failure will retry adding "-lgsl" to the LIBS options. If also this does not work, the gsl library will be disabled and some features will not be available. This procedure allows you to use libraries with custom names. So, if your gsl library is called /opt/local/lib/libmygsl.so you can link it with

```
> ./configure LDFLAGS=-L/opt/local/lib CPPFLAGS=-I/opt/local/include LIBS=-lmygsl
```

In this example, the linker will directly try to link `/opt/local/lib/libmygsl.so`. This rule is true for all the libraries, so that you will always be able to link a specific version of a library by specifying it using the LIBS variable.

Since version 2.3.2, the search for the library functions passing to the linker a flag with the standard library name (in the gsl example, it would be `-lgsl`) can be skipped by using the option `--disable-libsearch`. Notice that in this manner only libraries that are explicitly passed using the `LIBS` option will be linked. For instance

```
> ./configure --disable-libsearch LIBS=-lgsl
```

will make sure that only gsl is linked and, for instance, BLAS and LAPACK libraries are not. This might be useful when installing PLUMED within package managers such as MacPorts to make sure that only desired libraries are linked and thus to avoid to introduce spurious dependencies. The only exception to this rule is `-ldl`, which is anyway a system library on Linux.

Warning

On OSX it is common practice to hard code the full path to libraries in the libraries themselves. This means that, after having linked a shared library, that specific shared library will be searched in the same place (we do the same for the `libplumed.dylib` library, which has an install name hard coded). On the other hand, on Linux it is common practice not to hard code the full path. This means that if you use the `LDFLAGS` option to specify the path to the libraries you want to link to PLUMED (e.g. `./configure LDFLAGS="-L/path"`) these libraries might not be found later. The visible symptom is that `src/lib/plumed-shared` will not be linked correctly. Although the file 'src/lib/plumed-shared' is not necessary, being able to produce it means that it will be possible to link PLUMED dynamically with MD codes later. The easiest solution is to hard code the library search path in this way:

```
> ./configure LDFLAGS="-L/path -Wl,-rpath,/path"
```

Notice that as of PLUMED v2.4 it is possible to use the configure option `--enable-rpath` to automatically hard code the path defined in `LIBRARY_PATH`:

```
> ./configure LIBRARY_PATH=/path --enable-rpath
```

In this way, the search path used at link time (`LIBRARY_PATH`) and the one saved in the `libplumed.so` library will be consistent by construction. In a typical environment configured using module framework (http://modules.sourceforge.net), `LIBRARY_PATH` will be a variable containing the path to all the modules loaded at compilation time.

PLUMED needs BLAS and LAPACK. These are treated slightly different from other libraries. The search is done in the usual way (i.e., first look for them without any link flag, then add "-lblas" and "-llapack", respectively). As such if you want to use a specific version of BLAS or LAPACK you can make them available to configure by using

```
> ./configure LDFLAGS=-L/path/to/blas/lib LIBS=-lnameoflib
```

If the functions of these libraries are not found, the compiler looks for a version with a final underscore added. Finally, since BLAS and LAPACK are compulsory in PLUMED, you can use a internal version of these libraries that comes as part of PLUMED. If all else fails the internal version of BLAS and LAPACK are the ones that will be used by PLUMED. If you wish to disable any search for external libraries (e.g. because the system libraries have problems) this can be done with

```
> ./configure --disable-external-blas
```

Notice that you can also disable external LAPACK only, that is use internal LAPACK with external BLAS using

```
> ./configure --disable-external-lapack
```

Since typically it is the BLAS library that can be heavily optimized, this configuration should not provide significant slowing down and could be used on systems where native LAPACK libraries have problems.

As a final resort, you can also edit the resulting Makefile.conf file. Notable variables in this file include:

- DYNAMIC_LIB : these are the libraries needed to compile the PLUMED library (e.g. -L/path/to/gsl -lgsl etc). Notice that for the PLUMED shared library to be compiled properly these should be dynamic libraries. Also notice that PLUMED preferentially requires BLAS and LAPACK library; see BLAS and LAPACK for further info. Notice that the variables that you supply with `configure LIBS=something` will end up in this variable. This is a bit misleading but is required to keep the configuration files compatible with PLUMED 2.0.
- LIBS : these are the libraries needed when patching an MD code; typically only "-ldl" (needed to have functions for dynamic loading).
- CPPFLAGS : add here definition needed to enable specific optional functions; e.g. use -D__PLUMED_HA↩ S_GSL to enable the gsl library
- SOEXT : this gives the extension for shared libraries in your system, typically "so" on UNIX, "dylib" on mac; If your system does not support dynamic libraries or, for some other reason, you would like a static executable you can just set this variable to a blank ("SOEXT=").

3.2.1 BLAS and LAPACK

We tried to keep PLUMED as independent as possible from external libraries and as such those features that require external libraries are optional. However, to have a properly working version of plumed PLUMED you need BLAS and LAPACK libraries. We would strongly recommend you download these libraries and install them separately so as to have the most efficient possible implementations of the functions contained within them. However, if you cannot install BLAS and LAPACK, you can use the internal ones. Since version 2.1, PLUMED uses a configure script to detect libraries. In case system LAPACK or BLAS are not found on your system, PLUMED will use the internal replacement.

We have had a number of emails (and have struggled ourselves) with ensuring that PLUMED can link BLAS and LAPACK. The following describes some of the pitfalls that you can fall into and a set of sensible steps by which you can check whether or not you have set up the configuration correctly.

Notice first of all that the DYNAMIC_LIB variable in the Makefile.conf should contain the flag necessary to load the BLAS and LAPACK libraries. Typically this will be -llapack -lblas, in some case followed by -lgfortran. Full path specification with -L may be necessary and on some machines the BLAS and LAPACK libraries may not be called -llapack and -lblas. Everything will depend on your system configuration.

Some simple to fix further problems include:

- If the linker complains and suggests recompiling LAPACK with -fPIC, it means that you have static LAPACK libraries. Either install dynamic LAPACK libraries or switch to static compilation of PLUMED by stopping to set the SOEXT variable in the configuration file.
- If the linker complains about other missing functions (typically starting with "for_" prefix) then you should also link some Fortran libraries. PLUMED is written in C++ and often C++ linkers do not include Fortran libraries by default. These libraries are required for LAPACK and BLAS to work. Please check the documentation of your compiler.
- If the linker complains that dsyevr_ cannot be found, try adding -DF77_NO_UNDERSCORE to CPPFLAGS Notice that "./configure" should automatically try this solution.

3.2.2 VMD trajectory plugins

PLUMED source code already includes a few selected VMD molfile plugins so as to read a small number of additional trajectory formats (e.g., dcd, gromacs files, pdb, and amber files). If you configure PLUMED with the full set of VMD plugins you will be able to read many more trajectory formats, basically all of those supported by VMD. To this aim, you need to download the SOURCE of VMD, which contains a plugins directory. Adapt build.sh and compile it. At the end, you should get the molfile plugins compiled as a static library `libmolfile_plugin.a`. Locate said file and `libmolfile_plugin.h`, they should be in a directory called `/pathtovmdplugins/ARCH/molfile` (e.g. `/pathtovmdplugins/MACOSXX86_64/molfile`). Also locate file `molfile_plugin.h`, which should be in `/pathtovmdplugins/include`. Then customize the configure command with something along the lines of:

```
> ./configure LDFLAGS="-L/pathtovmdplugins/ARCH/molfile" CPPFLAGS="-I/pathtovmdplugins/include -I/pathtovmdplu
```

Notice that it might be necessary to add to `LDFLAGS` the path to your TCL interpreter, e.g.

```
> ./configure LDFLAGS="-ltcl8.5 -L/mypathtotcl -L/pathtovmdplugins/ARCH/molfile" \
            CPPFLAGS="-I/pathtovmdplugins/include -I/pathtovmdplugins/ARCH/molfile"
```

Then, rebuild plumed.

3.2.3 Additional Modules

PLUMED includes some additional modules that by default are not compiled, but can be enabled during configuration. You can use the option `--enable-modules` to activate some of them, e.g.

```
> ./configure --enable-modules=module1name+module2name
```

For more information on modules see List of modules.

3.3 Compiling PLUMED

Once configured, PLUMED can be compiled using the following command:

```
> make -j 4
```

This will compile the entire code and produce a number of files in the 'src/lib' directory, including the executable 'src/lib/plumed'. When shared libraries are enabled, a shared libraries called 'src/lib/libKernel.so' should also be present. Notice that the extension could be '.dylib' on a Mac.

In case you want to run PLUMED *without installing it* (i.e. from the compilation directory), you can use the file 'sourceme.sh' that has been created by the configure script in the main PLUMED directory. This file can be "sourced" (presently only working for bash shell) if you want to use PLUMED *before installing it* (i.e. from the compilation directory). It is a good idea to source it now, so that you can play with the just compiled PLUMED:

```
> source sourceme.sh
```

Now a "plumed" executable should be in your path. Try to type

```
> plumed -h
```

Warning

> If you are cross compiling, the plumed executable will not work. As a consequence, you won't be able to run regtests or compile the manual. This is not a problem.

You can also check if PLUMED is correctly compiled by performing our regression tests. Be warned that some of them fail because of the different numerical accuracy on different machines. As of version 2.4, in order to test the `plumed` executable that you just compiled (prior to installing it) you can use the following command

```
> make check
```

On the other hand, in order to test the `plumed` executable that you just installed (see Installing PLUMED) you should type

```
> make installcheck
```

In addition, similarly to previous versions of PLUMED, you can test the `plumed` executable that is in your current path with

```
> cd regtest
> make
```

You can check the exact version they will use by using the command

```
> which plumed
```

Thus, you can easily run the test suite using a different version of PLUMED (maybe an earlier version that you already installed), just making sure that it can be found in the path. Clearly, if you test a given version of PLUMED with a test suite from a different version you can expect two possible kinds of innocuous errors:

- If `plumed` executable is older than the test suite, the tests might fail since they rely on some feature introduced in PLUMED in a newer version.
- If `plumed` executable is newer than the test suite, the tests might fail since some non-backward compatible change was made in PLUMED. We try to keep the number of non-backward compatible changes small, but as you can see in the Change Log there are typically a few of them at every new major release.

Attention

Even though we regularly perform tests on Travis-CI, it is possible that aggressive optimization or even architecture dependent features trigger bugs that did not show up on travis. So please always perform the regtests when you install PLUMED.

Notice that the compiled executable, which now sits in 'src/lib/plumed', relies on other resource files present in the compilation directory. This directory should thus stay in the correct place. One should thus not rename or delete it. In fact the path to the PLUMED root directory is hard coded in the plumed executable as can be verified using

```
> plumed info --root
```

In case you try to use the plumed executable without the compilation directory in place (e.g. you move away the src/lib/plumed static executable and delete or rename the compilation directory) PLUMED will not work correctly and will give you an error message

```
> plumed help
ERROR: I cannot find /xxx/yyy/patches directory
```

You can force plumed to run anyway by using the option –standalone-executable:

```
> plumed --standalone-executable help
```

Many features will not be available if you run in this way. However, this is currently the only way to use the PLUMED static executable on Windows.

3.4 Installing PLUMED

It is strongly suggested to install PLUMED in a predefined location. This is done using

```
> make install
```

This will allow you to remove the original compilation directory, or to recompile a different PLUMED version in the same place.

To install PLUMED one should first decide the location:

```
> ./configure --prefix=$HOME/opt
> make
> make install
```

As of PLUMED 2.5 you cannot anymore change the location during install. If you didn't specify the `--prefix` option during configure PLUMED will be installed in /usr/local. The install command should be executed with root permissions (e.g. "sudo make install") if you want to install PLUMED on a system directory.

Notice that upon installation PLUMED might need to relink a library. This was always true until version 2.1, but in version 2.2 libraries should only be relinked if one changes the install prefix during when typing `make install`. If root user does not have access to compilers, "sudo -E make install" might solve the issue.

Upon install, the executable is copied to $prefix/bin, libraries to $prefix/lib, include files to $prefix/include, and documentation to $prefix/shared/doc/plumed. Additionally, a directory $prefix/lib/plumed is created containing several other files, including patch files, object files (for static patches), etc. Notice also that these path can be further customized using standard autoconf directories (e.g. `./configure --bindir=/usr/bin64`).

One should then set the environment properly. We suggest to do it using the module framework (http://modules.sourceforge.net). An ad hoc generated module file for PLUMED can be found in $prefix/lib/plumed/src/lib/modulefile Just edit it as you wish and put it in your modulefile directory. This will also allow you to install multiple PLUMED versions on your machine and to switch among them. If you do not want to use modules, you can still have a look at the modulefile we did so as to know which environment variables should be set for PLUMED to work correctly.

If the environment is properly configured one should be able to do the following things:

- use the "plumed" executable from the command line. This is also possible before installing.
- link against the PLUMED library using the "-lplumed" flag for the linker. This allows one to use PLUMED library in general purpose programs
- use PLUMED internal functionality (C++ classes) including header files such as "#include <plumed/tools/Vector.h>". This is useful as it may be expedient to exploit the PLUMED library in general purpose programs

As a final note, if you want to install several PLUMED versions without using modules then you should provide a different suffix and/or prefix at configure time:

```
> ./configure prefix=$HOME/opt --program-suffix=_2.2 --program-prefix=mpi-
> make install
```

This will install a plumed executable named "mpi-plumed_2.2". All the other files will be renamed similarly, e.g. the PLUMED library will be loaded with "-lmpi-plumed_2.2" and the PLUMED header files will be included with "#include <mpi-plumed_2.2/tools/Vector.h>". Notice that you can also use arbitrary scripts to edit the name of the executable with the option –program-transform-name=PROGRAM (see autoconf documentation for more info). These options are useful if you do not want to set up modules, but we believe that using modules as described above is more flexible.

3.5 Patching your MD code

A growing number of MD codes can use PLUMED without any modification. If you are using one of these codes, refer to its manual to know how to activate PLUMED. In case your MD code is not supporting PLUMED already, you should modify it. We provide scripts to adjust some of the most popular MD codes so as to provide PLUMED support. At the present times we support patching the following list of codes:

- gromacs-2020-7
- gromacs-2021-7
- gromacs-2022-5
- gromacs-2023
- namd-2-12
- namd-2-13
- namd-2-14
- qespresso-5-0-2
- qespresso-6-2
- qespresso-7-0
- qespresso-7-2

In the section Code specific notes you can find information specific for each MD code.

To patch your MD code, you should have already installed PLUMED properly. This is necessary as you need to have the command "plumed" in your execution path. As described above this executable will be in your paths if plumed was installed or if you have run sourceme.sh

Once you have a compiled and working version of plumed, follow these steps to add it to an MD code

- Configure and compile your MD engine (look for the instructions in its documentation).
- Test if the MD code is working properly.
- Go to the root directory for the source code of the MD engine.
- Patch with PLUMED using:

  ```
  > plumed patch -p
  ```

  The script will interactively ask which MD engine you are patching.
- Once you have patched recompile the MD code (if dependencies are set up properly in the MD engine, only modified files will be recompiled)

There are different options available when patching. You can check all of them using

```
> plumed patch --help
```

Particularly interesting options include:

- –static just link PLUMED as a collection of object files. This is only suggested if for external reasons you absolutely need a static executable. Notice that with this setting it is often more complicated to configure properly the MD code, since all the libraries that PLUMED depends on should be properly specified. The `./configure` script does its best in this sense, but sometime it cannot solve the problem. Additionally, this patching mode has been reported not to work properly on OSX.
- –shared (default) allows you to link PLUMED as a shared library. As a result when PLUMED is updated, there will be no need to recompile the MD code. This is way better than –static since the libraries that PLUMED depends on should be automatically linked. Notice that if you later remove the directory where PLUMED is installed also the MD code will not run anymore.
- –runtime allows you to choose the location of the PLUMED library at runtime by setting the variable PLUM↩ ED_KERNEL. This is probably the most flexible option, and we encourage system administrators to use this option when installing PLUMED on shared facilities. Indeed, using this setting it will be possible to update separately the PLUMED library and the MD code, leaving to the user the possibility to combine different versions at will. We also recommend to use the provided modulefile (see above) to properly set the runtime environment.

Notice that with PLUMED version <2.5 there was no possibility to link PLUMED as a static library (something like 'libplumed.a'). However, starting with PLUMED 2.5, the `./configure` script will try to set up the system so that a `libplumed.a` file is produced. Patching an MD code with `--static` with try to link against this static library. Creation of the `libplumed.a` library can be avoided with `./configure --disable-static-archive`.

If your MD code is not supported, you may want to implement an interface for it. Refer to the developer manual .

3.6 Cross compiling

If you are compiling an executable from a different machine, then `plumed` executable will not be available in the compilation environment. This means that you won't be able to perform regtests on the machine nor to compile the manual. You can try to run the regtests on the computing nodes, but this might require some tweak since often machines where people do cross compiling have architectures with limited capabilities on the compute nodes. Also notice that many of the `plumed` options (e.g. patch) are implemented as shell scripts launched from within the `plumed` executable. If the compute nodes have some limitation (e.g. they do not allow to fork new processes) these options will not work. Anyway, the PLUMED library in combination with an MD software should work if both PLUMED and the MD software have been properly compiled.

Also notice that it will not be possible to use the command `plumed patch` on the machine where you are compiling. You should thus use `plumed-patch` instead of `plumed patch` (notice that it should be written as a single word).

Try e.g.:

```
> plumed-patch --help
```

This script provides a "shell only" implementation of `plumed patch` that will skip the launch of the `plumed` executable.

Notice that other command line tools will be available in the directory `prefix/lib/progname/`. If configuring with default values this would be `/usr/local/lib/plumed/plumed-*`. These files are not included in the execution path (prefix/bin) to avoid clashes, but can be executed also when plumed is cross compiled and the main plumed executable cannot be launched.

3.7 Installing PLUMED with MacPorts

If you are using a Mac, notice that you can take advantage of a MacPorts package. Installing a working plumed should be as easy as:

- Install MacPorts
- Type `sudo port install plumed`

Notice that plumed comes with many variants that can be inspected with the command

```
> sudo port info plumed
```

Plumed uses variants to support different compilers. For instance, you can install plumed with mpich using

```
> sudo port install plumed +mpich
```

Using more recent clang instead of native compilers is recommended so as to take advantage of openMP

```
> sudo port install plumed +mpich +clang50
```

Notice that support for c++11 with gcc compilers is someway problematic within MacPorts due to impossibility to use the system c++ library. For this reason, only clang compilers are supported (see also this discussion).

Variants can be also used to compile with debug flags (`+debug`), to pick a linear algebra library (e.g. `+openblas`) and to enable all optional modules (`+allmodules`). Notice that the default variant installed with `sudo port install plumed` is shipped as a compiled binary, which is significantly faster to install.

In addition, we provide a developer version (typically: a later version not yet considered as stable) under the subport `plumed-devel` that can be installed with

```
> sudo port install plumed-devel
```

`plumed-devel` also supports the same variants as `plumed` in order to customize the compilation. `plumed-devel` and `plumed` cannot be installed at the same time.

It is also possible to install a plumed-patched version of gromacs. For instance, you can use the following command to install gromacs patched with plumed with clang-5.0 compiler and mpich:

```
> sudo port install plumed +mpich +clang50
> sudo port install gromacs-plumed +mpich +clang50
```

In case you want to combine gromacs with the unstable version of plumed, use this instead:

```
> sudo port install plumed-devel +mpich +clang50
> sudo port install gromacs-plumed +mpich +clang50
```

Notice that gromacs should be compiled using the same compiler variant as plumed (in this example `+mpich +clang50`). In case this is not true, compilation will fail.

Also notice that gromacs is patched with plumed in runtime mode but that the path of libplumedKernel.dylib in the MacPorts tree is hard coded. As a consequence:

- If gromacs is run with `PLUMED_KERNEL` environment variable unset (or set to empty), then the MacPorts plumed is used.
- If gromacs is run with `PLUMED_KERNEL` environment variable pointing to another instance of the plumed library, the other instance is used.

This is especially useful if you are developing PLUMED since you will be able to install gromacs once for all and combine it with your working version of PLUMED.

3.8 Installing PLUMED with conda

If you use the conda package manager you can install a pre-compiled PLUMED binary using the following command:

```
> conda install -c conda-forge plumed
```

Similarly, the python wrappers can be installed with

```
> conda install -c conda-forge py-plumed
```

These packages are part of `conda-forge` and as such should be binary compatible with other codes from the same distribution. Notice that it should also be possible to combine the installed plumed kernel with an MD code compiled outside of conda (or within a different conda environment) if plumed is linked in runtime mode. The only variable that you need to set in order to access to the installed plumed kernel is `PLUMED_KERNEL` (e.g., `export PLUMED_KERNEL=/conda/prefix/lib/libplumedKernel.so`).

Notice that binaries are only available for Linux and MacOS and that they have a limited number of features. In particular, they do not support MPI and do not include optional modules. However, they can be used to quickly install a working PLUMED version without the need to have a compiler.

Notice that there are additional conda packages on the `plumed` channel. Those packages are for testing only.

3.9 Installing PLUMED on a cluster

If you are installing PLUMED on a cluster and you want several users to take advantage of it consider the following suggestions.

First of all, we highly recommend using the module file that PLUMED provides to set up the environment. Just edit it as necessary to make it suitable for your environment.

Notice that PLUMED can take advantage of many additional features if specific libraries are available upon compiling it.

Try to patch all MD codes with the `--runtime` option. This will allow independent update of PLUMED and MD codes. Users will be able to combine any of the installed gromacs/amber/etc versions with any of the installed PLUMED versions. Notice that it is sometime claimed that statically linked codes are faster. In our experience, this is not true. In case you absolutely need a static executable, be ready to face non trivial linking issues. PLUMED is written in C++, thus required the appropriate C++ library to be linked, and might require additional libraries (e.g. libgsl).

Sometime we make small fixes on the patches. For this reason, keep track of which version of PLUMED you used to patch each of the MD code. Perhaps you can call the MD code modules with names such as `gromacs/4.6.7p1`, `gromacs/4.6.7p2` and write somewhere in the module file which version of PLUMED you used. Alternatively, call them something like `gromacs/4.6.7p2.2.0`. In this way, when we report a bug on the mailing list, users will know if the version they are using is affected by it.

Usually it is not necessary to install both a MPI and a non-MPI PLUMED version. PLUMED library only calls MPI functions when the MD code is compiled with MPI. PLUMED executable calls MPI functions only when it is invoked without `--no-mpi`. In many machines it is thus sufficient to run the plumed executable on the login node as

```
> plumed --no-mpi
```

even though PLUMED was compiled with MPI and the login node does not support MPI. The only case where you might need two different PLUMED installation for compute and login node is when you are cross compiling.

PLUMED needs to be well optimized to run efficiently. If you need a single PLUMED binary to run efficiency on machines with different levels of hardware (e.g.: some of your workstations support AVX and some do not), with intel compiler you can use something like

```
> ./configure CXX=mpicxx CXXFLAGS="-O3 -axSSE2,AVX"
```

It will take more time to compile but it will allow you to use a single module. Otherwise, you should install two PLUMED version with different optimization levels.

Using modules, it is not necessary to make the PLUMED module explicitly dependent on the used library. Imagine a scenario where you first installed a module `libgsl`, then load it while you compile PLUMED. If you provide the following option to configure `--enable-rpath`, the PLUMED executable and library will remember where libgsl is, without the need to load libgsl module at runtime. Notice that this trick often does not work for fundamental libraries such as C++ and MPI library. As a consequence, usually the PLUMED module should load the compiler and MPI modules.

Attention

In case you found out how to compile PLUMED on some fancy architecture please share your tricks! You can either post it in your blog, send it to the mailing list, or ask as to update this paragraph in the manual, we will be happy to do so.

3.10 Installing Python wrappers

As of PLUMED 2.5 it is possible to use the PLUMED library through Python wrappers. Notice that this is not something for end users but rather for developers. The interface is very similar to the one used in MD codes linked with PLUMED.

There are two ways to install Python wrappers.

3.10.1 Installing Python wrappers within PLUMED

If `./configure` finds a `python` executable that also has the `cython` module available, Python wrappers will be installed within `/prefix/lib/plumed/python`. In order to access them, you should add this directory to the environment variable `PYTHONPATH`. Notice that if your python interpreter has a different name you might have to pass it to `./configure` with `PYTHON_BIN=python3.6`. The whole thing would then be:

```
./configure PYTHON_BIN=python3.6 --prefix=$HOME/opt
make && make install
export PYTHONPATH="$HOME/opt/lib/plumed/python:$PYTHONPATH"
python3.6
>> import plumed
```

Notice that in this manner you will have to commit to a specific python version **before** installing PLUMED.

3.10.2 Installing Python wrappers outside PLUMED

If you use multiple python versions, you might find it easier to install the Python wrappers separately from PLUMED. The simplest way is to do it with `pip`:

```
pip3.6 install --user plumed
```

Here the `--user` flag allows you to install the packages on your home. Notice that you don't even need to download PLUMED in order to install the wrappers, but you will need PLUMED in order to use them. You can tell the wrappers where PLUMED is by setting the `PLUMED_KERNEL` environment variable:

```
export PLUMED_KERNEL=$HOME/opt/lib/libplumedKernel.so
python3.6
>> import plumed
```

Notice that by installing the wrappers in this manner you will download those that are packaged on `Pypi`. If you want to install using pip the development version of the wrappers you should download the PLUMED repository and use the following commands:

```
pip3.6 install --user cython # cython is required in this case
cd plumed2/python
make pip
pip3.6 install --user .
```

If you want to install the development version it is recommended to use a virtualenv so that it will not interfere with the released packages.

3.11 Other hints

We here collect a list of suggestions that might be useful on particular machines.

- On Blue Gene Q (likely on AIX) the prelinking made with `ld -r` is not working properly. There is no easy way to detect this at configure time. If during `make` you receive an error in the form

  ```
  ld: TOC section size exceeds 64k
  ```

  please configure plumed again with the following flag

```
> ./configure --disable-ld-r
```

- On Cray machines, you might have to set the following environment variable before configuring and building both PLUMED and the MD code that you want to patch with PLUMED (kindly reported by Marco De La Pierre):

```
> export CRAYPE_LINK_TYPE=dynamic
```

- Intel MPI seems to require the flags `-lmpi_mt -mt_mpi` for compiling and linking and the flag `-DMPICH_IGNORE_CXX_SEEK` for compiling (kindly reported by Abhishek Acharya). You might want to try to configure using

```
> ./configure LDFLAGS=-lmpi_mt CXXFLAGS="-DMPICH_IGNORE_CXX_SEEK -mt_mpi" STATIC_LIBS=-mt_mpi
```

  Adding libraries to `STATIC_LIBS` uses them for all the linking steps, whereas those in `LIBS` are only used when linking the PLUMED kernel library. See more at this thread.

3.12 Code specific notes

Here you can find instructions that are specific for patching each of the supported MD codes. Notice that MD codes with native PLUMED support are not listed here.

- gromacs-2020.7
- gromacs-2021.7
- gromacs-2022.5
- gromacs-2023
- namd-2.12
- namd-2.13
- namd-2.14
- qespresso-5.0.2
- qespresso-6.2
- qespresso-7.0
- qespresso-7.2

3.12.1 gromacs-2020.7

PLUMED can be incorporated into gromacs using the standard patching procedure. Patching must be done in the gromacs root directory *before* the cmake command is invoked.

On clusters you may want to patch gromacs using the static version of plumed, in this case building gromacs can result in multiple errors. One possible solution is to configure gromacs with these additional options:

cmake -DBUILD_SHARED_LIBS=OFF -DGMX_PREFER_STATIC_LIBS=ON

To enable PLUMED in a gromacs simulation one should use mdrun with an extra -plumed flag. The flag can be used to specify the name of the PLUMED input file, e.g.:

gmx mdrun -plumed plumed.dat

For more information on gromacs you should visit http://www.gromacs.org

3.12.2 gromacs-2021.7

PLUMED can be incorporated into gromacs using the standard patching procedure. Patching must be done in the gromacs root directory *before* the cmake command is invoked.

On clusters you may want to patch gromacs using the static version of plumed, in this case building gromacs can result in multiple errors. One possible solution is to configure gromacs with these additional options:

cmake -DBUILD_SHARED_LIBS=OFF -DGMX_PREFER_STATIC_LIBS=ON

To enable PLUMED in a gromacs simulation one should use mdrun with an extra -plumed flag. The flag can be used to specify the name of the PLUMED input file, e.g.:

gmx mdrun -plumed plumed.dat

For more information on gromacs you should visit http://www.gromacs.org

3.12.3 gromacs-2022.5

PLUMED can be incorporated into gromacs using the standard patching procedure. Patching must be done in the gromacs root directory *before* the cmake command is invoked.

On clusters you may want to patch gromacs using the static version of plumed, in this case building gromacs can result in multiple errors. One possible solution is to configure gromacs with these additional options:

cmake -DBUILD_SHARED_LIBS=OFF -DGMX_PREFER_STATIC_LIBS=ON

To enable PLUMED in a gromacs simulation one should use mdrun with an extra -plumed flag. The flag can be used to specify the name of the PLUMED input file, e.g.:

gmx mdrun -plumed plumed.dat

For more information on gromacs you should visit http://www.gromacs.org

3.12.4 gromacs-2023

PLUMED can be incorporated into gromacs using the standard patching procedure. Patching must be done in the gromacs root directory *before* the cmake command is invoked.

On clusters you may want to patch gromacs using the static version of plumed, in this case building gromacs can result in multiple errors. One possible solution is to configure gromacs with these additional options:

cmake -DBUILD_SHARED_LIBS=OFF -DGMX_PREFER_STATIC_LIBS=ON

To enable PLUMED in a gromacs simulation one should use mdrun with an extra -plumed flag. The flag can be used to specify the name of the PLUMED input file, e.g.:

gmx mdrun -plumed plumed.dat

For more information on gromacs you should visit http://www.gromacs.org

3.12.5 namd-2.12

To enable PLUMED in a NAMD simulation one should add the following lines in the NAMD configuration file (often named as xxx.namd or xxx.conf) and run the plumed-patched version of NAMD with it:

```
plumed on
plumedfile plumed.dat
```

The first line tells NAMD to run with PLUMED and the second line specifies the PLUMED input file.

Bug NAMD does not currently take into account virial contributions from PLUMED. Please use constant volume simulations only.

For more information on NAMD you should visit http://www.ks.uiuc.edu/Research/namd/

3.12.6 namd-2.13

To enable PLUMED in a NAMD simulation one should add the following lines in the NAMD configuration file (often named as xxx.namd or xxx.conf) and run the plumed-patched version of NAMD with it:

```
plumed on
plumedfile plumed.dat
```

The first line tells NAMD to run with PLUMED and the second line specifies the PLUMED input file.

Bug NAMD does not currently take into account virial contributions from PLUMED. Please use constant volume simulations only.

For more information on NAMD you should visit http://www.ks.uiuc.edu/Research/namd/

3.12.7 namd-2.14

To enable PLUMED in a NAMD simulation one should add the following lines in the NAMD configuration file (often named as xxx.namd or xxx.conf) and run the plumed-patched version of NAMD with it:

```
plumed on
plumedfile plumed.dat
```

The first line tells NAMD to run with PLUMED and the second line specifies the PLUMED input file.

Bug NAMD does not currently take into account virial contributions from PLUMED. Please use constant volume simulations only.

For more information on NAMD you should visit http://www.ks.uiuc.edu/Research/namd/

3.12.8 qespresso-5.0.2

For more information on Quantum Espresso you should visit http://www.quantum-espresso.org

3.12.9 qespresso-6.2

For more information on Quantum Espresso you should visit http://www.quantum-espresso.org This patch was kindly provided by Ralf Meyer, email: meyer.ralf(at)yahoo.com

3.12.10 qespresso-7.0

For more information on Quantum Espresso you should visit http://www.quantum-espresso.org To apply this patch configure Quantum Espresso by running ./configure first. The newer CMake installation workflow is not supported yet. To enable PLUMED on md runs use pw.x -plumed < md.in > md.out. A fixed PLUMED input file name 'plumed.dat' is used. This patch was kindly provided by Ralf Meyer, email: meyer.ralf(at)yahoo.com

3.12.11 qespresso-7.2

For more information on Quantum Espresso you should visit http://www.quantum-espresso.org To apply this patch configure Quantum Espresso by running ./configure first. The newer CMake installation workflow is not supported yet. To enable PLUMED on md runs use pw.x -plumed < md.in > md.out. A fixed PLUMED input file name 'plumed.dat' is used. This patch was kindly provided by Ralf Meyer, email: meyer.ralf(at)yahoo.com

Chapter 4

Getting Started

To run PLUMED you need to provide one input file.

You can follow the PLUMED Masterclass 21.1: PLUMED syntax and analysis to learn more about it.

In this input file you specify what it is that PLUMED should do during the course of the run. Typically this will involve calculating one or more collective variables, perhaps calculating a function of these CVs and then doing some analysis of values of your collective variables/functions or running some free energy method. A very brief introduction to the syntax used in the PLUMED input file is provided in this `10-minute video` .

Within this input file every line is an instruction for PLUMED to perform some particular action. This could be the calculation of a colvar, an occasional analysis of the trajectory or a biasing of the dynamics. The first word in these lines specify what particular action is to be performed. This is then followed by a number of keywords which provide PLUMED with more details as to how the action is to be performed. These keywords are either single words (in which they tell PLUMED to do the calculation in a particular way - for example NOPBC tells PLUMED to not use the periodic boundary conditions when calculating a particular colvar) or they can be words followed by an equals sign and a comma separated list *with no spaces* of numbers or characters (so for example ATOMS=1,2,3,4 tells PLUMED to use atom numbers 1,2,3 and 4 in the calculation of a particular colvar). The reason why spaces are not admitted is that PLUMED should be able to understand when the list of atoms ended and a new keyword should be expected. Space separated lists can be used instead of comma separated list if the entire list is enclosed in curly braces (e.g. ATOMS={1 2 3 4}). Please note that you can split commands over multiple lines by using Continuation lines.

The most important of these keywords is the label keyword as it is only by using these labels that we can pass data from one action to another. As an example if you do:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GettingStartedPP.md
DISTANCE ATOMS=1,2
```

Then PLUMED will do nothing other than read in your input file. In contrast if you do:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GettingStartedPP.md
DISTANCE ATOMS=1,2 LABEL=d1
PRINT ARG=d1 FILE=colvar STRIDE=10
```

then PLUMED will print out the value of the distance between atoms 1 and 2 every 10 steps to the file colvar as you have told PLUMED to take the value calculated by the action d1 and to print it. You can use any character string to label your actions as long as it does not begin with the symbol @. Strings beginning with @ are used by within PLUMED to reference special, code-generated groups of atoms and to give labels to any Actions for which the user does not provide a label in the input.

Notice that if a word followed by a column is added at the beginning of the line (e.g. pippo:), PLUMED automatically removes it and adds an equivalent label (LABEL=pippo). Thus, a completely equivalent result can be obtained with the following shortcut:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GettingStartedPP.md
d1: DISTANCE ATOMS=1,2
PRINT ARG=d1 FILE=colvar STRIDE=10
```

Also notice that all the actions can be labeled, and that many actions besides normal collective variables can define one or more value, which can be then referred using the corresponding label.

Actions can be referred also with POSIX regular expressions (see Regular Expressions) if regex library is available on your system and detected at configure time. You can also add Comments to the input or set up your input over multiple files and then create a composite input by Including other files.

More information on the input syntax as well as details on the the various trajectory analysis tools that come with PLUMED are given in:

- Collective Variables tells you about the ways that you can calculate functions of the positions of the atoms.
- Analysis tells you about the various forms of analysis you can run on trajectories using PLUMED.
- Bias tells you about the methods that you can use to bias molecular dynamics simulations with PLUMED.

4.1 Plumed units

By default the PLUMED inputs and outputs quantities in the following units:

- Energy - kJ/mol
- Length - nanometers
- Time - picoseconds

Unlike PLUMED 1 the units used are independent of the MD engine you are using. If you want to change these units you can do this using the UNITS keyword.

4.2 UNITS

	This is part of the setup module

This command sets the internal units for the code.

A new unit can be set by either specifying a conversion factor from the plumed default unit or by using a string corresponding to one of the defined units given below. This directive MUST appear at the BEGINNING of the plumed.dat file. The same units must be used throughout the plumed.dat file.

Notice that all input/output will then be made using the specified units. That is: all the input parameters, all the output files, etc. The only exceptions are file formats for which there is a specific convention concerning the units. For example, trajectories written in .gro format (with DUMPATOMS) are going to be always in nm.

The following strings can be used to specify units. Note that the strings are case sensitive.

- LENGTH: nm (default), A (for Angstrom), um (for micrometer), Bohr (0.052917721067 nm)

- ENERGY: kj/mol (default), j/mol, kcal/mol (4.184 kj/mol), eV (96.48530749925792 kj/mol), Ha (for Hartree, 2625.499638 kj/mol)
- TIME: ps (default), fs, ns, atomic (2.418884326509e-5 ps)
- MASS: amu (default)
- CHARGE: e (default)

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/UNITS.tmp
# this is using Angstrom - kj/mol - fs
UNITS LENGTH=A TIME=fs

# compute distance between atoms 1 and 4
d: DISTANCE ATOMS=1,4

# print time and distance on a COLVAR file
PRINT ARG=d FILE=COLVAR

# dump atoms 1 to 100 on a 'out.gro' file
DUMPATOMS FILE=out.gro STRIDE=10 ATOMS=1-100

# dump atoms 1 to 100 on a 'out.xyz' file
DUMPATOMS FILE=out.xyz STRIDE=10 ATOMS=1-100
```

In the `COLVAR` file, time and distance will appear in fs and A respectively, *irrespective* of which units you are using in the host MD code. The coordinates in the `out.gro` file will be expressed in nm, since `gro` files are by convention written in nm. The coordinates in the `out.xyz` file will be written in Angstrom *since we used the UNITS command setting Angstrom units*. Indeed, within PLUMED xyz files are using internal PLUMED units and not necessarily Angstrom!

If a number, x, is found instead of a string, the new unit is equal to x times the default units. Using the following command as first line of the previous example would have lead to an identical result:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/UNITS.tmp
UNITS LENGTH=0.1 TIME=0.001
```

Glossary of keywords and components

Options

NATURAL	(default=off) use natural units
LENGTH	the units of lengths. Either specify a conversion factor from the default, nm, or use one of the defined units, A (for angstroms), um (for micrometer), and Bohr.
ENERGY	the units of energy. Either specify a conversion factor from the default, kj/mol, or use one of the defined units, j/mol, kcal/mol and Ha (for Hartree)
TIME	the units of time. Either specify a conversion factor from the default, ps, or use one of the defined units, ns, fs, and atomic
MASS	the units of masses. Specify a conversion factor from the default, amu
CHARGE	the units of charges. Specify a conversion factor from the default, e

Chapter 5

Collective Variables

Chemical systems contain an enormous number atoms, which, in most cases makes it simply impossible for us to understand anything by monitoring the atom positions directly. Consequently, we introduce Collective variables (CVs) that describe the chemical processes we are interested in and monitor these simpler quantities instead. These CVs are used in many of the methods implemented in PLUMED - there values can be monitored using PRINT, Functions of them can be calculated or they can be analyzed or biased using the Analysis and Biasing methods implemented in PLUMED. Before doing any of these things however we first have to tell PLUMED how to calculate them.

The simplest collective variables that are implemented in PLUMED take in a set of atomic positions and output one or multiple scalar CV values. Information on these variables is given on the page entitled CV Documentation while information as to how sets of atoms can be selected can be found in the pages on Groups and Virtual Atoms. Please be aware that PLUMED contains implementations of many other collective variables but that the input for these variables may be less transparent when it is first encountered. In particular, the page on Distances from reference configurations describes the various ways that you can calculate the distance from a particular reference configuration. So you will find instructions on how to calculate the RMSD distance from the folded state of a protein here. Meanwhile, the page on Functions describes the various functions of collective variables that can be used in the code. This is a very powerful feature of PLUMED as you can use the Functions commands to calculate any function or combination of the simple collective variables listed on the page CV Documentation. Lastly the page on MultiColvar describes MultiColvars.
MultiColvars allow you to use many different colvars and allow us to implement all these collective variables without a large amount of code. For some things (e.g. DISTANCES GROUPA=1 GROUPB=2-100 LESS_THAN={RATIONAL R_0=3}) there are more computationally efficient options available in plumed (e.g. COORDINATION). However, MultiColvars are worth investigating as they provide a flexible syntax for many quite-complex CVs.

- Groups and Virtual Atoms
- CV Documentation
- Distances from reference configurations
- Functions
- MultiColvar
- Exploiting contact matrices

5.1 Groups and Virtual Atoms

5.1.1 Specifying Atoms

The vast majority of the CVs implemented in PLUMED are calculated from a list of atom positions. Within PLUMED atoms are specified using their numerical indices in the molecular dynamics input file.

In PLUMED lists of atoms can be either provided directly inside the definition of each collective variable, or predefined as a GROUP that can be reused multiple times. Lists of atoms can be written as:

- comma separated lists of numbers (`GROUP ATOMS=10,11,15,20 LABEL=g1`)
- numerical ranges. So `GROUP ATOMS=10-20 LABEL=g2` is equivalent to `GROUP ATOMS=10,11,12,13,14,15,16, LABEL=g2`
- numerical ranges with a stride. So `GROUP ATOMS=10-100:10 LABEL=g3 is equivalent to`GROUP ATOMS=10,20,30,40,50,60,70,80,90,100 LABEL=g3
- `atoms ranges with a negative stride. So`GROUP ATOMS=100-10:-10 LABEL=g4 is equivalent to `GROUP ATOMS=100,90,80,70,60,50,40,30,20,10 LABEL=g4`

In addition, there are a few shortcuts that can be used:

- `@mdatoms` indicate all the physical atoms present in the MD engine (e.g. `DUMPATOMS ATOMS=@mdatoms`).
- `@allatoms` indicates all atoms, including those defined only in PLUMED (e.g. `DUMPATOMS ATOMS=@allatoms`).

The list of the virtual atoms defined in PLUMED can be obtained by using the command `GROUP ATOMS=@allatoms REMOVE=@mdatoms`.

Other shortcuts are available if you loaded the structure of the molecule using the MOLINFO command.

All the above methods can be combined just putting one name after the other separated by a comma:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GroupPP.md
DUMPATOMS ATOMS=1,2,10-20,40-60:5,100-70:-2 LABEL=g5 FILE=test.xyz
```

Some collective variable must accept a fixed number of atoms, for example a DISTANCE is calculated using two atoms only, an ANGLE is calculated using either 3 or 4 atoms and TORSION is calculated using 4 atoms.

Additional material and examples can be also found in the tutorial Belfast tutorial: Analyzing CVs.

5.1.1.1 Molecules

In addition, for certain colvars, pdb files can be read in using the following keywords and used to select ATOMS:

MOLINFO	This command is used to provide information on the molecules that are present in your system.

5.1.1.2 Broken Molecules and PBC

PLUMED is designed so that for the majority of the CVs implemented the periodic boundary conditions are treated in the same manner as they would be treated in the host code. In some codes this can be problematic when the colvars you are using involve some property of a molecule. These codes allow the atoms in the molecules to become separated by periodic boundaries, a fact which PLUMED could only deal with were the topology passed from the MD code to PLUMED. Making this work would involve a lot laborious programming and goes against our original aim of having a general patch that can be implemented in a wide variety of MD codes. Consequentially, we have implemented a more pragmatic solution to this problem - the user specifies in input any molecules (or parts of molecules) that must be kept in tact throughout the simulation run. In PLUMED 1 this was done using the ALIGN_ATOMS keyword. In PLUMED 2 the same effect can be achieved using the WHOLEMOLECULES command.

The following input computes the end-to-end distance for a polymer of 100 atoms and keeps it at a value around 5.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GroupPP.md
WHOLEMOLECULES ENTITY0=1-100
e2e: DISTANCE ATOMS=1,100 NOPBC
RESTRAINT ARG=e2e KAPPA=1 AT=5
```

Notice that NOPBC is used to be sure in DISTANCE that if the end-to-end distance is larger than half the simulation box the distance is compute properly. Also notice that, since many MD codes break molecules across cell boundary, it might be necessary to use the WHOLEMOLECULES keyword (also notice that it should be before distance).

Notice that most expressions are invariant with respect to a change in the order of the atoms, but some of them depend on that order. E.g., with WHOLEMOLECULES it could be useful to specify atom lists in a reversed order.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GroupPP.md
# to see the effect, one could dump the atoms as they were before molecule reconstruction:
# DUMPATOMS FILE=dump-broken.xyz ATOMS=1-20
WHOLEMOLECULES STRIDE=1 ENTITY0=1-20
DUMPATOMS FILE=dump.xyz ATOMS=1-20
```

Notice that there are other ways to manipulate the coordinates stored within PLUMED:

- Using the FIT_TO_TEMPLATE they can be aligned to a template structure.
- Using WRAPAROUND you can bring a set of atom as close as possible to another set of atoms.
- Using RESET_CELL you can rotate the periodic cell.

5.1.2 Virtual Atoms

Sometimes, when calculating a colvar, you may not want to use the positions of a number of atoms directly. Instead you may wish to use the position of a virtual atom whose position is generated based on the positions of a collection of other atoms. For example you might want to use the center of mass of a group of atoms. Plumed has a number of routines for calculating the positions of these virtual atoms from lists of atoms:

CENTER	Calculate the center for a group of atoms, with arbitrary weights.
CENTER_OF_MULTICOLVAR	Calculate a a weighted average position based on the value of some multi-colvar.
COM	Calculate the center of mass for a group of atoms.
FIXEDATOM	Add a virtual atom in a fixed position.
GHOST	Calculate the absolute position of a ghost atom with fixed coordinates in the local reference frame formed by three atoms.

To specify to a colvar that you want to use the position of a virtual atom to calculate a colvar rather than one of the atoms in your system you simply use the label for your virtual atom in place of the usual numerical index. Virtual atoms and normal atoms can be mixed together in the input to colvars as shown below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GroupPP.md
COM ATOMS=1,10 LABEL=com1
DISTANCE ATOMS=11,com1
```

If you don't want to calculate CVs from the virtual atom. That is to say you just want to monitor the position of a virtual atom (or any set of atoms) over the course of your trajectory you can do this using DUMPATOMS.

5.1.3 GROUP

	This is part of the generic module

Define a group of atoms so that a particular list of atoms can be referenced with a single label in definitions of CVs or virtual atoms.

Atoms can be listed as comma separated numbers (i.e. `1,2,3,10,45,7,9`) , simple positive ranges (i.e. `20-40`), ranges with a stride either positive or negative (i.e. `20-40:2` or `80-50:-2`) or as comma separated combinations of all the former methods (`1,2,4,5,10-20,21-40:2,80-50:-2`).

Moreover, lists can be imported from ndx files (GROMACS format). Use `NDX_FILE` to set the name of the index file and `NDX_GROUP` to set the name of the group to be imported (default is first one).

It is also possible to remove atoms from a list and or sort them using keywords `REMOVE`, `SORT`, and `UNIQUE`. The flow is the following:

- If `ATOMS` is present, then take the ordered list of atoms from the `ATOMS` keyword as a starting list.
- Alternatively, if `NDX_FILE` is present, use the list obtained from the gromacs group.
- If `REMOVE` is present, then remove the first occurrence of each of these atoms from the list. If one tries to remove an atom that was not listed plumed adds a notice in the output. An atom that is present twice in the original list might be removed twice.
- If `SORT` is present, then the resulting list is sorted by increasing serial number.
- If `UNIQUE` is present, then the resulting list is sorted by increasing serial number *and* duplicate elements are removed.

Notice that this command just creates a shortcut, and does not imply any real calculation. So, having a huge group defined does not slow down your calculation in any way. It is just convenient to better organize input files. Might be used in combination with the INCLUDE command so as to store long group definitions in a separate file.

Examples

This command create a group of atoms containing atoms 1, 4, 7, 11 and 14 (labeled 'o'), and another containing atoms 2, 3, 5, 6, 8, 9, 12, and 13 (labeled 'h'):

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GROUP.tmp
o: GROUP ATOMS=1,4,7,11,14
h: GROUP ATOMS=2,3,5,6,8,9,12,13
# compute the coordination among the two groups
c: COORDINATION GROUPA=o GROUPB=h R_0=0.3
# same could have been obtained without GROUP, just writing:
# c: COORDINATION GROUPA=1,4,7,11,14 GROUPB=2,3,5,6,8,9,12,13

# print the coordination on file 'colvar'
PRINT ARG=c FILE=colvar
```

Groups can be conveniently stored in a separate file. E.g. one could create a file named `groups.dat` which reads

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GROUP.tmp
#SETTINGS FILENAME=groups.dat
# this is groups.dat
o: GROUP ATOMS=1,4,7,11,14
h: GROUP ATOMS=2,3,5,6,8,9,12,13
```

and then include it in the main 'plumed.dat' file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GROUP.tmp
INCLUDE FILE=groups.dat
# compute the coordination among the two groups
c: COORDINATION GROUPA=o GROUPB=h R_0=0.3
# print the coordination on file 'colvar'
PRINT ARG=c FILE=colvar
```

The `groups.dat` file could be very long and include lists of thousand atoms without cluttering the main plumed.↩ dat file.

A GROMACS index file such as the one shown below:

```
[ Protein ]
1 3 5 7 9
2 4 6 8 10
[ Group2 ]
30 31 32 33 34 35 36 37 38 39 40
5
```

can also be imported by using the GROUP keyword as shown below

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GROUP.tmp
# import group named 'Protein' from file index.ndx
pro: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein
# dump all the atoms of the protein on a trajectory file
DUMPATOMS ATOMS=pro FILE=traj.gro
```

A list can be edited with `REMOVE`. For instance, if you are using a water model with three atoms per molecule, you can easily construct the list of hydrogen atoms in this manner

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GROUP.tmp
# take one atom every three, that is oxygens
ox: GROUP ATOMS=1-90:3
# take the remaining atoms, that is hydrogens
hy: GROUP ATOMS=1-90 REMOVE=ox
DUMPATOMS ATOMS=ox FILE=ox.gro
DUMPATOMS ATOMS=hy FILE=hy.gro
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the numerical indexes for the set of atoms in the group. For more information on how to specify lists of atoms see Groups and Virtual Atoms
REMOVE	remove these atoms from the list. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

SORT	(default=off) sort the resulting list
UNIQUE	(default=off) sort atoms and remove duplicated ones
NDX_FILE	the name of index file (gromacs syntax)
NDX_GROUP	the name of the group to be imported (gromacs syntax) - first group found is used by default

5.1.4 MOLINFO

This is part of the generic module

This command is used to provide information on the molecules that are present in your system.

The information on the molecules in your system can either be provided in the form of a pdb file or as a set of lists of atoms that describe the various chains in your system. If a pdb file is used plumed the MOLINFO command will endeavor to recognize the various chains and residues that make up the molecules in your system using the chain↩ IDs and resnumbers from the pdb file. You can then use this information in later commands to specify atom lists in terms residues. For example using this command you can find the backbone atoms in your structure automatically. Starting with PLUMED 2.7 you can use multiple MOLINFO actions. Every time you perform an atom selection, the last available MOLINFO action will be used. This allows you to provide multiple PDB files, for instance using different naming conventions (see `#134`).

Warning

> Please be aware that the PDB parser in plumed is far from perfect. You should thus check the log file and examine what plumed is actually doing whenever you use the MOLINFO action. Also make sure that the atoms are listed in the pdb with the correct order. If you are using gromacs, the safest way is to use reference pdb file generated with `gmx editconf -f topol.tpr -o reference.pdb`.

More information of the PDB parser implemented in PLUMED can be found at this page.

Providing `MOLTYPE=protein`, `MOLTYPE=rna`, or `MOLTYPE=dna` will instruct plumed to look for known residues from these three types of molecule. In other words, this is available for historical reasons and to allow future extensions where alternative lists will be provided. As of now, you can just ignore this keyword.

Using MOLINFO extends the possibility of atoms selection using the @ special symbol. The following shortcuts are available that do not refer to one specific residue:

```
@nucleic : all atoms that are part of a DNA or RNA molecule
@protein : all atoms that are part of a protein
@water : all water molecules
@ions : all the ions
@hydrogens : all hydrogen atoms (those for which the first non-number in the name is a H)
@nonhydrogens : all non hydrogen atoms (those for which the first non-number in the name is not a H)
```

Warning

Be careful since these choices are based on common names used in PDB files. Always check if the selected atoms are correct.

In addition, atoms from a specific residue can be selected with a symbol in this form:

```
@"definition"-chain_residuenum
@"definition"-chainresiduenum
@"definition"-residuenum
```

So for example

```
@psi-1 will select the atoms defining the psi torsion of residue 1
@psi-C1  or @psi-C_1 will define the same torsion for residue 1 of chain C.
@psi-3_1 will define the same torsion for residue 1 of chain 3.
```

Using the underscore to separate chain and residue is available as of PLUMED 2.5 and allows selecting chains with a numeric id.

In the following are listed the current available definitions:

For protein residues, the following groups are available:

```
# quadruplets for dihedral angles
@phi-#
@psi-#
@omega-#
@chi1-#
@chi2-#
@chi3-#
@chi4-#
@chi5-#

# all sidechain atoms (excluding glycine, including all hydrogens)
@sidechain-#
# all backbone atoms (including hydrogens)
@back-#
```

that select the appropriate atoms that define each dihedral angle for residue #.

For DNA or RNA residues, the following groups are available:

```
# quadruplets for backbone dihedral angles
@alpha-#
@beta-#
@gamma-#
@delta-#
@epsilon-#
@zeta-#

# quadruplets for sugar dihedral angles
@v0-#
@v1-#
@v2-#
@v3-#
@v4-#

# quadruplet corresponding to the chi torsional angle
@chi-#
```

```
# backbone, sugar, and base heavy atoms
@back-#
@sugar-#
@base-#

# ordered triplets of atoms on the 6-membered ring of nucleobases
# namely:
#  C2/C4/C6 for pyrimidines
#  C2/C6/C4 for purines
@lcs-#
```

Notice that `zeta` and `epsilon` groups should not be used on 3' end residue and `alpha` and `beta` should not be used on 5' end residue.

Furthermore it is also possible to pick single atoms using the syntax `atom-chain_residuenum`, `@atom-chainresiduenum` or `@atom-residuenum`. As of PLUMED 2.5, this also works when the residue is not a protein/rna/dna residue. For instance, `@OW-100` will select oxygen of water molecule with residue number 100.

Finally, notice that other shortcuts are available even when not using the MOLINFO command (see Specifying Atoms).

Warning

If a residue-chain is repeated twice in the reference pdb only the first entry will be selected.

Bug At the moment the HA1 atoms in a GLY residues are treated as if they are the CB atoms. This may or may not be true - GLY is problematic for secondary structure residues as it is achiral.

Bug If you use WHOLEMOLECULES RESIDUES=1-10 for a 18 amino acid protein (18 amino acids + 2 terminal groups = 20 residues) the code will fail as it will not be able to interpret terminal residue 1.

Advanced atom selection with mdtraj or MDAnalysis

Since PLUMED 2.6 it is possible to use the expressive selection syntax of `mdtraj` and/or `MDAnalysis`:

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=helix.pdb PYTHON_BIN=python
g1: GROUP ATOMS=@mda:backbone
g2: GROUP ATOMS={@mda:{resnum 1 or resid 3:5}}
g3: GROUP ATOMS={@mda:{resid 3:5} @mda:{resnum 1}}
g4: GROUP ATOMS={@mdt:{protein and (backbone or resname ALA)}}
g5: GROUP ATOMS={@mdt:{mass 5.5 to 20}} # masses guessed by mdtraj based on atom type!
g6: GROUP ATOMS={@mda:{resid 3:5} @mda:{resnum 1} 1-10}
```

Here `@mda:` indicates that `MDAnalysis` language is used, whereas `@mdt:` indicates that `mdtraj` language is used. Notice that these languages typically select atoms in order. If you want to specify a different order, you can chain definitions as in `g3` above (compare with `g2`). Selections can be also chained with standard PLUMED selections (see `g6`).

The double braces are required due to the way PLUMED parses atom lists. In particular:

- The outer braces are needed to show PLUMED where the `ATOMS=...` option ends.

- The inner braces are needed to show PLUMED where each selector ends.

MDAnalysis also supports geometric selectors based on atomic coordinates. These selectors **are static** and return lists computed using the coordinates stored in the `MOLINFO` pdb file.

In order to use this syntax you should check the following points at runtime:

1. `plumed --no-mpi config has subprocess` prints `subprocess on` (should be ok on most UNIX systems).
2. You have a python interpreter with mdtraj and/or MDAnalysis installed. You can check using:
   - `python -c "import mdtraj"`
   - `python -c "import MDAnalysis"`

   In order to install these packages refer to their documentation. Pip or conda install should be ok, provided you make sure the correct python interpreter is in the execution PATH at runtime. Notice that you will only need the package(s) related to the syntax that you want to use.
3. In case you installed these modules on a python with a different name (e.g. `python3.6`), the correct check is:
   - `python3.6 -c "import mdtraj"`
   - `python3.6 -c "import MDAnalysis"`

   If this is the case, you should set the environment variable `export PYTHON_BIN=python3.6` or `export PLUMED_PYTHON_BIN=python3.6` (higher priority). Alternatively, directly provide the interpreter in the PLUMED input file using `MOLINFO PYTHON_BIN=python3.6` (even higher priority).
4. The PDB file that you provide to `MOLINFO` should have consecutive atom numbers starting from 1. This is currently enforced since reading atom numbers out of order (as PLUMED does) is not supported by other packages.

Advanced atom selection with VMD (experimental)

Similarly to the `@mda:` and `@mdt:` selectors above, you can use the two following selectors in order to access to VMD syntax for atoms selection:

- `@vmdexec:`: This selector launches an instance of VMD, so `vmd` executable should be in your execution path. Might be very slow or even crash your simulation. Notice that even if `vmd` executable is used, the implementation is still python based and so a working python interpreter should be provided.
- `@vmd:`: This selector tries to import the `vmd` python module. Notice that the best way to obtain this module is not within the standard VMD installer but rather by installing the python module that can be found at this link. The module is also available on conda. You should make sure the module is available in the python interpreter used by MOLINFO (check using the command `python -c "import vmd"`).

These two selectors are experimental and might be removed at some point.

Examples

In the following example the MOLINFO command is used to provide the information on which atoms are in the backbone of a protein to the ALPHARMSD CV.

```
BEGIN_PLUMED_FILE
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=reference.pdb
ALPHARMSD RESIDUES=all TYPE=DRMSD LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} LABEL=a
```

The following example prints the distance corresponding to the hydrogen bonds in a GC Watson-Crick pair.

```
BEGIN_PLUMED_FILE
#SETTINGS MOLFILE=regtest/basic/rt-ermsd/ref.pdb
MOLINFO STRUCTURE=reference.pdb MOLTYPE=dna
hb1: DISTANCE ATOMS=@N2-2,@O2-15
hb2: DISTANCE ATOMS=@N1-2,@N3-15
hb3: DISTANCE ATOMS=@O6-2,@N4-15
PRINT ARG=hb1,hb2,hb3
```

This example use MOLINFO to calculate torsion angles

```
BEGIN_PLUMED_FILE
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
t1: TORSION ATOMS=@phi-3
t2: TORSION ATOMS=@psi-4
PRINT ARG=t1,t2 FILE=colvar STRIDE=10
```

Glossary of keywords and components

The atoms involved can be specified using

CHAIN	(for masochists (mostly Davide Branduardi)) The atoms involved in each of the chains of interest in the structure.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

STRUCTURE	a file in pdb format containing a reference structure. This is used to defines the atoms in the various residues, chains, etc . For more details on the PDB file format visit http://www.↩ wwpdb.org/docs.html
MOLTYPE	(default=protein) what kind of molecule is contained in the pdb file - usually not needed since protein/RNA/DNA are compatible
PYTHON_BIN	(default=default) python interpreter

Options

WHOLE	(default=off) The reference structure is whole, i.e. not broken by PBC

5.1.5 WHOLEMOLECULES

	This is part of the generic module

This action is used to rebuild molecules that can become split by the periodic boundary conditions.

It is similar to the ALIGN_ATOMS keyword of plumed1, and is needed since some MD dynamics code (e.g. GR↩ OMACS) can break molecules during the calculation.

Running some CVs without this command can cause there to be discontinuities changes in the CV value and artifacts in the calculations. This command can be applied more than once. To see what effect is has use a variable without pbc or use the DUMPATOMS directive to output the atomic positions.

Attention

This directive modifies the stored position at the precise moment it is executed. This means that only collective variables which are below it in the input script will see the corrected positions. As a general rule, put it at the top of the input file. Also, unless you know exactly what you are doing, leave the default stride (1), so that this action is performed at every MD step.

The way WHOLEMOLECULES modifies each of the listed entities is this:

- First atom of the list is left in place
- Each atom of the list is shifted by a lattice vectors so that it becomes as close as possible to the previous one, iteratively.

In this way, if an entity consists of a list of atoms such that consecutive atoms in the list are always closer than half a box side the entity will become whole. This can be usually achieved selecting consecutive atoms (1-100), but it is also possible to skip some atoms, provided consecutive chosen atoms are close enough.

Examples

This command instructs plumed to reconstruct the molecule containing atoms 1-20 at every step of the calculation and dump them on a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WHOLEMOLECULES.tmp
# to see the effect, one could dump the atoms as they were before molecule reconstruction:
# DUMPATOMS FILE=dump-broken.xyz ATOMS=1-20
WHOLEMOLECULES ENTITY0=1-20
DUMPATOMS FILE=dump.xyz ATOMS=1-20
```

This command instructs plumed to reconstruct two molecules containing atoms 1-20 and 30-40

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WHOLEMOLECULES.tmp
WHOLEMOLECULES ENTITY0=1-20 ENTITY1=30-40
DUMPATOMS FILE=dump.xyz ATOMS=1-20,30-40
```

This command instructs plumed to reconstruct the chain of backbone atoms in a protein

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WHOLEMOLECULES.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES RESIDUES=all MOLTYPE=protein
```

Glossary of keywords and components

The atoms involved can be specified using

ENTITY	the atoms that make up a molecule that you wish to align. To specify multiple molecules use a list of ENTITY keywords: ENTITY0, ENTITY1,.... You can use multiple instances of this keyword i.e. ENTITY1, ENTITY2, ENTITY3...

Or alternatively by using

RESIDUES	this command specifies that the backbone atoms in a set of residues all must be aligned. It must be used in tandem with the MOLINFO action and the MOLTYPE keyword. If you wish to use all the residues from all the chains in your system you can do so by specifying all. Alternatively, if you wish to use a subset of the residues you can specify the particular residues you are interested in as a list of numbers

Compulsory keywords

STRIDE	(default=1) the frequency with which molecules are reassembled. Unless you are completely certain about what you are doing leave this set equal to 1!

Options

EMST	(default=off) Define atoms sequence in entities using an Euclidean minimum spanning tree
ADDREFERENCE	(default=off) Define the reference position of the first atom of each entity using a PDB file
MOLTYPE	the type of molecule that is under study. This is used to define the backbone atoms

5.1.6 FIT_TO_TEMPLATE

This is part of the generic module

This action is used to align a molecule to a template.

This can be used to move the coordinates stored in plumed so as to be aligned with a provided template in PDB format. Pdb should contain also weights for alignment (see the format of PDB files used e.g. for RMSD). Make sure your PDB file is correctly formatted as explained in this page. Weights for displacement are ignored, since no displacement is computed here. Notice that all atoms (not only those in the template) are aligned. To see what effect try the DUMPATOMS directive to output the atomic positions.

Also notice that PLUMED propagate forces correctly so that you can add a bias on a CV computed after alignment. For many CVs this has no effect, but in some case the alignment can change the result. Examples are:

- POSITION CV since it is affected by a rigid shift of the system.
- DISTANCE CV with COMPONENTS. Since the alignment could involve a rotation (with TYPE=OPTIMAL) the actual components could be different from the original ones.
- CELL components for a similar reason.
- DISTANCE from a FIXEDATOM, provided the fixed atom is introduced *after* the FIT_TO_TEMPLATE action.

Attention

The implementation of TYPE=OPTIMAL is available but should be considered in testing phase. Please report any strange behavior.

This directive modifies the stored position at the precise moment it is executed. This means that only collective variables which are below it in the input script will see the corrected positions. As a general rule, put it at the top of the input file. Also, unless you know exactly what you are doing, leave the default stride (1), so that this action is performed at every MD step.

When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.5, by considering the ordered list of atoms and rebuilding the molecules using a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Examples

Align the atomic position to a template then print them. The following example is only translating the system so as to align the center of mass of a molecule to the one in the reference structure `ref.pdb`:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIT_TO_TEMPLATE.tmp
# dump coordinates before fitting, to see the difference:
DUMPATOMS FILE=dump-before.xyz ATOMS=1-20

# fit coordinates to ref.pdb template
# this is a "TYPE=SIMPLE" fit, so that only translations are used.
FIT_TO_TEMPLATE STRIDE=1 REFERENCE=ref.pdb TYPE=SIMPLE

# dump coordinates after fitting, to see the difference:
DUMPATOMS FILE=dump-after.xyz ATOMS=1-20
```

The following example instead performs a rototranslational fit.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIT_TO_TEMPLATE.tmp
# dump coordinates before fitting, to see the difference:
DUMPATOMS FILE=dump-before.xyz ATOMS=1-20

# fit coordinates to ref.pdb template
# this is a "TYPE=OPTIMAL" fit, so that rototranslations are used.
FIT_TO_TEMPLATE STRIDE=1 REFERENCE=ref.pdb TYPE=OPTIMAL

# dump coordinates after fitting, to see the difference:
DUMPATOMS FILE=dump-after.xyz ATOMS=1-20
```

In both these cases the reference structure should be provided in a reference pdb file such as the one below:

```
ATOM      8  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
END
```

In the following example you see two completely equivalent way to restrain an atom close to a position that is defined in the reference frame of an aligned molecule. You could for instance use this command to calculate the position of the center of mass of a ligand after having aligned the atoms to the reference frame of the protein that is determined by aligning the atoms in the protein to the coordinates provided in the file ref.pdb

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIT_TO_TEMPLATE.tmp
# center of the ligand:
center: CENTER ATOMS=100-110

FIT_TO_TEMPLATE REFERENCE=ref.pdb TYPE=OPTIMAL

# place a fixed atom in the protein reference coordinates:
fix: FIXEDATOM AT=1.0,1.1,1.0

# take the distance between the fixed atom and the center of the ligand
d: DISTANCE ATOMS=center,fix

# apply a restraint
RESTRAINT ARG=d AT=0.0 KAPPA=100.0
```

Notice that you could have obtained an (almost) identical result by adding a fictitious atom to `ref.pdb` with the serial number corresponding to the atom labelled `center` (there is no automatic way to get it, but in this example it should be the number of atoms of the system plus one), and properly setting the weights for alignment and displacement in RMSD. There are two differences to be expected: (ab) FIT_TO_TEMPLATE might be slower since it has to rototranslate all the available atoms and (b) variables employing periodic boundary conditions (such as DISTANCE without `NOPBC`, as in the example above) are allowed after FIT_TO_TEMPLATE, whereas RMSD expects the issues related to the periodic boundary conditions to be already solved. The latter means that before the RMSD statement one should use WRAPAROUND or WHOLEMOLECULES to properly place the ligand.

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which molecules are reassembled. Unless you are completely certain about what you are doing leave this set equal to 1!
REFERENCE	a file in pdb format containing the reference structure and the atoms involved in the CV.
TYPE	(default=SIMPLE) the manner in which RMSD alignment is performed. Should be OPTIMAL or SIMPLE.

Options

NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances

5.1.7 WRAPAROUND

This is part of the generic module

Rebuild periodic boundary conditions around chosen atoms.

Modify position of atoms indicated by ATOMS by shifting them by lattice vectors so that they are as close as possible to the atoms indicated by AROUND. More precisely, for every atom i in the ATOMS list the following procedure is performed:

- The atom j among those in the AROUND list is searched that is closest to atom i.
- The atom i is replaced with its periodic image that is closest to atom j.

This action works similarly to WHOLEMOLECULES in that it replaces atoms coordinate. Notice that only atoms specified with ATOMS are replaced, and that, at variance with WHOLEMOLECULES, the order in which atoms are specified is irrelevant.

This is often convenient at a post processing stage (using the driver), but sometime it is required during the simulation if collective variables need atoms to be in a specific periodic image.

Attention

This directive modifies the stored position at the precise moment it is executed. This means that only collective variables which are below it in the input script will see the corrected positions. As a general rule, put it at the top of the input file. Also, unless you know exactly what you are doing, leave the default stride (1), so that this action is performed at every MD step.

Consider that the computational cost grows with the product of the size of the two lists (ATOMS and AROUND), so that this action can become very expensive. If you are using it to analyze a trajectory this is usually not a big problem. If you use it to analyze a simulation on the fly, e.g. with DUMPATOMS to store a properly wrapped trajectory, consider the possibility of using the STRIDE keyword here (with great care).

Examples

This command instructs plumed to move all the ions to their periodic image that is as close as possible to the rna group.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WRAPAROUND.tmp
rna: GROUP ATOMS=1-100
ions: GROUP ATOMS=101-110
# first make the rna molecule whole
WHOLEMOLECULES ENTITY0=rna
WRAPAROUND ATOMS=ions AROUND=rna
DUMPATOMS FILE=dump.xyz ATOMS=rna,ions
```

In case you want to do it during a simulation and you only care about wrapping the ions in the `dump.xyz` file, you can use the following:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WRAPAROUND.tmp
# add some restraint that do not require molecules to be whole:
a: TORSION ATOMS=1,2,10,11
RESTRAINT ARG=a AT=0.0 KAPPA=5


# then do the things that are required for dumping the trajectory
# notice that they are all done every 100 steps, so as not to
# unnecessarily overload the calculation

rna: GROUP ATOMS=1-100
ions: GROUP ATOMS=101-110
# first make the rna molecule whole
WHOLEMOLECULES ENTITY0=rna STRIDE=100
WRAPAROUND ATOMS=ions AROUND=rna STRIDE=100
DUMPATOMS FILE=dump.xyz ATOMS=rna,ions STRIDE=100
```

Notice that if the biased variable requires a molecule to be whole, you might have to put just the WHOLEMOLECULES command before computing that variable and leave the default STRIDE=1.

This command instructs plumed to center all atoms around the center of mass of a solute molecule.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WRAPAROUND.tmp
solute: GROUP ATOMS=1-100
all: GROUP ATOMS=1-1000
# center of the solute:
# notice that since plumed 2.2 this also works if the
# solute molecule is broken
com: COM ATOMS=solute
# notice that we wrap around a single atom. this should be fast
WRAPAROUND ATOMS=all AROUND=com
DUMPATOMS FILE=dump.xyz ATOMS=all
```

Notice that whereas WHOLEMOLECULES is designed to make molecules whole, WRAPAROUND can easily break molecules. In the last example, if solvent (atoms 101-1000) is made e.g. of water, then water molecules could be broken by WRAPAROUND (hydrogen could end up in an image and oxygen in another one). One solution is to use WHOLEMOLECULES on *all* the water molecules after WRAPAROUND. This is tedious. A better solution is to use the GROUPBY option which is going to consider the atoms listed in ATOMS as a list of groups each of size GROUPBY. The first atom of the group will be brought close to the AROUND atoms. The following atoms of the group will be just brought close to the first atom of the group. Assuming that oxygen is the first atom of each water molecules, in the following examples all the water oxygen atoms will be brought close to the solute, and all the hydrogen atoms will be kept close to their related oxygen.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WRAPAROUND.tmp
solute: GROUP ATOMS=1-100
water: GROUP ATOMS=101-1000
com: COM ATOMS=solute
# notice that we wrap around a single atom. this should be fast
WRAPAROUND ATOMS=solute AROUND=com
# notice that we wrap around a single atom. this should be fast
WRAPAROUND ATOMS=water AROUND=com GROUPBY=3
DUMPATOMS FILE=dump.xyz ATOMS=solute,water
```

Glossary of keywords and components

The atoms involved can be specified using

AROUND	reference atoms. For more information on how to specify lists of atoms see Groups and Virtual Atoms
ATOMS	wrapped atoms. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

STRIDE	(default=1) the frequency with which molecules are reassembled. Unless you are completely certain about what you are doing leave this set equal to 1!
GROUPBY	(default=1) group atoms so as not to break molecules

Options

PAIR	(default=off) Pair atoms in AROUND and ATOMS groups

5.1.8 RESET_CELL

This is part of the generic module

This action is used to rotate the full cell

This can be used to modify the periodic box. Notice that this is done at fixed scaled coordinates, so that also atomic coordinates for the entire system are affected. To see what effect try the DUMPATOMS directive to output the atomic positions.

Also notice that PLUMED propagate forces correctly so that you can add a bias on a CV computed after rotation. See also FIT_TO_TEMPLATE

Currently, only TYPE=TRIANGULAR is implemented, which allows one to reset the cell to a lower triangular one. Namely, a proper rotation is found that allows rotating the box so that the first lattice vector is in the form (ax,0,0), the second lattice vector is in the form (bx,by,0), and the third lattice vector is arbitrary.

Attention

The implementation of this action is available but should be considered in testing phase. Please report any strange behavior.

This directive modifies the stored position at the precise moment it is executed. This means that only collective variables which are below it in the input script will see the corrected positions. Unless you know exactly what you are doing, leave the default stride (1), so that this action is performed at every MD step.

Examples

Reset cell to be triangular after a rototranslational fit

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESET_CELL.tmp
DUMPATOMS FILE=dump-original.xyz ATOMS=1-20
FIT_TO_TEMPLATE STRIDE=1 REFERENCE=ref.pdb TYPE=OPTIMAL
DUMPATOMS FILE=dump-fit.xyz ATOMS=1-20
RESET_CELL TYPE=TRIANGULAR
DUMPATOMS FILE=dump-reset.xyz ATOMS=1-20
```

The reference file for the FIT_TO_TEMPLATE is just a normal pdb file with the format shown below:

```
ATOM      8  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
END
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which molecules are reassembled. Unless you are completely certain about what you are doing leave this set equal to 1!
TYPE	(default=TRIANGULAR) the manner in which the cell is reset

5.1.9 CENTER

This is part of the vatom module

Calculate the center for a group of atoms, with arbitrary weights.

The computed center is stored as a virtual atom that can be accessed in an atom list through the label for the CENTER action that creates it. Notice that the generated virtual atom has charge equal to the sum of the charges and mass equal to the sum of the masses. If used with the MASS flag, then it provides a result identical to COM.

When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.2, by considering the ordered list of atoms and rebuilding the molecule using a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Note

As an experimental feature, CENTER also supports a keyword PHASES. This keyword finds the center of mass for sets of atoms that have been split by the period boundaries by computing scaled coordinates and average trigonometric functions, similarly to CENTER_OF_MULTICOLVAR. Notice that by construction this center position is not invariant with respect to rotations of the atoms at fixed cell lattice. In addition, for symmetric Bravais lattices, it is not invariant with respect to special symmetries. E.g., if you have an hexagonal cell, the center will not be invariant with respect to rotations of 120 degrees. On the other hand, it might make the treatment of PBC easier in difficult cases.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CENTER.tmp
# a point which is on the line connecting atoms 1 and 10, so that its distance
# from 10 is twice its distance from 1:
c1: CENTER ATOMS=1,1,10
# this is another way of stating the same:
c1bis: CENTER ATOMS=1,10 WEIGHTS=2,1

# center of mass among these atoms:
c2: CENTER ATOMS=2,3,4,5 MASS

d1: DISTANCE ATOMS=c1,c2

PRINT ARG=d1
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the list of atoms which are involved the virtual atom's definition. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
MASS	(default=off) If set center is mass weighted
PHASES	(default=off) Compute center using trigonometric phases

WEIGHTS	Center is computed as a weighted average.
SET_CHARGE	Set the charge of the virtual atom to a given value.
SET_MASS	Set the mass of the virtual atom to a given value.

5.1.10 CENTER_OF_MULTICOLVAR

This is part of the multicolvar module

Calculate a a weighted average position based on the value of some multicolvar.

This action calculates the position of a new virtual atom using the following formula:

$$x_\alpha = \frac{1}{2\pi} \arctan\left[\frac{\sum_i w_i f_i \sin(2\pi x_{i,\alpha})}{\sum_i w_i f_i \cos(2\pi x_{i,\alpha})} \right]$$

Where in this expression the w_i values are a set of weights calculated within a multicolvar action and the f_i are the values of the multicolvar functions. The $x_{i,\alpha}$ values are the positions (in scaled coordinates) associated with each of the multicolvars calculated.

Bug The virial contribution for this type of virtual atom is not currently evaluated so do not use in bias functions unless the volume of the cell is fixed

Examples

Lets suppose that you are examining the formation of liquid droplets from gas. You may want to determine the center of mass of any of the droplets formed. In doing this calculation you recognize that the atoms in the liquid droplets will have a higher coordination number than those in the surrounding gas. As you want to calculate the position of the droplets you thus recognize that these atoms with high coordination numbers should have a high weight in the weighted average you are using to calculate the position of the droplet. You can thus calculate the position of the droplet using an input like the one shown below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CENTER_OF_MULTICOLVAR.tmp
c1: COORDINATIONNUMBER LOWMEM SPECIES=1-512 SWITCH={EXP D_0=4.0 R_0=0.5}
cc: CENTER_OF_MULTICOLVAR DATA=c1
```

The first line here calculates the coordination numbers of all the atoms in the system. The virtual atom then uses the values of the coordination numbers calculated by the action labelled c1 when it calculates the Berry Phase average described above. (N.B. the w_i in the above expression are all set equal to 1 in this case)

The above input is fine we can, however, refine this somewhat by making use of a multicolvar transform action as shown below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CENTER_OF_MULTICOLVAR.tmp
c1: COORDINATIONNUMBER SPECIES=1-512 SWITCH={EXP D_0=4.0 R_0=0.5}
cf: MTRANSFORM_MORE DATA=c1 SWITCH={RATIONAL D_0=2.0 R_0=0.1} LOWMEM
cc: CENTER_OF_MULTICOLVAR DATA=cf
```

This input once again calculates the coordination numbers of all the atoms in the system. The middle line then transforms these coordination numbers to numbers between 0 and 1. Essentially any atom with a coordination number larger than 2.0 is given a weight of one and below this value the transformed value decays to zero. It is these transformed coordination numbers that are used to calculate the Berry phase average described in the previous section.

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the list of atoms which are involved the virtual atom's definition. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	find the average value for a multicolvar

Options

COMPONENT	if your input multicolvar is a vector then specify which component you would like to use in calculating the weight

5.1.11 COM

This is part of the vatom module

Calculate the center of mass for a group of atoms.

The computed center of mass is stored as a virtual atom that can be accessed in an atom list through the label for the COM action that creates it.

For arbitrary weights (e.g. geometric center) see CENTER.

When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.2, by considering the ordered list of atoms and rebuilding the molecule using a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Examples

The following input instructs plumed to print the distance between the center of mass for atoms 1,2,3,4,5,6,7 and that for atoms 15,20:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COM.tmp
c1: COM ATOMS=1-7
c2: COM ATOMS=15,20
d1: DISTANCE ATOMS=c1,c2
PRINT ARG=d1
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the list of atoms which are involved the virtual atom's definition. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
MASS	(default=off) If set center is mass weighted
PHASES	(default=off) Compute center using trigonometric phases
WEIGHTS	Center is computed as a weighted average.
SET_CHARGE	Set the charge of the virtual atom to a given value.
SET_MASS	Set the mass of the virtual atom to a given value.

5.1.12 FIXEDATOM

	This is part of the vatom module

Add a virtual atom in a fixed position.

This action creates a virtual atom at a fixed position. The coordinates can be specified in Cartesian components (by default) or in scaled coordinates (SCALED_COMPONENTS). It is also possible to assign a predefined charge or mass to the atom.

Attention

Similar to POSITION this variable is not invariant for translation of the system. Adding a force on it can create serious troubles.

Notice that the distance between two atoms created using FIXEDATOM is invariant for translation. Additionally, if one first align atoms to a reference using FIT_TO_TEMPLATE, then it is safe to add further fixed atoms without breaking translational invariance.

Examples

The following input instructs plumed to compute the angle between distance of atoms 15 and 20 and the z axis and keeping it close to zero.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIXEDATOM.tmp
a: FIXEDATOM AT=0,0,0
b: FIXEDATOM AT=0,0,1
an: ANGLE ATOMS=a,b,15,20
RESTRAINT ARG=an AT=0.0 KAPPA=100.0
```

The following input instructs plumed to align a protein to a template and to then compute the distance between one of the atoms in the protein and the point (10,20,30).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIXEDATOM.tmp
FIT_TO_TEMPLATE STRIDE=1 REFERENCE=ref.pdb TYPE=SIMPLE
a: FIXEDATOM AT=10,20,30
d: DISTANCE ATOMS=a,20
PRINT ARG=d FILE=colvar
```

The reference structure to align to is provided in a pdb file called ref.pdb as shown below:

```
ATOM      8  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
END
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the list of atoms which are involved the virtual atom's definition. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

AT	coordinates of the virtual atom
SET_MASS	(default=1) mass of the virtual atom
SET_CHARGE	(default=0) charge of the virtual atom

Options

SCALED_COMPONENTS	(default=off) use scaled components

5.1.13 GHOST

This is part of the vatom module

Calculate the absolute position of a ghost atom with fixed coordinates in the local reference frame formed by three atoms.

The computed ghost atom is stored as a virtual atom that can be accessed in an atom list through the the label for the GHOST action that creates it.

Examples

The following input instructs plumed to print the distance between the ghost atom and the center of mass for atoms 15,20:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GHOST.tmp
c1: GHOST ATOMS=1,5,10 COORDINATES=10.0,10.0,10.0
c2: COM ATOMS=15,20
d1: DISTANCE ATOMS=c1,c2
PRINT ARG=d1
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the list of atoms which are involved the virtual atom's definition. For more information on how to specify lists of atoms see Groups and Virtual Atoms
COORDINATES	coordinates of the ghost atom in the local reference frame. For more information on how to specify lists of atoms see Groups and Virtual Atoms

5.2 CV Documentation

The following list contains descriptions of a number of the colvars that are currently implemented in PLUMED.

ADAPTIVE_PATH	Compute path collective variables that adapt to the lowest free energy path connecting states A and B.
ALPHABETA	Measures a distance including pbc between the instantaneous values of a set of torsional angles and set of reference values.
ALPHARMSD	Probe the alpha helical content of a protein structure.
ANGLE	Calculate an angle.
ANTIBETARMSD	Probe the antiparallel beta sheet content of your protein structure.
CELL	Calculate the components of the simulation cell
CONSTANT	Return one or more constant quantities with or without derivatives.
CONTACTMAP	Calculate the distances between a number of pairs of atoms and transform each distance by a switching function.
COORDINATION	Calculate coordination numbers.
DHENERGY	Calculate Debye-Huckel interaction energy among GROUPA and GROUPB.
DIHCOR	Measures the degree of similarity between dihedral angles.
DIMER	This CV computes the dimer interaction energy for a collection of dimers.
DIPOLE	Calculate the dipole moment for a group of atoms.
DISTANCE	Calculate the distance between a pair of atoms.
DISTANCE_FROM_CONTOUR	Calculate the perpendicular distance from a Willard-Chandler dividing surface.
EEFSOLV	Calculates EEF1 solvation free energy for a group of atoms.
ENERGY	Calculate the total potential energy of the simulation box.
ERMSD	Calculate eRMSD with respect to a reference structure.
EXTRACV	Allow PLUMED to use collective variables computed in the MD engine.
FAKE	This is a fake colvar container used by cltools or various other actions that supports input and period definitions
GHBFIX	Calculate the GHBFIX interaction energy among GROUPA and GROUP↩ Busing a potential defined in Kuhrova et al., Improving the performance of the AMBER RNA force field bytuning the hydrogen-bonding interactions, J↩ CTC, 2019. Essentially it is a switching function being -1 for small distances and 0 for large distances with a smooth interpolation in the middle. This can be scaled as desired by specifying interaction scaling parameters and energy units.
GPROPERTYMAP	Property maps but with a more flexible framework for the distance metric being used.
GYRATION	Calculate the radius of gyration, or other properties related to it.
PARABETARMSD	Probe the parallel beta sheet content of your protein structure.
PATH	Path collective variables with a more flexible framework for the distance metric being used.
PATHMSD	This Colvar calculates path collective variables.
PCAVARS	Projection on principal component eigenvectors or other high dimensional linear subspace
POSITION	Calculate the components of the position of an atom.
PROJECTION_ON_AXIS	Calculate a position based on the projection along and extension from a defined axis.
PROPERTYMAP	Calculate generic property maps.
PUCKERING	Calculate sugar pseudorotation coordinates.
TEMPLATE	This file provides a template for if you want to introduce a new CV.
TORSION	Calculate a torsional angle.
VOLUME	Calculate the volume of the simulation box.

In addition to the keywords above, by enabling optional modules you can access to the following keywords:

CS2BACKBONE	(from PLUMED-ISDB module) Calculates the backbone chemical shifts for a protein.
EMMI	(from PLUMED-ISDB module) Calculate the fit of a structure or ensemble of structures with a cryo-EM density map.
FRET	(from PLUMED-ISDB module) Calculates the FRET efficiency between a pair of atoms.The efficiency is calculated using the Forster relation:
FUNNEL_PS	(from Funnel-Metadynamics (FM) module) FUNNEL_PS implements the Funnel-Metadynamics (FM) technique in PLUMED 2.
FUSIONPOREEXPANSIONP	(from Membrane Fusion module) A CV for inducing the expansion of a fusion pore from a nucleated fusion pore.
FUSIONPORENUCLEATIONP	(from Membrane Fusion module) A CV for inducing the nucleation of the fusion pore from a hemifusion stalk.
JCOUPLING	(from PLUMED-ISDB module) Calculates ${}^3J$ coupling constants for a dihedral angle.
MEMFUSIONP	(from Membrane Fusion module) Calculate a CV that can induce the formation of the hemifusion stalk between two initially flat and planar bilayers.
NOE	(from PLUMED-ISDB module) Calculates NOE intensities as sums of 1/r^6, also averaging over multiple equivalent atoms or ambiguous NOE.
PCS	(from PLUMED-ISDB module) Calculates the Pseudo-contact shift of a nucleus determined by the presence of a metal ion susceptible to anisotropic magnetization.
PIV	(from PIV collective variable module) Calculates the PIV-distance.
PRE	(from PLUMED-ISDB module) Calculates the Paramagnetic Resonance Enhancement intensity ratio between a spin label atom and a list of atoms .
RDC	(from PLUMED-ISDB module) Calculates the (Residual) Dipolar Coupling between two atoms.
S2CM	(from S2 contact model collective variable module) S2 contact model CV.
SANS	(from PLUMED-ISDB module) Calculates SANS intensity.
SASA_HASEL	(from SASA collective variable module) Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.
SASA_LCPO	(from SASA collective variable module) Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.
SAXS	(from PLUMED-ISDB module) Calculates SAXS intensity.

5.2.1 ADAPTIVE_PATH

This is part of the mapping module

Compute path collective variables that adapt to the lowest free energy path connecting states A and B.

The Path Collective Variables developed by Branduardi and co-workers [5] allow one to compute the progress along a high-dimensional path and the distance from the high-dimensional path. The progress along the path (s) is computed using:

$$s = i_2 + \mathrm{sign}(i_2 - i_1)\frac{\sqrt{(\mathbf{v}_1\cdot\mathbf{v}_2)^2 - |\mathbf{v}_3|^2(|\mathbf{v}_1|^2 - |\mathbf{v}_2|^2)}}{2|\mathbf{v}_3|^2} - \frac{\mathbf{v}_1\cdot\mathbf{v}_3 - |\mathbf{v}_3|^2}{2|\mathbf{v}_3|^2}$$

In this expression $\mathbf{v}_1$ and $\mathbf{v}_3$ are the vectors connecting the current position to the closest and second closest node of the path, respectfully and i_1 and i_2 are the projections of the closest and second closest frames of the path. $\mathbf{v}_2$,

meanwhile, is the vector connecting the closest frame to the second closest frame. The distance from the path, z is calculated using:

$$z = \sqrt{\left[|\mathbf{v}_1|^2 - |\mathbf{v}_2| \left(\frac{\sqrt{(\mathbf{v}_1 \cdot \mathbf{v}_2)^2 - |\mathbf{v}_3|^2(|\mathbf{v}_1|^2 - |\mathbf{v}_2|^2)}}{2|\mathbf{v}_3|^2} - \frac{\mathbf{v}_1 \cdot \mathbf{v}_3 - |\mathbf{v}_3|^2}{2|\mathbf{v}_3|^2} \right) \right]^2}$$

Notice that these are the definitions of s and z that are used by PATH when the GPATH option is employed. The reason for this is that the adaptive path method implemented in this action was inspired by the work of Diaz and Ensing in which these formula were used [4]. To learn more about how the path is adapted we strongly recommend reading this paper.

Examples

The input below provides an example that shows how the adaptive path works. The path is updated every 50 steps of MD based on the data accumulated during the preceding 50 time steps.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ADAPTIVE_PATH.tmp
d1: DISTANCE ATOMS=1,2 COMPONENTS
pp: ADAPTIVE_PATH TYPE=EUCLIDEAN FIXED=2,5 UPDATE=50 WFILE=out-path.pdb WSTRIDE=50 REFERENCE=mypath.pdb
PRINT ARG=d1.x,d1.y,pp.* FILE=colvar
```

In the case above the distance between frames is calculated based on the x and y components of the vector connecting atoms 1 and 2. As such an extract from the input reference path (mypath.pdb) would look as follows:

```
REMARK ARG=d1.x,d1.y d1.x=1.12 d1.y=-.60
END
REMARK ARG=d1.x,d1.y d1.x=.99 d1.y=-.45
END
REMARK ARG=d1.x,d1.y d1.x=.86 d1.y=-.30
END
REMARK ARG=d1.x,d1.y d1.x=.73 d1.y=-.15
END
REMARK ARG=d1.x,d1.y d1.x=.60 d1.y=0
END
REMARK ARG=d1.x,d1.y d1.x=.47 d1.y=.15
END
```

Notice that one can also use RMSD frames in place of arguments like those above.

Glossary of keywords and components

Compulsory keywords

REFERENCE	a pdb file containing the set of reference configurations
TYPE	(default=OPTIMAL-FAST) the manner in which distances are calculated. More information on the different metrics that are available in PLUMED can be found in the section of the manual on Distances from reference configurations
FIXED	the positions in the list of input frames of the two path nodes whose positions remain fixed during the path optimization
HALFLIFE	(default=-1) the number of MD steps after which a previously measured path distance weighs only 50% in the average. This option may increase convergence by allowing to forget the memory of a bad initial guess path. The default is to set this to infinity

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
DISABLE_CHECKS	(default=off) disable checks on reference input structures.
WFILE	file on which to write out the path
WSTRIDE	frequency with which to write out the path

5.2.2 ALPHABETA

This is part of the multicolvar module

Measures a distance including pbc between the instantaneous values of a set of torsional angles and set of reference values.

This colvar calculates the following quantity.

$$s = \frac{1}{2} \sum_i \left[1 + \cos(\phi_i - \phi_i^{\mathrm{Ref}}) \right]$$

where the ϕ_i values are the instantaneous values for the TORSION angles of interest. The ϕ_i^{Ref} values are the user-specified reference values for the torsional angles.

Examples

The following provides an example of the input for an alpha beta similarity.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ALPHABETA.tmp
ALPHABETA ...
ATOMS1=168,170,172,188 REFERENCE1=3.14
ATOMS2=170,172,188,190 REFERENCE2=3.14
ATOMS3=188,190,192,230 REFERENCE3=3.14
LABEL=ab
... ALPHABETA
PRINT ARG=ab FILE=colvar STRIDE=10
```

Because all the reference values are the same we can calculate the same quantity using

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ALPHABETA.tmp
ALPHABETA ...
ATOMS1=168,170,172,188 REFERENCE=3.14
ATOMS2=170,172,188,190
ATOMS3=188,190,192,230
LABEL=ab
... ALPHABETA
PRINT ARG=ab FILE=colvar STRIDE=10
```

Writing out the atoms involved in all the torsion angles in this way can be rather tedious. Thankfully if you are working with protein you can avoid this by using the MOLINFO command. PLUMED uses the pdb file that you provide to this command to learn about the topology of the protein molecule. This means that you can specify torsion angles using the following syntax:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ALPHABETA.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
ALPHABETA ...
ATOMS1=@phi-3 REFERENCE=3.14
ATOMS2=@psi-3
ATOMS3=@phi-4
LABEL=ab
... ALPHABETA
PRINT ARG=ab FILE=colvar STRIDE=10
```

Here, @phi-3 tells plumed that you would like to calculate the ϕ angle in the third residue of the protein. Similarly @psi-4 tells plumed that you want to calculate the ψ angle of the fourth residue of the protein.

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the atoms involved in each of the alpha-beta variables you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one alpha-beta values will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of four atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATO↩ MS3...

Compulsory keywords

REFERENCE	the reference values for each of the torsional angles. If you use a single REFERENCE value the same reference value is used for all torsional angles. You can use multiple instances of this keyword i.e. REFERENCE1, REFERENCE2, REFERENCE3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

TIMINGS	(default=off) output information on the timings of the various parts of the calculation
COEFFICIENT	the coefficient for each of the torsional angles. If you use a single COEFFICIENT value the same reference value is used for all torsional angles. You can use multiple instances of this keyword i.e. COEFFICIENT1, COEFFICIENT2, COEFFICIENT3...

5.2.3 ALPHARMSD

This is part of the secondarystructure module

Probe the alpha helical content of a protein structure.

Any chain of six contiguous residues in a protein chain can form an alpha helix. This colvar thus generates the set of all possible six residue sections and calculates the RMSD distance between the configuration in which the residues find themselves and an idealized alpha helical structure. These distances can be calculated by either aligning the instantaneous structure with the reference structure and measuring each atomic displacement or by calculating differences between the set of inter-atomic distances in the reference and instantaneous structures.

This colvar is based on the following reference [6]. The authors of this paper use the set of distances from the alpha helix configurations to measure the number of segments that have an alpha helical configuration. This is done by calculating the following sum of functions of the rmsd distances:

$$s = \sum_i \frac{1 - \left(\frac{r_i - d_0}{r_0}\right)^n}{1 - \left(\frac{r_i - d_0}{r_0}\right)^m}$$

where the sum runs over all possible segments of alpha helix. By default the NN, MM and D_0 parameters are set equal to those used in [6]. The R_0 parameter must be set by the user - the value used in [6] was 0.08 nm.

If you change the function in the above sum you can calculate quantities such as the average distance from a purely the alpha helical configuration or the distance between the set of residues that is closest to an alpha helix and the reference configuration. To do these sorts of calculations you can use the AVERAGE and MIN keywords. In addition you can use the LESS_THAN keyword if you would like to change the form of the switching function. If you use any of these options you no longer need to specify NN, R_0, MM and D_0.

Please be aware that for codes like gromacs you must ensure that plumed reconstructs the chains involved in your CV when you calculate this CV using anything other than TYPE=DRMSD. For more details as to how to do this see WHOLEMOLECULES.

Examples

The following input calculates the number of six residue segments of protein that are in an alpha helical configuration.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ALPHARMSD.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=helix.pdb
alpha: ALPHARMSD RESIDUES=all
```

Here the same is done use RMSD instead of DRMSD

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ALPHARMSD.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES ENTITY0=1-100
alpha: ALPHARMSD RESIDUES=all TYPE=OPTIMAL R_0=0.1
```

Glossary of keywords and components

Description of components

By default this Action calculates the number of structural units that are within a certain distance of a idealized secondary structure element. This quantity can then be referenced elsewhere in the input by using the label of the action. However, this Action can also be used to calculate the following quantities by using the keywords as described below. The quantities then calculated can be referenced using the label of the action followed by a dot and then the name from the table below. Please note that you can use the LESS_THAN keyword more than once. The resulting components will be labelled *label*.lessthan-1, *label*.lessthan-2 and so on unless you exploit the fact that these labels can be given custom labels by using the LABEL keyword in the description of you LESS_THAN function that you are computing

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.

The atoms involved can be specified using

RESIDUES	this command is used to specify the set of residues that could conceivably form part of the secondary structure. It is possible to use residues numbers as the various chains and residues should have been identified else using an instance of the MOLINFO action. If you wish to use all the residues from all the chains in your system you can do so by specifying all. Alternatively, if you wish to use a subset of the residues you can specify the particular residues you are interested in as a list of numbers. Please be aware that to form secondary structure elements your chain must contain at least N residues, where N is dependent on the particular secondary structure you are interested in. As such if you define portions of the chain with fewer than N residues the code will crash.

Compulsory keywords

TYPE	(default=DRMSD) the manner in which RMSD alignment is performed. Should be OPTIMAL, SIMPLE or DRMSD. For more details on the OPTIMAL and SIMPLE methods see RMSD. For more details on the DRMSD method see DRMSD.
R_0	(default=0.08) The r_0 parameter of the switching function.
D_0	(default=0.0) The d_0 parameter of the switching function
NN	(default=8) The n parameter of the switching function
MM	(default=12) The m parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions
VERBOSE	(default=off) write a more detailed output
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.2.4 ANGLE

This is part of the colvar module

Calculate an angle.

This command can be used to compute the angle between three atoms. Alternatively if four atoms appear in the atom specification it calculates the angle between two vectors identified by two pairs of atoms.

If *three* atoms are given, the angle is defined as:

$$\theta = \arccos\left(\frac{\mathbf{r}_{21}\cdot\mathbf{r}_{23}}{|\mathbf{r}_{21}||\mathbf{r}_{23}|}\right)$$

Here $\mathbf{r}_{ij}$ is the distance vector among the ith and the jth listed atom.

If *four* atoms are given, the angle is defined as:

$$\theta = \arccos\left(\frac{\mathbf{r}_{21}\cdot\mathbf{r}_{34}}{|\mathbf{r}_{21}||\mathbf{r}_{34}|}\right)$$

Notice that angles defined in this way are non-periodic variables and their value is limited by definition between 0 and π.

The vectors $\mathbf{r}_{ij}$ are by default evaluated taking periodic boundary conditions into account. This behavior can be changed with the NOPBC flag.

Examples

This command tells plumed to calculate the angle between the vector connecting atom 1 to atom 2 and the vector connecting atom 2 to atom 3 and to print it on file COLVAR1. At the same time, the angle between vector connecting atom 1 to atom 2 and the vector connecting atom 3 to atom 4 is printed on file COLVAR2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ANGLE.tmp

a: ANGLE ATOMS=1,2,3
# equivalently one could state:
# a: ANGLE ATOMS=1,2,2,3

b: ANGLE ATOMS=1,2,3,4

PRINT ARG=a FILE=COLVAR1
PRINT ARG=b FILE=COLVAR2
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the list of atoms involved in this collective variable (either 3 or 4 atoms). For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances

5.2.5 ANTIBETARMSD

	This is part of the secondarystructure module

Probe the antiparallel beta sheet content of your protein structure.

Two protein segments containing three contiguous residues can form an antiparallel beta sheet. Although if the two segments are part of the same protein chain they must be separated by a minimum of 2 residues to make room for the turn. This colvar thus generates the set of all possible six residue sections that could conceivably form an antiparallel beta sheet and calculates the RMSD distance between the configuration in which the residues find themselves and an idealized antiparallel beta sheet structure. These distances can be calculated by either aligning the instantaneous structure with the reference structure and measuring each atomic displacement or by calculating differences between the set of inter-atomic distances in the reference and instantaneous structures.

This colvar is based on the following reference [6]. The authors of this paper use the set of distances from the anti parallel beta sheet configurations to measure the number of segments that have an configuration that resembles an anti parallel beta sheet. This is done by calculating the following sum of functions of the rmsd distances:

$$s = \sum_i \frac{1 - \left(\frac{r_i - d_0}{r_0}\right)^n}{1 - \left(\frac{r_i - d_0}{r_0}\right)^m}$$

where the sum runs over all possible segments of antiparallel beta sheet. By default the NN, MM and D_0 parameters are set equal to those used in [6]. The R_0 parameter must be set by the user - the value used in [6] was 0.08 nm.

If you change the function in the above sum you can calculate quantities such as the average distance from a purely configuration composed of pure anti-parallel beta sheets or the distance between the set of residues that is closest to an anti-parallel beta sheet and the reference configuration. To do these sorts of calculations you can use the AVERAGE and MIN keywords. In addition you can use the LESS_THAN keyword if you would like to change the form of the switching function. If you use any of these options you no longer need to specify NN, R_0, MM and D_0.

Please be aware that for codes like gromacs you must ensure that plumed reconstructs the chains involved in your CV when you calculate this CV using anything other than TYPE=DRMSD. For more details as to how to do this see WHOLEMOLECULES.

Examples

The following input calculates the number of six residue segments of protein that are in an antiparallel beta sheet configuration.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ANTIBETARMSD.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=beta.pdb
ab: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1
```

Here the same is done use RMSD instead of DRMSD

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ANTIBETARMSD.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES ENTITY0=1-100
hh: ANTIBETARMSD RESIDUES=all TYPE=OPTIMAL R_0=0.1  STRANDS_CUTOFF=1
```

Glossary of keywords and components

Description of components

By default this Action calculates the number of structural units that are within a certain distance of a idealized secondary structure element. This quantity can then be referenced elsewhere in the input by using the label of the action. However, this Action can also be used to calculate the following quantities by using the keywords as described below. The quantities then calculated can be referenced using the label of the action followed by a dot and then the name from the table below. Please note that you can use the LESS_THAN keyword more than once. The resulting components will be labelled *label*.lessthan-1, *label*.lessthan-2 and so on unless you exploit the fact that these labels can be given custom labels by using the LABEL keyword in the description of you LESS_THAN function that you are computing

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.

The atoms involved can be specified using

RESIDUES	this command is used to specify the set of residues that could conceivably form part of the secondary structure. It is possible to use residues numbers as the various chains and residues should have been identified else using an instance of the MOLINFO action. If you wish to use all the residues from all the chains in your system you can do so by specifying all. Alternatively, if you wish to use a subset of the residues you can specify the particular residues you are interested in as a list of numbers. Please be aware that to form secondary structure elements your chain must contain at least N residues, where N is dependent on the particular secondary structure you are interested in. As such if you define portions of the chain with fewer than N residues the code will crash.

Compulsory keywords

TYPE	(default=DRMSD) the manner in which RMSD alignment is performed. Should be OPTIMAL, SIMPLE or DRMSD. For more details on the OPTIMAL and SIMPLE methods see RMSD. For more details on the DRMSD method see DRMSD.
R_0	(default=0.08) The r_0 parameter of the switching function.
D_0	(default=0.0) The d_0 parameter of the switching function
NN	(default=8) The n parameter of the switching function
MM	(default=12) The m parameter of the switching function
STYLE	(default=all) Antiparallel beta sheets can either form in a single chain or from a pair of chains. If STYLE=all all chain configuration with the appropriate geometry are counted. If STYLE=inter only sheet-like configurations involving two chains are counted, while if STYLE=intra only sheet-like configurations involving a single chain are counted

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions
VERBOSE	(default=off) write a more detailed output
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
STRANDS_CUTOFF	If in a segment of protein the two strands are further apart then the calculation of the actual RMSD is skipped as the structure is very far from being beta-sheet like. This keyword speeds up the calculation enormously when you are using the LESS_THAN option. However, if you are using some other option, then this cannot be used

5.2.6 CELL

	This is part of the colvar module

Calculate the components of the simulation cell

Examples

The following input tells plumed to print the squared modulo of each of the three lattice vectors

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CELL.tmp
cell: CELL
aaa:    COMBINE ARG=cell.ax,cell.ay,cell.az POWERS=2,2,2 PERIODIC=NO
bbb:    COMBINE ARG=cell.bx,cell.by,cell.bz POWERS=2,2,2 PERIODIC=NO
ccc:    COMBINE ARG=cell.cx,cell.cy,cell.cz POWERS=2,2,2 PERIODIC=NO
PRINT ARG=aaa,bbb,ccc
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
ax	the ax component of the cell matrix
ay	the ay component of the cell matrix
az	the az component of the cell matrix
bx	the bx component of the cell matrix
by	the by component of the cell matrix
bz	the bz component of the cell matrix
cx	the cx component of the cell matrix
cy	the cy component of the cell matrix
cz	the cz component of the cell matrix

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

5.2.7 CONSTANT

	This is part of the colvar module

Return one or more constant quantities with or without derivatives.

Useful in combination with functions that takes in input constants or parameters.

Examples

The following input instructs plumed to compute the distance between atoms 1 and 2. If this distance is between 1.0 and 2.0, it is printed. If it is lower than 1.0 (larger than 2.0), 1.0 (2.0) is printed

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CONSTANT.tmp
cn: CONSTANT VALUES=1.0,2.0
dis: DISTANCE ATOMS=1,2
sss: SORT ARG=cn.v-0,dis,cn.v-1
PRINT ARG=sss.2
```

In case you want to pass a single value you can use VALUE:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CONSTANT.tmp
cn: CONSTANT VALUE=1.0
dis: DISTANCE ATOMS=1,2
sss: SORT ARG=cn,dis
PRINT ARG=sss.1
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
v	the # value

Options

NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
NODE↩ RIV	(default=off) Set to TRUE if you want values without derivatives.

VALUES	The values of the constants
VALUE	The value of the constant

5.2.8 CONTACTMAP

This is part of the colvar module

Calculate the distances between a number of pairs of atoms and transform each distance by a switching function.

The transformed distance can be compared with a reference value in order to calculate the squared distance between two contact maps. Each distance can also be weighted for a given value. CONTACTMAP can be used together with FUNCPATHMSD to define a path in the contactmap space.

The individual contact map distances related to each contact can be accessed as components named `cm.contact-1`, `cm.contact-2`, etc, assuming that the label of the CONTACTMAP is `cm`.

Examples

The following example calculates switching functions based on the distances between atoms 1 and 2, 3 and 4 and 4 and 5. The values of these three switching functions are then output to a file named colvar.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CONTACTMAP.tmp
CONTACTMAP ATOMS1=1,2 ATOMS2=3,4 ATOMS3=4,5 ATOMS4=5,6 SWITCH={RATIONAL R_0=1.5} LABEL=f1
PRINT ARG=f1.* FILE=colvar
```

The following example calculates the difference of the current contact map with respect to a reference provided. In this case REFERENCE is the fraction of contact that is formed (i.e. the distance between two atoms transformed with the SWITCH), while R_0 is the contact distance. WEIGHT gives the relative weight of each contact to the final distance measure.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CONTACTMAP.tmp
CONTACTMAP ...
ATOMS1=1,2 REFERENCE1=0.1 WEIGHT1=0.5
ATOMS2=3,4 REFERENCE2=0.5 WEIGHT2=1.0
ATOMS3=4,5 REFERENCE3=0.25 WEIGHT3=1.0
ATOMS4=5,6 REFERENCE4=0.0 WEIGHT4=0.5
SWITCH={RATIONAL R_0=1.5}
LABEL=cmap
CMDIST
... CONTACTMAP

PRINT ARG=cmap FILE=colvar
```

The next example calculates calculates fraction of native contacts (Q) for Trp-cage mini-protein. R_0 is the distance at which the switch function is guaranteed to be 1.0 – it doesn't really matter for Q and should be something very small, like 1 A. REF is the reference distance for the contact, e.g. the distance from a crystal structure. LAMBDA is the tolerance for the distance – if set to 1.0, the contact would have to have exactly the reference value to be formed; instead for lambda values of 1.5–1.8 are usually used to allow some slack. BETA is the softness of the switch function, default is 50nm. WEIGHT is the 1/(number of contacts) giving equal weight to each contact.

When using native contact Q switch function, please cite [7]

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CONTACTMAP.tmp
# The full (much-longer) example available in regtest/basic/rt72/

CONTACTMAP ...
ATOMS1=1,67 SWITCH1={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.4059} WEIGHT1=0.003597
ATOMS2=1,68 SWITCH2={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.4039} WEIGHT2=0.003597
ATOMS3=1,69 SWITCH3={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.3215} WEIGHT3=0.003597
ATOMS4=5,61 SWITCH4={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.4277} WEIGHT4=0.003597
ATOMS5=5,67 SWITCH5={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.3851} WEIGHT5=0.003597
ATOMS6=5,68 SWITCH6={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.3811} WEIGHT6=0.003597
ATOMS7=5,69 SWITCH7={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.3133} WEIGHT7=0.003597
LABEL=cmap
SUM
... CONTACTMAP

PRINT ARG=cmap FILE=colvar
```

(See also switchingfunction)

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
contact	By not using SUM or CMDIST each contact will be stored in a component

The atoms involved can be specified using

ATOMS	the atoms involved in each of the contacts you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one contact will be calculated for each ATOM keyword you specify.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Compulsory keywords

SWITCH	The switching functions to use for each of the contacts in your map. You can either specify a global switching function using SWITCH or one switching function for each contact. Details of the various switching functions you can use are provided on switchingfunction.. You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SUM	(default=off) calculate the sum of all the contacts in the input
CMDIST	(default=off) calculate the distance with respect to the provided reference contact map
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
REFERENCE	A reference value for a given contact, by default is 0.0 You can either specify a global reference value using REFERENCE or one reference value for each contact.. You can use multiple instances of this keyword i.e. REFERENCE1, REFERENCE2, REFERENCE3...
WEIGHT	A weight value for a given contact, by default is 1.0 You can either specify a global weight value using WEIGHT or one weight value for each contact.. You can use multiple instances of this keyword i.e. WEIGHT1, WEIGHT2, WEIG↩ HT3...

5.2.9 COORDINATION

This is part of the colvar module

Calculate coordination numbers.

This keyword can be used to calculate the number of contacts between two groups of atoms and is defined as

$$\sum_{i\in A}\sum_{i\in B} s_{ij}$$

where s_{ij} is 1 if the contact between atoms i and j is formed, zero otherwise. In actuality, s_{ij} is replaced with a switching function so as to ensure that the calculated CV has continuous derivatives. The default switching function is:

$$s_{ij} = \frac{1-\left(\frac{\mathbf{r}_{ij}-d_0}{r_0}\right)^n}{1-\left(\frac{\mathbf{r}_{ij}-d_0}{r_0}\right)^m}$$

but it can be changed using the optional SWITCH option.

To make your calculation faster you can use a neighbor list, which makes it that only a relevant subset of the pairwise distance are calculated at every step.

If GROUPB is empty, it will sum the $\frac{N(N-1)}{2}$ pairs in GROUPA. This avoids computing twice permuted indexes (e.g. pair (i,j) and (j,i)) thus running at twice the speed.

Notice that if there are common atoms between GROUPA and GROUPB the switching function should be equal to one. These "self contacts" are discarded by plumed (since version 2.1), so that they actually count as "zero".

Examples

The following example instructs plumed to calculate the total coordination number of the atoms in group 1-10 with the atoms in group 20-100. For atoms 1-10 coordination numbers are calculated that count the number of atoms from the second group that are within 0.3 nm of the central atom. A neighbor list is used to make this calculation faster, this neighbor list is updated every 100 steps.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COORDINATION.tmp
COORDINATION GROUPA=1-10 GROUPB=20-100 R_0=0.3 NLIST NL_CUTOFF=0.5 NL_STRIDE=100
```

The following is a dummy example which should compute the value 0 because the self interaction of atom 1 is skipped. Notice that in plumed 2.0 "self interactions" were not skipped, and the same calculation should return 1.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COORDINATION.tmp
c: COORDINATION GROUPA=1 GROUPB=1 R_0=0.3
PRINT ARG=c STRIDE=10
```

Here's an example that shows what happens when providing COORDINATION with a single group:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COORDINATION.tmp
# define some huge group:
group: GROUP ATOMS=1-1000
# Here's coordination of a group against itself:
c1: COORDINATION GROUPA=group GROUPB=group R_0=0.3
# Here's coordination within a single group:
x: COORDINATION GROUPA=group R_0=0.3
# This is just multiplying times 2 the variable x:
c2: COMBINE ARG=x COEFFICIENTS=2 PERIODIC=NO

# the two variables c1 and c2 should be identical, but the calculation of c2 is twice faster
# since it runs on half of the pairs.
PRINT ARG=c1,c2 STRIDE=10
```

Glossary of keywords and components

The atoms involved can be specified using

GROUPA	First list of atoms. For more information on how to specify lists of atoms see Groups and Virtual Atoms
GROUPB	Second list of atoms (if empty, N∗(N-1)/2 pairs in GROUPA are counted). For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN

D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
PAIR	(default=off) Pair only 1st element of the 1st group with 1st element in the second, etc
NLIST	(default=off) Use a neighbor list to speed up the calculation
NL_CUTOFF	The cutoff for the neighbor list
NL_STRIDE	The frequency with which we are updating the atoms in the neighbor list
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

5.2.10 DHENERGY

This is part of the colvar module

Calculate Debye-Huckel interaction energy among GROUPA and GROUPB.

This variable calculates the electrostatic interaction among GROUPA and GROUPB using a Debye-Huckel approximation defined as

$$\frac{1}{4\pi\epsilon_r\epsilon_0}\sum_{i\in A}\sum_{j\in B} q_i q_j \frac{e^{-\kappa|\mathbf{r}_{ij}|}}{|\mathbf{r}_{ij}|}$$

This collective variable can be used to analyze or induce electrostatically driven reactions [8]. Notice that the value of the DHENERGY is returned in plumed units (see UNITS).

If GROUPB is empty, it will sum the N∗(N-1)/2 pairs in GROUPA. This avoids computing twice permuted indexes (e.g. pair (i,j) and (j,i)) thus running at twice the speed.

Notice that if there are common atoms between GROUPA and GROUPB their interaction is discarded.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DHENERGY.tmp
# this is printing the electrostatic interaction between two groups of atoms
dh: DHENERGY GROUPA=1-10 GROUPB=11-20 EPSILON=80.0 I=0.1 TEMP=300.0
PRINT ARG=dh
```

Glossary of keywords and components

The atoms involved can be specified using

GROUPA	First list of atoms. For more information on how to specify lists of atoms see Groups and Virtual Atoms
GROUPB	Second list of atoms (if empty, N∗(N-1)/2 pairs in GROUPA are counted). For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

I	(default=1.0) Ionic strength (M)
TEMP	(default=300.0) Simulation temperature (K)
EPSILON	(default=80.0) Dielectric constant of solvent

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
PAIR	(default=off) Pair only 1st element of the 1st group with 1st element in the second, etc
NLIST	(default=off) Use a neighbor list to speed up the calculation
NL_CUTOFF	The cutoff for the neighbor list
NL_STRIDE	The frequency with which we are updating the atoms in the neighbor list

5.2.11 DIHCOR

This is part of the multicolvar module

Measures the degree of similarity between dihedral angles.

This colvar calculates the following quantity.

$$s = \frac{1}{2} \sum_i [1 + \cos(\phi_i - \psi_i)]$$

where the ϕ_i and ψ values and the instantaneous values for the TORSION angles of interest.

Examples

The following provides an example input for the DIHCOR action

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIHCOR.tmp
DIHCOR ...
  ATOMS1=1,2,3,4,5,6,7,8
  ATOMS2=5,6,7,8,9,10,11,12
  LABEL=dih
... DIHCOR
PRINT ARG=dih FILE=colvar STRIDE=10
```

In the above input we are calculating the correlation between the torsion angle involving atoms 1, 2, 3 and 4 and the torsion angle involving atoms 5, 6, 7 and 8. This is then added to the correlation between the torsion angle involving atoms 5, 6, 7 and 8 and the correlation angle involving atoms 9, 10, 11 and 12.

Writing out the atoms involved in all the torsion angles in this way can be rather tedious. Thankfully if you are working with protein you can avoid this by using the MOLINFO command. PLUMED uses the pdb file that you provide to this command to learn about the topology of the protein molecule. This means that you can specify torsion angles using the following syntax:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIHCOR.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
dih: DIHCOR ...
ATOMS1=@phi-3,@psi-3
ATOMS2=@psi-3,@phi-4
ATOMS3=@phi-4,@psi-4
...
PRINT ARG=dih FILE=colvar STRIDE=10
```

Here, @phi-3 tells plumed that you would like to calculate the ϕ angle in the third residue of the protein. Similarly @psi-4 tells plumed that you want to calculate the ψ angle of the fourth residue of the protein.

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the atoms involved in each of the dihedral correlation values you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one dihedral correlation will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of 8 atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.2.12 DIMER

This is part of the colvar module

This CV computes the dimer interaction energy for a collection of dimers.

Each dimer represents an atom, as described in the dimer paper [9]. A system of N atoms is thus represented with N dimers, each Dimer being composed of two beads and eventually a virtual site representing its center of mass.

A typical configuration for a dimerized system has the following ordering of atoms:

1 TAG1 X Y Z N atoms representing the first bead of each Dimer

2 TAG2 X Y Z

...

N TAGN X Y Z N atoms representing the second bead of each Dimer

N+1 TAG1 X Y Z

N+2 TAG2 X Y Z

...

2N TAGN X Y Z Optional: N atoms representing the center of mass of each Dimer

2N+1 TAG1 X Y Z

2N+2 TAG2 X Y Z

...

3N TAGN X Y Z The configuration might go on with un-dimerized atoms (like a solvent)

3N+1

3N+2

...

The Dimer interaction energy is defined between atoms x and N+x, for x=1,...,N and is characterized by two parameters Q and DSIGMA. These are passed as mandatory arguments along with the temperature of the system.

Examples

This line tells Plumed to compute the Dimer interaction energy for every dimer in the system.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIMER.tmp
dim: DIMER TEMP=300 Q=0.5 ALLATOMS DSIGMA=0.002
```

If the simulation doesn't use virtual sites for the dimers centers of mass, Plumed has to know in order to determine correctly the total number of dimers from the total number of atoms:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIMER.tmp
dim: DIMER TEMP=300 Q=0.5 ALLATOMS DSIGMA=0.002 NOVSITES
```

The NOVSITES flag is not required if one provides the atom serials of each Dimer. These are defined through two lists of atoms provided **instead** of the ALLATOMS keyword. For example, the Dimer interaction energy of dimers specified by beads (1;23),(5;27),(7;29) is:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIMER.tmp
dim: DIMER TEMP=300 Q=0.5 ATOMS1=1,5,7 ATOMS2=23,27,29 DSIGMA=0.002
```

Note that the ATOMS1,ATOMS2 keywords can support atom groups and interval notation as defined in GROUP.

In a Replica Exchange simulation the keyword DSIGMA can be used in two ways: if a plumed.n.dat file is provided for each replica, then DSIGMA is passed as a single value, like in the previous examples, and each replica will read its own DSIGMA value. If a unique plumed.dat is given, DSIGMA has to be a list containing a value for each replica. For 4 replicas:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIMER.tmp
#SETTINGS NREPLICAS=4
dim: DIMER TEMP=300 Q=0.5 ATOMS1=1,5,7 ATOMS2=23,27,29 DSIGMA=0.002,0.002,0.004,0.01
```

Usage of the CV

The dimer interaction is not coded in the driver program and has to be inserted in the Hamiltonian of the system as a linear RESTRAINT (see RESTRAINT):

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIMER.tmp
dim: DIMER TEMP=300 Q=0.5 ALLATOMS DSIGMA=0.002
RESTRAINT ARG=dim AT=0 KAPPA=0 SLOPE=1 LABEL=dimforces
```

In a replica exchange, Metadynamics (see METAD) can be used on the Dimer CV to reduce the number of replicas. Just keep in mind that METAD SIGMA values should be tuned in the standard way for each replica according to the value of DSIGMA.

Glossary of keywords and components

The atoms involved can be specified using

ATOMS1	The list of atoms representing the first bead of each Dimer being considered by this CV. Used if ALLATOMS flag is missing. For more information on how to specify lists of atoms see Groups and Virtual Atoms
ATOMS2	The list of atoms representing the second bead of each Dimer being considered by this CV. Used if ALLATOMS flag is missing. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DSIGMA	The interaction strength of the dimer bond.
Q	The exponent of the dimer potential.
TEMP	The temperature (in Kelvin) of the simulation.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
ALLATOMS	(default=off) Use EVERY atom of the system. Overrides ATOMS keyword.
NOVSITES	(default=off) If present the configuration is without virtual sites at the centroid positions.

5.2.13 DIPOLE

	This is part of the colvar module

Calculate the dipole moment for a group of atoms.

When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.5, by considering the ordered list of atoms and rebuilding the molecule with a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Examples

The following tells plumed to calculate the dipole of the group of atoms containing the atoms from 1-10 and print it every 5 steps

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DIPOLE.tmp
d: DIPOLE GROUP=1-10
PRINT FILE=output STRIDE=5 ARG=d
```

Attention

If the total charge Q of the group in non zero, then a charge Q/N will be subtracted to every atom, where N is the number of atoms. This implies that the dipole (which for a charged system depends on the position) is computed on the geometric center of the group.

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Keyword	Description
x	**COMPONENTS**	the x-component of the dipole
y	**COMPONENTS**	the y-component of the dipole
z	**COMPONENTS**	the z-component of the dipole

The atoms involved can be specified using

GROUP	the group of atoms we are calculating the dipole moment for. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
COMPONENTS	(default=off) calculate the x, y and z components of the dipole separately and store them as label.x, label.y and label.z

5.2.14 DISTANCE

	This is part of the colvar module

Calculate the distance between a pair of atoms.

By default the distance is computed taking into account periodic boundary conditions. This behavior can be changed with the NOPBC flag. Moreover, single components in Cartesian space (x,y, and z, with COMPONENTS) or single components projected to the three lattice vectors (a,b, and c, with SCALED_COMPONENTS) can be also computed.

Notice that Cartesian components will not have the proper periodicity! If you have to study e.g. the permeation of a molecule across a membrane, better to use SCALED_COMPONENTS.

Examples

The following input tells plumed to print the distance between atoms 3 and 5, the distance between atoms 2 and 4 and the x component of the distance between atoms 2 and 4.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCE.tmp
d1:  DISTANCE ATOMS=3,5
d2:  DISTANCE ATOMS=2,4
d2c: DISTANCE ATOMS=2,4 COMPONENTS
PRINT ARG=d1,d2,d2c.x
```

The following input computes the end-to-end distance for a polymer of 100 atoms and keeps it at a value around 5.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCE.tmp
WHOLEMOLECULES ENTITY0=1-100
e2e: DISTANCE ATOMS=1,100 NOPBC
RESTRAINT ARG=e2e KAPPA=1 AT=5
```

Notice that NOPBC is used to be sure that if the end-to-end distance is larger than half the simulation box the distance is compute properly. Also notice that, since many MD codes break molecules across cell boundary, it might be necessary to use the WHOLEMOLECULES keyword (also notice that it should be *before* distance). The list of atoms provided to WHOLEMOLECULES here contains all the atoms between 1 and 100. Strictly speaking, this is not necessary. If you know for sure that atoms with difference in the index say equal to 10 are *not* going to be farther than half cell you can e.g. use

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCE.tmp
WHOLEMOLECULES ENTITY0=1,10,20,30,40,50,60,70,80,90,100
e2e: DISTANCE ATOMS=1,100 NOPBC
RESTRAINT ARG=e2e KAPPA=1 AT=5
```

Just be sure that the ordered list provide to WHOLEMOLECULES has the following properties:

- Consecutive atoms should be closer than half-cell throughout the entire simulation.
- Atoms required later for the distance (e.g. 1 and 100) should be included in the list

The following example shows how to take into account periodicity e.g. in z-component of a distance

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCE.tmp
# this is a center of mass of a large group
c: COM ATOMS=1-100
# this is the distance between atom 101 and the group
d: DISTANCE ATOMS=c,101 COMPONENTS
# this makes a new variable, dd, equal to d and periodic, with domain -10,10
# this is the right choise if e.g. the cell is orthorombic and its size in
# z direction is 20.
dz: COMBINE ARG=d.z PERIODIC=-10,10
# metadynamics on dd
METAD ARG=dz SIGMA=0.1 HEIGHT=0.1 PACE=200
```

Using SCALED_COMPONENTS this problem should not arise because they are always periodic with domain (-0.↩
5,+0.5).

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Keyword	Description
x	**COMPONENTS**	the x-component of the vector connecting the two atoms
y	**COMPONENTS**	the y-component of the vector connecting the two atoms
z	**COMPONENTS**	the z-component of the vector connecting the two atoms
a	**SCALED_COMPONENTS**	the normalized projection on the first lattice vector of the vector connecting the two atoms
b	**SCALED_COMPONENTS**	the normalized projection on the second lattice vector of the vector connecting the two atoms
c	**SCALED_COMPONENTS**	the normalized projection on the third lattice vector of the vector connecting the two atoms

The atoms involved can be specified using

ATOMS	the pair of atom that we are calculating the distance between. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
COMPONENTS	(default=off) calculate the x, y and z components of the distance separately and store them as label.x, label.y and label.z
SCALED_COMPONENTS	(default=off) calculate the a, b and c scaled components of the distance separately and store them as label.a, label.b and label.c

5.2.15 DISTANCE_FROM_CONTOUR

	This is part of the multicolvar module

Calculate the perpendicular distance from a Willard-Chandler dividing surface.

Suppose that you have calculated a multicolvar. By doing so you have calculated a set of colvars, s_i, and each of these colvars has a well defined position in space (x_i, y_i, z_i). You can use this information to calculate a phase-field model of the colvar density using:

$$p(x,y,x) = \sum_i s_i K \left[\frac{x - x_i}{\sigma_x}, \frac{y - y_i}{\sigma_y}, \frac{z - z_i}{\sigma_z} \right]$$

In this expression σ_x, σ_y and σ_z are bandwidth parameters and K is one of the kernelfunctions. This is what is done within MULTICOLVARDENS

The Willard-Chandler surface is a surface of constant density in the above phase field $p(x, y, z)$. In other words, it is a set of points, (x', y', z'), in your box which have:

$$p(x', y', z') = \rho$$

where ρ is some target density. This action calculates the distance projected on the x, y or z axis between the position of some test particle and this surface of constant field density.

Examples

In this example atoms 2-100 are assumed to be concentrated along some part of the z axis so that you an interface between a liquid/solid and the vapor. The quantity dc measures the distance between the surface at which the density of 2-100 atoms is equal to 0.2 and the position of the test particle atom 1.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCE_FROM_CONTOUR.tmp
dens: DENSITY SPECIES=2-100
dc: DISTANCE_FROM_CONTOUR DATA=dens ATOM=1 BANDWIDTH=0.5,0.5,0.5 DIR=z CONTOUR=0.2
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Description
dist1	the distance between the reference atom and the nearest contour
dist2	the distance between the reference atom and the other contour
qdist	the differentiable (squared) distance between the two contours (see above)
thickness	the distance between the two contours on the line from the reference atom

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The atoms involved can be specified using

ATOM	The atom whose perpendicular distance we are calculating from the contour. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The input base multicolvar which is being used to calculate the contour
BANDWIDTH	the bandwidths for kernel density estimation
KERNEL	(default=gaussian) the kernel function you are using. More details on the kernels available in plumed plumed can be found in kernelfunctions.
DIR	the direction perpendicular to the contour that you are looking for
CONTOUR	the value we would like for the contour
TOLERANCE	(default=0.1) this parameter is used to manage periodic boundary conditions. The problem here is that we can be between contours even when we are not within the membrane because of periodic boundary conditions. When we are in the contour, however, we should have it so that the sums of the absolute values of the distances to the two contours is approximately the distance between the two contours. There can be numerical errors in these calculations, however, so we specify a small tolerance here

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.2.16 EEFSOLV

This is part of the colvar module

Calculates EEF1 solvation free energy for a group of atoms.

EEF1 is a solvent-accessible surface area based model, where the free energy of solvation is computed using a pairwise interaction term for non-hydrogen atoms:

$$\Delta G_i^{\rm solv} = \Delta G_i^{\rm ref} - \sum_{j \neq i} f_i(r_{ij}) V_j$$

where $\Delta G_i^{\rm solv}$ is the free energy of solvation, $\Delta G_i^{\rm ref}$ is the reference solvation free energy, V_j is the volume of atom j and

$$f_i(r) 4\pi r^2 = \frac{2}{\sqrt{\pi}} \frac{\Delta G_i^{\rm free}}{\lambda_i} \exp\left\{ -\frac{(r-R_i)^2}{\lambda_i^2} \right\}$$

where $\Delta G_i^{\rm free}$ is the solvation free energy of the isolated group, λ_i is the correlation length equal to the width of the first solvation shell and R_i is the van der Waals radius of atom i.

The output from this collective variable, the free energy of solvation, can be used with the BIASVALUE keyword to provide implicit solvation to a system. All parameters are designed to be used with a modified CHARMM36 force

field. It takes only non-hydrogen atoms as input, these can be conveniently specified using the GROUP action with the NDX_GROUP parameter. To speed up the calculation, EEFSOLV internally uses a neighbor list with a cutoff dependent on the type of atom (maximum of 1.95 nm). This cutoff can be extended further by using the NL_BUFFER keyword.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EEFSOLV.tmp
#SETTINGS MOLFILE=regtest/basic/rt77/peptide.pdb
MOLINFO MOLTYPE=protein STRUCTURE=peptide.pdb
WHOLEMOLECULES ENTITY0=1-111

# This allows us to select only non-hydrogen atoms
#SETTINGS AUXFILE=regtest/basic/rt77/index.ndx
protein-h: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein-H

# We extend the cutoff by 0.1 nm and update the neighbor list every 40 steps
solv: EEFSOLV ATOMS=protein-h

# Here we actually add our calculated energy back to the potential
bias: BIASVALUE ARG=solv

PRINT ARG=solv FILE=SOLV
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	The atoms to be included in the calculation, e.g. the whole protein.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NL_BUFFER	(default=0.1) The buffer to the intrinsic cutoff used when calculating pairwise interactions.
NL_STRIDE	(default=40) The frequency with which the neighbor list is updated.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
TEMP_CORRECTION	(default=off) Correct free energy of solvation constants for temperatures different from 298.15 K

5.2.17 ENERGY

This is part of the colvar module

Calculate the total potential energy of the simulation box.

The potential energy can be biased e.g. with umbrella sampling [10] or with well-tempered metadynamics [11].

Notice that this CV could be unavailable with some MD code. When it is available, and when also replica exchange is available, metadynamics applied to ENERGY can be used to decrease the number of required replicas.

Bug This ENERGY does not include long tail corrections. Thus when using e.g. LAMMPS "`pair_modify tail yes`" or GROMACS "`DispCorr Ener`" (or "`DispCorr EnerPres`"), the potential energy from ENERGY will be slightly different form the one of the MD code. You should still be able to use ENERGY and then reweight your simulation with the correct MD energy value.

Bug Acceptance for replica exchange when ENERGY is biased is computed correctly only if all the replicas have the same potential energy function. This is for instance not true when using GROMACS with lambda replica exchange or with plumed-hrex branch.

Examples

The following input instructs plumed to print the energy of the system

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ENERGY.tmp
ene: ENERGY
PRINT ARG=ene
```

Glossary of keywords and components

5.2.18 ERMSD

This is part of the colvar module

Calculate eRMSD with respect to a reference structure.

eRMSD is a metric developed for measuring distances between three-dimensional RNA structures. The standard RMSD measure is highly inaccurate when measuring distances among three-dimensional structures of nucleic acids. It is not unusual, for example, that two RNA structures with low RMSD (i.e. less than 0.4nm) display a completely different network of base-base interactions.

eRMSD measures the distance between structures by considering only the relative positions and orientations of nucleobases. The eRMSD can be considered as a vectorial version of contact maps and it is calculated as follows:

1. Set up a local reference system in the center of the six-membered ring of each nucleobase in a molecule. The xy plane lies on the plane of the nucleobase, and it is oriented such that the Watson-Crick interaction is always at $\theta \approx 60°$.
2. Calculate all pairwise distance vectors $\vec{r}_{i,j}$ among base centers.
3. Rescale distance vectors as $\tilde{\vec{r}}_{i,j} = (r_x/a, r_y/a, r_z/b)$, where a=b=5 , c=3 . This rescaling has the effect of weighting more deviations on the z-axis with respect to the x/y directions.
4. Calculate the G vectors

$$\vec{G}(\tilde{\vec{r}}) = (\sin(\gamma\tilde{r})\tilde{r}_x/\tilde{r}, \sin(\gamma\tilde{r})\tilde{r}_y/\tilde{r}, \sin(\gamma\tilde{r})\tilde{r}_z/\tilde{r}, 1+\cos(\gamma\tilde{r})) \times \frac{\Theta(\tilde{r}_{cutoff} - \tilde{r})}{\gamma}$$

Here, $\gamma = \pi/\tilde{r}_{cutoff}$ and Θ is the Heaviside step function. The default cutoff is set to 2.4.

1. The eRMSD between two structures α and β reads

$$eRMSD = \sqrt{\frac{1}{N}\sum_{j,k} |\vec{G}(\tilde{\vec{r}}^{\alpha}_{jk}) - \vec{G}(\tilde{\vec{r}}^{\beta}_{jk})|^2}$$

Using the default cutoff, two structures with eRMSD of 0.7 or lower can be considered as significantly similar. A full description of the eRMSD can be found in [12]

ERMSD is computed using the position of three atoms on the 6-membered ring of each involved nucleobase. The atoms should be:

- C2,C4,C6 for pyrimdines
- C2,C6,C4 for purines

The different order for purines and pyrimidines is fundamental and allows you to compute ERMSD between structures with different sequences as well! Notice that the simplest way to avoid mistakes in choosing these atoms is to use the `@lcs-#` strings as shown in the examples (see also MOLINFO).

Warning

Notice that the ERMSD implemented here is not integrated with the other metrics in plumed. As a consequence, it is not (yet) possible to e.g. build path collective variables using ERMSD

Notice that ERMSD expect a single molecule and makes coordinate whole before anything else. As such, results might be unexpected for a multi molecular system.

Examples

Calculate the eRMSD from reference structure reference.pdb using the default cutoff (2.4). The list of residues involved in the calculation has to be specified. In this example, the eRMSD is calculated considering residues 1,2,3,4,5,6.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ERMSD.tmp
#SETTINGS MOLFILE=regtest/basic/rt-ermsd/ref.pdb
MOLINFO STRUCTURE=reference.pdb
eRMSD1: ERMSD REFERENCE=reference.pdb ATOMS=@lcs-1,@lcs-2,@lcs-3,@lcs-4,@lcs-5,@lcs-6
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the list of atoms (use lcs). For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

REFERENCE	a file in pdb format containing the reference structure and the atoms involved in the CV.
CUTOFF	(default=2.4) only pairs of atoms closer than CUTOFF are considered in the calculation.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
PAIRS	List of pairs considered. All pairs are considered if this value is not specified.

5.2.19 EXTRACV

This is part of the colvar module

Allow PLUMED to use collective variables computed in the MD engine.

This feature requires the MD engine to use special instructions to pass to PLUMED the value of some pre-computed collective variable. Check the documentation of the MD code to find out which collective variables can be computed and passed to PLUMED. These variables can then be accessed by name using the EXTRACV action.

Examples

This example takes the lambda variable pre-computed in GROMACS and apply to it a restraint to keep it close to the value 3.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EXTRACV.tmp
l: EXTRACV NAME=lambda
RESTRAINT ARG=l KAPPA=10 AT=3
```

Glossary of keywords and components

Compulsory keywords

NAME	name of the CV as computed by the MD engine

5.2.20 FAKE

	This is part of the colvar module

This is a fake colvar container used by cltools or various other actions that supports input and period definitions

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FAKE.tmp
FAKE ATOMS=1 PERIODIC=-3.14,3.14   LABEL=d2
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the fake atom index, a number is enough. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

PERIODIC	if the output of your function is periodic then you should specify the periodicity of the function. If the output is not periodic you must state this using PERIODIC=NO,NO (one for the lower and the other for the upper boundary). For multicomponents then it is PERIOD↩ IC=mincomp1,maxcomp1,mincomp2,maxcomp2 etc

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
COMPONENTS	additional components that this variable is supposed to have. Periodicity is ruled by PERIODIC keyword

5.2.21 GHBFIX

	This is part of the colvar module

Calculate the GHBFIX interaction energy among GROUPA and GROUPB using a potential defined in Kuhrova et al., Improving the performance of the AMBER RNA force field by tuning the hydrogen-bonding interactions, JCTC, 2019. Essentially it is a switching function being -1 for small distances and 0 for large distances with a smooth interpolation in the middle. This can be scaled as desired by specifying interaction scaling parameters and energy units.

This collective variable can be used to analyze hydrogen bond interactions, or to generate bias potentials. Notice that the value of the GHBFIX is returned in plumed units (see UNITS), if not specified differently via ENERGY_U↩ NITS.

Examples

This example prints the GHBFIX interaction in kcal/mol between two groups of atoms using D_0, D_MAX and C It is applied in the functional form introduced in the pioneering paper. The types of atoms 1-6 should be defined in typesTable_examples.dat while their interaction parameters should be defined in scalingParameters↩ _examples.dat in kBT units.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/GHBFIX.tmp
#SETTINGS AUXFOLDER=regtest/basic/rt-ghbfix
gh: GHBFIX PAIR GROUPA=1,2,3 GROUP=4,5,6 D_0=0.2 D_MAX=0.3 C=0.8 TYPES=typesTable_examples.dat PARAMS=scalingP
PRINT FILE=output ARG=gh
```

Glossary of keywords and components

The atoms involved can be specified using

GROUPA	First list of atoms. For more information on how to specify lists of atoms see Groups and Virtual Atoms
GROUPB	Second list of atoms (if empty, N∗(N-1)/2 pairs in GROUPA are counted). For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

TYPES	the value of TYPES in the switching function
PARAMS	the value of PARAMS in the switching function
D_MAX	the value of D_MAX in the switching function
D_0	the value of D_0 in the switching function
C	the value of C in the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
PAIR	(default=off) Pair only 1st element of the 1st group with 1st element in the second, etc
NLIST	(default=off) Use a neighbor list to speed up the calculation
NL_CUTOFF	The cutoff for the neighbor list
NL_STRIDE	The frequency with which we are updating the atoms in the neighbor list
ENERGY_UNITS	the value of ENERGY_UNITS in the switching function

5.2.22 GPROPERTYMAP

This is part of the mapping module

Property maps but with a more flexible framework for the distance metric being used.

This colvar calculates a property map using the formalism developed by Spiwok [13]. In essence if you have the value of some property, X_i, that it takes at a set of high-dimensional positions then you calculate the value of the property at some arbitrary point in the high-dimensional space using:

$$X = \frac{\sum_i X_i * \exp(-\lambda D_i(x))}{\sum_i \exp(-\lambda D_i(x))}$$

Within PLUMED there are multiple ways to define the distance from a high-dimensional configuration, D_i. You could calculate the RMSD distance or you could calculate the amount by which a set of collective variables change. As such this implementation of the property map allows one to use all the different distance metric that are discussed in Distances from reference configurations. This is as opposed to the alternative implementation PROPERTYMAP which is a bit faster but which only allows one to use the RMSD distance.

Examples

The input shown below can be used to calculate the interpolated values of two properties called X and Y based on the values that these properties take at a set of reference configurations and using the formula above. For this input the distances between the reference configurations and the instantaneous configurations are calculated using the OPTIMAL metric that is discussed at length in the manual pages on RMSD.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GPROPERTYMAP.tmp
p2: GPROPERTYMAP REFERENCE=allv.pdb PROPERTY=X,Y LAMBDA=69087
PRINT ARG=p2.X,p2.Y,p2.zpath STRIDE=1 FILE=colvar
```

The additional input file for this calculation, which contains the reference frames and the values of X and Y at these reference points has the following format.

```
REMARK X=1 Y=2
ATOM      1  CL  ALA     1      -3.171   0.295   2.045  1.00  1.00
ATOM      5  CLP ALA     1      -1.819  -0.143   1.679  1.00  1.00
ATOM      6  OL  ALA     1      -1.177  -0.889   2.401  1.00  1.00
ATOM      7  NL  ALA     1      -1.313   0.341   0.529  1.00  1.00
ATOM      8  HL  ALA     1      -1.845   0.961  -0.011  1.00  1.00
ATOM      9  CA  ALA     1      -0.003  -0.019   0.021  1.00  1.00
ATOM     10  HA  ALA     1       0.205  -1.051   0.259  1.00  1.00
ATOM     11  CB  ALA     1       0.009   0.135  -1.509  1.00  1.00
ATOM     15  CRP ALA     1       1.121   0.799   0.663  1.00  1.00
ATOM     16  OR  ALA     1       1.723   1.669   0.043  1.00  1.00
ATOM     17  NR  ALA     1       1.423   0.519   1.941  1.00  1.00
ATOM     18  HR  ALA     1       0.873  -0.161   2.413  1.00  1.00
ATOM     19  CR  ALA     1       2.477   1.187   2.675  1.00  1.00
END
FIXED
REMARK X=2 Y=3
ATOM      1  CL  ALA     1      -3.175   0.365   2.024  1.00  1.00
ATOM      5  CLP ALA     1      -1.814  -0.106   1.685  1.00  1.00
ATOM      6  OL  ALA     1      -1.201  -0.849   2.425  1.00  1.00
ATOM      7  NL  ALA     1      -1.296   0.337   0.534  1.00  1.00
ATOM      8  HL  ALA     1      -1.807   0.951  -0.044  1.00  1.00
ATOM      9  CA  ALA     1       0.009  -0.067   0.033  1.00  1.00
ATOM     10  HA  ALA     1       0.175  -1.105   0.283  1.00  1.00
ATOM     11  CB  ALA     1       0.027   0.046  -1.501  1.00  1.00
ATOM     15  CRP ALA     1       1.149   0.725   0.654  1.00  1.00
ATOM     16  OR  ALA     1       1.835   1.491  -0.011  1.00  1.00
ATOM     17  NR  ALA     1       1.380   0.537   1.968  1.00  1.00
ATOM     18  HR  ALA     1       0.764  -0.060   2.461  1.00  1.00
ATOM     19  CR  ALA     1       2.431   1.195   2.683  1.00  1.00
END
```

Glossary of keywords and components

Compulsory keywords

REFERENCE	a pdb file containing the set of reference configurations
PROPERTY	the property to be used in the index. This should be in the REMARK of the reference
TYPE	(default=OPTIMAL-FAST) the manner in which distances are calculated. More information on the different metrics that are available in PLUMED can be found in the section of the manual on Distances from reference configurations
LAMBDA	(default=0) the value of the lambda parameter for paths

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
DISABLE_CHECKS	(default=off) disable checks on reference input structures.
NOZPATH	(default=off) do not calculate the zpath position
NOMAPPING	(default=off) do not calculate the position on the manifold

5.2.23 GYRATION

This is part of the colvar module

Calculate the radius of gyration, or other properties related to it.

The different properties can be calculated and selected by the TYPE keyword: the Radius of Gyration (RADIUS); the Trace of the Gyration Tensor (TRACE); the Largest Principal Moment of the Gyration Tensor (GTPC_1); the middle Principal Moment of the Gyration Tensor (GTPC_2); the Smallest Principal Moment of the Gyration Tensor (GTPC_3); the Asphericiry (ASPHERICITY); the Acylindricity (ACYLINDRICITY); the Relative Shape Anisotropy (KAPPA2); the Smallest Principal Radius Of Gyration (GYRATION_3); the Middle Principal Radius of Gyration (GYRATION_2); the Largest Principal Radius of Gyration (GYRATION_1). A derivation of all these different variants can be found in [14]

The radius of gyration is calculated using:

$$s_{\mathrm{Gyr}} = \left(\frac{\sum_i^n m_i |r_i - r_{\mathrm{COM}}|^2}{\sum_i^n m_i} \right)^{1/2}$$

with the position of the center of mass r_{COM} given by:

$$r_{\mathrm{COM}} = \frac{\sum_i^n r_i \, m_i}{\sum_i^n m_i}$$

The radius of gyration usually makes sense when atoms used for the calculation are all part of the same molecule. When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.2, by considering the ordered list of atoms and rebuilding the broken entities using a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Examples

The following input tells plumed to print the radius of gyration of the chain containing atoms 10 to 20.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GYRATION.tmp
GYRATION TYPE=RADIUS ATOMS=10-20 LABEL=rg
PRINT ARG=rg STRIDE=1 FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the group of atoms that you are calculating the Gyration Tensor for. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

TYPE	(default=RADIUS) The type of calculation relative to the Gyration Tensor you want to perform

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
MASS_WEIGHTED	(default=off) set the masses of all the atoms equal to one

5.2.24 PARABETARMSD

	This is part of the secondarystructure module

Probe the parallel beta sheet content of your protein structure.

Two protein segments containing three contiguous residues can form a parallel beta sheet. Although if the two segments are part of the same protein chain they must be separated by a minimum of 3 residues to make room for the turn. This colvar thus generates the set of all possible six residue sections that could conceivably form a parallel beta sheet and calculates the RMSD distance between the configuration in which the residues find themselves and an idealized parallel beta sheet structure. These distances can be calculated by either aligning the instantaneous structure with the reference structure and measuring each atomic displacement or by calculating differences between the set of inter-atomic distances in the reference and instantaneous structures.

This colvar is based on the following reference [6]. The authors of this paper use the set of distances from the parallel beta sheet configurations to measure the number of segments whose configuration resembles a parallel beta sheet. This is done by calculating the following sum of functions of the rmsd distances:

$$s = \sum_i \frac{1 - \left(\frac{r_i - d_0}{r_0}\right)^n}{1 - \left(\frac{r_i - d_0}{r_0}\right)^m}$$

where the sum runs over all possible segments of parallel beta sheet. By default the NN, MM and D_0 parameters are set equal to those used in [6]. The R_0 parameter must be set by the user - the value used in [6] was 0.08 nm.

If you change the function in the above sum you can calculate quantities such as the average distance from a structure composed of only parallel beta sheets or the distance between the set of residues that is closest to a parallel beta sheet and the reference configuration. To do these sorts of calculations you can use the AVERAGE and MIN keywords. In addition you can use the LESS_THAN keyword if you would like to change the form of the switching function. If you use any of these options you no longer need to specify NN, R_0, MM and D_0.

Please be aware that for codes like gromacs you must ensure that plumed reconstructs the chains involved in your CV when you calculate this CV using anything other than TYPE=DRMSD. For more details as to how to do this see WHOLEMOLECULES.

Examples

The following input calculates the number of six residue segments of protein that are in an parallel beta sheet configuration.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PARABETARMSD.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=beta.pdb
pb: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1
```

Here the same is done use RMSD instead of DRMSD

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PARABETARMSD.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES ENTITY0=1-100
hh: PARABETARMSD RESIDUES=all TYPE=OPTIMAL R_0=0.1  STRANDS_CUTOFF=1
```

Glossary of keywords and components

Description of components

By default this Action calculates the number of structural units that are within a certain distance of a idealized secondary structure element. This quantity can then be referenced elsewhere in the input by using the label of the action. However, this Action can also be used to calculate the following quantities by using the keywords as described below. The quantities then calculated can be referenced using the label of the action followed by a dot and then the name from the table below. Please note that you can use the LESS_THAN keyword more than once. The resulting components will be labelled *label*.lessthan-1, *label*.lessthan-2 and so on unless you exploit the fact that these labels can be given custom labels by using the LABEL keyword in the description of you LESS_THAN function that you are computing

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.

The atoms involved can be specified using

RESIDUES	this command is used to specify the set of residues that could conceivably form part of the secondary structure. It is possible to use residues numbers as the various chains and residues should have been identified else using an instance of the MOLINFO action. If you wish to use all the residues from all the chains in your system you can do so by specifying all. Alternatively, if you wish to use a subset of the residues you can specify the particular residues you are interested in as a list of numbers. Please be aware that to form secondary structure elements your chain must contain at least N residues, where N is dependent on the particular secondary structure you are interested in. As such if you define portions of the chain with fewer than N residues the code will crash.

Compulsory keywords

TYPE	(default=DRMSD) the manner in which RMSD alignment is performed. Should be OPTIMAL, SIMPLE or DRMSD. For more details on the OPTIMAL and SIMPLE methods see RMSD. For more details on the DRMSD method see DRMSD.
R_0	(default=0.08) The r_0 parameter of the switching function.
D_0	(default=0.0) The d_0 parameter of the switching function
NN	(default=8) The n parameter of the switching function
MM	(default=12) The m parameter of the switching function
STYLE	(default=all) Parallel beta sheets can either form in a single chain or from a pair of chains. If STYLE=all all chain configuration with the appropriate geometry are counted. If STYLE=inter only sheet-like configurations involving two chains are counted, while if STYLE=intra only sheet-like configurations involving a single chain are counted

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions
VERBOSE	(default=off) write a more detailed output
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...

ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
STRANDS_CUTOFF	If in a segment of protein the two strands are further apart then the calculation of the actual RMSD is skipped as the structure is very far from being beta-sheet like. This keyword speeds up the calculation enormously when you are using the LESS_THAN option. However, if you are using some other option, then this cannot be used

5.2.25 PATH

This is part of the mapping module

Path collective variables with a more flexible framework for the distance metric being used.

The Path Collective Variables developed by Branduardi and co-workers [5] allow one to compute the progress along a high-dimensional path and the distance from the high-dimensional path. The progress along the path (s) is computed using:

$$s = \frac{\sum_{i=1}^{N} i \exp(-\lambda R[X - X_i])}{\sum_{i=1}^{N} \exp(-\lambda R[X - X_i])}$$

while the distance from the path (z) is measured using:

$$z = -\frac{1}{\lambda} \ln \left[\sum_{i=1}^{N} \exp(-\lambda R[X - X_i]) \right]$$

In these expressions N high-dimensional frames (X_i) are used to describe the path in the high-dimensional space. The two expressions above are then functions of the distances from each of the high-dimensional frames $R[X-X_i]$. Within PLUMED there are multiple ways to define the distance from a high-dimensional configuration. You could calculate the RMSD distance or you could calculate the amount by which a set of collective variables change. As such this implementation of the path CV allows one to use all the difference distance metrics that are discussed in Distances from reference configurations. This is as opposed to the alternative implementation of path (PATHMSD) which is a bit faster but which only allows one to use the RMSD distance.

The s and z variables are calculated using the above formulas by default. However, there is an alternative method of calculating these collective variables, which is detailed in [15]. This alternative method uses the tools of geometry (as opposed to algebra, which is used in the equations above). In this alternative formula the progress along the path s is calculated using:

$$s = i_2 + \mathrm{sign}(i_2 - i_1)\frac{\sqrt{(\mathbf{v}_1\cdot\mathbf{v}_2)^2 - |\mathbf{v}_3|^2(|\mathbf{v}_1|^2 - |\mathbf{v}_2|^2)}}{2|\mathbf{v}_3|^2} - \frac{\mathbf{v}_1\cdot\mathbf{v}_3 - |\mathbf{v}_3|^2}{2|\mathbf{v}_3|^2}$$

where $\mathbf{v}_1$ and $\mathbf{v}_3$ are the vectors connecting the current position to the closest and second closest node of the path, respectfully and i_1 and i_2 are the projections of the closest and second closest frames of the path. $\mathbf{v}_2$, meanwhile, is the vector connecting the closest frame to the second closest frame. The distance from the path, z is calculated using:

$$z = \sqrt{\left[|\mathbf{v}_1|^2 - |\mathbf{v}_2| \left(\frac{\sqrt{(\mathbf{v}_1 \cdot \mathbf{v}_2)^2 - |\mathbf{v}_3|^2(|\mathbf{v}_1|^2 - |\mathbf{v}_2|^2)}}{2|\mathbf{v}_3|^2} - \frac{\mathbf{v}_1 \cdot \mathbf{v}_3 - |\mathbf{v}_3|^2}{2|\mathbf{v}_3|^2} \right) \right]^2}$$

The symbols here are as they were for s. If you would like to use these equations to calculate s and z then you should use the GPATH flag. The values of s and z can then be referenced using the gspath and gzpath labels.

Examples

In the example below the path is defined using RMSD distance from frames.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PATH.tmp
p1: PATH REFERENCE=file.pdb TYPE=OPTIMAL LAMBDA=500.0
PRINT ARG=p1.spath,p1.zpath STRIDE=1 FILE=colvar FMT=%8.4f
```

The reference frames in the path are defined in the pdb file shown below. In this frame each configuration in the path is separated by a line containing just the word END.

```
ATOM      1  CL  ALA     1      -3.171   0.295   2.045  1.00  1.00
ATOM      5  CLP ALA     1      -1.819  -0.143   1.679  1.00  1.00
ATOM      6  OL  ALA     1      -1.177  -0.889   2.401  1.00  1.00
ATOM      7  NL  ALA     1      -1.313   0.341   0.529  1.00  1.00
END
ATOM      1  CL  ALA     1      -3.175   0.365   2.024  1.00  1.00
ATOM      5  CLP ALA     1      -1.814  -0.106   1.685  1.00  1.00
ATOM      6  OL  ALA     1      -1.201  -0.849   2.425  1.00  1.00
ATOM      7  NL  ALA     1      -1.296   0.337   0.534  1.00  1.00
END
ATOM      1  CL  ALA     1      -2.990   0.383   2.277  1.00  1.00
ATOM      5  CLP ALA     1      -1.664  -0.085   1.831  1.00  1.00
ATOM      6  OL  ALA     1      -0.987  -0.835   2.533  1.00  1.00
ATOM      7  NL  ALA     1      -1.227   0.364   0.646  1.00  1.00
END
```

In the example below the path is defined using the values of two torsional angles (t1 and t2). In addition, the s and z are calculated using the geometric expressions described above rather than the algebraic expressions that are used by default.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PATH.tmp
t1: TORSION ATOMS=5,7,9,15
t2: TORSION ATOMS=7,9,15,17
pp: PATH TYPE=EUCLIDEAN REFERENCE=epath.pdb GPATH NOSPATH NOZPATH
PRINT ARG=pp.* FILE=colvar
```

Notice that the LAMBDA parameter is not required here as we are not calculating s and s using the algebraic formulas defined earlier. The positions of the frames in the path are defined in the file epath.pdb. An extract from this file looks as shown below.

```
REMARK ARG=t1,t2 t1=-4.25053  t2=3.88053
END
REMARK ARG=t1,t2 t1=-4.11     t2=3.75
END
REMARK ARG=t1,t2 t1=-3.96947  t2=3.61947
END
```

The remarks in this pdb file tell PLUMED the labels that are being used to define the position in the high dimensional space and the values that these arguments have at each point on the path.

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Keyword	Description
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry

Compulsory keywords

REFERENCE	a pdb file containing the set of reference configurations
TYPE	(default=OPTIMAL-FAST) the manner in which distances are calculated. More information on the different metrics that are available in PLUMED can be found in the section of the manual on Distances from reference configurations
LAMBDA	(default=0) the value of the lambda parameter for paths

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
DISABLE_CHECKS	(default=off) disable checks on reference input structures.
NOZPATH	(default=off) do not calculate the zpath position
NOSPATH	(default=off) do not calculate the spath position
GPATH	calculate the position on the path using trigonometry The final value can be referenced using *label*.gpath. You can use multiple instances of this keyword i.e. GPATH1, GPATH2, GPATH3... The corresponding values are then referenced using *label*.gpath-1, *label*.gpath-2, *label*.gpath-3...

5.2.26 PATHMSD

This is part of the colvar module

This Colvar calculates path collective variables.

This is the Path Collective Variables implementation (see [5]). This variable computes the progress along a given set of frames that is provided in input ("sss" component) and the distance from them ("zzz" component). (see below).

When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.5, by considering the ordered list of atoms and rebuilding molecules with a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Examples

Here below is a case where you have defined three frames and you want to calculate the progress along the path and the distance from it in p1

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PATHMSD.tmp
p1: PATHMSD REFERENCE=file.pdb  LAMBDA=500.0 NEIGH_STRIDE=4 NEIGH_SIZE=8
PRINT ARG=p1.sss,p1.zzz STRIDE=1 FILE=colvar FMT=%8.4f
```

note that NEIGH_STRIDE=4 NEIGH_SIZE=8 control the neighbor list parameter (optional but recommended for performance) and states that the neighbor list will be calculated every 4 steps and consider only the closest 8 member to the actual md snapshots.

This input must be accompanied by a REFERENCE PDB file in which the positions of each of the frames are specified separated using either END or ENDMDL as shown below:

```
ATOM      1  CL  ALA     1      -3.171   0.295   2.045  1.00  1.00
ATOM      5  CLP ALA     1      -1.819  -0.143   1.679  1.00  1.00
ATOM      6  OL  ALA     1      -1.177  -0.889   2.401  1.00  1.00
ATOM      7  NL  ALA     1      -1.313   0.341   0.529  1.00  1.00
END
ATOM      1  CL  ALA     1      -3.175   0.365   2.024  1.00  1.00
ATOM      5  CLP ALA     1      -1.814  -0.106   1.685  1.00  1.00
ATOM      6  OL  ALA     1      -1.201  -0.849   2.425  1.00  1.00
ATOM      7  NL  ALA     1      -1.296   0.337   0.534  1.00  1.00
END
ATOM      1  CL  ALA     1      -2.990   0.383   2.277  1.00  1.00
ATOM      5  CLP ALA     1      -1.664  -0.085   1.831  1.00  1.00
ATOM      6  OL  ALA     1      -0.987  -0.835   2.533  1.00  1.00
ATOM      7  NL  ALA     1      -1.227   0.364   0.646  1.00  1.00
END
```

Note

The implementation of this collective variable and of PROPERTYMAP is shared, as well as most input options.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
sss	the position on the path
zzz	the distance from the path

Compulsory keywords

LAMBDA	the lambda parameter is needed for smoothing, is in the units of plumed
REFERENCE	the pdb is needed to provide the various milestones

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
NEIGH_SIZE	size of the neighbor list
NEIGH_STRIDE	how often the neighbor list needs to be calculated in time units
EPSILON	(default=-1) the maximum distance between the close and the current structure, the positive value turn on the close structure method
LOG_CLOSE	(default=0) value 1 enables logging regarding the close structure
DEBUG_CLOSE	(default=0) value 1 enables extensive debugging info regarding the close structure, the simulation will run much slower

5.2.27 PCAVARS

This is part of the mapping module

Projection on principal component eigenvectors or other high dimensional linear subspace

The collective variables described in Distances from reference configurations allow one to calculate the distance between the instantaneous structure adopted by the system and some high-dimensional, reference configuration. The problem with doing this is that, as one gets further and further from the reference configuration, the distance from it becomes a progressively poorer and poorer collective variable. This happens because the ``number" of structures at a distance d from a reference configuration is proportional to d^N in an N dimensional space. Consequently, when d is small the distance from the reference configuration may well be a good collective variable. However, when d is large it is unlikely that the distance from the reference structure is a good CV. When the distance is large there will almost certainly be markedly different configuration that have the same CV value and hence barriers in transverse degrees of freedom.

For these reasons dimensionality reduction is often employed so a projection $\mathbf{s}$ of a high-dimensional configuration $\mathbf{X}$ in a lower dimensionality space using a function:

$$\mathbf{s} = F(\mathbf{X} - \mathbf{X}^{ref})$$

where here we have introduced some high-dimensional reference configuration $\mathbf{X}^{ref}$. By far the simplest way to do this is to use some linear operator for F. That is to say we find a low-dimensional projection by rotating the basis vectors using some linear algebra:

$$\mathbf{s}_i = \sum_k A_{ik}(X_k - X_k^{ref})$$

Here A is a d by D matrix where D is the dimensionality of the high dimensional space and d is the dimensionality of the lower dimensional subspace. In plumed when this kind of projection you can use the majority of the metrics detailed on Distances from reference configurations to calculate the displacement, $\mathbf{X} - \mathbf{X}^{ref}$, from the reference configuration. The matrix A can be found by various means including principal component analysis and normal mode analysis. In both these methods the rows of A would be the principle eigenvectors of a square matrix. For PCA the covariance while for normal modes the Hessian.

Bug It is not possible to use the DRMSD metric with this variable. You can get around this by listing the set of distances you wish to calculate for your DRMSD in the plumed file explicitly and using the EUCLIDEAN metric. MAHALONOBIS and NORM-EUCLIDEAN also do not work with this variable but using these options makes little sense when projecting on a linear subspace.

Examples

The following input calculates a projection on a linear subspace where the displacements from the reference configuration are calculated using the OPTIMAL metric. Consequently, both translation of the center of mass of the atoms and rotation of the reference frame are removed from these displacements. The matrix A and the reference configuration R^{ref} are specified in the pdb input file reference.pdb and the value of all projections (and the residual) are output to a file called colvar2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PCAVARS.tmp
PCAVARS REFERENCE=reference.pdb TYPE=OPTIMAL LABEL=pca2
PRINT ARG=pca2.* FILE=colvar2
```

The reference configurations can be specified using a pdb file. The first configuration that you provide is the reference configuration, which is referred to in the above as X^{ref} subsequent configurations give the directions of row vectors that are contained in the matrix A above. These directions can be specified by specifying a second configuration - in this case a vector will be constructed by calculating the displacement of this second configuration from the reference configuration. A pdb input prepared in this way would look as follows:

```
REMARK TYPE=OPTIMAL
ATOM      2  CH3 ACE     1      12.932 -14.718  -6.016  1.00  1.00
ATOM      5  C   ACE     1      21.312  -9.928  -5.946  1.00  1.00
ATOM      9  CA  ALA     2      19.462 -11.088  -8.986  1.00  1.00
ATOM     13  HB2 ALA     2      21.112 -10.688 -12.476  1.00  1.00
ATOM     15  C   ALA     2      19.422   7.978 -14.536  1.00  1.00
ATOM     20 HH31 NME     3      20.122  -9.928 -17.746  1.00  1.00
ATOM     21 HH32 NME     3      18.572 -13.148 -16.346  1.00  1.00
END
REMARK TYPE=OPTIMAL
ATOM      2  CH3 ACE     1      13.932 -14.718  -6.016  1.00  1.00
ATOM      5  C   ACE     1      20.312  -9.928  -5.946  1.00  1.00
ATOM      9  CA  ALA     2      18.462 -11.088  -8.986  1.00  1.00
ATOM     13  HB2 ALA     2      20.112 -11.688 -12.476  1.00  1.00
ATOM     15  C   ALA     2      19.422   7.978 -12.536  1.00  1.00
ATOM     20 HH31 NME     3      20.122  -9.928 -17.746  1.00  1.00
ATOM     21 HH32 NME     3      18.572 -13.148 -16.346  1.00  1.00
END
```

Alternatively, the second configuration can specify the components of A explicitly. In this case you need to include the keyword TYPE=DIRECTION in the remarks to the pdb as shown below.

```
ATOM      2  CH3 ACE     1      12.932 -14.718  -6.016  1.00  1.00
ATOM      5  C   ACE     1      21.312  -9.928  -5.946  1.00  1.00
ATOM      9  CA  ALA     2      19.462 -11.088  -8.986  1.00  1.00
ATOM     13  HB2 ALA     2      21.112 -10.688 -12.476  1.00  1.00
ATOM     15  C   ALA     2      19.422   7.978 -14.536  1.00  1.00
ATOM     20 HH31 NME     3      20.122  -9.928 -17.746  1.00  1.00
ATOM     21 HH32 NME     3      18.572 -13.148 -16.346  1.00  1.00
END
REMARK TYPE=DIRECTION
ATOM      2  CH3 ACE     1       0.1414  0.3334 -0.0302  1.00  0.00
ATOM      5  C   ACE     1       0.0893 -0.1095 -0.1434  1.00  0.00
ATOM      9  CA  ALA     2       0.0207 -0.321   0.0321  1.00  0.00
ATOM     13  HB2 ALA     2       0.0317 -0.6085  0.0783  1.00  0.00
ATOM     15  C   ALA     2       0.1282 -0.4792  0.0797  1.00  0.00
ATOM     20 HH31 NME     3       0.0053 -0.465   0.0309  1.00  0.00
ATOM     21 HH32 NME     3      -0.1019 -0.4261 -0.0082  1.00  0.00
END
```

If your metric involves arguments the labels of these arguments in your plumed input file should be specified in the REMARKS for each of the frames of your path. An input file in this case might look like this:

```
DESCRIPTION: a pca eigenvector specified using the start point and direction in the HD space.
REMARK WEIGHT=1.0
REMARK ARG=d1,d2
REMARK d1=1.0 d2=1.0
END
REMARK TYPE=DIRECTION
REMARK ARG=d1,d2
REMARK d1=0.1 d2=0.25
END
```

Here we are working with the EUCLIDEAN metric and notice that we have specified the components of A using D↩ IRECTION. Consequently, the values of d1 and d2 in the second frame above do not specify a particular coordinate in the high-dimensional space as in they do in the first frame. Instead these values are the coefficients that can be used to construct a linear combination of d1 and d2. If we wanted to specify the direction in this metric using the start and end point of the vector we would write:

```
DESCRIPTION: a pca eigenvector specified using the start and end point of a vector in the HD space.
REMARK WEIGHT=1.0
REMARK ARG=d1,d2
REMARK d1=1.0 d2=1.0
END
REMARK ARG=d1,d2
REMARK d1=1.1 d2=1.25
END
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
eig	the projections on each eigenvalue are stored on values labeled eig-1, eig-2, ...
residual	the distance of the configuration from the linear subspace defined by the vectors, eig-1, eig2, ... that are contained in the rows of A.

Compulsory keywords

REFERENCE	a pdb file containing the reference configuration and configurations that define the directions for each eigenvector
TYPE	(default=OPTIMAL) The method we are using for alignment to the reference structure

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

5.2.28 POSITION

This is part of the colvar module

Calculate the components of the position of an atom.

Notice that single components will not have the proper periodicity! If you need the values to be consistent through PBC you should use SCALED_COMPONENTS, which defines values that by construction are in the -0.5,0.↩ 5 domain. This is similar to the equivalent flag for DISTANCE. Also notice that by default the minimal image distance from the origin is considered (can be changed with NOPBC).

Attention

This variable should be used with extreme care since it allows to easily go into troubles. See comments below.

This variable can be safely used only if Hamiltonian is not invariant for translation (i.e. there are other absolute positions which are biased, e.g. by position restraints) and cell size and shapes are fixed through the simulation.

If you are not in this situation and still want to use the absolute position of an atom you should first fix the reference frame. This can be done e.g. using FIT_TO_TEMPLATE.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/POSITION.tmp
# align to a template
FIT_TO_TEMPLATE REFERENCE=ref.pdb
p: POSITION ATOM=3
PRINT ARG=p.x,p.y,p.z
```

The reference position is specified in a pdb file like the one shown below

```
ATOM      3  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
END
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
x	the x-component of the atom position
y	the y-component of the atom position
z	the z-component of the atom position

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
a	**SCALED_COMPONENTS**	the normalized projection on the first lattice vector of the atom position
b	**SCALED_COMPONENTS**	the normalized projection on the second lattice vector of the atom position
c	**SCALED_COMPONENTS**	the normalized projection on the third lattice vector of the atom position

The atoms involved can be specified using

ATOM	the atom number. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SCALED_COMPONENTS	(default=off) calculate the a, b and c scaled components of the position separately and store them as label.a, label.b and label.c

5.2.29 PROJECTION_ON_AXIS

This is part of the colvar module

Calculate a position based on the projection along and extension from a defined axis.

This variable takes 3 input atoms or pseudoatoms, using the two AXIS_ATOMS to define a linear vector. The position of the ATOM is then calculated relative to this vector, with two output components. The projection on the axis (proj) is the distance along the axis from the ATOM to the origin. The extension (ext) is the orthogonal distance between the ATOM and the axis.

Examples

This command tells plumed to define an axis, by calculating a vector that passes through atom 1 and atom 2. The position of atom 3 as a projection along this vector is calculated and printed to COLVAR1. At the same time, the perpendicular distance of atom 3 from the axis, the extension, is printed to COLVAR2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PROJECTION_ON_AXIS.tmp
poa: PROJECTION_ON_AXIS AXIS_ATOMS=1,2 ATOM=3
PRINT ARG=poa.proj FILE=COLVAR1
PRINT ARG=poa.ext FILE=COLVAR2
```

A particular application of this variable could be to study the motion of a ligand relative to its binding pocket on a protein. In this set of commands, the anchor points a1 and a2 are defined using example atom numbers within the protein. As a2 is attempting to be as close as possible to the center of the binding pocket, a COM is used when there are no suitable protein atoms. Similarly, a COM is used to define the position of the ligand in lig1. The calculated projection of lig1 along the axis defined between a1 and a2 is printed to COLVAR1. The calculated perpendicular extension of lig1 from the axis defined between a1 and a2 is printed to COLVAR2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PROJECTION_ON_AXIS.tmp
a1: GROUP ATOMS=3754            # Anchor point 1
a2: COM ATOMS=3019,4329,4744    # Anchor point 2
lig1: COM ATOMS=5147-5190       # Ligand
pp: PROJECTION_ON_AXIS AXIS_ATOMS=a1,a2 ATOM=lig1
PRINT ARG=pp.proj FILE=COLVAR1
PRINT ARG=pp.ext FILE=COLVAR2
```

Glossary of keywords and components

The atoms involved can be specified using

AXIS_ATOMS	The atoms that define the direction of the axis of interest. For more information on how to specify lists of atoms see Groups and Virtual Atoms
ATOM	The atom whose position we want to project on the axis of interest. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances

5.2.30 PROPERTYMAP

This is part of the colvar module

Calculate generic property maps.

This Colvar calculates the property maps according to the work of Spiwok [13].

Basically it calculates

$$X = \frac{\sum_i X_i * \exp(-\lambda D_i(x))}{\sum_i \exp(-\lambda D_i(x))}$$
$$Y = \frac{\sum_i Y_i * \exp(-\lambda D_i(x))}{\sum_i \exp(-\lambda D_i(x))}$$
$$\dots$$
$$zzz = -\frac{1}{\lambda}\log(\sum_i \exp(-\lambda D_i(x)))$$

where the parameters X_i and Y_i are provided in the input pdb (allv.pdb in this case) and $D_i(x)$ is the mean squared displacement after optimal alignment calculated on the pdb frames you input (see Kearsley).

When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.5, by considering the ordered list of atoms and rebuilding molecules using a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PROPERTYMAP.tmp
p3: PROPERTYMAP REFERENCE=allv.pdb PROPERTY=X,Y LAMBDA=69087 NEIGH_SIZE=8 NEIGH_STRIDE=4
PRINT ARG=p3.X,p3.Y,p3.zzz STRIDE=1 FILE=colvar FMT=%8.4f
```

note that NEIGH_STRIDE=4 NEIGH_SIZE=8 control the neighbor list parameter (optional but recommended for performance) and states that the neighbor list will be calculated every 4 steps and consider only the closest 8 member to the actual md snapshots.

In this case the input line instructs plumed to look for two properties X and Y with attached values in the REMARK line of the reference pdb (Note: No spaces from X and = and 1 !!!!). e.g.

```
REMARK X=1 Y=2
ATOM      1  CL  ALA     1      -3.171   0.295   2.045  1.00  1.00
ATOM      5  CLP ALA     1      -1.819  -0.143   1.679  1.00  1.00
END
REMARK X=2 Y=3
ATOM      1  CL  ALA     1      -3.175   0.365   2.024  1.00  1.00
ATOM      5  CLP ALA     1      -1.814  -0.106   1.685  1.00  1.00
END
```

Note

The implementation of this collective variable and of PATHMSD is shared, as well as most input options.

Glossary of keywords and components

Description of components

The names of the components in this action can be customized by the user in the actions input file. However, in addition to the components that can be customized the following quantities will always be output

Quantity	Description
zzz	the minimum distance from the reference points

Compulsory keywords

LAMBDA	the lambda parameter is needed for smoothing, is in the units of plumed
REFERENCE	the pdb is needed to provide the various milestones
PROPERTY	the property to be used in the indexing: this goes in the REMARK field of the reference

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
NEIGH_SIZE	size of the neighbor list
NEIGH_STRIDE	how often the neighbor list needs to be calculated in time units
EPSILON	(default=-1) the maximum distance between the close and the current structure, the positive value turn on the close structure method
LOG_CLOSE	(default=0) value 1 enables logging regarding the close structure
DEBUG_CLOSE	(default=0) value 1 enables extensive debugging info regarding the close structure, the simulation will run much slower

5.2.31 PUCKERING

This is part of the colvar module

Calculate sugar pseudorotation coordinates.

This command can be used to calculate ring's pseudorotations in sugars (puckers). It works for both 5-membered and 6-membered rings. Notice that there are two different implementations depending if one passes 5 or 6 atoms in the ATOMS keyword.

For 5-membered rings the implementation is the one discussed in [16] . This implementation is simple and can be used in RNA to distinguish C2'-endo and C3'-endo conformations. Both the polar coordinates (phs and amp) and the Cartesian coordinates (Zx and Zy) are provided. C2'-endo conformations have negative Zx, whereas C3'-endo conformations have positive Zy. Notation is consistent with [16] . The five atoms should be provided as C4',O4',C1',C2',C3'. Notice that this is the same order that can be obtained using the MOLINFO syntax (see example below).

For 6-membered rings the implementation is the general Cremer-Pople one [17] as also discussed in [18] . This implementation provides both a triplet with Cartesian components (qx, qy, and qz) and a triplet of polar components (amplitude, phi, and theta). Applications of this particular implementation are to be published (paper in preparation).

Note

The 6-membered ring implementation distributed with previous versions of PLUMED lead to qx and qy values that had an opposite sign with respect to those originally defined in [17]. The bug is fixed in version 2.5.

Components of this action are:

Examples

This input tells plumed to print the puckering phase angle of the second nucleotide of a RNA molecule on file COLVAR.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PUCKERING.tmp
 #SETTINGS MOLFILE=regtest/basic/rt65/AA.pdb
 MOLINFO STRUCTURE=rna.pdb MOLTYPE=rna
 PUCKERING ATOMS=@sugar-2 LABEL=puck
 PRINT ARG=puck.phs FILE=COLVAR
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
phs	Pseudorotation phase (5 membered rings)
amp	Pseudorotation amplitude (5 membered rings)
Zx	Pseudorotation x Cartesian component (5 membered rings)
Zy	Pseudorotation y Cartesian component (5 membered rings)
phi	Pseudorotation phase (6 membered rings)
theta	Theta angle (6 membered rings)
amplitude	Pseudorotation amplitude (6 membered rings)
qx	Cartesian component x (6 membered rings)
qy	Cartesian component y (6 membered rings)
qz	Cartesian component z (6 membered rings)

The atoms involved can be specified using

ATOMS	the five or six atoms of the sugar ring in the proper order. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

5.2.32 TEMPLATE

	This is part of the colvar module

This file provides a template for if you want to introduce a new CV.

5.2.33 TORSION

This is part of the colvar module

Calculate a torsional angle.

This command can be used to compute the torsion between four atoms or alternatively to calculate the angle between two vectors projected on the plane orthogonal to an axis.

Examples

This input tells plumed to print the torsional angle between atoms 1, 2, 3 and 4 on file COLVAR.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TORSION.tmp
t: TORSION ATOMS=1,2,3,4
# this is an alternative, equivalent, definition:
# t: TORSION VECTOR1=2,1 AXIS=2,3 VECTOR2=3,4
PRINT ARG=t FILE=COLVAR
```

If you are working with a protein you can specify the special named torsion angles ϕ, ψ, ω and χ_1 by using TORSION in combination with the MOLINFO command. This can be done by using the following syntax.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TORSION.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
t1: TORSION ATOMS=@phi-3
t2: TORSION ATOMS=@psi-4
PRINT ARG=t1,t2 FILE=colvar STRIDE=10
```

Here, @phi-3 tells plumed that you would like to calculate the ϕ angle in the third residue of the protein. Similarly @psi-4 tells plumed that you want to calculate the ψ angle of the fourth residue of the protein.

Both of the previous examples specify that the torsion angle should be calculated based on the position of four atoms. For the first example in particular the assumption when the torsion is specified in this way is that there are chemical bonds between atoms 1 and 2, atoms 2 and 3 and atoms 3 and 4. In general, however, a torsional angle measures the angle between two planes, which have at least one vector in common. As shown below, there is thus an alternate, more general, way through which we can define a torsional angle:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TORSION.tmp
t1: TORSION VECTOR1=1,2 AXIS=3,4 VECTOR2=5,6
PRINT ARG=t1 FILE=colvar STRIDE=20
```

This input instructs PLUMED to calculate the angle between the plane containing the vector connecting atoms 1 and 2 and the vector connecting atoms 3 and 4 and the plane containing this second vector and the vector connecting atoms 5 and 6. We can even use PLUMED to calculate the torsional angle between two bond vectors around the z-axis as shown below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TORSION.tmp
a0: FIXEDATOM AT=0,0,0
az: FIXEDATOM AT=0,0,1
t1: TORSION VECTOR1=1,2 AXIS=a0,az VECTOR2=5,6
PRINT ARG=t1 FILE=colvar STRIDE=20
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the four atoms involved in the torsional angle

Or alternatively by using

AXIS	two atoms that define an axis. You can use this to find the angle in the plane perpendicular to the axis between the vectors specified using the VECTOR1 and VECTOR2 keywords.
VECTOR1	two atoms that define a vector. You can use this in combination with VECTOR2 and AXIS
VECTOR2	two atoms that define a vector. You can use this in combination with VECTOR1 and AXIS

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
COSINE	(default=off) calculate cosine instead of dihedral

5.2.34 VOLUME

	This is part of the colvar module

Calculate the volume of the simulation box.

Examples

The following input tells plumed to print the volume of the system

```
BEGIN_PLUMED_FILE working DATADIR=example-check/VOLUME.tmp
vol: VOLUME
PRINT ARG=vol
```

Glossary of keywords and components

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

5.3 Distances from reference configurations

One colvar that has been shown to be very successful in studying protein folding is the distance between the instantaneous configuration and a reference configuration - often the structure of the folded state. When the free energy of a protein is shown as a function of this collective variable there is a minima for low values of the CV, which is due to the folded state of the protein. There is then a second minima at higher values of the CV, which is the minima corresponding to the unfolded state.

A slight problem with this sort of collective variable is that there are many different ways of calculating the distance from a particular reference structure. The simplest - adding together the distances by which each of the atoms has been translated in going from the reference configuration to the instantaneous configuration - is not particularly sensible. A distance calculated in this way does not neglect translation of the center of mass of the molecule and rotation of the frame of reference. A common practice is thus to remove these components by calculating the RMSD distance between the reference and instantaneous configurations. This is not the only way to calculate the distance, however. One could also calculate the total amount by which a large number of collective variables change in moving from the reference to the instantaneous configurations. One could even combine RMSD distances with the amount the collective variables change. A full list of the ways distances can be measured in PLUMED is given below:

DRMSD	Calculate the distance RMSD with respect to a reference structure.
MULTI_RMSD	Calculate the RMSD distance moved by a number of separated domains from their positions in a reference structure.
PCARMSD	Calculate the PCA components for a number of provided eigenvectors and an average structure.
RMSD	Calculate the RMSD with respect to a reference structure.
TARGET	This function measures the Pythagorean distance from a particular structure measured in the space defined by some set of collective variables.

These options for calculating distances are re-used in a number of places in the code. For instance they are used in some of the analysis algorithms that are implemented in PLUMED and in PATH collective variables. Notice that most of these actions read the reference configuration from a PDB file. Be sure you understand how to format properly a PDB file to use used in PLUMED (see pdbreader).

5.3.1 DRMSD

This is part of the colvar module

Calculate the distance RMSD with respect to a reference structure.

To calculate the root-mean-square deviation between the atoms in two configurations you must first superimpose the two structures in some ways. Obviously, it is the internal vibrational motions of the structure - i.e. not the translations and rotations - that are interesting. However, aligning two structures by removing the translational and rotational motions is not easy. Furthermore, in some cases there can be alignment issues caused by so-called frame-fitting problems. It is thus often cheaper and easier to calculate the distances between all the pairs of atoms. The distance between the two structures, $\mathbf{X}^a$ and $\mathbf{X}^b$ can then be measured as:

$$d(\mathbf{X}^A,\mathbf{X}^B) = \sqrt{\frac{1}{N(N-1)}\sum_{i\neq j}[d(\mathbf{x}_i^a,\mathbf{x}_j^a) - d(\mathbf{x}_i^b,\mathbf{x}_j^b)]^2}$$

where N is the number of atoms and $d(\mathbf{x}_i,\mathbf{x}_j)$ represents the distance between atoms i and j. Clearly, this representation of the configuration is invariant to translation and rotation. However, it can become expensive to

calculate when the number of atoms is large. This can be resolved within the DRMSD colvar by setting LOWER↩ _CUTOFF and UPPER_CUTOFF. These keywords ensure that only pairs of atoms that are within a certain range are incorporated into the above sum.

In PDB files the atomic coordinates and box lengths should be in Angstroms unless you are working with natural units. If you are working with natural units then the coordinates should be in your natural length unit. For more details on the PDB file format visit http://www.wwpdb.org/docs.html

Examples

The following tells plumed to calculate the distance RMSD between the positions of the atoms in the reference file and their instantaneous position. Only pairs of atoms whose distance in the reference structure is within 0.1 and 0.8 nm are considered.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DRMSD.tmp
DRMSD REFERENCE=file1.pdb LOWER_CUTOFF=0.1 UPPER_CUTOFF=0.8
```

The reference file is a PDB file that looks like this

```
ATOM      8  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
END
```

The following tells plumed to calculate a DRMSD value for a pair of molecules.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DRMSD.tmp
DRMSD REFERENCE=file2.pdb LOWER_CUTOFF=0.1 UPPER_CUTOFF=0.8 TYPE=INTER-DRMSD
```

In the input reference file (file.pdb) the atoms in each of the two molecules are separated by a TER command as shown below.

```
ATOM      8  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
TER
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
END
```

In this example the INTER-DRMSD type ensures that the set of distances from which the final quantity is computed involve one atom from each of the two molecules. If this is replaced by INTRA-DRMSD then only those distances involving pairs of atoms that are both in the same molecule are computed.

Glossary of keywords and components

Compulsory keywords

REFERENCE	a file in pdb format containing the reference structure and the atoms involved in the CV.
LOWER_CUTOFF	only pairs of atoms further than LOWER_CUTOFF are considered in the calculation.
UPPER_CUTOFF	only pairs of atoms closer than UPPER_CUTOFF are considered in the calculation.
TYPE	(default=DRMSD) what kind of DRMSD would you like to calculate. You can use either the normal DRMSD involving all the distances between the atoms in your molecule. Alternatively, if you have multiple molecules you can use the type INTER-DRMSD to compute DRMSD values involving only those distances between the atoms at least two molecules or the type INTRA-DRMSD to compute DRMSD values involving only those distances between atoms in the same molecule

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances

5.3.2 MULTI_RMSD

This is part of the colvar module

Calculate the RMSD distance moved by a number of separated domains from their positions in a reference structure.

When you have large proteins the calculation of the root mean squared deviation between all the atoms in a reference structure and the instantaneous configuration becomes prohibitively expensive. You may thus instead want to calculate the RMSD between the atoms in a set of domains of your protein and your reference structure. That is to say:

$$d(X, X_r) = \sqrt{\sum_i w_i |X_i - X_i'|^2}$$

where here the sum is over the domains of the protein, X_i represents the positions of the atoms in domain i in the instantaneous configuration and X_i' is the positions of the atoms in domain i in the reference configuration. w_i is an optional weight.

The distances for each of the domains in the above sum can be calculated using the DRMSD or RMSD measures or using a combination of these distance. The reference configuration is specified in a pdb file like the one below:

```
ATOM      2  O   ALA     2      -0.926  -2.447  -0.497  1.00  1.00      DIA  O
ATOM      4  HNT ALA     2       0.533  -0.396   1.184  1.00  1.00      DIA  H
ATOM      6  HT1 ALA     2      -0.216  -2.590   1.371  1.00  1.00      DIA  H
ATOM      7  HT2 ALA     2      -0.309  -1.255   2.315  1.00  1.00      DIA  H
ATOM      8  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
TER
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
ATOM     16  HN  ALA     2       1.713   1.021  -0.873  1.00  1.00      DIA  H
ATOM     18  HA  ALA     2       0.099  -0.774  -2.218  1.00  1.00      DIA  H
ATOM     19  CB  ALA     2       2.063  -1.223  -1.276  1.00  1.00      DIA  C
ATOM     20  HB1 ALA     2       2.670  -0.716  -2.057  1.00  1.00      DIA  H
ATOM     21  HB2 ALA     2       2.556  -1.051  -0.295  1.00  1.00      DIA  H
ATOM     22  HB3 ALA     2       2.070  -2.314  -1.490  1.00  1.00      DIA  H
END
```

with the TER keyword being used to separate the various domains in you protein.

Examples

The following tells plumed to calculate the RMSD distance between the positions of the atoms in the reference file and their instantaneous position. The Kearsley algorithm for each of the domains.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MULTI_RMSD.tmp
MULTI_RMSD REFERENCE=file1.pdb TYPE=MULTI-OPTIMAL
```

The following tells plumed to calculate the RMSD distance between the positions of the atoms in the domains of reference the reference structure and their instantaneous positions. Here distances are calculated using the DRMSD measure.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MULTI_RMSD.tmp
MULTI_RMSD REFERENCE=file1.pdb TYPE=MULTI-DRMSD
```

in this case it is possible to use the following DRMSD options in the pdb file using the REMARK syntax:

```
NOPBC to calculate distances without PBC
LOWER_CUTOFF=# only pairs of atoms further than LOWER_CUTOFF are considered in the calculation
UPPER_CUTOFF=# only pairs of atoms further than UPPER_CUTOFF are considered in the calculation
```

as shown in the following example

```
REMARK NOPBC
REMARK LOWER_CUTOFF=0.1
REMARK UPPER_CUTOFF=0.8
ATOM      2  O   ALA     2      -0.926  -2.447  -0.497  1.00  1.00      DIA  O
ATOM      4  HNT ALA     2       0.533  -0.396   1.184  1.00  1.00      DIA  H
ATOM      6  HT1 ALA     2      -0.216  -2.590   1.371  1.00  1.00      DIA  H
ATOM      7  HT2 ALA     2      -0.309  -1.255   2.315  1.00  1.00      DIA  H
ATOM      8  HT3 ALA     2      -1.480  -1.560   1.212  1.00  1.00      DIA  H
ATOM      9  CAY ALA     2      -0.096   2.144  -0.669  1.00  1.00      DIA  C
ATOM     10  HY1 ALA     2       0.871   2.385  -0.588  1.00  1.00      DIA  H
TER
ATOM     12  HY3 ALA     2      -0.520   2.679  -1.400  1.00  1.00      DIA  H
ATOM     14  OY  ALA     2      -1.139   0.931  -0.973  1.00  1.00      DIA  O
ATOM     16  HN  ALA     2       1.713   1.021  -0.873  1.00  1.00      DIA  H
ATOM     18  HA  ALA     2       0.099  -0.774  -2.218  1.00  1.00      DIA  H
ATOM     19  CB  ALA     2       2.063  -1.223  -1.276  1.00  1.00      DIA  C
ATOM     20  HB1 ALA     2       2.670  -0.716  -2.057  1.00  1.00      DIA  H
ATOM     21  HB2 ALA     2       2.556  -1.051  -0.295  1.00  1.00      DIA  H
ATOM     22  HB3 ALA     2       2.070  -2.314  -1.490  1.00  1.00      DIA  H
END
```

Glossary of keywords and components

Compulsory keywords

REFERENCE	a file in pdb format containing the reference structure and the atoms involved in the CV.
TYPE	(default=MULTI-SIMPLE) the manner in which RMSD alignment is performed. Should be MULTI-OPTIMAL, MULTI-OPTIMAL-FAST, MULTI-SIMPLE or MULTI-DRMSD.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SQUARED	(default=off) This should be set if you want the mean squared displacement instead of the root mean squared displacement

5.3.3 PCARMSD

This is part of the colvar module

Calculate the PCA components for a number of provided eigenvectors and an average structure.

For information on this method (see [19] and [20]). Performs optimal alignment at every step and reports the rmsd so you know if you are far or close from the average structure. It takes the average structure and eigenvectors in form of a pdb. Note that beta and occupancy values in the pdb are neglected and all the weights are placed to 1 (differently from the RMSD colvar for example)

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PCARMSD.tmp
PCARMSD AVERAGE=file.pdb EIGENVECTORS=eigenvectors.pdb
```

The input is taken so to be compatible with the output you get from g_covar utility of gromacs (suitably adapted to have a pdb input format). The reference configuration (file.pdb) will thus be in a file that looks something like this:

```
TITLE     Average structure
MODEL        1
ATOM      1  CL  ALA     1       1.042  -3.070   0.946  1.00  0.00
ATOM      5  CLP ALA     1       0.416  -2.033   0.132  1.00  0.00
ATOM      6  OL  ALA     1       0.415  -2.082  -0.976  1.00  0.00
ATOM      7  NL  ALA     1      -0.134  -1.045   0.677  1.00  0.00
ATOM      9  CA  ALA     1      -0.774   0.053   0.003  1.00  0.00
TER
ENDMDL
```

while the eigenvectors will be in a pdb file (eigenvectors.pdb) that looks something like this:

```
TITLE     frame t= -1.000
MODEL        1
ATOM      1  CL  ALA     1       1.194  -2.988   0.724  1.00  0.00
ATOM      5  CLP ALA     1      -0.996   0.042   0.144  1.00  0.00
ATOM      6  OL  ALA     1      -1.246  -0.178  -0.886  1.00  0.00
ATOM      7  NL  ALA     1      -2.296   0.272   0.934  1.00  0.00
ATOM      9  CA  ALA     1      -0.436   2.292   0.814  1.00  0.00
TER
ENDMDL
TITLE     frame t= 0.000
MODEL        1
ATOM      1  CL  ALA     1       1.042  -3.070   0.946  1.00  0.00
ATOM      5  CLP ALA     1      -0.774   0.053   0.003  1.00  0.00
ATOM      6  OL  ALA     1      -0.849  -0.166  -1.034  1.00  0.00
ATOM      7  NL  ALA     1      -2.176   0.260   0.563  1.00  0.00
ATOM      9  CA  ALA     1       0.314   1.825   0.962  1.00  0.00
TER
ENDMDL
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
eig	the projections on each eigenvalue are stored on values labeled eig-1, eig-2, ...
residual	the distance of the present configuration from the configuration supplied as AVERAGE in terms of mean squared displacement after optimal alignment

Compulsory keywords

AVERAGE	a file in pdb format containing the reference structure and the atoms involved in the CV.
EIGENVECTORS	a file in pdb format containing the reference structure and the atoms involved in the CV.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SQUARED_ROOT	(default=off) This should be set if you want RMSD instead of mean squared displacement

5.3.4 RMSD

This is part of the colvar module

Calculate the RMSD with respect to a reference structure.

The aim with this colvar it to calculate something like:

$$d(X, X') = |X - X'|$$

where X is the instantaneous position of all the atoms in the system and X' is the positions of the atoms in some reference structure provided as input. $d(X, X')$ thus measures the distance all the atoms have moved away from this reference configuration. Oftentimes, it is only the internal motions of the structure - i.e. not the translations of the center of mass or the rotations of the reference frame - that are interesting. Hence, when calculating the the root-mean-square deviation between the atoms in two configurations you must first superimpose the two structures in some way. At present PLUMED provides two distinct ways of performing this superposition. The first method is applied when you use TYPE=SIMPLE in the input line. This instruction tells PLUMED that the root mean square deviation is to be calculated after the positions of the geometric centers in the reference and instantaneous configurations are aligned. In other words $d(X, x')$ is to be calculated using:

$$d(X, X') = \sqrt{\sum_i \sum_\alpha^{x,y,z} \frac{w_i}{\sum_j w_j} (X_{i,\alpha} - com_\alpha(X) - X'_{i,\alpha} + com_\alpha(X'))^2}$$

with

$$com_\alpha(X) = \sum_i \frac{w'_i}{\sum_j w'_j} X_{i,\alpha}$$

and

$$com_\alpha(X') = \sum_i \frac{w'_i}{\sum_j w'_j} X'_{i,\alpha}$$

Obviously, $com_\alpha(X)$ and $com_\alpha(X')$ represent the positions of the center of mass in the reference and instantaneous configurations if the weights w' are set equal to the atomic masses. If the weights are all set equal to one, however, $com_\alpha(X)$ and $com_\alpha(X')$ are the positions of the geometric centers. Notice that there are sets of weights: w' and w. The first is used to calculate the position of the center of mass (so it determines how the atoms are *aligned*). Meanwhile, the second is used when calculating how far the atoms have actually been *displaced*. These weights are assigned in the reference configuration that you provide as input (i.e. the appear in the input file to this action that you set using REFERENCE=whatever.pdb). This input reference configuration consists of a simple pdb file containing the set of atoms for which you want to calculate the RMSD displacement and their positions in the reference configuration. It is important to note that the indices in this pdb need to be set correctly. The indices in this file determine the indices of the instantaneous atomic positions that are used by PLUMED when calculating this colvar. As such if you want to calculate the RMSD distance moved by the first, fourth, sixth and twenty eighth atoms in the MD codes input file then the indices of the corresponding reference positions in this pdb file should be set equal to 1, 4, 6 and 28.

The pdb input file should also contain the values of w and w'. In particular, the OCCUPANCY column (the first column after the coordinates) is used provides the values of w' that are used to calculate the position of the center of mass. The BETA column (the second column after the Cartesian coordinates) is used to provide the w values which are used in the the calculation of the displacement. Please note that it is possible to use fractional values for beta and for the occupancy. However, we recommend you only do this when you really know what you are doing however as the results can be rather strange.

In PDB files the atomic coordinates and box lengths should be in Angstroms unless you are working with natural units. If you are working with natural units then the coordinates should be in your natural length unit. For more

details on the PDB file format visit http://www.wwpdb.org/docs.html. Make sure your PDB file is correctly formatted as explained in this page.

A different method is used to calculate the RMSD distance when you use TYPE=OPTIMAL on the input line. In this case the root mean square deviation is calculated after the positions of geometric centers in the reference and instantaneous configurations are aligned AND after an optimal alignment of the two frames is performed so that motion due to rotation of the reference frame between the two structures is removed. The equation for $d(X,X')$ in this case reads:

$$d(X,X') = \sqrt{\sum_i \sum_\alpha^{x,y,z} \frac{w_i}{\sum_j w_j} [X_{i,\alpha} - com_\alpha(X) - \sum_\beta M(X,X',w')_{\alpha,\beta}(X'_{i,\beta} - com_\beta(X'))]^2}$$

where $M(X,X',w')$ is the optimal alignment matrix which is calculated using the Kearsley [21] algorithm. Again different sets of weights are used for the alignment (w') and for the displacement calculations (w). This gives a great deal of flexibility as it allows you to use a different sets of atoms (which may or may not overlap) for the alignment and displacement parts of the calculation. This may be very useful when you want to calculate how a ligand moves about in a protein cavity as you can use the protein as a reference system and do no alignment of the ligand.

(Note: when this form of RMSD is used to calculate the secondary structure variables (ALPHARMSD, ANTIBETARMSD and PARABETARMSD all the atoms in the segment are assumed to be part of both the alignment and displacement sets and all weights are set equal to one)

Please note that there are a number of other methods for calculating the distance between the instantaneous configuration and a reference configuration that are available in plumed. More information on these various methods can be found in the section of the manual on Distances from reference configurations.

When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.5, by considering the ordered list of atoms and rebuilding molecules using a procedure that is equivalent to that done in WHOLEMOLECULES . Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.

In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.

Examples

The following tells plumed to calculate the RMSD distance between the positions of the atoms in the reference file and their instantaneous position. The Kearsley algorithm is used so this is done optimally.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RMSD.tmp
RMSD REFERENCE=file.pdb TYPE=OPTIMAL
```

The reference configuration is specified in a pdb file that will have a format similar to the one shown below:

```
ATOM      1  CL  ALA     1      -3.171   0.295   2.045  1.00  1.00
ATOM      5  CLP ALA     1      -1.819  -0.143   1.679  1.00  1.00
ATOM      6  OL  ALA     1      -1.177  -0.889   2.401  1.00  1.00
ATOM      7  NL  ALA     1      -1.313   0.341   0.529  1.00  1.00
ATOM      8  HL  ALA     1      -1.845   0.961  -0.011  1.00  1.00
END
```

...

Glossary of keywords and components

Compulsory keywords

REFERENCE	a file in pdb format containing the reference structure and the atoms involved in the CV.
TYPE	(default=SIMPLE) the manner in which RMSD alignment is performed. Should be OPTIMAL or SIMPLE.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SQUARED	(default=off) This should be set if you want mean squared displacement instead of RMSD

5.3.5 TARGET

This is part of the function module

This function measures the Pythagorean distance from a particular structure measured in the space defined by some set of collective variables.

This collective variable can be used to calculate something akin to:

$$d(X,X') = |X - X'|$$

where X is the instantaneous values for a set of collective variables for the system and X' is the values that these self-same set of collective variables take in some reference structure provided as input. If we call our set of collective variables $\{s_i\}$ then this CV computes:

$$d = \sqrt{\sum_{i=1}^{N}(s_i - s_i^{(ref)})^2}$$

where $s_i^{(ref)}$ are the values of the CVs in the reference structure and N is the number of input CVs.

We can also calculate normalized euclidean differences using this action and the METRIC=NORM-EUCLIDEAN flag. In other words, we can compute:

$$d = \sqrt{\sum_{i=1}^{N}\sigma_i(s_i - s_i^{(ref)})^2}$$

where σ_i is a vector of weights. Lastly, by using the METRIC=MAHALONOBIS we can compute Mahalonobis distances using:

$$d = \left(\mathbf{s} - \mathbf{s}^{(ref)}\right)^T \mathbf{\Sigma} \left(\mathbf{s} - \mathbf{s}^{(ref)}\right)$$

where $\mathbf{s}$ is a column vector containing the values of all the CVs and $\mathbf{s}^{(ref)}$ is a column vector containing the values of the CVs in the reference configuration. $\mathbf{\Sigma}$ is then an $N \times N$ matrix that is specified in the input.

Examples

The following input calculates the distance between a reference configuration and the instantaneous position of the system in the trajectory. The position of the reference configuration is specified by providing the values of the distance between atoms 1 and 2 and atoms 3 and 4.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TARGET.tmp
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=3,4
t1: TARGET REFERENCE=reference.pdb TYPE=EUCLIDEAN
PRINT ARG=t1 FILE=colvar
```

The contents of the file containing the reference structure (reference.pdb) is shown below. As you can see you must provide information on the labels of the CVs that are being used to define the position of the reference configuration in this file together with the values that these quantities take in the reference configuration.

```
DESCRIPTION: a reference point.
REMARK WEIGHT=1.0
REMARK ARG=d1,d2
REMARK d1=1.0 d2=1.0
END
```

Glossary of keywords and components

Compulsory keywords

TYPE	(default=EUCLIDEAN) the manner in which the distance should be calculated
REFERENCE	a file in pdb format containing the reference structure. In the PDB file the atomic coordinates and box lengths should be in Angstroms unless you are working with natural units. If you are working with natural units then the coordinates should be in your natural length unit. The charges and masses of the atoms (if required) should be inserted in the beta and occupancy columns respectively. For more details on the PDB file format visit http://www.↩ wwpdb.org/docs.html

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

5.4 Functions

When performing biased dynamics or analyzing a trajectory you may wish to analyze/bias the value of some function of a set of collective variables rather than the values of the collective variables directly. You can do this with PLUMED

by using any one of the following list of functions.

Notice that in many functions you should explicitly say to PLUMED whether the result is a periodic variable or not using the keyword PERIODIC. This is crucial to allow a variable to be properly based. To know if a function is periodic of not you should answer to the following question:

- Can my function change with a discontinuity when I move my atoms in a continuous manner?

In case the answer is no, than you should use PERIODIC=NO. In case the answer is yes, then you should consider the following question:

- Are the values of the function at the discontinuity always the same or do they change?

In case the answer is that they are the same, you should use PERIODIC=A,B where A is the smallest value and B is the largest value. In case the answer is that the values at the discontinuity are not always the same, then you cannot construct a variable that can be biased with PLUMED. Consider the following examples:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FunctionsPP.md
t: TORSION ATOMS=1,2,3,4
# When atoms are moved, t could jump suddenly from -pi to +pi

c: MATHEVAL ARG=t FUNC=x*x*x PERIODIC=-31.0062766802998,31.0062766802998
# When atoms are moved, c could jump suddenly from -pi**3 to +pi**3

# equivalently, we could have used:
# c: COMBINE ARG=t POWERS=3 PERIODIC=-31.0062766802998,31.0062766802998

# compute x/y/z components of the distance between atoms 1 and 10
d: DISTANCE ATOMS=1,10 COMPONENTS

# make a new variable equal to d.z but with the correct periodicity
dz: COMBINE ARG=d.z PERIODIC=-10,10
# here we assumed the system is in a orthorhombic box with z side = 20
```

COMBINE	Calculate a polynomial combination of a set of other variables.
CUSTOM	Calculate a combination of variables using a custom expression.
ENSEMBLE	Calculates the replica averaging of a collective variable over multiple replicas.
FUNCPATHGENERAL	This function calculates path collective variables (PCVs) using an arbitrary combination of collective variables.
FUNCPATHMSD	This function calculates path collective variables.
FUNCSUMHILLS	This function is intended to be called by the command line tool sum_hills. It is meant to integrate a HILLS file or an HILLS file interpreted as a histogram in a variety of ways. It is, therefore, not expected that you use this during your dynamics (it will crash!)
LOCALENSEMBLE	Calculates the average over multiple arguments.
MATHEVAL	An alias to the CUSTOM function.
PIECEWISE	Compute a piece wise straight line through its arguments that passes through a set of ordered control points.
SORT	This function can be used to sort colvars according to their magnitudes.
STATS	Calculates statistical properties of a set of collective variables with respect to a set of reference values.

In addition to the keywords above, by enabling optional modules you can access to the following keywords:

PYTORCH_MODEL	(from PYTORCH (Machine Learning Collective Variables) module) Load a PyTorch model compiled with TorchScript.
SELECT	(from PLUMED-ISDB module) Selects an argument based on the value of a SELECTOR.

5.4.1 COMBINE

This is part of the function module

Calculate a polynomial combination of a set of other variables.

The functional form of this function is

$$C = \sum_{i=1}^{N_{arg}} c_i (x_i - a_i)^{p_i}$$

The coefficients c, the parameters a and the powers p are provided as vectors.

Notice that COMBINE is not able to predict which will be periodic domain of the computed value automatically. The user is thus forced to specify it explicitly. Use PERIODIC=NO if the resulting variable is not periodic, and PERIODIC=A,B where A and B are the two boundaries if the resulting variable is periodic.

Examples

The following input tells plumed to print the distance between atoms 3 and 5 its square (as computed from the x,y,z components) and the distance again as computed from the square root of the square.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COMBINE.tmp
DISTANCE LABEL=dist      ATOMS=3,5 COMPONENTS
COMBINE  LABEL=distance2 ARG=dist.x,dist.y,dist.z POWERS=2,2,2 PERIODIC=NO
COMBINE  LABEL=distance  ARG=distance2 POWERS=0.5 PERIODIC=NO
PRINT ARG=distance,distance2
```

(See also PRINT and DISTANCE).

The following input tells plumed to add a restraint on the cube of a dihedral angle. Notice that since the angle has a periodic domain -pi,pi its cube has a domain -pi**3,pi**3.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COMBINE.tmp
t: TORSION ATOMS=1,3,5,7
c: COMBINE ARG=t POWERS=3 PERIODIC=-31.0062766802998,31.0062766802998
RESTRAINT ARG=c KAPPA=10 AT=0
```

Glossary of keywords and components

Compulsory keywords

PERIODIC	if the output of your function is periodic then you should specify the periodicity of the function. If the output is not periodic you must state this using PERIODIC=NO
COEFFICIENTS	(default=1.0) the coefficients of the arguments in your function
PARAMETERS	(default=0.0) the parameters of the arguments in your function
POWERS	(default=1.0) the powers to which you are raising each of the arguments in your function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NORMALIZE	(default=off) normalize all the coefficients so that in total they are equal to one
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

5.4.2 CUSTOM

This is part of the function module

Calculate a combination of variables using a custom expression.

This action computes an arbitrary function of one or more collective variables. Arguments are chosen with the ARG keyword, and the function is provided with the FUNC string. Notice that this string should contain no space. Within FUNC, one can refer to the arguments as x,y,z, and t (up to four variables provided as ARG). This names can be customized using the VAR keyword (see examples below).

This function is implemented using the Lepton library, that allows to evaluate algebraic expressions and to automatically differentiate them.

If you want a function that depends not only on collective variables but also on time you can use the TIME action.

Examples

The following input tells plumed to perform a metadynamics using as a CV the difference between two distances.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CUSTOM.tmp
dAB: DISTANCE ATOMS=10,12
dAC: DISTANCE ATOMS=10,15
diff: CUSTOM ARG=dAB,dAC FUNC=y-x PERIODIC=NO
# notice: the previous line could be replaced with the following
# diff: COMBINE ARG=dAB,dAC COEFFICIENTS=-1,1
METAD ARG=diff SIGMA=0.1 HEIGHT=0.5 BIASFACTOR=10 PACE=100
```

(see also DISTANCE, COMBINE, and METAD). Notice that forces applied to diff will be correctly propagated to atoms 10, 12, and 15. Also notice that since CUSTOM is used without the VAR option the two arguments should be referred to as x and y in the expression FUNC. For simple functions such as this one it is possible to use COMBINE.

The following input tells plumed to print the angle between vectors identified by atoms 1,2 and atoms 2,3 its square (as computed from the x,y,z components) and the distance again as computed from the square root of the square.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CUSTOM.tmp
DISTANCE LABEL=d1 ATOMS=1,2 COMPONENTS
DISTANCE LABEL=d2 ATOMS=2,3 COMPONENTS
CUSTOM ...
  LABEL=theta
  ARG=d1.x,d1.y,d1.z,d2.x,d2.y,d2.z
  VAR=ax,ay,az,bx,by,bz
  FUNC=acos((ax*bx+ay*by+az*bz)/sqrt((ax*ax+ay*ay+az*az)*(bx*bx+by*by+bz*bz)))
  PERIODIC=NO
... CUSTOM
PRINT ARG=theta
```

(See also PRINT and DISTANCE).

Notice that this action implements a large number of functions (trigonometric, exp, log, etc). Among the useful functions, have a look at the step function (that is the Heaviside function). `step(x)` is defined as 1 when `x` is positive and `0` when x is negative. This allows for a straightforward implementation of if clauses.

For example, imagine that you want to implement a restraint that only acts when a distance is larger than 0.5. You can do it with

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CUSTOM.tmp
d: DISTANCE ATOMS=10,15
m: CUSTOM ARG=d FUNC=0.5*step(0.5-x)+x*step(x-0.5) PERIODIC=NO
# check the function you are applying:
PRINT ARG=d,m FILE=checkme
RESTRAINT ARG=d AT=0.5 KAPPA=10.0
```

(see also DISTANCE, PRINT, and RESTRAINT)

The meaning of the function `0.5*step(0.5-x)+x*step(x-0.5)` is:

- If x<0.5 (step(0.5-x)!=0) use 0.5
- If x>0.5 (step(x-0.5)!=0) use x Notice that the same could have been obtained using an UPPER_WALLS However, with CUSTOM you can create way more complex definitions.

Warning

If you apply forces on the variable (as in the previous example) you should make sure that the variable is continuous! Conversely, if you are just analyzing a trajectory you can safely use discontinuous variables.

A possible continuity check with gnuplot is

```
# this allow to step function to be used in gnuplot:
gnuplot> step(x)=0.5*(erf(x*10000000)+1)
# here you can test your function
gnuplot> p 0.5*step(0.5-x)+x*step(x-0.5)
```

Also notice that you can easily make logical operations on the conditions that you create. The equivalent of the AND operator is the product: `step(1.0-x)*step(x-0.5)` is only equal to 1 when x is between 0.5 and 1.0. By combining negation and AND you can obtain an OR. That is, `1-step(1.0-x)*step(x-0.5)` is only equal to 1 when x is outside the 0.5-1.0 interval.

CUSTOM can be used in combination with DISTANCE to implement variants of the DISTANCE keyword that were present in PLUMED 1.3 and that allowed to compute the distance of a point from a line defined by two other points, or the progression along that line.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CUSTOM.tmp
# take center of atoms 1 to 10 as reference point 1
p1: CENTER ATOMS=1-10
# take center of atoms 11 to 20 as reference point 2
p2: CENTER ATOMS=11-20
# take center of atoms 21 to 30 as reference point 3
p3: CENTER ATOMS=21-30

# compute distances
d12: DISTANCE ATOMS=p1,p2
d13: DISTANCE ATOMS=p1,p3
d23: DISTANCE ATOMS=p2,p3

# compute progress variable of the projection of point p3
# along the vector joining p1 and p2
# notice that progress is measured from the middle point
onaxis: CUSTOM ARG=d13,d23,d12 FUNC=(0.5*(y^2-x^2)/z) PERIODIC=NO

# compute between point p3 and the vector joining p1 and p2
fromaxis: CUSTOM ARG=d13,d23,d12,onaxis VAR=x,y,z,o FUNC=(0.5*(y^2+x^2)-o^2-0.25*z^2) PERIODIC=NO

PRINT ARG=onaxis,fromaxis
```

Notice that these equations have been used to combine RMSD from different snapshots of a protein so as to define progression (S) and distance (Z) variables [22].

Glossary of keywords and components

Compulsory keywords

PERIODIC	if the output of your function is periodic then you should specify the periodicity of the function. If the output is not periodic you must state this using PERIODIC=NO
FUNC	the function you wish to evaluate

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
VAR	the names to give each of the arguments in the function. If you have up to three arguments in your function you can use x, y and z to refer to them. Otherwise you must use this flag to give your variables names.

5.4.2.1 TIME

	This is part of the generic module

retrieve the time of the simulation to be used elsewhere

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TIME.tmp
TIME            LABEL=t1
PRINT ARG=t1
```

Glossary of keywords and components

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

5.4.3 ENSEMBLE

This is part of the function module

Calculates the replica averaging of a collective variable over multiple replicas.

Each collective variable is averaged separately and stored in a component labelled *label*.cvlabel.

Examples

The following input tells plumed to calculate the distance between atoms 3 and 5 and the average it over the available replicas.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ENSEMBLE.tmp
dist: DISTANCE ATOMS=3,5
ens: ENSEMBLE ARG=dist
PRINT ARG=dist,ens.dist
```

Glossary of keywords and components

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
REWEIGHT	(default=off) simple REWEIGHT using the latest ARG as energy
CENTRAL	(default=off) calculate a central moment instead of a standard moment
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
TEMP	the system temperature - this is only needed if you are reweighting
MOMENT	the moment you want to calculate in alternative to the mean or the variance
POWER	the power of the mean (and moment)

5.4.4 FUNCPATHGENERAL

This is part of the function module

This function calculates path collective variables (PCVs) using an arbitrary combination of collective variables.

The method used to calculate the PCVs that is used in this method is described in [23].

This variable computes the progress along a given set of frames that is provided in an input file ("s" component) and the distance from them ("z" component). The input file could be a colvar file generated with plumed driver on a trajectory containing the frames.

The metric for the path collective variables takes the following form:

$$R[X - X_i] = \sum_{j=1}^{M} c_j^2 (x_j - x_{i,j})^2 \, .$$

Here, the coefficients c_j determine the relative weights of the collective variables c_j in the metric. A value for the lambda coefficient also needs to be provided, typically chosen in such a way that it ensures a smooth variation of the "s" component.

Examples

This command calculates the PCVs using the values from the file COLVAR_TRAJ and the provided values for the lambda and the coefficients. Since the columns in the file were not specified, the first one will be ignored (assumed to correspond to the time) and the rest used.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FUNCPATHGENERAL.tmp
FUNCPATHGENERAL ...
LABEL=path
LAMBDA=12.2
REFERENCE=COLVAR_TRAJ
COEFFICIENTS=0.3536,0.3536,0.3536,0.3536,0.7071
ARG=d1,d2,d,t,drmsd
... FUNCPATHGENERAL
```

The command below is a variation of the previous one, specifying a subset of the collective variables and using a neighbor list. The columns are zero-indexed. The neighbor list will include the 10 closest frames and will be recalculated every 20 steps.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FUNCPATHGENERAL.tmp
FUNCPATHGENERAL ...
LABEL=path
LAMBDA=5.0
REFERENCE=COLVAR_TRAJ
COLUMNS=2,3,4
COEFFICIENTS=0.3536,0.3536,0.3536
ARG=d2,d,t
NEIGH_SIZE=10
NEIGH_STRIDE=20
... FUNCPATHGENERAL
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
s	Position on the path
z	Distance from the path

Compulsory keywords

LAMBDA	Lambda parameter required for smoothing
COEFFICIENTS	Coefficients to be assigned to the CVs
REFERENCE	Colvar file needed to provide the CV milestones

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
COLUMNS	List of columns in the reference colvar file specifying the CVs
NEIGH_SIZE	Size of the neighbor list
NEIGH_STRIDE	How often the neighbor list needs to be calculated in time units

5.4.5 FUNCPATHMSD

This is part of the function module

This function calculates path collective variables.

This is the Path Collective Variables implementation (see [5]). This variable computes the progress along a given set of frames that is provided in input ("s" component) and the distance from them ("z" component). It is a function of mean squared displacement that are obtained by the joint use of mean squared displacement variables with the SQUARED flag (see below).

Examples

Here below is a case where you have defined three frames and you want to calculate the progress along the path and the distance from it in p1

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FUNCPATHMSD.tmp
t1: RMSD REFERENCE=frame_1.pdb TYPE=OPTIMAL SQUARED
t2: RMSD REFERENCE=frame_21.pdb TYPE=OPTIMAL SQUARED
t3: RMSD REFERENCE=frame_42.pdb TYPE=OPTIMAL SQUARED
p1: FUNCPATHMSD ARG=t1,t2,t3 LAMBDA=500.0
PRINT ARG=t1,t2,t3,p1.s,p1.z STRIDE=1 FILE=colvar FMT=%8.4f
```

For this input you would then define the position of the reference coordinates in three separate pdb files. The contents of the file frame_1.pdb are shown below:

```
ATOM      1  CL  ALA     1      -3.171   0.295   2.045  1.00  1.00
ATOM      5  CLP ALA     1      -1.819  -0.143   1.679  1.00  1.00
ATOM      6  OL  ALA     1      -1.177  -0.889   2.401  1.00  1.00
ATOM      7  NL  ALA     1      -1.313   0.341   0.529  1.00  1.00
ATOM      8  HL  ALA     1      -1.845   0.961  -0.011  1.00  1.00
END
```

This is then frame.21.pdb:

```
ATOM      1  CL  ALA     1      -3.089   1.850   1.546  1.00  1.00
ATOM      5  CLP ALA     1      -1.667   1.457   1.629  1.00  1.00
ATOM      6  OL  ALA     1      -0.974   1.868   2.533  1.00  1.00
ATOM      7  NL  ALA     1      -1.204   0.683   0.642  1.00  1.00
ATOM      8  HL  ALA     1      -1.844   0.360  -0.021  1.00  1.00
END
```

and finally this is frame_42.pdb:

```
ATOM      1  CL  ALA     1      -3.257   1.605   1.105  1.00  1.00
ATOM      5  CLP ALA     1      -1.941   1.459   0.447  1.00  1.00
ATOM      6  OL  ALA     1      -1.481   2.369  -0.223  1.00  1.00
ATOM      7  NL  ALA     1      -1.303   0.291   0.647  1.00  1.00
ATOM      8  HL  ALA     1      -1.743  -0.379   1.229  1.00  1.00
END
```

This second example shows how to define a PATH in CONTACTMAP space:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FUNCPATHMSD.tmp
CONTACTMAP ...
ATOMS1=1,2 REFERENCE1=0.1
ATOMS2=3,4 REFERENCE2=0.5
ATOMS3=4,5 REFERENCE3=0.25
ATOMS4=5,6 REFERENCE4=0.0
SWITCH={RATIONAL R_0=1.5}
LABEL=c1
CMDIST
... CONTACTMAP

CONTACTMAP ...
ATOMS1=1,2 REFERENCE1=0.3
ATOMS2=3,4 REFERENCE2=0.9
ATOMS3=4,5 REFERENCE3=0.45
ATOMS4=5,6 REFERENCE4=0.1
```

```
SWITCH={RATIONAL R_0=1.5}
LABEL=c2
CMDIST
... CONTACTMAP

CONTACTMAP ...
ATOMS1=1,2 REFERENCE1=1.0
ATOMS2=3,4 REFERENCE2=1.0
ATOMS3=4,5 REFERENCE3=1.0
ATOMS4=5,6 REFERENCE4=1.0
SWITCH={RATIONAL R_0=1.5}
LABEL=c3
CMDIST
... CONTACTMAP

p1: FUNCPATHMSD ARG=c1,c2,c3 LAMBDA=500.0
PRINT ARG=c1,c2,c3,p1.s,p1.z STRIDE=1 FILE=colvar FMT=%8.4f
```

This third example shows how to define a PATH in PIV space:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FUNCPATHMSD.tmp
PIV ...
LABEL=c1
PRECISION=1000
NLIST
REF_FILE=Ref1.pdb
PIVATOMS=2
ATOMTYPES=A,B
ONLYDIRECT
SFACTOR=1.0,0.2
SORT=1,1
SWITCH1={RATIONAL R_0=0.6 MM=12 NN=4}
SWITCH2={RATIONAL R_0=0.5 MM=10 NN=5}
NL_CUTOFF=1.2,1.2
NL_STRIDE=10,10
NL_SKIN=0.1,0.1
... PIV
PIV ...
LABEL=c2
PRECISION=1000
NLIST
REF_FILE=Ref2.pdb
PIVATOMS=2
ATOMTYPES=A,B
ONLYDIRECT
SFACTOR=1.0,0.2
SORT=1,1
SWITCH1={RATIONAL R_0=0.6 MM=12 NN=4}
SWITCH2={RATIONAL R_0=0.4 MM=10 NN=5}
NL_CUTOFF=1.2,1.2
NL_STRIDE=10,10
NL_SKIN=0.1,0.1
... PIV

p1: FUNCPATHMSD ARG=c1,c2 LAMBDA=0.180338
METAD ARG=p1.s,p1.z SIGMA=0.01,0.2 HEIGHT=0.8 PACE=500   LABEL=res
PRINT ARG=c1,c2,p1.s,p1.z,res.bias STRIDE=500  FILE=colvar FMT=%15.6f
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
s	the position on the path
z	the distance from the path

Compulsory keywords

LAMBDA	the lambda parameter is needed for smoothing, is in the units of plumed

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
NEIGH_SIZE	size of the neighbor list
NEIGH_STRIDE	how often the neighbor list needs to be calculated in time units

5.4.6 FUNCSUMHILLS

This is part of the function module

This function is intended to be called by the command line tool sum_hills. It is meant to integrate a HILLS file or an HILLS file interpreted as a histogram in a variety of ways. It is, therefore, not expected that you use this during your dynamics (it will crash!)

In the future one could implement periodic integration during the metadynamics or straightforward MD as a tool to check convergence

Examples

There are currently no examples for this keyword.

Glossary of keywords and components

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ISCLTOOL	(default=off) use via plumed command line: calculate at read phase and then go
PARALLELREAD	(default=off) read parallel HILLS file
NEGBIAS	(default=off) dump negative bias (-bias) instead of the free energy: needed in well tempered with flexible hills
NOHISTORY	(default=off) to be used with INITSTRIDE: it splits the bias/histogram in pieces without previous history
MINTOZERO	(default=off) translate the resulting bias/histogram to have the minimum to zero
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
HILLSFILES	source file for hills creation(may be the same as HILLS)
HISTOFILES	source file for histogram creation(may be the same as HILLS)
HISTOSIGMA	sigmas for binning when the histogram correction is needed
PROJ	only with sumhills: the projection on the CVs
KT	only with sumhills: the kt factor when projection on CVs
GRID_MIN	the lower bounds for the grid
GRID_MAX	the upper bounds for the grid
GRID_BIN	the number of bins for the grid
GRID_SPACING	the approximate grid spacing (to be used as an alternative or together with GRID_BIN)
INTERVAL	set one dimensional INTERVAL
OUTHILLS	output file for hills
OUTHISTO	output file for histogram
INITSTRIDE	stride if you want an initial dump
STRIDE	stride when you do it on the fly
FMT	the format that should be used to output real numbers

5.4.7 LOCALENSEMBLE

	This is part of the function module

Calculates the average over multiple arguments.

If more than one collective variable is given for each argument then they are averaged separately. The average is stored in a component labelled *label*.cvlabel.

Examples

The following input tells plumed to calculate the chemical shifts for four different proteins in the same simulation box then average them, calculated the sum of the squared deviation with respect to the experimental values and applies a linear restraint.

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=data/template.pdb

chaina: GROUP ATOMS=1-1640
chainb: GROUP ATOMS=1641-3280
chainc: GROUP ATOMS=3281-4920
chaind: GROUP ATOMS=4921-6560

WHOLEMOLECULES ENTITY0=chaina ENTITY1=chainb ENTITY2=chainc ENTITY3=chaind

csa: CS2BACKBONE ATOMS=chaina NRES=100 DATA=data/ TEMPLATE=chaina.pdb NOPBC
csb: CS2BACKBONE ATOMS=chainb NRES=100 DATA=data/ TEMPLATE=chainb.pdb NOPBC
csc: CS2BACKBONE ATOMS=chainc NRES=100 DATA=data/ TEMPLATE=chainc.pdb NOPBC
csd: CS2BACKBONE ATOMS=chaind NRES=100 DATA=data/ TEMPLATE=chaind.pdb NOPBC

ensca: LOCALENSEMBLE NUM=4 ARG1=(csa\.ca_.*) ARG2=(csb\.ca_.*) ARG3=(csc\.ca_.*) ARG4=(csd\.ca_.*)
enscb: LOCALENSEMBLE NUM=4 ARG1=(csa\.cb_.*) ARG2=(csb\.cb_.*) ARG3=(csc\.cb_.*) ARG4=(csd\.cb_.*)
ensco: LOCALENSEMBLE NUM=4 ARG1=(csa\.co_.*) ARG2=(csb\.co_.*) ARG3=(csc\.co_.*) ARG4=(csd\.co_.*)
enshn: LOCALENSEMBLE NUM=4 ARG1=(csa\.hn_.*) ARG2=(csb\.hn_.*) ARG3=(csc\.hn_.*) ARG4=(csd\.hn_.*)
ensnh: LOCALENSEMBLE NUM=4 ARG1=(csa\.nh_.*) ARG2=(csb\.nh_.*) ARG3=(csc\.nh_.*) ARG4=(csd\.nh_.*)

stca: STATS ARG=(ensca\.csa\.ca_.*) PARARG=(csa\.expca_.*) SQDEVSUM
stcb: STATS ARG=(enscb\.csa\.cb_.*) PARARG=(csa\.expcb_.*) SQDEVSUM
stco: STATS ARG=(ensco\.csa\.co_.*) PARARG=(csa\.expco_.*) SQDEVSUM
sthn: STATS ARG=(enshn\.csa\.hn_.*) PARARG=(csa\.exphn_.*) SQDEVSUM
stnh: STATS ARG=(ensnh\.csa\.nh_.*) PARARG=(csa\.expnh_.*) SQDEVSUM

res: RESTRAINT ARG=stca.*,stcb.*,stco.*,sthn.*,stnh.* AT=0.,0.,0.,0.,0. KAPPA=0.,0.,0.,0.,0 SLOPE=16.,16.,12.,
```

Glossary of keywords and components

Compulsory keywords

NUM	the number of local replicas

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

5.4.8 MATHEVAL

This is part of the function module

An alias to the CUSTOM function.

This alias is kept in order to maintain compatibility with previous PLUMED versions. However, notice that as of PLUMED 2.5 the libmatheval library is not linked anymore, and the MATHEVAL function is implemented using the Lepton library.

Examples

Just replace CUSTOM with MATHEVAL.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MATHEVAL.tmp
d: DISTANCE ATOMS=10,15
m: MATHEVAL ARG=d FUNC=0.5*step(0.5-x)+x*step(x-0.5) PERIODIC=NO
# check the function you are applying:
PRINT ARG=d,m FILE=checkme
RESTRAINT ARG=d AT=0.5 KAPPA=10.0
```

(see also DISTANCE, PRINT, and RESTRAINT)

Glossary of keywords and components

Compulsory keywords

PERIODIC	if the output of your function is periodic then you should specify the periodicity of the function. If the output is not periodic you must state this using PERIODIC=NO
FUNC	the function you wish to evaluate

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
VAR	the names to give each of the arguments in the function. If you have up to three arguments in your function you can use x, y and z to refer to them. Otherwise you must use this flag to give your variables names.

5.4.9 PIECEWISE

This is part of the function module

Compute a piece wise straight line through its arguments that passes through a set of ordered control points.

For variables less than the first (greater than the last) point, the value of the first (last) point is used.

$$\frac{y_{i+1}-y_i}{x_{i+1}-x_i}(s-x_i)+y_i; if x_i < s < x_{i+1}$$

$$y_N; if x > x_{N-1}$$

$$y_1; if x < x_0$$

Control points are passed using the POINT0=... POINT1=... syntax as in the example below

If one argument is supplied, it results in a scalar quantity. If multiple arguments are supplied, it results in a vector of values. Each value will be named as the name of the original argument with suffix _pfunc.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PIECEWISE.tmp
dist1: DISTANCE ATOMS=1,10
dist2: DISTANCE ATOMS=2,11

pw: PIECEWISE POINT0=1,10 POINT1=2,PI POINT2=3,10 ARG=dist1
ppww: PIECEWISE POINT0=1,10 POINT1=2,PI POINT2=3,10 ARG=dist1,dist2
PRINT ARG=pw,ppww.dist1_pfunc,ppww.dist2_pfunc
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
_pfunc	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will be named with the arguments of the function followed by the character string _pfunc. These quantities tell the user the values of the piece wise functions of each of the arguments.

Compulsory keywords

POINT	This keyword is used to specify the various points in the function above.. You can use multiple instances of this keyword i.e. POINT1, POINT2, POINT3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

5.4.10 SORT

This is part of the function module

This function can be used to sort colvars according to their magnitudes.

Description of components

This function sorts its arguments according to their magnitudes. The lowest argument will be labelled *label*.1, the second lowest will be labelled *label*.2 and so on.

Examples

The following input tells plumed to print the distance of the closest and of the farthest atoms to atom 1, chosen among atoms from 2 to 5

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SORT.tmp
d12:  DISTANCE ATOMS=1,2
d13:  DISTANCE ATOMS=1,3
d14:  DISTANCE ATOMS=1,4
d15:  DISTANCE ATOMS=1,5
sort: SORT ARG=d12,d13,d14,d15
PRINT ARG=sort.1,sort.4
```

Glossary of keywords and components

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

5.4.11 STATS

This is part of the function module

Calculates statistical properties of a set of collective variables with respect to a set of reference values.

In particular it calculates and stores as components the sum of the squared deviations, the correlation, the slope and the intercept of a linear fit.

The reference values can be either provided as values using PARAMETERS or using value without derivatives from other actions using PARARG (for example using experimental values from collective variables such as CS2BACKBONE, RDC, NOE, PRE).

Examples

The following input tells plumed to print the distance between three couple of atoms and compare them with three reference distances.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/STATS.tmp
d1: DISTANCE ATOMS=10,50
d2: DISTANCE ATOMS=1,100
d3: DISTANCE ATOMS=45,75
st: STATS ARG=d1,d2,d3 PARAMETERS=1.5,4.0,2.0
PRINT ARG=d1,d2,d3,st.*
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
sqdevsum	the sum of the squared deviations between arguments and parameters
corr	the correlation between arguments and parameters
slope	the slope of a linear fit between arguments and parameters
intercept	the intercept of a linear fit between arguments and parameters

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
sqd	**SQDEV**	the squared deviations between arguments and parameters

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
SQDEVSUM	(default=off) calculates only SQDEVSUM
SQDEV	(default=off) calculates and store the SQDEV as components
UPPERDISTS	(default=off) calculates and store the SQDEV as components
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
PARARG	the input for this action is the scalar output from one or more other actions without derivatives.
PARAMETERS	the parameters of the arguments in your function

5.5 MultiColvar

Oftentimes, when you do not need one of the collective variables described elsewhere in the manual, what you want instead is a function of a distribution of collective variables of a particular type. In other words, you would like to calculate a function something like this:

$$s = \sum_i g[f(\{X\}_i)]$$

In this expression g is a function that takes in one argument and f is a function that takes a set of atomic positions as argument. The symbol $\{X\}_i$ is used to indicate the fact that the function f is evaluated for a number of different sets of atoms. If you would just like to output the values of all the various f functions you should use the command DUMPMULTICOLVAR

This functionality is useful if you need to calculate a minimum distance or the number of coordination numbers greater than a 3.0.
To avoid duplicating the code to calculate an angle or distance many times and to make it easier to implement very complex collective variables PLUMED provides these sort of collective variables using so-called MultiColvars. MultiColvars are named in this way because a single PLUMED action can be used to calculate a number of different collective variables. For instance the DISTANCES action can be used to calculate the minimum distance, the number of distances less than a certain value, the number of distances within a certain range... A more detailed introduction to multicolvars is provided in this 10-minute video. Descriptions of the various multicolvars that are implemented in PLUMED 2 are given below:

ANGLES	Calculate functions of the distribution of angles .
BOND_DIRECTIONS	Calculate the vectors connecting atoms that are within cutoff defined using a switching function.
BRIDGE	Calculate the number of atoms that bridge two parts of a structure
COORDINATIONNUMBER	Calculate the coordination numbers of atoms so that you can then calculate functions of the distribution of coordination numbers such as the minimum, the number less than a certain quantity and so on.
DENSITY	Calculate functions of the density of atoms as a function of the box. This allows one to calculate the number of atoms in half the box.
DISTANCES	Calculate the distances between one or many pairs of atoms. You can then calculate functions of the distribution of distances such as the minimum, the number less than a certain quantity and so on.
ENVIRONMENTSIMILARITY	Measure how similar the environment around atoms is to that found in some reference crystal structure.
FCCUBIC	Measure how similar the environment around atoms is to that found in a FCC structure.
HBPAMM_SH	Number of HBPAMM hydrogen bonds formed by each hydrogen atom in the system
INPLANEDISTANCES	Calculate distances in the plane perpendicular to an axis
MOLECULES	Calculate the vectors connecting a pair of atoms in order to represent the orientation of a molecule.
PLANES	Calculate the plane perpendicular to two vectors in order to represent the orientation of a planar molecule.
Q3	Calculate 3rd order Steinhardt parameters.
Q4	Calculate fourth order Steinhardt parameters.
Q6	Calculate sixth order Steinhardt parameters.
SIMPLECUBIC	Calculate whether or not the coordination spheres of atoms are arranged as they would be in a simple cubic structure.
TETRAHEDRAL	Calculate the degree to which the environment about ions has a tetrahedral order.
TORSIONS	Calculate whether or not a set of torsional angles are within a particular range.
XANGLES	Calculate the angles between the vector connecting two atoms and the x axis.
XDISTANCES	Calculate the x components of the vectors connecting one or many pairs of atoms.You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.
XYDISTANCES	Calculate distance between a pair of atoms neglecting the z-component.
XYTORSIONS	Calculate the torsional angle around the x axis from the positive y direction.
XZDISTANCES	Calculate distance between a pair of atoms neglecting the y-component.
XZTORSIONS	Calculate the torsional angle around the x axis from the positive z direction.
YANGLES	Calculate the angles between the vector connecting two atoms and the y axis.
YDISTANCES	Calculate the y components of the vectors connecting one or many pairs of atoms.You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.
YXTORSIONS	Calculate the torsional angle around the y axis from the positive x direction.

YZDISTANCES	Calculate distance between a pair of atoms neglecting the x-component.
YZTORSIONS	Calculate the torsional angle around the y axis from the positive z direction.
ZANGLES	Calculate the angles between the vector connecting two atoms and the z axis.
ZDISTANCES	Calculate the z components of the vectors connecting one or many pairs of atoms.You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.
ZXTORSIONS	Calculate the torsional angle around the z axis from the positive x direction.
ZYTORSIONS	Calculate the torsional angle around the z axis from the positive y direction.

To instruct PLUMED to calculate a multicolvar you give an instruction that looks something like this:

```
NAME <atoms involved> <parameters> <what am I calculating> TOL=0.001 LABEL=label
```

Oftentimes the simplest way to specify the atoms involved is to use multiple instances of the ATOMS keyword i.e. ATOMS1, ATOMS2, ATOMS3,... Separate instances of the quantity specified by NAME are then calculated for each of the sets of atoms. For example if the command issued contains the following:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CollectiveVariablesPP.md
DISTANCES ATOMS1=1,2 ATOMS2=3,4 ATOMS3=5,6
```

The distances between atoms 1 and 2, atoms 3 and 4, and atoms 5 and 6 are calculated. Obviously, generating this sort of input is rather tedious so short cuts are also available many of the collective variables. These are described on the manual pages for the actions.

After specifying the atoms involved you sometimes need to specify some parameters that required in the calculation. For instance, for COORDINATIONNUMBER - the number of atoms in the first coordination sphere of each of the atoms in the system - you need to specify the parameters for a switchingfunction that will tell us whether or not an atom is in the first coordination sphere. Details as to how to do this are provided on the manual pages.

One of the most important keywords for multicolvars is the TOL keyword. This specifies that terms in sums that contribute less than a certain value can be ignored. In addition, it is assumed that the derivative with respect to these terms are essentially zero. By increasing the TOL parameter you can increase the speed of the calculation. Be aware, however, that this increase in speed is only possible because you are lowering the accuracy with which you are computing the quantity of interest.

Once you have specified the base quantities that are to be calculated from the atoms involved and any parameters you need to specify what function of these base quantities is to be calculated. For most multicolvars you can calculate the minimum, the number less than a target value, the number within a certain range, the number more than a target value and the average value directly.

5.5.1 MultiColvar functions

It is possible to use multicolvars to calculate complicated collective variables by exploiting the fact that the output from one multicolvar can be used as input to a second multicolvar. One simple way of exploiting this functionality is to filter the atoms based on the value they have for a symmetry function. For example you might want to consider only those atoms that with a COORDINATIONNUMBER higher that a certain threshold when calculating some particularly expensive symmetry function such as Q6. The following methods can thus all be used to filter the values of multicolvars in this way:

MFILTER_BETWEEN	This action can be used to filter the colvar values calculated by a mcolvso that one can compute the mean and so on for only those multicolvars within a certain range.
MFILTER_LESS	This action can be used to filter the distribution of colvar values in a mcolvso that one can compute the mean and so on for only those multicolvars less than a tolerance.
MFILTER_MORE	This action can be used to filter the distribution of colvar values in a mcolvso that one

An alternative way of filtering atoms is to consider only those atoms in a particular part of the simulation box. This can be done by exploiting the following methods

AROUND	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.
CAVITY	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a box defined by the positions of four atoms.
INCYLINDER	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.
INENVELOPE	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a region where the density of a certain type of atom is high.
INSPHERE	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.
TETRAHEDRALPORE	This quantity can be used to calculate functions of the distribution of collective variables for the atoms lie that lie in a box defined by the positions of four atoms at the corners of a tetrahedron.

The idea with these methods is that function of the form:

$$s = \sum_i w(\{X\}_i) g[f(\{X\}_i)]$$

can be evaluated where once again g is a function with one argument and g is a function of a set of atomic positions.
The difference from the more general function described earlier is that we now have a weight w which is again a function of the atomic positions. This weight varies between zero and one and it is this weight that is calculated in the list of filtering methods and volume methods described in the lists above.

In addition to these volume and filtering methods it is also possible to calculate the local average of a quantities in the manner described in [24] using the LOCAL_AVERAGE method. Furthermore, in many cases Q6, MOLECULES and PLANES the symmetry function being evaluated is a vector. You can thus construct a variety of novel collective variables by taking dot products of vectors on adjacent atoms as described below:

GRADIENT	Calculate the gradient of the average value of a multicolvar value
INTERMOLECULARTORSIONS	Calculate torsion angles between vectors on adjacent molecules
LOCAL_AVERAGE	Calculate averages over spherical regions centered on atoms
LOCAL_Q3	Calculate the local degree of order around an atoms by taking the average dot product between the q_3 vector on the central atom and the q_3 vector on the atoms in the first coordination sphere.
LOCAL_Q4	Calculate the local degree of order around an atoms by taking the average dot product between the q_4 vector on the central atom and the q_4 vector on the atoms in the first coordination sphere.
LOCAL_Q6	Calculate the local degree of order around an atoms by taking the average dot product between the q_6 vector on the central atom and the q_6 vector on the atoms in the first coordination sphere.
MCOLV_COMBINE	Calculate linear combinations of multiple multicolvars
MCOLV_PRODUCT	Calculate a product of multiple multicolvars
NLINKS	Calculate number of pairs of atoms/molecules that are linked
PAMM	Probabilistic analysis of molecular motifs.
POLYMER_ANGLES	Calculate a function to investigate the relative orientations of polymer angles
SMAC	Calculate a variant on the SMAC collective variable

The final set of functions that you can apply on multicolvars are functions that transform all the colvars calculated

using a multicolvar using a function. This can be useful if you are calculating some complicated derived quantity of some simpler quantity. It is also useful if you are calculating a Willard Chandler surface or a histogram. The actions that you can use to perform these transforms are:

MTRANSFORM_BETWEEN	This action can be used to transform the colvar values calculated by a Multi↩ Colvar using a histogram bead
MTRANSFORM_LESS	This action can be used to transform the colvar values calculated by a multicovar using a switching function
MTRANSFORM_MORE	This action can be used to transform the colvar values calculated by a multicolvar using one minus a switching function

5.5.2 MultiColvar bias

There may be occasions when you want add restraints on many collective variables. For instance if you are studying a cluster you might want to add a wall on the distances between each of the atoms and the center of mass of the cluster in order to prevent the cluster subliming. Alternatively, you may wish to insist that a particular set of atoms in your system all have a coordination number greater than 2. You can add these sorts of restraints by employing the following biases, which all act on the set of collective variable values calculated by a multicolvar. So for example the following set of commands:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CollectiveVariablesPP.md
COM ATOMS=1-20 LABEL=c1
DISTANCES GROUPA=c1 GROUPB=1-20 LABEL=d1
UWALLS DATA=d1 AT=2.5 KAPPA=0.2 LABEL=sr
```

creates the aforementioned set of restraints on the distances between the 20 atoms in a cluster and the center of mass of the cluster.

The list of biases of this type are as follows:

LWALLS	Add LOWER_WALLS restraints on all the multicolvar values
UWALLS	Add UPPER_WALLS restraints on all the multicolvar values

Notice that (in theory) you could also use this functionality to add additional terms to your force field or to implement your force field.

5.5.3 Extracting all the base quantities

There may be occasions where you want to get information on all the individual colvar values that you have calculated. For example you might want to output the values of all the coordination numbers calculated by a COORDINATIONNUMBER action. You can thus use the following command to extract this sort of information, DUMPMULTICOLVAR.

5.5.4 ANGLES

	This is part of the multicolvar module

Calculate functions of the distribution of angles .

You can use this command to calculate functions such as:

$$f(x) = \sum_{ijk} g(\theta_{ijk})$$

Alternatively you can use this command to calculate functions such as:

$$f(x) = \sum_{ijk} s(r_{ij}) s(r_{jk}) g(\theta_{ijk})$$

where $s(r)$ is a switchingfunction. This second form means that you can use this to calculate functions of the angles in the first coordination sphere of an atom / molecule [25].

Examples

The following example instructs plumed to find the average of two angles and to print it to a file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ANGLES.tmp
ANGLES ATOMS1=1,2,3 ATOMS2=4,5,6 MEAN LABEL=a1
PRINT ARG=a1.mean FILE=colvar
```

The following example tells plumed to calculate all angles involving at least one atom from GROUPA and two atoms from GROUPB in which the distances are less than 1.0. The number of angles between $\frac{\pi}{4}$ and $\frac{3\pi}{4}$ is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ANGLES.tmp
ANGLES GROUPA=1-10 GROUPB=11-100 BETWEEN={GAUSSIAN LOWER=0.25pi UPPER=0.75pi} SWITCH={GAUSSIAN R_0=1.0} LABEL=
PRINT ARG=a1.between FILE=colvar
```

This final example instructs plumed to calculate all the angles in the first coordination spheres of the atoms. The bins for a normalized histogram of the distribution is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ANGLES.tmp
ANGLES GROUP=1-38 HISTOGRAM={GAUSSIAN LOWER=0.0 UPPER=pi NBINS=20} SWITCH={GAUSSIAN R_0=1.0} LABEL=a1
PRINT ARG=a1.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one angle will be calculated for each ATOM keyword you specify (all ATOM keywords should provide the indices of three atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate angles for each distinct set of three atoms in the group

Or alternatively by using

GROUPA	A group of central atoms about which angles should be calculated
GROUPB	When used in conjunction with GROUPA this keyword instructs plumed to calculate all distinct angles involving one atom from GROUPA and two atoms from GROUPB. The atom from GROUPA is the central atom.

Or alternatively by using

GROUPC	This must be used in conjunction with GROUPA and GROUPB. All angles involving one atom from GROUPA, one atom from GROUPB and one atom from GROUPC are calculated. The GROUPA atoms are assumed to be the central atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
SWITCH	A switching function that ensures that only angles between atoms that are within a certain fixed cutoff are calculated. The following provides information on the switchingfunction that are available.
SWITCHA	A switching function on the distance between the atoms in group A and the atoms in group B
SWITCHB	A switching function on the distance between the atoms in group A and the atoms in group B

5.5.5 BOND_DIRECTIONS

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the vectors connecting atoms that are within cutoff defined using a switching function.

Examples

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of vectors calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. All of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set the label for the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name. In addition, you can calculate all of these scalar functions for one particular component of the calculated vector by making use of the COMPONENT keyword. The first component is used to refer to the norm of the vector. The individual components can then be referenced using the numbers 2, 3, and so on. So as an example MEAN1={COMPON↩ ENT=1} calculates the average vector norm. MEAN2={COMPONENT=2} by contrast calculates the mean for all of the first components of the vectors.

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum

The atoms involved can be specified using

ATOMS	the atoms involved in each of the vectors you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one vector will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Compulsory keywords

NN	(default=12) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
VSUM	calculate the norm of the sum of vectors. The final value can be referenced using *label*.vsum. You can use multiple instances of this keyword i.e. VSU↩ M1, VSUM2, VSUM3... The corresponding values are then referenced using *label*.vsum-1, *label*.vsum-2, *label*.vsum-3...

5.5.6 BRIDGE

This is part of the multicolvar module

Calculate the number of atoms that bridge two parts of a structure

This quantity calculates:

$$f(x) = \sum_{ijk} s_A(r_{ij}) s_B(r_{ik})$$

where the sum over i is over all the ``bridging atoms" and s_A and s_B are switchingfunction.

Examples

The following example instructs plumed to calculate the number of water molecules that are bridging between atoms 1-10 and atoms 11-20 and to print the value to a file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BRIDGE.tmp
w1: BRIDGE BRIDGING_ATOMS=100-200 GROUPA=1-10 GROUPB=11-20 SWITCH={RATIONAL R_0=0.2}
PRINT ARG=w1 FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

BRIDGING_ATOMS	The list of atoms that can form the bridge between the two interesting parts of the structure.
GROUPA	The list of atoms that are in the first interesting part of the structure
GROUPB	The list of atoms that are in the second interesting part of the structure

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

SWITCH	The parameters of the two switchingfunction in the above formula
SWITCHA	The switchingfunction on the distance between bridging atoms and the atoms in group A
SWITCHB	The switchingfunction on the distance between the bridging atoms and the atoms in group B

5.5.7 COORDINATIONNUMBER

This is part of the multicolvar module

Calculate the coordination numbers of atoms so that you can then calculate functions of the distribution of coordination numbers such as the minimum, the number less than a certain quantity and so on.

So that the calculated coordination numbers have continuous derivatives the following function is used:

$$s = \frac{1-\left(\frac{r-d_0}{r_0}\right)^n}{1-\left(\frac{r-d_0}{r_0}\right)^m}$$

If R_POWER is set, this will use the product of pairwise distance raised to the R_POWER with the coordination number function defined above. This was used in White and Voth [26] as a way of indirectly biasing radial distribution functions. Note that in that reference this function is referred to as moments of coordination number, but here we call them powers to distinguish from the existing MOMENTS keyword of Multicolvars.

Examples

The following input tells plumed to calculate the coordination numbers of atoms 1-100 with themselves. The minimum coordination number is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COORDINATIONNUMBER.tmp
COORDINATIONNUMBER SPECIES=1-100 R_0=1.0 MIN={BETA=0.1}
```

The following input tells plumed to calculate how many atoms from 1-100 are within 3.0 of each of the atoms from 101-110. In the first 101 is the central atom, in the second 102 is the central atom and so on. The number of coordination numbers more than 6 is then computed.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COORDINATIONNUMBER.tmp
COORDINATIONNUMBER SPECIESA=101-110 SPECIESB=1-100 R_0=3.0 MORE_THAN={RATIONAL R_0=6.0 NN=6 MM=12 D_0=0}
```

The following input tells plumed to calculate the mean coordination number of all atoms with themselves and its powers. An explicit cutoff is set for each of 8.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COORDINATIONNUMBER.tmp
cn0: COORDINATIONNUMBER SPECIES=1-10 SWITCH={RATIONAL R_0=1.0 D_MAX=8} MEAN
cn1: COORDINATIONNUMBER SPECIES=1-10 SWITCH={RATIONAL R_0=1.0 D_MAX=8} R_POWER=1 MEAN
cn2: COORDINATIONNUMBER SPECIES=1-10 SWITCH={RATIONAL R_0=1.0 D_MAX=8} R_POWER=2 MEAN
PRINT ARG=cn0.mean,cn1.mean,cn2.mean STRIDE=1 FILE=cn_out
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
R_POWER	Multiply the coordination number function by a power of r, as done in White and Voth (see note above, default: no)
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...

LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.8 DENSITY

This is part of the multicolvar module

Calculate functions of the density of atoms as a function of the box. This allows one to calculate the number of atoms in half the box.

Examples

The following example calculates the number of atoms in one half of the simulation box.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DENSITY.tmp
DENSITY SPECIES=1-100 LABEL=d
AROUND ATOM=101 DATA=d SIGMA=0.1 XLOWER=0.0 XUPPER=0.5 LABEL=d1
PRINT ARG=d1.* FILE=colvar1 FMT=%8.4f
```

Glossary of keywords and components

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.5.9 DISTANCES

This is part of the multicolvar module

Calculate the distances between one or many pairs of atoms. You can then calculate functions of the distribution of distances such as the minimum, the number less than a certain quantity and so on.

Examples

The following input tells plumed to calculate the distances between atoms 3 and 5 and between atoms 1 and 2 and to print the minimum for these two distances.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCES.tmp
d1: DISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1}
PRINT ARG=d1.min
```

(See also PRINT).

The following input tells plumed to calculate the distances between atoms 3 and 5 and between atoms 1 and 2 and then to calculate the number of these distances that are less than 0.1 nm. The number of distances less than 0.1nm is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCES.tmp
d1: DISTANCES ATOMS1=3,5 ATOMS2=1,2 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lessthan
```

(See also PRINT switchingfunction).

The following input tells plumed to calculate all the distances between atoms 1, 2 and 3 (i.e. the distances between atoms 1 and 2, atoms 1 and 3 and atoms 2 and 3). The average of these distances is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCES.tmp
d1: DISTANCES GROUP=1-3 MEAN
PRINT ARG=d1.mean
```

(See also PRINT)

The following input tells plumed to calculate all the distances between the atoms in GROUPA and the atoms in GROUPB. In other words the distances between atoms 1 and 2 and the distance between atoms 1 and 3. The number of distances more than 0.1 is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCES.tmp
d1: DISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.morethan
```

(See also PRINT switchingfunction)

Calculating minimum distances

To calculate and print the minimum distance between two groups of atoms you use the following commands

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCES.tmp
d1: DISTANCES GROUPA=1-10 GROUPB=11-20 MIN={BETA=500.}
PRINT ARG=d1.min FILE=colvar STRIDE=10
```

(see DISTANCES and PRINT)

In order to ensure that the minimum value has continuous derivatives we use the following function:

$$s = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$$

where β is a user specified parameter.

This input is used rather than a separate MINDIST colvar so that the same routine and the same input style can be used to calculate minimum coordination numbers (see COORDINATIONNUMBER), minimum angles (see ANGLES) and many other variables.

This new way of calculating mindist is part of plumed 2's multicolvar functionality. These special actions allow you to calculate multiple functions of a distribution of simple collective variables. As an example you can calculate the number of distances less than 1.0, the minimum distance, the number of distances more than 2.0 and the number of distances between 1.0 and 2.0 by using the following command:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DISTANCES.tmp
d1: DISTANCES ...
 GROUPA=1-10 GROUPB=11-20
 LESS_THAN={RATIONAL R_0=1.0}
 MORE_THAN={RATIONAL R_0=2.0}
 BETWEEN={GAUSSIAN LOWER=1.0 UPPER=2.0}
 MIN={BETA=500.}
...
PRINT ARG=d1.lessthan,d1.morethan,d1.between,d1.min FILE=colvar STRIDE=10
```

(see DISTANCES and PRINT)

A calculation performed this way is fast because the expensive part of the calculation - the calculation of all the distances - is only done once per step. Furthermore, it can be made faster by using the TOL keyword to discard those distance that make only a small contributions to the final values together with the NL_STRIDE keyword, which ensures that the distances that make only a small contribution to the final values aren't calculated at every step.

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the distances you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one distance will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.10 ENVIRONMENTSIMILARITY

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Measure how similar the environment around atoms is to that found in some reference crystal structure.

This CV was introduced in this article [27]. The starting point for the definition of the CV is the local atomic density around an atom. We consider an environment χ around this atom and we define the density by

$$\rho_\chi(\mathbf{r}) = \sum_{i\in\chi} \exp\left(-\frac{|\mathbf{r}_i-\mathbf{r}|^2}{2\sigma^2}\right),$$

where i runs over the neighbors in the environment χ, σ is a broadening parameter, and $\mathbf{r}_i$ are the coordinates of the neighbors relative to the central atom. We now define a reference environment or template χ_0 that contains n reference positions $\{\mathbf{r}_1^0,...,\mathbf{r}_n^0\}$ that describe, for instance, the nearest neighbors in a given lattice. σ is set using the SIGMA keyword and χ_0 is chosen with the CRYSTAL_STRUCTURE keyword. If only the SPECIES keyword is given then the atoms defined there will be the central and neighboring atoms. If instead the SPECIESA and SPECIESB keywords are given then SPECIESA determines the central atoms and SPECIESB the neighbors.

The environments χ and χ_0 are compared using the kernel,

$$k_{\chi_0}(\chi) = \int d\mathbf{r}\rho_\chi(\mathbf{r})\rho_{\chi_0}(\mathbf{r}).$$

Combining the two equations above and performing the integration analytically we obtain,

$$k_{\chi_0}(\chi) = \sum_{i\in\chi}\sum_{j\in\chi_0} \pi^{3/2}\sigma^3 \exp\left(-\frac{|\mathbf{r}_i-\mathbf{r}_j^0|^2}{4\sigma^2}\right).$$

The kernel is finally normalized,

$$\tilde{k}_{\chi_0}(\chi) = \frac{1}{n}\sum_{i\in\chi}\sum_{j\in\chi_0} \exp\left(-\frac{|\mathbf{r}_i-\mathbf{r}_j^0|^2}{4\sigma^2}\right),$$

such that $\tilde{k}_{\chi_0}(\chi_0)=1$. The above kernel is computed for each atom in the SPECIES or SPECIESA keywords. This quantity is a multicolvar so you can compute it for multiple atoms using a single PLUMED action and then compute

the average value for the atoms in your system, the number of atoms that have an $\tilde{k}_{\chi_0}$ value that is more that some target and so on.

The kernel can be generalized to crystal structures described as a lattice with a basis of more than one atom. In this case there is more than one type of environment. We consider the case of M environments $X = \chi_1, \chi_2, ..., \chi_M$ and we define the kernel through a best match strategy:

$$\tilde{k}_X(\chi) = \frac{1}{\lambda} \log \left(\sum_{l=1}^{M} \exp \left(\lambda \, \tilde{k}_{\chi_l}(\chi) \right) \right).$$

For a large enough λ this expression will select the largest $\tilde{k}_{\chi_l}(\chi)$ with $\chi_l \in X$. This approach can be used, for instance, to target the hexagonal closed packed (HCP keyword) or the diamond structure (DIAMOND keyword).

The CRYSTAL_STRUCTURE keyword can take the values SC (simple cubic), BCC (body centered cubic), FCC (face centered cubic), HCP (hexagonal closed pack), DIAMOND (cubic diamond), and CUSTOM (user defined). All options follow the same conventions as in the lattice command of LAMMPS. If a CRYSTAL_STRUCTURE other than CUSTOM is used, then the lattice constants have to be specified using the keyword LATTICE_CONS↩ TANTS. One value has to be specified for SC, BCC, FCC, and DIAMOND and two values have to be set for HCP (a and c lattice constants in that order).

If the CUSTOM option is used then the reference environments have to be specified by the user. The reference environments are specified in pdb files containing the distance vectors from the central atom to the neighbors. Make sure your PDB file is correctly formatted as explained in this page If only one reference environment is specified then the filename should be given as argument of the keyword REFERENCE. If instead several reference environments are given, then they have to be provided in separate pdb files and given as arguments of the keywords REFERENCE_1, REFERENCE_2, etc. If you have a reference crystal structure configuration you can use the Environment Finder app to determine the reference environments that you should use.

If multiple chemical species are involved in the calculation, it is possible to provide the atom types (names) both for atoms in the reference environments and in the simulation box. This information is provided in pdb files using the atom name field. The comparison between environments is performed taking into account whether the atom names match.

Examples

The following input calculates the ENVIRONMENTSIMILARITY kernel for 250 atoms in the system using the BCC atomic environment as target, and then calculates and prints the average value for this quantity.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ENVIRONMENTSIMILARITY.tmp
ENVIRONMENTSIMILARITY SPECIES=1-250 SIGMA=0.05 LATTICE_CONSTANTS=0.423 CRYSTAL_STRUCTURE=BCC MEAN LABEL=es

PRINT ARG=es.mean FILE=COLVAR
```

The next example compares the environments of the 96 selected atoms with a user specified reference environment. The reference environment is contained in the env1.pdb file. Once the kernel is computed the average and the number of atoms with a kernel larger than 0.5 are computed.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/ENVIRONMENTSIMILARITY.tmp
ENVIRONMENTSIMILARITY ...
 SPECIES=1-288:3
 SIGMA=0.05
 CRYSTAL_STRUCTURE=CUSTOM
 REFERENCE=env1.pdb
 LABEL=es
 MEAN
 MORE_THAN={RATIONAL R_0=0.5 NN=12 MM=24}
... ENVIRONMENTSIMILARITY

PRINT ARG=es.mean,es.morethan FILE=COLVAR
```

The next example is similar to the one above but in this case 4 reference environments are specified. Each reference environment is given in a separate pdb file.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/ENVIRONMENTSIMILARITY.tmp
ENVIRONMENTSIMILARITY ...
 SPECIES=1-288:3
 SIGMA=0.05
 CRYSTAL_STRUCTURE=CUSTOM
 REFERENCE_1=env1.pdb
 REFERENCE_2=env2.pdb
 REFERENCE_3=env3.pdb
 REFERENCE_4=env4.pdb
 LABEL=es
 MEAN
 MORE_THAN={RATIONAL R_0=0.5 NN=12 MM=24}
... ENVIRONMENTSIMILARITY

PRINT ARG=es.mean,es.morethan FILE=COLVAR
```

The following examples illustrates the use of pdb files to provide information about different chemical species:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/ENVIRONMENTSIMILARITY.tmp
ENVIRONMENTSIMILARITY ...
 SPECIES=1-6
 SIGMA=0.05
 CRYSTAL_STRUCTURE=CUSTOM
 REFERENCE=env.pdb
 LABEL=es
 MEAN
 MORE_THAN={RATIONAL R_0=0.5 NN=12 MM=24}
 ATOM_NAMES_FILE=atom-names.pdb
... ENVIRONMENTSIMILARITY
```

Here the file env.pdb is:

```
ATOM      1    O MOL     1      -2.239  -1.296  -0.917  1.00  0.00           O
ATOM      2    O MOL     1       0.000   0.000   2.751  1.00  0.00           O
```

where atoms are of type O, and the atom-names.pdb file is:

```
ATOM      1  O       X   1       0.000   2.593   4.126  0.00  0.00           O
ATOM      2  H       X   1       0.000   3.509   3.847  0.00  0.00           H
ATOM      3  H       X   1       0.000   2.635   5.083  0.00  0.00           H
ATOM      4  O       X   1       0.000   2.593  11.462  0.00  0.00           O
ATOM      5  H       X   1       0.000   3.509  11.183  0.00  0.00           H
ATOM      6  H       X   1       0.000   2.635  12.419  0.00  0.00           H
```

where atoms are of type O and H. In this case, all atoms are used as centers, but only neighbors of type O are taken into account.

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

SIGMA	(default=0.1) Broadening parameter
CRYSTAL_STRUCTURE	(default=FCC) Targeted crystal structure. Options are: SC: simple cubic, BCC: body center cubic, FCC: face centered cubic, HCP: hexagonal closed pack, DIAMOND: cubic diamond, CUSTOM: user defined
LAMBDA	(default=100) Lambda parameter

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LATTICE_CONSTANTS	Lattice constants. Two comma separated values for HCP, one value for all other CRYSTAL_STRUCTURES.
REFERENCE	PDB file with relative distances from central atom. Use this keyword if you are targeting a single reference environment.
REFERENCE_	PDB files with relative distances from central atom. Each file corresponds to one template. Use these keywords if you are targeting more than one reference environment.. You can use multiple instances of this keyword i.e. REFERENCE_1, REFERENCE_2, REFERENCE_3...
ATOM_NAMES_FILE	PDB file with atom names for all atoms in SPECIES. Atoms in reference environments will be compared only if atom names match.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...

MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.11 FCCUBIC

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Measure how similar the environment around atoms is to that found in a FCC structure.

This CV was introduced in this article [28] and again in this article [29] This CV essentially determines whether the environment around any given atom is similar to that found in the FCC structure or not. The function that is used to make this determination is as follows:

$$s_i = \frac{\sum_{i\neq j}\sigma(r_{ij})\left\{a\left[\frac{(x_{ij}y_{ij})^4+(x_{ij}z_{ij})^4+(y_{ij}z_{ij})^4}{r_{ij}^8} - \frac{\alpha(x_{ij}y_{ij}z_{ij})^4}{r_{ij}^{12}}\right]+b\right\}}{\sum_{i\neq j}\sigma(r_{ij})}$$

In this expression x_{ij}, y_{ij} and z_{ij} are the x, y and z components of the vector connecting atom i to atom j and r_{ij} is the magnitude of this vector. $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atom i and atom j and its inclusion in the numerator and the denominator of the above expression as well as the fact that we are summing over all of the other atoms in the system ensures that we are calculating an average of the function of x_{ij}, y_{ij} and z_{ij} for the atoms in the first coordination sphere around atom i. Lastly, α is a parameter that can be set by the user, which by default is equal to three. The values of a and b are calculated from α using:

$$a = \frac{80080}{2717 + 16\alpha} \qquad \text{and} \qquad b = \frac{16(\alpha - 143)}{2717 + 16\alpha}$$

This quantity is once again a multicolvar so you can compute it for multiple atoms using a single PLUMED action and then compute the average value for the atoms in your system, the number of atoms that have an s_i value that is more that some target and so on. Notice also that you can rotate the reference frame if you are using a non-standard unit cell.

Examples

The following input calculates the FCCUBIC parameter for the 64 atoms in the system and then calculates and prints the average value for this quantity.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FCCUBIC.tmp
FCCUBIC SPECIES=1-64 SWITCH={RATIONAL D_0=3.0 R_0=1.5} MEAN LABEL=d
PRINT ARG=d.* FILE=colv
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function
PHI	(default=0.0) The Euler rotational angle phi
THETA	(default=0.0) The Euler rotational angle theta
PSI	(default=0.0) The Euler rotational angle psi
ALPHA	(default=3.0) The alpha parameter of the angular function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
UNORMALIZED	(default=off) calculate the sum of the components of the vector rather than the mean
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...

MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.12 HBPAMM_SH

	This is part of the pamm module
	It is only available if you configure PLUMED with ./configure –enable-modules=pamm . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Number of HBPAMM hydrogen bonds formed by each hydrogen atom in the system

Examples

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The atoms involved can be specified using

HYDROGENS	The list of hydrogen atoms that can form part of a hydrogen bond. The atoms must be specified using a comma separated list.

SITES	The list of atoms which can be part of a hydrogen bond. When this command is used the set of atoms that can donate a hydrogen bond is assumed to be the same as the set of atoms that can form hydrogen bonds. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices

Or alternatively by using

DONORS	The list of atoms which can donate a hydrogen bond. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices
ACCEPTORS	The list of atoms which can accept a hydrogen bond. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices

Compulsory keywords

CLUSTERS	the name of the file that contains the definitions of all the kernels for PAMM. You can use multiple instances of this keyword i.e. CLUSTERS1, CLUSTERS2, CLUSTERS3...
REGULARISE	(default=0.001) don't allow the denominator to be smaller then this value

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...

MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SUM2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

5.5.13 INPLANEDISTANCES

This is part of the multicolvar module

Calculate distances in the plane perpendicular to an axis

Each quantity calculated in this CV uses the positions of two atoms, this indices of which are specified using the VECTORSTART and VECTOREND keywords, to specify the orientation of a vector, $\mathbf{n}$. The perpendicular distance between this vector and the position of some third atom is then computed using:

$$x_j = |\mathbf{r}_j| \sin(\theta_j)$$

where $\mathbf{r}_j$ is the distance between one of the two atoms that define the vector $\mathbf{n}$ and a third atom (atom j) and where θ_j is the angle between the vector $\mathbf{n}$ and the vector $\mathbf{r}_j$. The x_j values for each of the atoms specified using the GROUP keyword are calculated. Keywords such as MORE_THAN and LESS_THAN can then be used to calculate the number of these quantities that are more or less than a given cutoff.

Examples

The following input can be used to calculate the number of atoms that have indices greater than 3 and less than 101 that are within a cylinder with a radius of 0.3 nm that has its long axis aligned with the vector connecting atoms 1 and 2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INPLANEDISTANCES.tmp
d1: INPLANEDISTANCES VECTORSTART=1 VECTOREND=2 GROUP=3-100 LESS_THAN={RATIONAL D_0=0.2 R_0=0.1}
PRINT ARG=d1.lessthan FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

VECTORSTART	The first atom position that is used to define the normal to the plane of interest. For more information on how to specify lists of atoms see Groups and Virtual Atoms
VECTOREND	The second atom position that is used to define the normal to the plane of interest. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Or alternatively by using

GROUP	The set of atoms for which you wish to calculate the in plane distance

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

TIMINGS	(default=off) output information on the timings of the various parts of the calculation
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...

MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.14 MOLECULES

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the vectors connecting a pair of atoms in order to represent the orientation of a molecule.

At its simplest this command can be used to calculate the average length of an internal vector in a collection of different molecules. When used in conjunction with MutiColvarFunctions in can be used to do a variety of more complex tasks.

Examples

The following input tells plumed to calculate the distances between two of the atoms in a molecule. This is done for the same set of atoms four different molecules and the average separation is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MOLECULES.tmp
MOLECULES MOL1=1,2 MOL2=3,4 MOL3=5,6 MOL4=7,8 MEAN LABEL=mm
PRINT ARG=mm.mean FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of vectors calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. All of them can be calculated multiple times

with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set the label for the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name. In addition, you can calculate all of these scalar functions for one particular component of the calculated vector by making use of the COMPONENT keyword. The first component is used to refer to the norm of the vector. The individual components can then be referenced using the numbers 2, 3, and so on. So as an example MEAN1={COMPON↩ ENT=1} calculates the average vector norm. MEAN2={COMPONENT=2} by contrast calculates the mean for all of the first components of the vectors.

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean

The atoms involved can be specified using

MOL	The numerical indices of the atoms in the molecule. The orientation of the molecule is equal to the vector connecting the first two atoms specified. If a third atom is specified its position is used to specify where the molecule is. If a third atom is not present the molecule is assumed to be at the center of the vector connecting the first two atoms.. You can use multiple instances of this keyword i.e. MOL1, MOL2, MOL3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...

5.5.15 PLANES

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the plane perpendicular to two vectors in order to represent the orientation of a planar molecule.

Examples

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of vectors calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. All of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set the label for the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name. In addition, you can calculate all of these scalar functions for one particular component of the calculated vector by making use of the COMPONENT keyword. The first component is used to refer to the norm of the vector. The individual components can then be referenced using the numbers 2, 3, and so on. So as an example MEAN1={COMPONENT=1} calculates the average vector norm. MEAN2={COMPONENT=2} by contrast calculates the mean for all of the first components of the vectors.

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean

The atoms involved can be specified using

MOL	The numerical indices of the atoms in the molecule. If three atoms are specified the orientation of the molecule is taken as the normal to the plane containing the vector connecting the first and second atoms and the vector connecting the second and third atoms. If four atoms are specified the orientation of the molecule is taken as the normal to the plane containing the vector connecting the first and second atoms and the vector connecting the third and fourth atoms. The molecule is always assumed to lie at the geometric center for the three/four atoms.. You can use multiple instances of this keyword i.e. MOL1, MOL2, MOL3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

TIMINGS	(default=off) output information on the timings of the various parts of the calculation
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...

5.5.16 Q3

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate 3rd order Steinhardt parameters.

The 3rd order Steinhardt parameters allow us to measure the degree to which the first coordination shell around an atom is ordered. The Steinhardt parameter for atom, i is complex vector whose components are calculated using the following formula:

$$q_{3m}(i) = \frac{\sum_j \sigma(r_{ij}) Y_{3m}(\mathbf{r}_{ij})}{\sum_j \sigma(r_{ij})}$$

where Y_{3m} is one of the 3rd order spherical harmonics so m is a number that runs from -3 to $+3$. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. The parameters of this function should be set so that it the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise.

The Steinhardt parameters can be used to measure the degree of order in the system in a variety of different ways. The simplest way of measuring whether or not the coordination sphere is ordered is to simply take the norm of the above vector i.e.

$$Q_3(i) = \sqrt{\sum_{m=-3}^{3} q_{3m}(i)^* q_{3m}(i)}$$

This norm is small when the coordination shell is disordered and larger when the coordination shell is ordered. Furthermore, when the keywords LESS_THAN, MIN, MAX, HISTOGRAM, MEAN and so on are used with this colvar it is the distribution of these normed quantities that is investigated.

Other measures of order can be taken by averaging the components of the individual q_3 vectors individually or by taking dot products of the q_3 vectors on adjacent atoms. More information on these variables can be found in the documentation for LOCAL_Q3, LOCAL_AVERAGE and NLINKS.

Examples

The following command calculates the average Q3 parameter for the 64 atoms in a box of Lennard Jones and prints this quantity to a file called colvar:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q3.tmp
Q3 SPECIES=1-64 D_0=1.3 R_0=0.2 MEAN LABEL=q3
PRINT ARG=q3.mean FILE=colvar
```

The following command calculates the histogram of Q3 parameters for the 64 atoms in a box of Lennard Jones and prints these quantities to a file called colvar:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q3.tmp
Q3 SPECIES=1-64 D_0=1.3 R_0=0.2 HISTOGRAM={GAUSSIAN LOWER=0.0 UPPER=1.0 NBINS=20 SMEAR=0.1} LABEL=q3
PRINT ARG=q3.* FILE=colvar
```

The following command could be used to measure the Q3 parameters that describe the arrangement of chlorine ions around the sodium atoms in sodium chloride. The imagined system here is composed of 64 NaCl formula units and the atoms are arranged in the input with the 64 Na^+ ions followed by the 64 Cl^- ions. Once again the average Q3 parameter is calculated and output to a file called colvar

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q3.tmp
Q3 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN LABEL=q3
PRINT ARG=q3.mean FILE=colvar
```

If you simply want to examine the values of the Q3 parameters for each of the atoms in your system you can do so by exploiting the command DUMPMULTICOLVAR as shown in the example below. The following output file will output a file in an extended xyz format called q3.xyz for each frame of the analyzed MD trajectory. The first column in this file will contain a dummy name for each of the atoms, columns 2-4 will then contain the x, y and z positions of the atoms, column 5 will contain the value of the Q3 parameter, columns 6-12 will contain the real parts of the director of the q_{3m} vector while columns 12-19 will contain the imaginary parts of this director.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q3.tmp
q3: Q3 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN
DUMPMULTICOLVAR DATA=q3 FILE=q3.xyz
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of vectors calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. All of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set the label for the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name. In addition, you can calculate all of these scalar functions for one particular component of the calculated vector by making use of the COMPONENT keyword. The first component is used to refer to the norm of the vector. The individual components can then be referenced using the numbers 2, 3, and so on. So as an example MEAN1={COMPON↩ ENT=1} calculates the average vector norm. MEAN2={COMPONENT=2} by contrast calculates the mean for all of the first components of the vectors.

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=12) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.17 Q4

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate fourth order Steinhardt parameters.

The fourth order Steinhardt parameters allow us to measure the degree to which the first coordination shell around an atom is ordered. The Steinhardt parameter for atom, i is complex vector whose components are calculated using the following formula:

$$q_{4m}(i) = \frac{\sum_j \sigma(r_{ij}) Y_{4m}(\mathbf{r}_{ij})}{\sum_j \sigma(r_{ij})}$$

where Y_{4m} is one of the fourth order spherical harmonics so m is a number that runs from -4 to $+4$. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. The parameters of this function

should be set so that it the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise.

The Steinhardt parameters can be used to measure the degree of order in the system in a variety of different ways. The simplest way of measuring whether or not the coordination sphere is ordered is to simply take the norm of the above vector i.e.

$$Q_4(i) = \sqrt{\sum_{m=-4}^{4} q_{4m}(i)^* q_{4m}(i)}$$

This norm is small when the coordination shell is disordered and larger when the coordination shell is ordered. Furthermore, when the keywords LESS_THAN, MIN, MAX, HISTOGRAM, MEAN and so on are used with this colvar it is the distribution of these normed quantities that is investigated.

Other measures of order can be taken by averaging the components of the individual q_4 vectors individually or by taking dot products of the q_4 vectors on adjacent atoms. More information on these variables can be found in the documentation for LOCAL_Q4, LOCAL_AVERAGE and NLINKS.

Examples

The following command calculates the average Q4 parameter for the 64 atoms in a box of Lennard Jones and prints this quantity to a file called colvar:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q4.tmp
Q4 SPECIES=1-64 D_0=1.3 R_0=0.2 MEAN LABEL=q4
PRINT ARG=q4.mean FILE=colvar
```

The following command calculates the histogram of Q4 parameters for the 64 atoms in a box of Lennard Jones and prints these quantities to a file called colvar:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q4.tmp
Q4 SPECIES=1-64 D_0=1.3 R_0=0.2 HISTOGRAM={GAUSSIAN LOWER=0.0 UPPER=1.0 NBINS=20 SMEAR=0.1} LABEL=q4
PRINT ARG=q4.* FILE=colvar
```

The following command could be used to measure the Q4 parameters that describe the arrangement of chlorine ions around the sodium atoms in sodium chloride. The imagined system here is composed of 64 NaCl formula units and the atoms are arranged in the input with the 64 Na^+ ions followed by the 64 Cl^- ions. Once again the average Q4 parameter is calculated and output to a file called colvar

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q4.tmp
Q4 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN LABEL=q4
PRINT ARG=q4.mean FILE=colvar
```

If you simply want to examine the values of the Q4 parameters for each of the atoms in your system you can do so by exploiting the command DUMPMULTICOLVAR as shown in the example below. The following output file will output a file in an extended xyz format called q$.xyz for each frame of the analyzed MD trajectory. The first column in this file will contain a dummy name for each of the atoms, columns 2-4 will then contain the x, y and z positions of the atoms, column 5 will contain the value of the Q4 parameter, columns 6-15 will contain the real parts of the director of the q_{6m} vector while columns 15-24 will contain the imaginary parts of this director.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q4.tmp
q4: Q4 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN
DUMPMULTICOLVAR DATA=q4 FILE=q4.xyz
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of vectors calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. All of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set the label for the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name. In addition, you can calculate all of these scalar functions for one particular component of the calculated vector by making use of the COMPONENT keyword. The first component is used to refer to the norm of the vector. The individual components can then be referenced using the numbers 2, 3, and so on. So as an example MEAN1={COMPON↩ ENT=1} calculates the average vector norm. MEAN2={COMPONENT=2} by contrast calculates the mean for all of the first components of the vectors.

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=12) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...

LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.18 Q6

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate sixth order Steinhardt parameters.

The sixth order Steinhardt parameters allow us to measure the degree to which the first coordination shell around an atom is ordered. The Steinhardt parameter for atom, i is complex vector whose components are calculated using the following formula:

$$q_{6m}(i) = \frac{\sum_j \sigma(r_{ij}) Y_{6m}(\mathbf{r}_{ij})}{\sum_j \sigma(r_{ij})}$$

where Y_{6m} is one of the sixth order spherical harmonics so m is a number that runs from -6 to $+6$. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. The parameters of this function should be set so that it the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise.

The Steinhardt parameters can be used to measure the degree of order in the system in a variety of different ways. The simplest way of measuring whether or not the coordination sphere is ordered is to simply take the norm of the above vector i.e.

$$Q_6(i) = \sqrt{\sum_{m=-6}^{6} q_{6m}(i)^* q_{6m}(i)}$$

This norm is small when the coordination shell is disordered and larger when the coordination shell is ordered. Furthermore, when the keywords LESS_THAN, MIN, MAX, HISTOGRAM, MEAN and so on are used with this colvar it is the distribution of these normed quantities that is investigated.

Other measures of order can be taken by averaging the components of the individual q_6 vectors individually or by taking dot products of the q_6 vectors on adjacent atoms. More information on these variables can be found in the documentation for LOCAL_Q6, LOCAL_AVERAGE and NLINKS.

Examples

The following command calculates the average Q6 parameter for the 64 atoms in a box of Lennard Jones and prints this quantity to a file called colvar:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q6.tmp
Q6 SPECIES=1-64 D_0=1.3 R_0=0.2 MEAN LABEL=q6
PRINT ARG=q6.mean FILE=colvar
```

The following command calculates the histogram of Q6 parameters for the 64 atoms in a box of Lennard Jones and prints these quantities to a file called colvar:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q6.tmp
Q6 SPECIES=1-64 D_0=1.3 R_0=0.2 HISTOGRAM={GAUSSIAN LOWER=0.0 UPPER=1.0 NBINS=20 SMEAR=0.1} LABEL=q6
PRINT ARG=q6.* FILE=colvar
```

The following command could be used to measure the Q6 parameters that describe the arrangement of chlorine ions around the sodium atoms in sodium chloride. The imagined system here is composed of 64 NaCl formula units and the atoms are arranged in the input with the 64 Na^+ ions followed by the 64 Cl^- ions. Once again the average Q6 parameter is calculated and output to a file called colvar

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q6.tmp
Q6 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN LABEL=q6
PRINT ARG=q6.mean FILE=colvar
```

If you simply want to examine the values of the Q6 parameters for each of the atoms in your system you can do so by exploiting the command DUMPMULTICOLVAR as shown in the example below. The following output file will output a file in an extended xyz format called q6.xyz for each frame of the analyzed MD trajectory. The first column in this file will contain a dummy name for each of the atoms, columns 2-4 will then contain the x, y and z positions of the atoms, column 5 will contain the value of the Q6 parameter, columns 6-19 will contain the real parts of the director of the q_{6m} vector while columns 20-33 will contain the imaginary parts of this director.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/Q6.tmp
q6: Q6 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN
DUMPMULTICOLVAR DATA=q6 FILE=q6.xyz
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of vectors calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. All of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set the label for the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name. In addition, you can calculate all of these scalar functions for one particular component of the calculated vector by making use of the COMPONENT keyword. The first component is used to refer to the norm of the vector. The individual components can then be referenced using the numbers 2, 3, and so on. So as an example MEAN1={COMPON↩ ENT=1} calculates the average vector norm. MEAN2={COMPONENT=2} by contrast calculates the mean for all of the first components of the vectors.

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=12) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.19 SIMPLECUBIC

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate whether or not the coordination spheres of atoms are arranged as they would be in a simple cubic structure.

We can measure how similar the environment around atom i is to a simple cubic structure is by evaluating the following quantity:

$$s_i = \frac{\sum_{i\neq j}\sigma(r_{ij})\left[\frac{x_{ij}^4+y_{ij}^4+z_{ij}^4}{r_{ij}^4}\right]}{\sum_{i\neq j}\sigma(r_{ij})}$$

In this expression x_{ij}, y_{ij} and z_{ij} are the x, y and z components of the vector connecting atom i to atom j and r_{ij} is the magnitude of this vector. $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atom i and atom

j and its inclusion in the numerator and the denominator of the above expression as well as the fact that we are summing over all of the other atoms in the system ensures that we are calculating an average of the function of x_{ij}, y_{ij} and z_{ij} for the atoms in the first coordination sphere around atom i. This quantity is once again a multicolvar so you can compute it for multiple atoms using a single PLUMED action and then compute the average value for the atoms in your system, the number of atoms that have an s_i value that is more that some target and so on. Notice also that you can rotate the reference frame if you are using a non-standard unit cell.

Examples

The following input tells plumed to calculate the simple cubic parameter for the atoms 1-100 with themselves. The mean value is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SIMPLECUBIC.tmp
SIMPLECUBIC SPECIES=1-100 R_0=1.0 MEAN
```

The following input tells plumed to look at the ways atoms 1-100 are within 3.0 are arranged about atoms from 101-110. The number of simple cubic parameters that are greater than 0.8 is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SIMPLECUBIC.tmp
SIMPLECUBIC SPECIESA=101-110 SPECIESB=1-100 R_0=3.0 MORE_THAN={RATIONAL R_0=0.8 NN=6 MM=12 D_0=0}
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function
PHI	(default=0.0) The Euler rotational angle phi
THETA	(default=0.0) The Euler rotational angle theta
PSI	(default=0.0) The Euler rotational angle psi

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
UNORMALIZED	(default=off) calculate the sum of the components of the vector rather than the mean
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.20 TETRAHEDRAL

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the degree to which the environment about ions has a tetrahedral order.

We can measure the degree to which the atoms in the first coordination shell around any atom, i is is arranged like a tetrahedron using the following function.

$$s(i) = \frac{1}{\sum_j \sigma(r_{ij})}\sum_j \sigma(r_{ij})\left[\frac{(x_{ij}+y_{ij}+z_{ij})^3}{r_{ij}^3}+\frac{(x_{ij}-y_{ij}-z_{ij})^3}{r_{ij}^3}+\frac{(-x_{ij}+y_{ij}-z_{ij})^3}{r_{ij}^3}+\frac{(-x_{ij}-y_{ij}+z_{ij})^3}{r_{ij}^3}\right]$$

Here r_{ij} is the magnitude of the vector connecting atom i to atom j and x_{ij}, y_{ij} and z_{ij} are its three components. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. The parameters of this function should be set so that the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise.

Examples

The following command calculates the average value of the TETRAHEDRAL parameter for a set of 64 atoms all of the same type and outputs this quantity to a file called colvar.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TETRAHEDRAL.tmp
tt: TETRAHEDRAL SPECIES=1-64 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN
PRINT ARG=tt.mean FILE=colvar
```

The following command calculates the number of TETRAHEDRAL parameters that are greater than 0.8 in a set of 10 atoms. In this calculation it is assumed that there are two atom types A and B and that the first coordination sphere of the 10 atoms of type A contains atoms of type B. The formula above is thus calculated for ten different A atoms and within it the sum over j runs over 40 atoms of type B that could be in the first coordination sphere.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TETRAHEDRAL.tmp
tt: TETRAHEDRAL SPECIESA=1-10 SPECIESB=11-40 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MORE_THAN={RATIONAL R_0=0.8}
PRINT ARG=tt.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.

mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function
PHI	(default=0.0) The Euler rotational angle phi
THETA	(default=0.0) The Euler rotational angle theta
PSI	(default=0.0) The Euler rotational angle psi

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
UNORMALIZED	(default=off) calculate the sum of the components of the vector rather than the mean
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.21 TORSIONS

This is part of the multicolvar module

Calculate whether or not a set of torsional angles are within a particular range.

Examples

The following provides an example of the input for the TORSIONS command

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TORSIONS.tmp
TORSIONS ...
ATOMS1=168,170,172,188
ATOMS2=170,172,188,190
ATOMS3=188,190,192,230
BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
LABEL=ab
... TORSIONS
PRINT ARG=ab.* FILE=colvar STRIDE=10
```

Writing out the atoms involved in all the torsion angles in this way can be rather tedious. Thankfully if you are working with protein you can avoid this by using the MOLINFO command. PLUMED uses the pdb file that you provide to this command to learn about the topology of the protein molecule. This means that you can specify torsion angles using the following syntax:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TORSIONS.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
TORSIONS ...
ATOMS1=@phi-3
ATOMS2=@psi-3
ATOMS3=@phi-4
BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
LABEL=ab
... TORSIONS
PRINT ARG=ab.* FILE=colvar STRIDE=10
```

Here, @phi-3 tells plumed that you would like to calculate the ϕ angle in the third residue of the protein. Similarly @psi-4 tells plumed that you want to calculate the ψ angle of the fourth residue of the protein.

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the torsion angles you wish to calculate. Keywords like ATOMS1, A↩ TOMS2, ATOMS3,... should be listed and one torsion will be calculated for each ATOM keyword you specify (all ATOM keywords should provide the indices of four atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...

5.5.22 XANGLES

This is part of the multicolvar module

Calculate the angles between the vector connecting two atoms and the x axis.

Examples

The following input tells plumed to calculate the angles between the x-axis and the vector connecting atom 3 to atom 5 and between the x-axis and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XANGLES.tmp
XANGLES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one angle will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...

LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.23 XDISTANCES

This is part of the multicolvar module

Calculate the x components of the vectors connecting one or many pairs of atoms. You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.

Examples

The following input tells plumed to calculate the x-component of the vector connecting atom 3 to atom 5 and the x-component of the vector connecting atom 1 to atom 2. The minimum of these two quantities is then printed

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XDISTANCES.tmp
d1: XDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1}
PRINT ARG=d1.min
```

(See also PRINT).

The following input tells plumed to calculate the x-component of the vector connecting atom 3 to atom 5 and the x-component of the vector connecting atom 1 to atom 2. The number of values that are less than 0.1nm is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XDISTANCES.tmp
d1: XDISTANCES ATOMS1=3,5 ATOMS2=1,2 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lessthan
```

(See also PRINT switchingfunction).

The following input tells plumed to calculate the x-components of all the distinct vectors that can be created between atoms 1, 2 and 3 (i.e. the vectors between atoms 1 and 2, atoms 1 and 3 and atoms 2 and 3). The average of these quantities is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XDISTANCES.tmp
d1: XDISTANCES GROUP=1-3 MEAN
PRINT ARG=d1.mean
```

(See also PRINT)

The following input tells plumed to calculate all the vectors connecting the the atoms in GROUPA to the atoms in GROUPB. In other words the vector between atoms 1 and 2 and the vector between atoms 1 and 3. The number of values more than 0.1 is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XDISTANCES.tmp
d1: XDISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.morethan
```

(See also PRINT switchingfunction)

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the distances you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one distance will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.24 XYDISTANCES

This is part of the multicolvar module

Calculate distance between a pair of atoms neglecting the z-component.

You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.

Examples

The following input tells plumed to calculate the projection of the length of the vector connecting atom 3 to atom 5 projected in the xy-plane and the projection of the length of the vector the vector connecting atom 1 to atom 2 in the xy-plane. The minimum of these two quantities is then printed

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XYDISTANCES.tmp
d1: XYDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1}
PRINT ARG=d1.min
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the distances you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one distance will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.25 XYTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the x axis from the positive y direction.

Examples

The following input tells plumed to calculate the angle around the x direction between the positive y-axis and the vector connecting atom 3 to atom 5 and the angle around the x direction between the positive y axis and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XYTORSIONS.tmp
d1: XYTORSIONS ATOMS1=3,5 ATOMS2=1,2 BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
PRINT ARG=d1.between
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the torsion angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one torsion will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...

MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.26 XZDISTANCES

This is part of the multicolvar module

Calculate distance between a pair of atoms neglecting the y-component.

You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.

Examples

The following input tells plumed to calculate the projection of the length of the vector connecting atom 3 to atom 5 projected in the xz-plane and the projection of the length of the vector the vector connecting atom 1 to atom 2 in the xz-plane. The minimum of these two quantities is then printed

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XZDISTANCES.tmp
d1: XZDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1}
PRINT ARG=d1.min
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the distances you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one distance will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.27 XZTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the x axis from the positive z direction.

Examples

The following input tells plumed to calculate the angle around the x direction between the positive z-axis and the vector connecting atom 3 to atom 5 and the angle around the x direction between the positive z direction and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/XZTORSIONS.tmp
d1: XZTORSIONS ATOMS1=3,5 ATOMS2=1,2 BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
PRINT ARG=d1.*
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the torsion angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one torsion will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...

MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.28 YANGLES

This is part of the multicolvar module

Calculate the angles between the vector connecting two atoms and the y axis.

Examples

The following input tells plumed to calculate the angles between the y-axis and the vector connecting atom 3 to atom 5 and between the y-axis and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YANGLES.tmp
YANGLES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one angle will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.29 YDISTANCES

This is part of the multicolvar module

Calculate the y components of the vectors connecting one or many pairs of atoms. You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.

Examples

The following input tells plumed to calculate the y-component of the vector connecting atom 3 to atom 5 and the y-component of the vector connecting atom 1 to atom 2. The minimum of these two quantities is then printed

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YDISTANCES.tmp
d1: YDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1}
PRINT ARG=d1.min
```

(See also PRINT).

The following input tells plumed to calculate the y-component of the vector connecting atom 3 to atom 5 and the y-component of the vector connecting atom 1 to atom 2. The number of values that are less than 0.1nm is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YDISTANCES.tmp
d1: YDISTANCES ATOMS1=3,5 ATOMS2=1,2 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lessthan
```

(See also PRINT switchingfunction).

The following input tells plumed to calculate the y-components of all the distinct vectors that can be created between atoms 1, 2 and 3 (i.e. the vectors between atoms 1 and 2, atoms 1 and 3 and atoms 2 and 3). The average of these quantities is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YDISTANCES.tmp
d1: YDISTANCES GROUP=1-3 MEAN
PRINT ARG=d1.mean
```

(See also PRINT)

The following input tells plumed to calculate all the vectors connecting the the atoms in GROUPA to the atoms in GROUPB. In other words the vector between atoms 1 and 2 and the vector between atoms 1 and 3. The number of values more than 0.1 is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YDISTANCES.tmp
d1: YDISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.morethan
```

(See also PRINT switchingfunction)

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the distances you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one distance will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.30 YXTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the y axis from the positive x direction.

Examples

The following input tells plumed to calculate the angle around the y direction between the positive x-direction and the vector connecting atom 3 to atom 5 and the angle around the y direction between the positive x axis and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YXTORSIONS.tmp
d1: YXTORSIONS ATOMS1=3,5 ATOMS2=1,2 BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
PRINT ARG=d1.*
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the torsion angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one torsion will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.31 YZDISTANCES

This is part of the multicolvar module

Calculate distance between a pair of atoms neglecting the x-component.

You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.

Examples

The following input tells plumed to calculate the projection of the length of the vector connecting atom 3 to atom 5 in the yz-plane and the projection of the length of the vector the vector connecting atom 1 to atom 2 in the yz-plane. The minimum of these two quantities is then printed

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YZDISTANCES.tmp
d1: YZDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1}
PRINT ARG=d1.min
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the distances you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one distance will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.32 YZTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the y axis from the positive z direction.

Examples

The following input tells plumed to calculate the angle around the y direction between the positive z-direction and the vector connecting atom 3 to atom 5 and the angle around the y direction between the positive z direction and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/YZTORSIONS.tmp
d1: YZTORSIONS ATOMS1=3,5 ATOMS2=1,2 BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
PRINT ARG=d1.*
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the torsion angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one torsion will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.33 ZANGLES

	This is part of the multicolvar module

Calculate the angles between the vector connecting two atoms and the z axis.

Examples

The following input tells plumed to calculate the angles between the z-axis and the vector connecting atom 3 to atom 5 and between the z-axis and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ZANGLES.tmp
ZANGLES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one angle will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.34 ZDISTANCES

This is part of the multicolvar module

Calculate the z components of the vectors connecting one or many pairs of atoms. You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.

Examples

The following input tells plumed to calculate the z-component of the vector connecting atom 3 to atom 5 and the z-component of the vector connecting atom 1 to atom 2. The minimum of these two quantities is then printed

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ZDISTANCES.tmp
d1: ZDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1}
PRINT ARG=d1.min
```

(See also PRINT).

The following input tells plumed to calculate the z-component of the vector connecting atom 3 to atom 5 and the z-component of the vector connecting atom 1 to atom 2. The number of values that are less than 0.1nm is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ZDISTANCES.tmp
d1: ZDISTANCES ATOMS1=3,5 ATOMS2=1,2 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lessthan
```

(See also PRINT switchingfunction).

The following input tells plumed to calculate the z-components of all the distinct vectors that can be created between atoms 1, 2 and 3 (i.e. the vectors between atoms 1 and 2, atoms 1 and 3 and atoms 2 and 3). The average of these quantities is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ZDISTANCES.tmp
d1: ZDISTANCES GROUP=1-3 MEAN
PRINT ARG=d1.mean
```

(See also PRINT)

The following input tells plumed to calculate all the vectors connecting the the atoms in GROUPA to the atoms in GROUPB. In other words the vector between atoms 1 and 2 and the vector between atoms 1 and 3. The number of values more than 0.1 is then printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ZDISTANCES.tmp
d1: ZDISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.morethan
```

(See also PRINT switchingfunction)

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the distances you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one distance will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...

HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.35 ZXTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the z axis from the positive x direction.

Examples

The following input tells plumed to calculate the angle around the z direction between the positive x-direction and the vector connecting atom 3 to atom 5 and the angle around the z direction between the positive x-direction and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ZXTORSIONS.tmp
d1: ZXTORSIONS ATOMS1=3,5 ATOMS2=1,2 BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
PRINT ARG=d1.*
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the torsion angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one torsion will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...

MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.36 ZYTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the z axis from the positive y direction.

Examples

The following input tells plumed to calculate the angle around the z direction between the positive y-axis and the vector connecting atom 3 to atom 5 and the angle around the z direction between the positive y axis and the vector connecting atom 1 to atom 2. The minimum of these two quantities is then output

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ZYTORSIONS.tmp
d1: ZYTORSIONS ATOMS1=3,5 ATOMS2=1,2 BETWEEN={GAUSSIAN LOWER=0 UPPER=pi SMEAR=0.1}
PRINT ARG=d1.*
```

(See also PRINT).

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

The atoms involved can be specified using

ATOMS	the atoms involved in each of the torsion angles you wish to calculate. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one torsion will be calculated for each ATOM keyword you specify (all ATOM keywords should specify the indices of two atoms). The eventual number of quantities calculated by this action will depend on what functions of the distribution you choose to calculate.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Or alternatively by using

GROUP	Calculate the distance between each distinct pair of atoms in the group

Or alternatively by using

GROUPA	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjunction with GROUPA.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...

MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
SWITCH	A switching function that ensures that only angles are only computed when atoms are within are within a certain fixed cutoff. The following provides information on the switchingfunction that are available.

5.5.37 MFILTER_BETWEEN

This is part of the multicolvar module

This action can be used to filter the colvar values calculated by a MultiColvar so that one can compute the mean and so on for only those multicolvars within a certain range.

This action can be used to create a dynamic group of atom based on the value of a multicolvar. In this action a multicolvar is within the dynamic group if its value lies in a particular range. In actuality a weight, w_i is ascribed to each colvar, s_i calculated by a multicolvar and this weight measures the degree to which a colvar is a member of the group. This weight is calculated using a histogrambead so it is given by:

$$w_i = \int_a^b K\left(\frac{s - s_i}{w}\right)$$

where a, b and w are parameters. If one calculates a function of the set of multicolvars these weights are included in the calculation. As such if one calculates the MEAN, μ of a filtered multicolvar what is computed is the following:

$$\mu = \frac{\sum_i w_i s_i}{\sum_i w_i}$$

One is thus calculating the mean for those colvars that are within the range of interest.

Examples

The example shown below calculates the mean for those distances that are between 0 and 3 nm in length

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MFILTER_BETWEEN.tmp
DISTANCES GROUPA=1 GROUPB=2-50 MEAN LABEL=d1
MFILTER_BETWEEN DATA=d1 LOWER=0 UPPER=3.0 SMEAR=0.0001 MEAN LABEL=d4
```

More complicated things can be done by using the label of a filter as input to a new multicolvar as shown in the example below. Here the coordination numbers of all atoms are computed. The atoms with a coordination number between 4 and 6 are then identified using the filter. This reduced list of atoms is then used as input to a second coordination number calculation. This second coordination number thus measures the number of atoms 4-6 coordinated atoms each of the 4-6 coordination atoms is bound to.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MFILTER_BETWEEN.tmp
c1: COORDINATIONNUMBER SPECIES=1-150 SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
cf: MFILTER_BETWEEN DATA=c1 LOWER=4 UPPER=6 SMEAR=0.5 LOWMEM
c2: COORDINATIONNUMBER SPECIES=cf SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0} MORE_THAN={RATIONAL D_0=2.0 R_0=0.1}
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
LOWER	the lower boundary for the range of interest
UPPER	the upper boundary for the range of interest
SMEAR	(default=0.5) the amount by which to smear the value for kernel density estimation

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BEAD	This keywords is used if you want to employ an alternative to the function defined above. The following provides information on the histogrambead that are available. When this keyword is present you no longer need the LOWER, U↩ PPER and SMEAR keywords.

5.5.38 MFILTER_LESS

This is part of the multicolvar module

This action can be used to filter the distribution of colvar values in a MultiColvar so that one can compute the mean and so on for only those multicolvars less than a tolerance.

This action can be used to create a dynamic group of atom based on the value of a multicolvar. In this action a multicolvar is within the dynamic group if its value is less than a target. In actuality a weight, w_i is ascribed to each colvar, s_i calculated by a multicolvar and this weight measures the degree to which a colvar is a member of the group. This weight is a number between 0 and 1 that is calculated using a switchingfunction , σ. If one calculates a function of the set of multicolvars these weights are included in the calculation. As such if one calculates the MEAN, μ of a filtered multicolvar what is computed is the following:

$$\mu = \frac{\sum_i w_i s_i}{\sum_i w_i}$$

One is thus calculating the mean for those colvars that are less than the target.

Examples

The example shown below calculates the mean for those distances that less than 1.5 nm in length

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MFILTER_LESS.tmp
DISTANCES GROUPA=1 GROUPB=2-50 MEAN LABEL=d1
MFILTER_LESS DATA=d1 SWITCH={GAUSSIAN D_0=1.5 R_0=0.00001} MEAN LABEL=d4
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...

MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

5.5.39 MFILTER_MORE

This is part of the multicolvar module

This action can be used to filter the distribution of colvar values in a MultiColvar so that one can compute the mean and so on for only those multicolvars more than a tolerance.

This action can be used to create a dynamic group of atom based on the value of a multicolvar. In this action a multicolvar is within the dynamic group if its value is greater than a target. In actuality a weight, w_i is ascribed to each colvar, s_i calculated by a multicolvar and this weight measures the degree to which a colvar is a member of the group. This weight is calculated using a switchingfunction , σ so it is given by:

$$w_i = 1 - \sigma(s_i)$$

If one calculates a function of the set of multicolvars these weights are included in the calculation. As such if one calculates the MEAN, μ of a filtered multicolvar what is computed is the following:

$$\mu = \frac{\sum_i w_i s_i}{\sum_i w_i}$$

One is thus calculating the mean for those colvars that are greater than the target.

Examples

The example shown below calculates the mean for those distances that greater than 1.5 nm in length

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MFILTER_MORE.tmp
DISTANCES GROUPA=1 GROUPB=2-50 MEAN LABEL=d1
MFILTER_MORE DATA=d1 SWITCH={GAUSSIAN D_0=1.5 R_0=0.00001} MEAN LABEL=d4
```

More complicated things can be done by using the label of a filter as input to a new multicolvar as shown in the example below. Here the coordination numbers of all atoms are computed. The atoms with a coordination number greater than 2 are then identified using the filter. This reduced list of atoms is then used as input to a second coordination number calculation. This second coordination number thus measures the number of two-coordinated atoms that each of the two-coordinated atoms is bound to.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MFILTER_MORE.tmp
c1: COORDINATIONNUMBER SPECIES=1-150 SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
cf: MFILTER_MORE DATA=c1 SWITCH={RATIONAL D_0=2.0 R_0=0.1} LOWMEM
c2: COORDINATIONNUMBER SPECIES=cf SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0} MORE_THAN={RATIONAL D_0=2.0 R_0=0.1}
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...

MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

5.5.40 AROUND

This is part of the multicolvar module

This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.

Each of the base quantities calculated by a multicolvar can can be assigned to a particular point in three dimensional space. For example, if we have the coordination numbers for all the atoms in the system each coordination number can be assumed to lie on the position of the central atom. Because each base quantity can be assigned to a particular point in space we can calculate functions of the distribution of base quantities in a particular part of the box by using:

$$\overline{s}_\tau = \frac{\sum_i f(s_i) w(x_i,y_i,z_i)}{\sum_i w(x_i,y_i,z_i)}$$

where the sum is over the collective variables, s_i, each of which can be thought to be at (x_i, y_i, z_i). The function $w(x_i, y_i, z_i)$ measures whether or not the system is in the subregion of interest. It is equal to:

$$w(x_i, y_i, z_i) = \int_{xl}^{xu} \int_{yl}^{yu} \int_{zl}^{zu} \mathrm{d}x\mathrm{d}y\mathrm{d}z K\left(\frac{x - x_i}{\sigma}\right) K\left(\frac{y - y_i}{\sigma}\right) K\left(\frac{z - z_i}{\sigma}\right)$$

where K is one of the kernel functions described on histograbead and σ is a bandwidth parameter. The function (s_i) can be any of the usual LESS_THAN, MORE_THAN, WITHIN etc that are used in all other multicolvars.

When AROUND is used with the DENSITY action the number of atoms in the specified region is calculated

Examples

The following commands tell plumed to calculate the average coordination number for the atoms that have x (in fractional coordinates) within 2.0 nm of the com of mass c1. The final value will be labeled s.mean.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AROUND.tmp
COM ATOMS=1-100 LABEL=c1
COORDINATIONNUMBER SPECIES=1-100 R_0=1.0 LABEL=c
AROUND DATA=c ATOM=c1 XLOWER=-2.0 XUPPER=2.0 SIGMA=0.1 MEAN LABEL=s
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

ATOM	the atom whose vicinity we are interested in examining. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
SIGMA	the width of the function to be used for kernel density estimation
KERNEL	(default=gaussian) the type of kernel function to be used
XLOWER	(default=0.0) the lower boundary in x relative to the x coordinate of the atom (0 indicates use full extent of box).
XUPPER	(default=0.0) the upper boundary in x relative to the x coordinate of the atom (0 indicates use full extent of box).
YLOWER	(default=0.0) the lower boundary in y relative to the y coordinate of the atom (0 indicates use full extent of box).
YUPPER	(default=0.0) the upper boundary in y relative to the y coordinate of the atom (0 indicates use full extent of box).
ZLOWER	(default=0.0) the lower boundary in z relative to the z coordinate of the atom (0 indicates use full extent of box).
ZUPPER	(default=0.0) the upper boundary in z relative to the z coordinate of the atom (0 indicates use full extent of box).

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
OUTSIDE	(default=off) calculate quantities for colvars that are on atoms outside the region of interest
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...

LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SUM2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

5.5.41 CAVITY

This is part of the multicolvar module

This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a box defined by the positions of four atoms.

Each of the base quantities calculated by a multicolvar can can be assigned to a particular point in three dimensional space. For example, if we have the coordination numbers for all the atoms in the system each coordination number can be assumed to lie on the position of the central atom. Because each base quantity can be assigned to a particular point in space we can calculate functions of the distribution of base quantities in a particular part of the box by using:

$$\overline{s}_\tau = \frac{\sum_i f(s_i) w(u_i, v_i, w_i)}{\sum_i w(u_i, v_i, w_i)}$$

where the sum is over the collective variables, s_i, each of which can be thought to be at (u_i, v_i, z_i). The function (s_i) can be any of the usual LESS_THAN, MORE_THAN, WITHIN etc that are used in all other multicolvars. Notice that here (at variance with what is done in AROUND) we have transformed from the usual (x_i, y_i, z_i) position to a position in (u_i, v_i, z_i). This is done using a rotation matrix as follows:

$$(u_i v_i w_i) = \mathbf{R}\,(x_i - x_o y_i - y_o z_i - z_o)$$

where $\mathbf{R}$ is a rotation matrix that is calculated by constructing a set of three orthonormal vectors from the reference positions specified by the user. The first of these unit vectors points from the first reference atom to the second. The second is then the normal to the plane containing atoms 1,2 and 3 and the the third is the unit vector orthogonal to these first two vectors. (x_o, y_o, z_o), meanwhile, specifies the position of the first reference atom.

In the previous function $w(u_i, v_i, w_i)$ measures whether or not the system is in the subregion of interest. It is equal to:

$$w(u_i, v_i, w_i) = \int_0^{u'} \int_0^{v'} \int_0^{w'} \mathrm{d}u\mathrm{d}v\mathrm{d}w K\left(\frac{u-u_i}{\sigma}\right) K\left(\frac{v-v_i}{\sigma}\right) K\left(\frac{w-w_i}{\sigma}\right)$$

where K is one of the kernel functions described on histogrambead and σ is a bandwidth parameter. The vector connecting atom 1 to atom 4 is used to define the extent of the box in each of the u, v and w directions. Essentially the vector connecting atom 1 to atom 4 is projected onto the three unit vectors described above and the resulting projections determine the u', v' and w' parameters in the above expression.

Examples

The following commands tell plumed to calculate the number of atoms in an ion channel in a protein. The extent of the channel is calculated from the positions of atoms 1, 4, 5 and 11. The final value will be labeled cav.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CAVITY.tmp
d1: DENSITY SPECIES=20-500
CAVITY DATA=d1 ATOMS=1,4,5,11 SIGMA=0.1 LABEL=cav
```

The following command tells plumed to calculate the coordination numbers (with other water molecules) for the water molecules in the protein channel described above. The average coordination number and the number of coordination numbers more than 4 is then calculated. The values of these two quantities are given the labels cav.mean and cav.morethan

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CAVITY.tmp
d1: COORDINATIONNUMBER SPECIES=20-500 R_0=0.1
CAVITY DATA=d1 ATOMS=1,4,5,11 SIGMA=0.1 MEAN MORE_THAN={RATIONAL R_0=4} LABEL=cav
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The atoms involved can be specified using

ATOMS	the positions of four atoms that define spatial extent of the cavity. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
SIGMA	the width of the function to be used for kernel density estimation
KERNEL	(default=gaussian) the type of kernel function to be used

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
OUTSIDE	(default=off) calculate quantities for colvars that are on atoms outside the region of interest
PRINT_BOX	(default=off) write out the positions of the corners of the box to an xyz file
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...
FILE	the file on which to write out the box coordinates
UNITS	(default=nm) the units in which to write out the corners of the box

5.5.42 INCYLINDER

	This is part of the multicolvar module

This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.

Each of the base quantities calculated by a multicolvar can can be assigned to a particular point in three dimensional space. For example, if we have the coordination numbers for all the atoms in the system each coordination number can be assumed to lie on the position of the central atom. Because each base quantity can be assigned to a particular point in space we can calculate functions of the distribution of base quantities in a particular part of the box by using:

$$\overline{s}_\tau = \frac{\sum_i f(s_i)\sigma(r_{xy})}{\sum_i \sigma(r_{xy})}$$

where the sum is over the collective variables, s_i, each of which can be thought to be at (x_i, y_i, z_i). The function σ is a switchingfunction that acts on the distance between the point at which the collective is located (x_i, y_i, z_i) and

the position of the atom that was specified using the ORIGIN keyword projected in the xy plane if DIRECTION=z is used. In other words:

$$r_{xy} = sqrt(x_i - x_0)^2 + (y_i - y_0)^2$$

In short this function, $\sigma(r_{xy})$, measures whether or not the CV is within a cylinder that runs along the axis specified using the DIRECTION keyword and that is centered on the position of the atom specified using ORIGIN.

The function (s_i) can be any of the usual LESS_THAN, MORE_THAN, WITHIN etc that are used in all other multicolvars.

When INCYLINDER is used with the DENSITY action the number of atoms in the specified region is calculated

Examples

The input below can be use to calculate the average coordination numbers for those atoms that are within a cylindrical tube of radius 1.5 nm that is centered on the position of atom 101 and that has its long axis parallel to the z-axis.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCYLINDER.tmp
c1: COORDINATIONNUMBER SPECIES=1-100 SWITCH={RATIONAL R_0=0.1}
d2: INCYLINDER ATOM=101 DATA=c1 DIRECTION=Z RADIUS={TANH R_0=1.5} SIGMA=0.1 LOWER=-0.1 UPPER=0.1 MEAN
PRINT ARG=d2.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous.

The atoms involved can be specified using

ATOM	the atom whose vicinity we are interested in examining. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
KERNEL	(default=gaussian) the type of kernel function to be used
DIRECTION	the direction of the long axis of the cylinder. Must be x, y or z
RADIUS	a switching function that gives the extent of the cylinder in the plane perpendicular to the direction
LOWER	(default=0.0) the lower boundary on the direction parallel to the long axis of the cylinder
UPPER	(default=0.0) the upper boundary on the direction parallel to the long axis of the cylinder

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
OUTSIDE	(default=off) calculate quantities for colvars that are on atoms outside the region of interest
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...

BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...
SIGMA	the width of the function to be used for kernel density estimation

5.5.43 INENVELOPE

This is part of the multicolvar module

This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a region where the density of a certain type of atom is high.

This collective variable can be used to determine whether colvars are within region where the density of a particular atom is high. This is achieved by calculating the following function at the point where the atom is located (x,y,z):

$$w_j = 1 - \sigma\left[\sum_{i=1}^{N} K\left(\frac{x-x_i}{\sigma_x}, \frac{y-y_i}{\sigma_y}, \frac{z-z_i}{\sigma_z}\right)\right]$$

Here σ is a switchingfunction and K is a kernelfunctions. The sum runs over the atoms specified using the ATOMS keyword and a w_j value is calculated for each of the central atoms of the input multicolvar.

Examples

The input below calculates a density field from the positions of atoms 1-14400. The number of the atoms that are specified in the DENSITY action that are within a region where the density field is greater than 2.0 is then calculated.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INENVELOPE.tmp
d1: DENSITY SPECIES=14401-74134:3 LOWMEM
fi: INENVELOPE DATA=d1 ATOMS=1-14400 CONTOUR={RATIONAL D_0=2.0 R_0=1.0} BANDWIDTH=0.1,0.1,0.1 LOWMEM
PRINT ARG=fi FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The atoms involved can be specified using

ATOMS	the atom whose positions we are constructing a field from. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
KERNEL	(default=gaussian) the type of kernel function to be used
BANDWIDTH	the bandwidths for kernel density estimation
CONTOUR	a switching function that tells PLUMED how large the density should be

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
OUTSIDE	(default=off) calculate quantities for colvars that are on atoms outside the region of interest
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SUM2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

5.5.44 INSPHERE

	This is part of the multicolvar module

This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.

Each of the base quantities calculated by a multicolvar can can be assigned to a particular point in three dimensional space. For example, if we have the coordination numbers for all the atoms in the system each coordination number can be assumed to lie on the position of the central atom. Because each base quantity can be assigned to a particular point in space we can calculate functions of the distribution of base quantities in a particular part of the box by using:

$$\overline{s}_\tau = \frac{\sum_i f(s_i)\sigma(r)}{\sum_i \sigma(r)}$$

where the sum is over the collective variables, s_i, each of which can be thought to be at (x_i, y_i, z_i). The function σ is a switchingfunction that acts on the distance between the point at which the collective is located (x_i, y_i, z_i) and the position of the atom that was specified using the ORIGIN keyword. In other words:

$$r = sqrt(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2$$

In short this function, $\sigma(r_{xy})$, measures whether or not the CV is within a sphere that is centered on the position of the atom specified using the keyword ORIGIN.

The function (s_i) can be any of the usual LESS_THAN, MORE_THAN, WITHIN etc that are used in all other multicolvars.

When INCYLINDER is used with the DENSITY action the number of atoms in the specified region is calculated

Examples

The input below can be use to calculate the average coordination numbers for those atoms that are within a sphere of radius 1.5 nm that is centered on the position of atom 101.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INSPHERE.tmp
c1: COORDINATIONNUMBER SPECIES=1-100 SWITCH={RATIONAL R_0=0.1}
d2: INSPHERE ATOM=101 DATA=c1 RADIUS={TANH R_0=1.5} MEAN
PRINT ARG=d2.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The atoms involved can be specified using

ATOM	the atom whose vicinity we are interested in examining. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
KERNEL	(default=gaussian) the type of kernel function to be used
RADIUS	the switching function that tells us the extent of the spherical region of interest

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
OUTSIDE	(default=off) calculate quantities for colvars that are on atoms outside the region of interest
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

5.5.45 TETRAHEDRALPORE

This is part of the multicolvar module

This quantity can be used to calculate functions of the distribution of collective variables for the atoms lie that lie in a box defined by the positions of four atoms at the corners of a tetrahedron.

Each of the base quantities calculated by a multicolvar can can be assigned to a particular point in three dimensional space. For example, if we have the coordination numbers for all the atoms in the system each coordination number can be assumed to lie on the position of the central atom. Because each base quantity can be assigned to a particular point in space we can calculate functions of the distribution of base quantities in a particular part of the box by using:

$$\overline{s}_\tau = \frac{\sum_i f(s_i) w(u_i, v_i, w_i)}{\sum_i w(u_i, v_i, w_i)}$$

where the sum is over the collective variables, s_i, each of which can be thought to be at (u_i, v_i, z_i). The function (s_i) can be any of the usual LESS_THAN, MORE_THAN, WITHIN etc that are used in all other multicolvars. Notice that here (at variance with what is done in AROUND) we have transformed from the usual (x_i, y_i, z_i) position to a position in (u_i, v_i, z_i). This is done using a rotation matrix as follows:

$$(u_i v_i w_i) = \mathbf{R} (x_i - x_o y_i - y_o z_i - z_o)$$

where $\mathbf{R}$ is a rotation matrix that is calculated by constructing a set of three orthonormal vectors from the reference positions specified by the user. Initially unit vectors are found by calculating the bisector, $\mathbf{b}$, and cross product, $\mathbf{c}$, of the vectors connecting atoms 1 and 2. A third unit vector, $\mathbf{p}$ is then found by taking the cross product between the cross product calculated during the first step, $\mathbf{c}$ and the bisector, $\mathbf{b}$. From this second cross product $\mathbf{p}$ and the bisector $\mathbf{b}$ two new vectors are calculated using:

$$v_1 = \cos\left(\frac{\pi}{4}\right)\mathbf{b} + \sin\left(\frac{\pi}{4}\right)\mathbf{p} \qquad \text{and} \qquad v_2 = \cos\left(\frac{\pi}{4}\right)\mathbf{b} - \sin\left(\frac{\pi}{4}\right)\mathbf{p}$$

In the previous function $w(u_i, v_i, w_i)$ measures whether or not the system is in the subregion of interest. It is equal to:

$$w(u_i, v_i, w_i) = \int_0^{u'}\int_0^{v'}\int_0^{w'} \mathrm{d}u\mathrm{d}v\mathrm{d}w K\left(\frac{u-u_i}{\sigma}\right) K\left(\frac{v-v_i}{\sigma}\right) K\left(\frac{w-w_i}{\sigma}\right)$$

where K is one of the kernel functions described on histograмbead and σ is a bandwidth parameter. The values of u' and v' are found by finding the projections of the vectors connecting atoms 1 and 2 and 1 and 3 v_1 and v_2. This gives four projections: the largest two projections are used in the remainder of the calculations. w' is calculated by taking the projection of the vector connecting atoms 1 and 4 on the vector $\mathbf{c}$. Notice that the manner by which this box is constructed differs from the way this is done in CAVITY. This is in fact the only point of difference between these two actions.

Examples

The following commands tell plumed to calculate the number of atom inside a tetrahedral cavity. The extent of the tetrahedral cavity is calculated from the positions of atoms 1, 4, 5, and 11, The final value will be labeled cav.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TETRAHEDRALPORE.tmp
d1: DENSITY SPECIES=20-500
TETRAHEDRALPORE DATA=d1 ATOMS=1,4,5,11 SIGMA=0.1 LABEL=cav
```

The following command tells plumed to calculate the coordination numbers (with other water molecules) for the water molecules in the tetrahedral cavity described above. The average coordination number and the number of coordination numbers more than 4 is then calculated. The values of these two quantities are given the labels cav.mean and cav.morethan

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TETRAHEDRALPORE.tmp
d1: COORDINATIONNUMBER SPECIES=20-500 R_0=0.1
CAVITY DATA=d1 ATOMS=1,4,5,11 SIGMA=0.1 MEAN MORE_THAN={RATIONAL R_0=4} LABEL=cav
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The atoms involved can be specified using

ATOMS	the positions of four atoms that define spatial extent of the cavity. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
SIGMA	the width of the function to be used for kernel density estimation
KERNEL	(default=gaussian) the type of kernel function to be used

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
OUTSIDE	(default=off) calculate quantities for colvars that are on atoms outside the region of interest
PRINT_BOX	(default=off) write out the positions of the corners of the box to an xyz file
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...
FILE	the file on which to write out the box coordinates
UNITS	(default=nm) the units in which to write out the corners of the box

5.5.46 GRADIENT

	This is part of the crystallization module

It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the gradient of the average value of a multicolvar value

This command allows you to calculate the collective variable discussed in [30].

Examples

The input below calculates the gradient of the density of atoms in the manner described in [30] in order to detect whether or not atoms are distributed uniformly along the x-axis of the simulation cell.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GRADIENT.tmp
d1: DENSITY SPECIES=1-50
s1: GRADIENT ORIGIN=1 DATA=d1 DIR=x NBINS=4 SIGMA=1.0
PRINT ARG=s1 FILE=colvar
```

The input below calculates the coordination numbers of the 50 atoms in the simulation cell. The gradient of this quantity is then evaluated in the manner described using the equation above to detect whether the average values of the coordination number are uniformly distributed along the x-axis of the simulation cell.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/GRADIENT.tmp
d2: COORDINATIONNUMBER SPECIES=1-50 SWITCH={RATIONAL R_0=2.0} MORE_THAN={EXP R_0=4.0}
s2: GRADIENT ORIGIN=1 DATA=d2 DIR=x NBINS=4 SIGMA=1.0
PRINT ARG=s2 FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

ORIGIN	we will use the position of this atom as the origin in our calculation. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
DIR	(default=xyz) the directions in which we are calculating the gradient. Should be x, y, z, xy, xz, yz or xyz
NBINS	number of bins to use in each direction for the calculation of the gradient
SIGMA	(default=1.0) the width of the function to be used for kernel density estimation
KERNEL	(default=gaussian) the type of kernel function to be used

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.5.47 INTERMOLECULARTORSIONS

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate torsion angles between vectors on adjacent molecules

This variable can be used to calculate the average torsional angles between vectors. In other words, it can be used to compute quantities like this:

$$s = \frac{\sum_{i \neq j} \sigma(r_{ij}) \theta_{ij}}{\sum_{i \neq j} \sigma(r_{ij})}$$

Here the sums run over all pairs of molecules. $\sigma(r_{ij})$ is a switchingfunction that action on the distance between the centers of molecules i and j. θ_{ij} is then the torsional angle between an orientation vector for molecule i and molecule j.

This command can be used to calculate the intermolecular torsional angles between the orientations of nearby molecules. The orientation of a molecule can be calculated by using either the MOLECULES or the PLANES commands. These two commands calculate the orientation of a bond in the molecule or the orientation of a plane containing three of the molecule's atoms. Furthermore, when we use these commands we think of molecules as objects that lie at a point in space and that have an orientation. This command calculates the torsional angles between the orientations of these objects. We can then calculates functions of a large number of these torsional angles that measures things such as the number of torsional angles that are within a particular range. Because it is often useful to only consider the torsional angles between objects that are within a certain distance of each other we can, when calculating these sums, perform a weighted sum and use a switchingfunction to ensure that we focus on molecules that are close together.

Examples

The example input below is necessarily but gives you an idea of what can be achieved using this action. The orientations and positions of four molecules are defined using the MOLECULES action as the position of the centers of mass of the two atoms specified and the direction of the vector connecting the two atoms that were specified. The torsional angles between the molecules are then calculated by the INTERMOLECULARTORSIONS

command labelled torsion_p. We then compute a HISTOGRAM that shows the distribution that these torsional angles take in the structure. The weight a given torsional angle contributes to this HISTOGRAM is determined using a switchingfunction that acts on the distance between the two molecules. As such the torsional angles between molecules that are close together contribute a high weight to the histogram while the torsional angles between molecules that are far apart does not contribute to the histogram. The histogram is averaged over the whole trajectory and output once all the trajectory frames have been read.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INTERMOLECULARTORSIONS.tmp
m1: MOLECULES MOL1=1,2 MOL2=3,4 MOL3=5,6 MOL4=7,8
torsion_p: INTERMOLECULARTORSIONS MOLS=m1 SWITCH={RATIONAL R_0=0.25 D_0=2.0 D_MAX=3.0}
htt_p: HISTOGRAM DATA=torsion_p GRID_MIN=-pi GRID_MAX=pi BANDWIDTH=0.1 GRID_BIN=200 STRIDE=1
DUMPGRID GRID=htt_p FILE=myhist.out
```

Glossary of keywords and components

The atoms involved can be specified using

MOLS	The molecules you would like to calculate the torsional angles between. This should be the label/s of MOLECULES or PLANES actions. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Or alternatively by using

MOLSA	In this version of the input the torsional angles between all pairs of atoms including one atom from MOLSA one atom from MOLSB will be computed. This should be the label/s of MOLECULES or PLANES actions
MOLSB	In this version of the input the torsional angles between all pairs of atoms including one atom from MOLSA one atom from MOLSB will be computed. This should be the label/s of MOLECULES or PLANES actions

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOWMEM	(default=off) lower the memory requirements
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

5.5.48 LOCAL_AVERAGE

This is part of the multicolvar module

Calculate averages over spherical regions centered on atoms

As is explained in this video certain multicolvars calculate one scalar quantity or one vector for each of the atoms in the system. For example COORDINATIONNUMBER measures the coordination number of each of the atoms in the system and Q4 measures the fourth order Steinhardt parameter for each of the atoms in the system. These quantities provide tell us something about the disposition of the atoms in the first coordination sphere of each of the atoms of interest. Lechner and Dellago [24] have suggested that one can probe local order in a system by taking the average value of such symmetry functions over the atoms within a spherical cutoff of each of these atoms in the systems. When this is done with Steinhardt parameters they claim this gives a coordinate that is better able to distinguish solid and liquid configurations of Lennard-Jones atoms.

You can calculate such locally averaged quantities within plumed by using the LOCAL_AVERAGE command. This command calculates the following atom-centered quantities:

$$s_i = \frac{c_i + \sum_j \sigma(r_{ij}) c_j}{1 + \sum_j \sigma(r_{ij})}$$

where the c_i and c_j values can be for any one of the symmetry functions that can be calculated using plumed multicolvars. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. Lechner and Dellago suggest that the parameters of this function should be set so that it the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise.

The s_i quantities calculated using the above command can be again thought of as atom-centered symmetry functions. They thus operate much like multicolvars. You can thus calculate properties of the distribution of s_i values using MEAN, LESS_THAN, HISTOGRAM and so on. You can also probe the value of these averaged variables in regions of the box by using the command in tandem with the AROUND command.

Examples

This example input calculates the coordination numbers for all the atoms in the system. These coordination numbers are then averaged over spherical regions. The number of averaged coordination numbers that are greater than 4 is then output to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_AVERAGE.tmp
COORDINATIONNUMBER SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=d1
LOCAL_AVERAGE SPECIES=d1 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MORE_THAN={RATIONAL R_0=4} LABEL=la
PRINT ARG=la.* FILE=colvar
```

This example input calculates the q_4 (see Q4) vectors for each of the atoms in the system. These vectors are then averaged component by component over a spherical region. The average value for this quantity is then output to a file. This calculates the quantities that were used in the paper by Lechner and Dellago [24]

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_AVERAGE.tmp
Q4 SPECIES=1-64 SWITCH={RATIONAL D_0=1.3 R_0=0.2} LABEL=q4
LOCAL_AVERAGE SPECIES=q4 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LABEL=la
PRINT ARG=la.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOWMEM	(default=off) lower the memory requirements
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
VSUM	calculate the norm of the sum of vectors. The final value can be referenced using *label*.vsum. You can use multiple instances of this keyword i.e. VSU↩ M1, VSUM2, VSUM3... The corresponding values are then referenced using *label*.vsum-1, *label*.vsum-2, *label*.vsum-3...

5.5.49 LOCAL_Q3

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the local degree of order around an atoms by taking the average dot product between the q_3 vector on the central atom and the q_3 vector on the atoms in the first coordination sphere.

The Q3 command allows one to calculate one complex vectors for each of the atoms in your system that describe the degree of order in the coordination sphere around a particular atom. The difficulty with these vectors comes when combining the order parameters from all of the individual atoms/molecules so as to get a measure of the global degree of order for the system. The simplest way of doing this - calculating the average Steinhardt parameter - can be problematic. If one is examining nucleation say only the order parameters for those atoms in the nucleus will change significantly when the nucleus forms. The order parameters for the atoms in the surrounding liquid will remain pretty much the same. As such if one models a small nucleus embedded in a very large amount of solution/melt any change in the average order parameter will be negligible. Substantial changes in the value of this average can be observed in simulations of nucleation but only because the number of atoms is relatively small.

When the average Q3 parameter is used to bias the dynamics a problems can occur. These averaged coordinates cannot distinguish between the correct, single-nucleus pathway and a concerted pathway in which all the atoms rearrange themselves into their solid-like configuration simultaneously. This second type of pathway would be impossible in reality because there is a large entropic barrier that prevents concerted processes like this from happening. However, in the finite sized systems that are commonly simulated this barrier is reduced substantially. As a result in simulations where average Steinhardt parameters are biased there are often quite dramatic system size effects

If one wants to simulate nucleation using some form on biased dynamics what is really required is an order parameter that measures:

- Whether or not the coordination spheres around atoms are ordered
- Whether or not the atoms that are ordered are clustered together in a crystalline nucleus

LOCAL_AVERAGE and NLINKS are variables that can be combined with the Steinhardt parameters allow to calculate variables that satisfy these requirements. LOCAL_Q3 is another variable that can be used in these sorts of calculations. The LOCAL_Q3 parameter for a particular atom is a number that measures the extent to which the orientation of the atoms in the first coordination sphere of an atom match the orientation of the central atom. It does this by calculating the following quantity for each of the atoms in the system:

$$s_i = \frac{\sum_j \sigma(r_{ij}) \sum_{m=-3}^{3} q^*_{3m}(i) q_{3m}(j)}{\sum_j \sigma(r_{ij})}$$

where $q_{3m}(i)$ and $q_{3m}(j)$ are the 3rd order Steinhardt vectors calculated for atom i and atom j respectively and the asterisk denotes complex conjugation. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. The parameters of this function should be set so that it the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise. The sum in the numerator of this expression is the dot product of the Steinhardt parameters for atoms i and j and thus measures the degree to which the orientations of these adjacent atoms is correlated.

Examples

The following command calculates the average value of the LOCAL_Q3 parameter for the 64 Lennard Jones atoms in the system under study and prints this quantity to a file called colvar.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q3.tmp
Q3 SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=q3
LOCAL_Q3 SPECIES=q3 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LABEL=lq3
PRINT ARG=lq3.mean FILE=colvar
```

The following input calculates the distribution of LOCAL_Q3 parameters at any given time and outputs this information to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q3.tmp
Q3 SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=q3
LOCAL_Q3 SPECIES=q3 SWITCH={RATIONAL D_0=1.3 R_0=0.2} HISTOGRAM={GAUSSIAN LOWER=0.0 UPPER=1.0 NBINS=20 SMEAR=0
PRINT ARG=lq3.* FILE=colvar
```

The following calculates the LOCAL_Q3 parameters for atoms 1-5 only. For each of these atoms comparisons of the geometry of the coordination sphere are done with those of all the other atoms in the system. The final quantity is the average and is outputted to a file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q3.tmp
Q3 SPECIESA=1-5 SPECIESB=1-64 D_0=1.3 R_0=0.2 LABEL=q3a
Q3 SPECIESA=6-64 SPECIESB=1-64 D_0=1.3 R_0=0.2 LABEL=q3b

LOCAL_Q3 SPECIES=q3a,q3b SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LOWMEM LABEL=w3
PRINT ARG=w3.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.50 LOCAL_Q4

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the local degree of order around an atoms by taking the average dot product between the q_4 vector on the central atom and the q_4 vector on the atoms in the first coordination sphere.

The Q4 command allows one to calculate one complex vectors for each of the atoms in your system that describe the degree of order in the coordination sphere around a particular atom. The difficulty with these vectors comes when combining the order parameters from all of the individual atoms/molecules so as to get a measure of the global degree of order for the system. The simplest way of doing this - calculating the average Steinhardt parameter - can be problematic. If one is examining nucleation say only the order parameters for those atoms in the nucleus will change significantly when the nucleus forms. The order parameters for the atoms in the surrounding liquid will remain pretty much the same. As such if one models a small nucleus embedded in a very large amount of solution/melt any change in the average order parameter will be negligible. Substantial changes in the value of this average can be observed in simulations of nucleation but only because the number of atoms is relatively small.

When the average Q4 parameter is used to bias the dynamics a problems can occur. These averaged coordinates cannot distinguish between the correct, single-nucleus pathway and a concerted pathway in which all the atoms rearrange themselves into their solid-like configuration simultaneously. This second type of pathway would be impossible in reality because there is a large entropic barrier that prevents concerted processes like this from happening. However, in the finite sized systems that are commonly simulated this barrier is reduced substantially. As a result in simulations where average Steinhardt parameters are biased there are often quite dramatic system size effects

If one wants to simulate nucleation using some form on biased dynamics what is really required is an order parameter that measures:

- Whether or not the coordination spheres around atoms are ordered
- Whether or not the atoms that are ordered are clustered together in a crystalline nucleus

LOCAL_AVERAGE and NLINKS are variables that can be combined with the Steinhardt parameters allow to calculate variables that satisfy these requirements. LOCAL_Q4 is another variable that can be used in these sorts of calculations. The LOCAL_Q4 parameter for a particular atom is a number that measures the extent to which the orientation of the atoms in the first coordination sphere of an atom match the orientation of the central atom. It does this by calculating the following quantity for each of the atoms in the system:

$$s_i = \frac{\sum_j \sigma(r_{ij}) \sum_{m=-4}^{4} q^*_{4m}(i) q_{4m}(j)}{\sum_j \sigma(r_{ij})}$$

where $q_{4m}(i)$ and $q_{4m}(j)$ are the fourth order Steinhardt vectors calculated for atom i and atom j respectively and the asterisk denotes complex conjugation. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. The parameters of this function should be set so that it the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise. The sum in the numerator of this expression is the dot product of the Steinhardt parameters for atoms i and j and thus measures the degree to which the orientations of these adjacent atoms is correlated.

Examples

The following command calculates the average value of the LOCAL_Q4 parameter for the 64 Lennard Jones atoms in the system under study and prints this quantity to a file called colvar.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q4.tmp
Q4 SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=q4
LOCAL_Q4 SPECIES=q4 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LABEL=lq4
PRINT ARG=lq4.mean FILE=colvar
```

The following input calculates the distribution of LOCAL_Q4 parameters at any given time and outputs this information to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q4.tmp
Q4 SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=q4
LOCAL_Q4 SPECIES=q4 SWITCH={RATIONAL D_0=1.3 R_0=0.2} HISTOGRAM={GAUSSIAN LOWER=0.0 UPPER=1.0 NBINS=20 SMEAR=0
PRINT ARG=lq4.* FILE=colvar
```

The following calculates the LOCAL_Q4 parameters for atoms 1-5 only. For each of these atoms comparisons of the geometry of the coordination sphere are done with those of all the other atoms in the system. The final quantity is the average and is outputted to a file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q4.tmp
Q4 SPECIESA=1-5 SPECIESB=1-64 D_0=1.3 R_0=0.2 LABEL=q4a
Q4 SPECIESA=6-64 SPECIESB=1-64 D_0=1.3 R_0=0.2 LABEL=q4b

LOCAL_Q4 SPECIES=q4a,q4b SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LOWMEM LABEL=w4
PRINT ARG=w4.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.51 LOCAL_Q6

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate the local degree of order around an atoms by taking the average dot product between the q_6 vector on the central atom and the q_6 vector on the atoms in the first coordination sphere.

The Q6 command allows one to calculate one complex vectors for each of the atoms in your system that describe the degree of order in the coordination sphere around a particular atom. The difficulty with these vectors comes when combining the order parameters from all of the individual atoms/molecules so as to get a measure of the global degree of order for the system. The simplest way of doing this - calculating the average Steinhardt parameter - can be problematic. If one is examining nucleation say only the order parameters for those atoms in the nucleus will change significantly when the nucleus forms. The order parameters for the atoms in the surrounding liquid will remain pretty much the same. As such if one models a small nucleus embedded in a very large amount of solution/melt any change in the average order parameter will be negligible. Substantial changes in the value of this average can be observed in simulations of nucleation but only because the number of atoms is relatively small.

When the average Q6 parameter is used to bias the dynamics a problems can occur. These averaged coordinates cannot distinguish between the correct, single-nucleus pathway and a concerted pathway in which all the atoms rearrange themselves into their solid-like configuration simultaneously. This second type of pathway would be impossible in reality because there is a large entropic barrier that prevents concerted processes like this from happening. However, in the finite sized systems that are commonly simulated this barrier is reduced substantially. As a result in simulations where average Steinhardt parameters are biased there are often quite dramatic system size effects

If one wants to simulate nucleation using some form on biased dynamics what is really required is an order parameter that measures:

- Whether or not the coordination spheres around atoms are ordered
- Whether or not the atoms that are ordered are clustered together in a crystalline nucleus

LOCAL_AVERAGE and NLINKS are variables that can be combined with the Steinhardt parameters allow to calculate variables that satisfy these requirements. LOCAL_Q6 is another variable that can be used in these sorts of calculations. The LOCAL_Q6 parameter for a particular atom is a number that measures the extent to which the orientation of the atoms in the first coordination sphere of an atom match the orientation of the central atom. It does this by calculating the following quantity for each of the atoms in the system:

$$s_i = \frac{\sum_j \sigma(r_{ij}) \sum_{m=-6}^{6} q^*_{6m}(i) q_{6m}(j)}{\sum_j \sigma(r_{ij})}$$

where $q_{6m}(i)$ and $q_{6m}(j)$ are the sixth order Steinhardt vectors calculated for atom i and atom j respectively and the asterisk denotes complex conjugation. The function $\sigma(r_{ij})$ is a switchingfunction that acts on the distance between atoms i and j. The parameters of this function should be set so that it the function is equal to one when atom j is in the first coordination sphere of atom i and is zero otherwise. The sum in the numerator of this expression is the dot product of the Steinhardt parameters for atoms i and j and thus measures the degree to which the orientations of these adjacent atoms is correlated.

Examples

The following command calculates the average value of the LOCAL_Q6 parameter for the 64 Lennard Jones atoms in the system under study and prints this quantity to a file called colvar.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q6.tmp
Q6 SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=q6
LOCAL_Q6 SPECIES=q6 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LABEL=lq6
PRINT ARG=lq6.mean FILE=colvar
```

The following input calculates the distribution of LOCAL_Q6 parameters at any given time and outputs this information to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q6.tmp
Q6 SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=q6
LOCAL_Q6 SPECIES=q6 SWITCH={RATIONAL D_0=1.3 R_0=0.2} HISTOGRAM={GAUSSIAN LOWER=0.0 UPPER=1.0 NBINS=20 SMEAR=0
PRINT ARG=lq6.* FILE=colvar
```

The following calculates the LOCAL_Q6 parameters for atoms 1-5 only. For each of these atoms comparisons of the geometry of the coordination sphere are done with those of all the other atoms in the system. The final quantity is the average and is outputted to a file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOCAL_Q6.tmp
Q6 SPECIESA=1-5 SPECIESB=1-64 D_0=1.3 R_0=0.2 LABEL=q6a
Q6 SPECIESA=6-64 SPECIESB=1-64 D_0=1.3 R_0=0.2 LABEL=q6b

LOCAL_Q6 SPECIES=q6a,q6b SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LOWMEM LABEL=w6
PRINT ARG=w6.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.52 MCOLV_COMBINE

This is part of the multicolvar module

Calculate linear combinations of multiple multicolvars

Examples

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

DATA	the multicolvars you are calculating linear combinations for
COEFFICIENTS	(default=1.0) the coefficients to use for the various multicolvars

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...

LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histograbead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.53 MCOLV_PRODUCT

This is part of the multicolvar module

Calculate a product of multiple multicolvars

Examples

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

DATA	the multicolvars you are calculating the product of

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1,
	label.morethan-2, *label*.morethan-3...

SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SUM2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta} \log \sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.54 NLINKS

	This is part of the multicolvar module

Calculate number of pairs of atoms/molecules that are linked

In its simplest guise this coordinate calculates a coordination number. Each pair of atoms is assumed "linked" if they are within some cutoff of each other. In more complex applications each entity is a vector and this quantity measures whether pairs of vectors are (a) within a certain cutoff and (b) if the two vectors have similar orientations. The vectors on individual atoms could be Steinhardt parameters (see Q3, Q4 and Q6) or they could describe some internal vector in a molecule.

Examples

The following calculates how many bonds there are in a system containing 64 atoms and outputs this quantity to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/NLINKS.tmp
DENSITY SPECIES=1-64 LABEL=d1
NLINKS GROUP=d1 SWITCH={RATIONAL D_0=1.3 R_0=0.2} LABEL=dd
PRINT ARG=dd FILE=colvar
```

The following calculates how many pairs of neighboring atoms in a system containing 64 atoms have similar dispositions for the atoms in their coordination sphere. This calculation uses the dot product of the Q6 vectors on adjacent atoms to measure whether or not two atoms have the same ``orientation"

```
BEGIN_PLUMED_FILE working DATADIR=example-check/NLINKS.tmp
Q6 SPECIES=1-64 SWITCH={RATIONAL D_0=1.3 R_0=0.2} LABEL=q6
NLINKS GROUP=q6 SWITCH={RATIONAL D_0=1.3 R_0=0.2} LABEL=dd
PRINT ARG=dd FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

GROUP	. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Or alternatively by using

GROUPA	
GROUPB	

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOWMEM	(default=off) lower the memory requirements
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

5.5.55 PAMM

	This is part of the pamm module
	It is only available if you configure PLUMED with ./configure –enable-modules=pamm . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Probabilistic analysis of molecular motifs.

Probabilistic analysis of molecular motifs (PAMM) was introduced in this paper **[pamm]**. The essence of this approach involves calculating some large set of collective variables for a set of atoms in a short trajectory and fitting this data using a Gaussian Mixture Model. The idea is that modes in these distributions can be used to identify features such as hydrogen bonds or secondary structure types.

The assumption within this implementation is that the fitting of the Gaussian mixture model has been done elsewhere by a separate code. You thus provide an input file to this action which contains the means, covariance matrices and weights for a set of Gaussian kernels, $\{\phi\}$. The values and derivatives for the following set of quantities is then computed:

$$s_k = \frac{\phi_k}{\sum_i \phi_i}$$

Each of the ϕ_k is a Gaussian function that acts on a set of quantities calculated within a MultiColvar . These might be TORSIONS, DISTANCES, ANGLES or any one of the many symmetry functions that are available within

MultiColvar actions. These quantities are then inserted into the set of n kernels that are in the the input file. This will be done for multiple sets of values for the input quantities and a final quantity will be calculated by summing the above s_k values or some transformation of the above. This sounds less complicated than it is and is best understood by looking through the example given below.

Warning

Mixing MultiColvar actions that are periodic with variables that are not periodic has not been tested

Examples

In this example I will explain in detail what the following input is computing:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PAMM.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO MOLTYPE=protein STRUCTURE=M1d.pdb
psi: TORSIONS ATOMS1=@psi-2 ATOMS2=@psi-3 ATOMS3=@psi-4
phi: TORSIONS ATOMS1=@phi-2 ATOMS2=@phi-3 ATOMS3=@phi-4
p: PAMM DATA=phi,psi CLUSTERS=clusters.pamm MEAN1={COMPONENT=1} MEAN2={COMPONENT=2}
PRINT ARG=p.mean-1,p.mean-2 FILE=colvar
```

The best place to start our explanation is to look at the contents of the clusters.pamm file

```
#! FIELDS height phi psi sigma_phi_phi sigma_phi_psi sigma_psi_phi sigma_psi_psi
#! SET multivariate von-misses
#! SET kerneltype gaussian
      2.97197455E-0001     -1.91983118E+0000      2.25029540E+0000      2.45960237E-0001     -1.30615381E-0001
      2.29131448E-0002      1.39809354E+0000      9.54585380E-0002      9.61755708E-0002     -3.55657919E-0002
      5.06676398E-0001     -1.09648066E+0000     -7.17867907E-0001      1.40523052E-0001     -1.05385552E-0001
```

This files contains the parameters of two two-dimensional Gaussian functions. Each of these Gaussian kernels has a weight, w_k, a vector that specifies the position of its center, $\mathbf{c}_k$, and a covariance matrix, Σ_k. The ϕ_k functions that we use to calculate our PAMM components are thus:

$$\phi_k = \frac{w_k}{N_k} \exp\left(-(\mathbf{s}-\mathbf{c}_k)^T \Sigma_k^{-1} (\mathbf{s}-\mathbf{c}_k)\right)$$

In the above N_k is a normalization factor that is calculated based on Σ. The vector $\mathbf{s}$ is a vector of quantities that are calculated by the TORSIONS actions. This vector must be two dimensional and in this case each component is the value of a torsion angle. If we look at the two TORSIONS actions in the above we are calculating the ϕ and ψ backbone torsional angles in a protein (Note the use of MOLINFO to make specification of atoms straightforward). We thus calculate the values of our 2 $\{\phi\}$ kernels 3 times. The first time we use the ϕ and ψ angles in the second residue of the protein, the second time it is the ϕ and ψ angles of the third residue of the protein and the third time it is the ϕ and ψ angles of the fourth residue in the protein. The final two quantities that are output by the print command, p.mean-1 and p.mean-2, are the averages over these three residues for the quantities:

$$s_1 = \frac{\phi_1}{\phi_1 + \phi_2}$$

and

$$s_2 = \frac{\phi_2}{\phi_1 + \phi_2}$$

There is a great deal of flexibility in this input. We can work with, and examine, any number of components, we can use any set of collective variables and compute these PAMM variables and we can transform the PAMM variables themselves in a large number of different ways when computing these sums.

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the set of PAMM variables will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly from the underlying set of PAMM variables. These quantities are calculated by employing the keywords listed below and they can be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. The particular PAMM variable that should be averaged in a MEAN command or transformed by a switching function in a LESS↩ _THAN command is specified using the COMPONENT keyword. COMPONENT=1 refers to the PAMM variable in which the first kernel in your input file is on the numerator, COMPONENT=2 refers to PAMM variable in which the second kernel in the input file is on the numerator and so on. The same quantity can be calculated multiple times for different PAMM components by a single PAMM action. In this case the relevant keyword must appear multiple times on the input line followed by a numerical identifier i.e. MEAN1, MEAN2, ... The quantities calculated when multiple MEAN commands appear on the input line can be reference elsewhere in the input file by using the name of the quantity followed followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. Alternatively, you can customize the labels of the quantities by using the LABEL keyword in the description of the keyword.

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

DATA	the multicolvars from which the pamm coordinates are calculated
CLUSTERS	the name of the file that contains the definitions of all the clusters
REGULARISE	(default=0.001) don't allow the denominator to be smaller then this value

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...

LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histograмbead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.5.56 POLYMER_ANGLES

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate a function to investigate the relative orientations of polymer angles

This CV takes the vectors calculated by a PLANES action as input and computes the following function of the relative angles, θ, between the vectors that are normal to the pairs of input vectors:

$$s = \frac{3\cos^2\theta - 1}{2}$$

This average of this quantity over all the vectors in the first coordination sphere around each of the PLANES specified is then calculated.

Examples

The example below calculates a set of vectors using the PLANES action. The average number for the function s defined above is then computed over the first coordination sphere of each of the centers of mass of the molecules that were used to define the planes. Finally the average of these quantities is computed an printed to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/POLYMER_ANGLES.tmp
PLANES ...
MOL1=9,10,11
MOL2=89,90,91
MOL3=473,474,475
MOL4=1161,1162,1163
MOL5=1521,1522,1523
MOL6=1593,1594,1595
MOL7=1601,1602,1603
MOL8=2201,2202,2203
LABEL=m3
... PLANES

s3: POLYMER_ANGLES SPECIES=m3 LOWMEM SWITCH={RATIONAL R_0=0.6} MEAN
PRINT ARG=s3.mean FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D↩ _0	(default=0.0) The d_0 parameter of the switching function
R↩ _0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.57 SMAC

	This is part of the crystallization module
	It is only available if you configure PLUMED with ./configure –enable-modules=crystallization . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate a variant on the SMAC collective variable

This variable is discussed in [31]

The SMAC collective variable can be used to study the formation of molecular solids from either the melt or from solution. The idea behind this variable is that what differentiates a molecular solid from a molecular liquid is an alignment of internal vectors in neighboring molecules. In other words, the relative orientation of neighboring molecules is no longer random as it is in a liquid. In a solid particular torsional angles between molecules are preferred. As such this CV calculates the following average:

$$s_i = \frac{\left\{1 - \psi\left[\sum_{j\neq i}\sigma(r_{ij})\right]\right\}\sum_{j\neq i}\sigma(r_{ij})\sum_n K_n(\theta_{ij})}{\sum_{j\neq i}\sigma(r_{ij})}$$

In this expression r_{ij} is the distance between molecule i and molecule j and $\sigma(r_{ij})$ is a switchingfunction that acts on this distance. By including this switching function in the second summation in the numerator and in the denominator we are thus ensuring that we calculate an average over the molecules in the first coordination sphere of molecule i. All molecules in higher coordination sphere will essentially contribute zero to the sums in the above expression because their $\sigma(r_{ij})$ will be very small. ψ is also a switching function. The term including ψ in the numerator is there to ensure that only those molecules that are attached to a reasonably large number of molecules. It is important to include this "more than" switching function when you are simulating nucleation from solution with this CV. Lastly, the $K_n functions are kernelfunctions that take the torsion angle, θ_{ij}, between the internal orientation vectors for molecules i and j as input. These kernel functions should be set so that they are equal to one when the relative orientation of the molecules are as they are in the solid and equal to zero otherwise. The final s_i quantity thus measures whether (on average) the molecules in the first coordination sphere around molecule i are oriented as they would be in the solid. Furthermore, this Action is a multicolvar so you can calculate the s_i values for all the molecules in your system simultaneously and then determine the average, the number less than and so on.

Examples

In the example below the orientation of the molecules in the system is determined by calculating the vector that connects a pair of atoms. SMAC is then used to determine whether the molecules are sitting in a solid or liquid like environment. We can determine whether the environment is solid or liquid like because in the solid the torsional angle between the bond vectors on adjacent molecules is close to 0 or π. The final quantity that is output to the colvar file measures the number of molecules that have a SMAC parameter that is greater than 0.7. N.B. By using the indices of three atoms for each of the MOL keywords below we are telling PLUMED to use the first two numbers to determine the orientation of the molecule that will ultimately be used when calculating the θ_{ij} terms in the formula above. The atom with the third index meanwhile is used when we calculate r_{ij}.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SMAC.tmp
MOLECULES ...
MOL1=9,10,9
MOL2=89,90,89
MOL3=473,474,473
MOL4=1161,1162,1161
MOL5=1521,1522,1521
MOL6=1593,1594,1593
MOL7=1601,1602,1601
MOL8=2201,2202,2201
LABEL=m3
... MOLECULES

SMAC ...
   SPECIES=m3 LOWMEM
   KERNEL1={GAUSSIAN CENTER=0 SIGMA=0.480} KERNEL2={GAUSSIAN CENTER=pi SIGMA=0.480}
   SWITCH={RATIONAL R_0=0.6} MORE_THAN={RATIONAL R_0=0.7} SWITCH_COORD={EXP R_0=4}
   LABEL=s2
... SMAC

PRINT ARG=s2.* FILE=colvar
```

This second example works in a way that is very similar to the previous command. Now, however, the orientation of the molecules is determined by finding the plane that contains the positions of three atoms.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SMAC.tmp
PLANES ...
MOL1=9,10,11
MOL2=89,90,91
MOL3=473,474,475
MOL4=1161,1162,1163
MOL5=1521,1522,1523
MOL6=1593,1594,1595
MOL7=1601,1602,1603
MOL8=2201,2202,2203
VMEAN
LABEL=m3
... PLANES

SMAC ...
   SPECIES=m3 LOWMEM
   KERNEL1={GAUSSIAN CENTER=0 SIGMA=0.480} KERNEL2={GAUSSIAN CENTER=pi SIGMA=0.480}
   SWITCH={RATIONAL R_0=0.6} MORE_THAN={RATIONAL R_0=0.7} SWITCH_COORD={EXP R_0=3.0}
   LABEL=s2
... SMAC

PRINT ARG=s2.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean

min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

The atoms involved can be specified using

SPECIES	this keyword is used for colvars such as coordination number. In that context it specifies that plumed should calculate one coordination number for each of the atoms specified. Each of these coordination numbers specifies how many of the other specified atoms are within a certain cutoff of the central atom. You can specify the atoms here as another multicolvar action or using a MultiColvar↩ Filter or ActionVolume action. When you do so the quantity is calculated for those atoms specified in the previous multicolvar. This is useful if you would like to calculate the Steinhardt parameter for those atoms that have a coordination number more than four for example

Or alternatively by using

SPECIESA	this keyword is used for colvars such as the coordination number. In that context it species that plumed should calculate one coordination number for each of the atoms specified in SPECIESA. Each of these coordination numbers specifies how many of the atoms specifies using SPECIESB is within the specified cutoff. As with the species keyword the input can also be specified using the label of another multicolvar
SPECIESB	this keyword is used for colvars such as the coordination number. It must appear with SPECIESA. For a full explanation see the documentation for that keyword

Compulsory keywords

NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function
KERNEL	The kernels used in the function of the angle. You can use multiple instances of this keyword i.e. KERNEL1, KERNEL2, KERNEL3...
SWITCH_COORD	This keyword is used to define the coordination switching function.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.5.58 MTRANSFORM_BETWEEN

This is part of the multicolvar module

This action can be used to transform the colvar values calculated by a MultiColvar using a histogram bead

In this action each colvar, s_i, calculated by MultiColvar is transformed by a histogrambead function that is equal to one if the colvar is within a certain range and which is equal to zero otherwise. In other words, we compute:

$$f_i = \int_a^b K\left(\frac{s - s_i}{w}\right)$$

where a, b and w are parameters.

It is important to understand the distinction between what is done here and what is done by MFILTER_BETWEEN. In MFILTER_BETWEEN a weight, w_i for the colvar is calculated using the histogrambead. If one calculates the MEAN for MFILTER_BETWEEN one is thus calculating:

$$\mu = \frac{\sum_i f_i s_i}{\sum_i f_i}$$

In this action by contrast the colvar is being transformed by the histogrambead. If one thus calculates a MEAN for this action one computes:

$$\mu = \frac{\sum_{i=1}^{N} f_i}{N}$$

In other words, you are calculating the mean for the transformed colvar.

Examples

The following input gives an example of how a MTRANSFORM_BETWEEN action can be used to duplicate functionality that is elsewhere in PLUMED.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MTRANSFORM_BETWEEN.tmp
DISTANCES ...
 GROUPA=1-10 GROUPB=11-20
 LABEL=d1
... DISTANCES
MTRANSFORM_BETWEEN DATA=d1 LOWER=1.0 UPPER=2.0 SMEAR=0.5
```

In this case you can achieve the same result by using:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MTRANSFORM_BETWEEN.tmp
DISTANCES ...
 GROUPA=1-10 GROUPB=11-20
 BETWEEN={GAUSSIAN LOWER=1.0 UPPER=2.0}
... DISTANCES
```

(see DISTANCES)

The advantage of MTRANSFORM_BETWEEN comes, however, if you want to use transformed colvars as input for MULTICOLVARDENS

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
LOWER	the lower boundary for the range of interest
UPPER	the upper boundary for the range of interest
SMEAR	(default=0.5) the amount by which to smear the value for kernel density estimation

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...

LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
BEAD	This keywords is used if you want to employ an alternative to the function defined above. The following provides information on the histograbead that are available. When this keyword is present you no longer need the LOWER, U↩ PPER and SMEAR keywords.

5.5.59 MTRANSFORM_LESS

	This is part of the multicolvar module

This action can be used to transform the colvar values calculated by a multicovar using a switching function

In this action each colvar, s_i, calculated by MultiColvar is transformed by a switchingfunction function that is equal to one if the colvar is less than a certain target value and which is equal to zero otherwise. It is important to understand the distinction between what is done here and what is done by MFILTER_LESS. In MFILTER_LESS a weight, w_i for the colvar is calculated using the switchingfunction. If one calculates the MEAN for MFILTER_LESS one is thus calculating:

$$\mu = \frac{\sum_i \sigma(s_i) s_i}{\sum_i (s_i)}$$

where σ is the switchingfunction. In this action by contrast the colvar is being transformed by the switchingfunction. If one thus calculates a MEAN for this action one computes:

$$\mu = \frac{\sum_{i=1}^{N} (s_i)}{N}$$

In other words, you are calculating the mean for the transformed colvar.

Examples

The following input gives an example of how a MTRANSFORM_LESS action can be used to duplicate functionality that is elsewhere in PLUMED.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MTRANSFORM_LESS.tmp
DISTANCES ...
 GROUPA=1-10 GROUPB=11-20
 LABEL=d1
... DISTANCES
MTRANSFORM_LESS DATA=d1 SWITCH={GAUSSIAN D_0=1.5 R_0=0.00001}
```

In this case you can achieve the same result by using:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MTRANSFORM_LESS.tmp
DISTANCES ...
 GROUPA=1-10 GROUPB=11-20
 LESS_THAN={GAUSSIAN D_0=1.5 R_0=0.00001}
... DISTANCES
```

(see DISTANCES)

The advantage of MTRANSFORM_LESS comes, however, if you want to use transformed colvars as input for MULTICOLVARDENS

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

5.5.60 MTRANSFORM_MORE

This is part of the multicolvar module

This action can be used to transform the colvar values calculated by a multicolvar using one minus a switching function

In this action each colvar, s_i, calculated by MultiColvar is transformed by a switchingfunction function that is equal to one if the colvar is greater than a certain target value and which is equal to zero otherwise. It is important to understand the distinction between what is done here and what is done by MFILTER_MORE. In MFILTER_MORE a weight, w_i for the colvar is calculated using the histograbead. If one calculates the MEAN for MFILTER_MORE one is thus calculating:

$$\mu = \frac{\sum_i [1-\sigma(s_i)] s_i}{\sum_i [1-\sigma(s_i)]}$$

where σ is the switchingfunction. In this action by contrast the colvar is being transformed by the switchingfunction. If one thus calculates a MEAN for this action one computes:

$$\mu = \frac{\sum_{i=1}^{N} 1-\sigma(s_i)}{N}$$

In other words, you are calculating the mean for the transformed colvar.

Examples

The following input gives an example of how a MTRANSFORM_MORE action can be used to duplicate functionality that is elsewhere in PLUMED.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MTRANSFORM_MORE.tmp
DISTANCES ...
 GROUPA=1-10 GROUPB=11-20
 LABEL=d1
... DISTANCES
MTRANSFORM_MORE DATA=d1 SWITCH={GAUSSIAN D_0=1.5 R_0=0.00001}
```

In this case you can achieve the same result by using:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MTRANSFORM_MORE.tmp
DISTANCES ...
 GROUPA=1-10 GROUPB=11-20
 MORE_THAN={GAUSSIAN D_0=1.5 R_0=0.00001}
... DISTANCES
```

(see DISTANCES)

The advantage of MTRANSFORM_MORE comes, however, if you want to use transformed colvars as input for MULTICOLVARDENS

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

Compulsory keywords

DATA	The multicolvar that calculates the set of base quantities that we are interested in
NN	(default=6) The n parameter of the switching function
MM	(default=0) The m parameter of the switching function; 0 implies 2∗NN
D_0	(default=0.0) The d_0 parameter of the switching function
R_0	The r_0 parameter of the switching function

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i-\overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. When this keyword is present you no longer need the NN, MM, D_0 and R_0 keywords.

5.5.61 LWALLS

	This is part of the manyrestraints module
	It is only available if you configure PLUMED with ./configure –enable-modules=manyrestraints . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Add LOWER_WALLS restraints on all the multicolvar values

This action takes the set of values calculated by the colvar specified by label in the DATA keyword and places a restraint on each quantity, x, with the following functional form:

$$k((x-a+o)/s)^e$$

k (KAPPA) is an energy constant in internal unit of the code, s (EPS) a rescaling factor and e (EXP) the exponent determining the power law. By default: EXP = 2, EPS = 1.0, OFF = 0.

Examples

The following set of commands can be used to stop any of the 800 atoms in group A from moving more than 2.46425 nm in the z direction from atom 34137. This is done by adding a lower wall on the z-distance between all the atoms in group A and the position of 34137.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LWALLS.tmp
l: ZDISTANCES GROUPA=1-800 GROUPB=34137 NOPBC
LWALLS DATA=l AT=2.46465 KAPPA=150.0 EXP=2 EPS=1 OFFSET=0 LABEL=lwall
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potentials

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient

vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

DATA	certain actions in plumed work by calculating a list of variables and summing over them. This particular action can be used to calculate functions of these base variables or prints them to a file. This keyword thus takes the label of one of those such variables as input.
AT	the radius of the sphere
KAPPA	the force constant for the wall. The k_i in the expression for a wall.
OFFSET	(default=0.0) the offset for the start of the wall. The o_i in the expression for a wall.
EXP	(default=2.0) the powers for the walls. The e_i in the expression for a wall.
EPS	(default=1.0) the values for s_i in the expression for a wall

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.5.62 UWALLS

	This is part of the manyrestraints module
	It is only available if you configure PLUMED with ./configure –enable-modules=manyrestraints . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Add UPPER_WALLS restraints on all the multicolvar values

This action takes the set of values calculated by the colvar specified by label in the DATA keyword and places a restraint on each quantity, x, with the following functional form:

$$k((x-a+o)/s)^e$$

k (KAPPA) is an energy constant in internal unit of the code, s (EPS) a rescaling factor and e (EXP) the exponent determining the power law. By default: EXP = 2, EPS = 1.0, OFF = 0.

Examples

The following set of commands can be used to stop a cluster composed of 20 atoms subliming. The position of the center of mass of the cluster is calculated by the COM command labelled c1. The DISTANCES command labelled d1 is then used to calculate the distance between each of the 20 atoms in the cluster and the center of mass of the cluster. These distances are then passed to the UWALLS command, which adds a UPPER_WALLS restraint on each of them and thereby prevents each of them from moving very far from the center of mass of the cluster.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/UWALLS.tmp
COM ATOMS=1-20 LABEL=c1
DISTANCES GROUPA=c1 GROUPB=1-20 LABEL=d1
UWALLS DATA=d1 AT=2.5 KAPPA=0.2 LABEL=sr
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potentials

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

DATA	certain actions in plumed work by calculating a list of variables and summing over them. This particular action can be used to calculate functions of these base variables or prints them to a file. This keyword thus takes the label of one of those such variables as input.
AT	the radius of the sphere
KAPPA	the force constant for the wall. The k_i in the expression for a wall.
OFFSET	(default=0.0) the offset for the start of the wall. The o_i in the expression for a wall.
EXP	(default=2.0) the powers for the walls. The e_i in the expression for a wall.
EPS	(default=1.0) the values for s_i in the expression for a wall

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.6 Exploiting contact matrices

A contact matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. There are various ways of defining whether a pair of atoms/molecules are adjacent or not. For example we can say two atoms are adjacent if the distance between them is less than some cutoff. Alternatively, if we have a have a pair of molecules, we might state they are adjacent if their centers of mass are within a certain cutoff and if the two molecules have the same orientation. Two electronegative atoms might be said to be adjacent if there is a hydrogen bond between them. For these reasons then PLUMED contains all of the following methods for calculating an adjacency matrix

ALIGNED_MATRIX	Adjacency matrix in which two molecule are adjacent if they are within a certain cutoff and if they have the same orientation.
CONTACT_MATRIX	Adjacency matrix in which two atoms are adjacent if they are within a certain cutoff.
HBOND_MATRIX	Adjacency matrix in which two atoms are adjacent if there is a hydrogen bond between them.
HBPAMM_MATRIX	Adjacency matrix in which two electronegative atoms are adjacent if they are hydrogen bonded
SMAC_MATRIX	Adjacency matrix in which two molecules are adjacent if they are within a certain cutoff and if the angle between them is within certain ranges.
TOPOLOGY_MATRIX	Adjacency matrix in which two atoms are adjacent if they are connected topologically

Once you have calculated an adjacency matrix you can then perform any one of the following operations on this object in order to reduce it to a scalar number or a set of connected components.

CLUSTER_WITHSURFACE	Take a connected component that was found using a clustering algorithm and create a new cluster that contains those atoms that are in the cluster together with those atoms that are within a certain cutoff of the cluster.
COLUMNSUMS	Sum the columns of a contact matrix
DFSCLUSTERING	Find the connected components of the matrix using the depth first search clustering algorithm.
ROWSUMS	Sum the rows of a adjacency matrix.
SPRINT	Calculate SPRINT topological variables from an adjacency matrix.

If the function you have chosen reduces your contact matrix to a set of connected components you then need a method to convert these connected components into a scalar number or to output this information to a file. The various things that you can do with a set of connected components are listed below:

CLUSTER_DIAMETER	Print out the diameter of one of the connected components
CLUSTER_DISTRIBUTION	Calculate functions of the distribution of properties in your connected components.

CLUSTER_NATOMS	Gives the number of atoms in the connected component
CLUSTER_PROPERTIES	Calculate properties of the distribution of some quantities that are part of a connected component
DUMPGRAPH	Write out the connectivity of the nodes in the graph in dot format.
OUTPUT_CLUSTER	Output the indices of the atoms in one of the clusters identified by a clustering object

5.6.1 ALIGNED_MATRIX

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Adjacency matrix in which two molecule are adjacent if they are within a certain cutoff and if they have the same orientation.

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. These matrices can then be further analyzed using a number of other algorithms as is detailed in [32].

For this action the elements of the adjacency matrix are calculated using:

$$a_{ij} = \sigma_1(|\mathbf{r}_{ij}|)\sigma_2(\mathbf{v}_i.\mathbf{v}_j)$$

This form of adjacency matrix can only be used if the input species are objects that lie at a point in space and that have an orientation, $\mathbf{v}$. These orientations might represent the orientation of a molecule, which could be calculated using MOLECULES or PLANES, or it might be the complex vectors calculated using the Steinhardt parameters Q3, Q4 or Q6. In the expression above $\mathbf{r}_{ij}$ is the vector connecting the points in space where objects i and j find themselves and σ_1 is a switchingfunction that acts upon the magnitude of this vector. σ_2 is a second switchingfunction that acts on the dot product of the directors of the vectors that define the orientations of objects i and j.

Examples

The example input below is necessarily but gives you an idea of what can be achieved using this action. The orientations and positions of four molecules are defined using the MOLECULES action as the position of the centers of mass of the two atoms specified and the direction of the vector connecting the two atoms that were specified. A 4×4 matrix is then computed using the formula above. The ij-element of this matrix tells us whether or not atoms i and j are within 0.1 nm of each other and whether or not the dot-product of their orientation vectors is greater than 0.5. The sum of the rows of this matrix are then computed. The sums of the ith row of this matrix tells us how many of the molecules that are within the first coordination sphere of molecule i have an orientation that is similar to that of molecule i. We thus calculate the number of these "coordination numbers" that are greater than 1.0 and output this quantity to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ALIGNED_MATRIX.tmp
m1: MOLECULES MOL1=1,2 MOL2=3,4 MOL3=5,6 MOL4=7,8
mat: ALIGNED_MATRIX ATOMS=m1 SWITCH={RATIONAL R_0=0.1} ORIENTATION_SWITCH={RATIONAL R_0=0.1 D_MAX=0.5}
row: ROWSUMS MATRIX=mat MORE_THAN={RATIONAL D_0=1.0 R_0=0.1}
PRINT ARG=row.* FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	The list of molecules for which you would like to calculate the contact matrix. The molecules involved must have an orientation so your list will be a list of the labels of MultiColvar or MultiColvar functions as PLUMED calculates the orientations of molecules within these operations. Please note also that the majority of MultiColvar and MultiColvar functions do not calculate a molecular orientation.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Or alternatively by using

ATOMSA	The list of molecules that you would like to use for the rows of the contact matrix. The molecules involved must have an orientation so your list will be a list of the labels of MultiColvar or MultiColvar functions as PLUMED calculates the orientations of molecules within these operations. Please note also that the majority of MultiColvar and MultiColvar functions do not calculate a molecular orientation.
ATOMSB	The list of molecules that you would like to use for the columns of the contact matrix. The molecules involved must have an orientation so your list will be a list of the labels of MultiColvar or MultiColvar functions as PLUMED calculates the orientations of molecules within these operations. Please note also that the majority of MultiColvar and MultiColvar functions do not calculate a molecular orientation.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
HIGHMEM	(default=off) use a more memory intensive version of this collective variable
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available.. You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...
ORIENTATION_SWITCH	A switching function that transforms the dot product of the input vectors.. You can use multiple instances of this keyword i.e. ORIENTATION_SWITCH1, O↩ RIENTATION_SWITCH2, ORIENTATION_SWITCH3...

5.6.2 CONTACT_MATRIX

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Adjacency matrix in which two atoms are adjacent if they are within a certain cutoff.

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. These matrices can then be further analyzed using a number of other algorithms as is detailed in [32].

For this action the elements of the contact matrix are calculated using:

$$a_{ij} = \sigma(|\mathbf{r}_{ij}|)$$

where $|\mathbf{r}_{ij}|$ is the magnitude of the vector connecting atoms i and j and where σ is a switchingfunction.

Examples

The input shown below calculates a 6×6 matrix whose elements are equal to one if atom i and atom j are within 0.3 nm of each other and which is zero otherwise. The columns in this matrix are then summed so as to give the coordination number for each atom. The final quantity output in the colvar file is thus the average coordination number.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CONTACT_MATRIX.tmp
mat: CONTACT_MATRIX ATOMS=1-6 SWITCH={EXP D_0=0.2 R_0=0.1 D_MAX=0.66}
COLUMNSUMS MATRIX=mat MEAN LABEL=csums
PRINT ARG=csums.* FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	The list of atoms for which you would like to calculate the contact matrix. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
HIGHMEM	(default=off) use a more memory intensive version of this collective variable
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available. . You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...

5.6.3 HBOND_MATRIX

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Adjacency matrix in which two atoms are adjacent if there is a hydrogen bond between them.

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. These matrices can then be further analyzed using a number of other algorithms as is detailed in [32].

For this action the elements of the adjacency matrix are calculated using:

$$a_{ij} = \sigma_{oo}(|\mathbf{r}_{ij}|) \sum_{k=1}^{N} \sigma_{oh}(|\mathbf{r}_{ik}|)\sigma_{\theta}(\theta_{kij})$$

This expression was derived by thinking about how to detect if there is a hydrogen bond between atoms i and j. The notion is that if the hydrogen bond is present atoms i and j should be within a certain cutoff distance. In addition, there should be a hydrogen within a certain cutoff distance of atom i and this hydrogen should lie on or close to the vector connecting atoms i and j. As such $\sigma_{oo}(|\mathbf{r}_{ij}|)$ is a switchingfunction that acts on the modulus of the vector connecting atom i to atom j. The sum over k then runs over all the hydrogen atoms that are specified using using HYDROGEN keyword. $\sigma_{oh}(|\mathbf{r}_{ik}|)$ is a switchingfunction that acts on the modulus of the vector connecting atom i to atom k and $\sigma_{\theta}(\theta_{kij})$ is a switchingfunction that acts on the angle between the vector connecting atoms i and j and the vector connecting atoms i and k.

It is important to note that hydrogen bonds, unlike regular bonds, are asymmetric. In other words, the hydrogen atom does not sit at the mid point between the two other atoms in this three-center bond. As a result of this adjacency matrices calculated using HBOND_MATRIX are not symmetric like those calculated by CONTACT_MATRIX. One consequence of this fact is that the quantities found by performing ROWSUMS and COLUMNSUMS on a square HBOND_MATRIX are not the same as they would be if you performed ROWSUMS and COLUMNSUMS on a square CONTACT_MATRIX.

Examples

The following input can be used to analyze the number of hydrogen bonds each of the oxygen atoms in a box of water participates in. Each water molecule can participate in a hydrogen bond in one of two ways. It can either donate one of its hydrogen atom to the neighboring oxygen or it can accept a bond between the hydrogen of a neighboring water molecule and its own oxygen. The input below allows you to output information on the number of hydrogen bonds each of the water molecules donates and accepts. This information is output in two xyz files which each contain five columns of data. The first four of these columns are a label for the atom and the x, y and z position of the oxygen. The last column is then the number of accepted/donated hydrogen bonds.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/HBOND_MATRIX.tmp
mat: HBOND_MATRIX ATOMS=1-192:3 HYDROGENS=2-192:3,3-192:3 SWITCH={RATIONAL R_0=3.20} HSWITCH={RATIONAL R_0=2.3
rsums: ROWSUMS MATRIX=mat MEAN
csums: COLUMNSUMS MATRIX=mat MEAN
DUMPMULTICOLVAR DATA=rsums FILE=donors.xyz
DUMPMULTICOLVAR DATA=csums FILE=acceptors.xyz
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
sum	**SUM**	the sum of values

The atoms involved can be specified using

ATOMS	The list of atoms which can be part of a hydrogen bond. When this command is used the set of atoms that can donate a hydrogen bond is assumed to be the same as the set of atoms that can form hydrogen bonds. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices. For more information on how to specify lists of atoms see Groups and Virtual Atoms
HYDROGENS	The list of hydrogen atoms that can form part of a hydrogen bond. The atoms must be specified using a comma separated list, an index range or by using a GROUP. A list of hydrogen atoms is always required even if you specify the other atoms using DONORS and ACC↩ EPTORS as described below.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Or alternatively by using

DONORS	The list of atoms which can donate a hydrogen bond. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices
ACCEPTORS	The list of atoms which can accept a hydrogen bond. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
HIGHMEM	(default=off) use a more memory intensive version of this collective variable
SWITCH	The switchingfunction that specifies how close a pair of atoms must be together for there to be a hydrogen bond between them. You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...
HSWITCH	The switchingfunction that specifies how close the hydrogen must be to the donor atom of the hydrogen bond for it to be considered a hydrogen bond. You can use multiple instances of this keyword i.e. HSWITCH1, HSWITCH2, HSWITCH3...

ASWITCH	A switchingfunction that is used to specify what the angle between the vector connecting the donor atom to the acceptor atom and the vector connecting the donor atom to the hydrogen must be in order for it considered to be a hydrogen bond. You can use multiple instances of this keyword i.e. ASWITCH1, ASWI↩ TCH2, ASWITCH3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

5.6.4 HBPAMM_MATRIX

	This is part of the pamm module
	It is only available if you configure PLUMED with ./configure –enable-modules=pamm . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Adjacency matrix in which two electronegative atoms are adjacent if they are hydrogen bonded

Examples

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
sum	**SUM**	the sum of values

The atoms involved can be specified using

SITES	The list of atoms which can be part of a hydrogen bond. When this command is used the set of atoms that can donate a hydrogen bond is assumed to be the same as the set of atoms that can form hydrogen bonds. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices

Or alternatively by using

DONORS	The list of atoms which can donate a hydrogen bond. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices
ACCEPTORS	The list of atoms which can accept a hydrogen bond. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices

Or alternatively by using

HYDROGENS	The list of hydrogen atoms that can form part of a hydrogen bond. The atoms must be specified using a comma separated list, an index range or by using a GROUP. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

CLUSTERS	the name of the file that contains the definitions of all the kernels for PAMM. You can use multiple instances of this keyword i.e. CLUSTERS1, CLUSTERS2, CLUSTERS3...
REGULARISE	(default=0.001) don't allow the denominator to be smaller then this value

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
HIGHMEM	(default=off) use a more memory intensive version of this collective variable
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

5.6.5 SMAC_MATRIX

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Adjacency matrix in which two molecules are adjacent if they are within a certain cutoff and if the angle between them is within certain ranges.

In this case the elements of the adjacency matrix are calculated using:

$$A_{ij} = \sigma(r_{ij}) \sum_n K_n(\theta_{ij})$$

In this expression r_{ij} is the distance between molecule i and molecule j and $\sigma(r_{ij}$ is a switchingfunction that acts on this distance. The $K_n functions are kernelfunctions that take the torsion angle, θ_{ij}, between the internal orientation vectors for molecules i and j as input. These kernel functions should be set so that they are equal to one when the relative orientation of the molecules are as they are in the solid and equal to zero otherwise. As the above matrix element is a product of functions it is only equal to one when the centers of mass of molecules i and j are with a certain distance of each other and when the molecules are aligned in some desirable way.

Examples

In the following example an adjacency matrix is constructed in which the (i,j) element is equal to one if molecules i and j are within 6 angstroms of each other and if the torsional angle between the orientations of these molecules is close to 0 or π. The various connected components of this matrix are determined using the DFSCLUSTERING algorithm and then the size of the largest cluster of connects molecules is output to a colvar file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SMAC_MATRIX.tmp
UNITS LENGTH=A

MOLECULES ...
MOL1=1,2,1
MOL2=5,6,5
MOL3=9,10,9
MOL4=13,14,13
MOL5=17,18,17
```

```
LABEL=m1
... MOLECULES

SMAC_MATRIX ...
   ATOMS=m1 SWITCH={RATIONAL D_0=5.99 R_0=0.1 D_MAX=6.0}
   KERNEL1={TRIANGULAR CENTER=0 SIGMA=1.0} KERNEL2={TRIANGULAR CENTER=pi SIGMA=0.6}
   LABEL=smac
... SMAC_MATRIX

dfs1: DFSCLUSTERING MATRIX=smac
cc2: CLUSTER_NATOMS CLUSTERS=dfs1 CLUSTER=1
PRINT ARG=cc2 FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	The list of molecules for which you would like to calculate the contact matrix. The molecules involved must have an orientation so your list will be a list of the labels of MultiColvar or MultiColvar functions as PLUMED calculates the orientations of molecules within these operations. Please note also that the majority of MultiColvar and MultiColvar functions do not calculate a molecular orientation.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Or alternatively by using

ATOMSA	The list of molecules that you would like to use for the rows of the contact matrix. The molecules involved must have an orientation so your list will be a list of the labels of MultiColvar or MultiColvar functions as PLUMED calculates the orientations of molecules within these operations. Please note also that the majority of MultiColvar and MultiColvar functions do not calculate a molecular orientation.
ATOMSB	The list of molecules that you would like to use for the columns of the contact matrix. The molecules involved must have an orientation so your list will be a list of the labels of MultiColvar or MultiColvar functions as PLUMED calculates the orientations of molecules within these operations. Please note also that the majority of MultiColvar and MultiColvar functions do not calculate a molecular orientation.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
HIGHMEM	(default=off) use a more memory intensive version of this collective variable

SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available.. You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...
KERNEL	The various kernels that are used to determine whether or not the molecules are aligned. You can use multiple instances of this keyword i.e. KERNEL1, KERNEL2, KERNEL3...

5.6.6 TOPOLOGY_MATRIX

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Adjacency matrix in which two atoms are adjacent if they are connected topologically

Examples

Glossary of keywords and components

The atoms involved can be specified using

NODES	The list of atoms for which you would like to calculate the contact matrix. The atoms involved must be specified as a list of labels of MultiColvar or labels of a MultiColvar functions actions. If you would just like to use the atomic positions you can use a DENSITY command to specify a group of atoms. Specifying your atomic positions using labels of other MultiColvar or MultiColvar functions commands is useful, however, as you can then exploit a much wider variety of functions of the contact matrix as described in Exploiting contact matrices. For more information on how to specify lists of atoms see Groups and Virtual Atoms
ATOMS	. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DENSITY_THRESHOLD	
SIGMA	the width of the function to be used for kernel density estimation
KERNEL	(default=gaussian) the type of kernel function to be used

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
HIGHMEM	(default=off) use a more memory intensive version of this collective variable
SWITCH	This keyword is used if you want to employ an alternative to the continuous switching function defined above. The following provides information on the switchingfunction that are available.. You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...
RADIUS	. You can use multiple instances of this keyword i.e. RADIUS1, RADIUS2, RADIUS3...
CYLINDER_SWITCH	a switching function on the distance from the center of the cylinder. You can use multiple instances of this keyword i.e. CYLINDER_SWITCH1, CYLINDE↩ R_SWITCH2, CYLINDER_SWITCH3...
BIN_SIZE	the size to use for the bins. You can use multiple instances of this keyword i.e. BIN_SIZE1, BIN_SIZE2, BIN_SIZE3...

5.6.7 CLUSTER_WITHSURFACE

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Take a connected component that was found using a clustering algorithm and create a new cluster that contains those atoms that are in the cluster together with those atoms that are within a certain cutoff of the cluster.

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. When analyzing these matrix we can treat them as a graph and find connected components using some clustering algorithm. This action is used in tandem with this form of analysis and takes one of the connected components that was found during this analysis and creates a new cluster that includes all the atoms within the connected component that was found together that were within a certain cutoff distance of the atoms in the connected component. This form of analysis has been used successfully in the forward flux sampling simulations described in this paper [33]

Examples

The following input uses PLUMED to calculate a adjacency matrix that connects a pair of atoms if they both have a coordination number that is less than 13.5 and if they are within 0.38 nm of each other. Depth first search clustering is used to find the connected components in this matrix. The number of atoms with indices that are between 1 and 1996 and that are either in the second largest cluster or that are within within 0.3 nm of one of the atoms within the the second largest cluster are then counted and this number of atoms is output to a file called size. In addition the indices of the atoms that were counted are output to a file called dfs2.dat.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CLUSTER_WITHSURFACE.tmp
c1: COORDINATIONNUMBER SPECIES=1-1996 SWITCH={CUBIC D_0=0.34 D_MAX=0.38}
cf: MFILTER_LESS DATA=c1 SWITCH={CUBIC D_0=13 D_MAX=13.5}
mat: CONTACT_MATRIX ATOMS=cf SWITCH={CUBIC D_0=0.34 D_MAX=0.38}
dfs: DFSCLUSTERING MATRIX=mat
clust2a: CLUSTER_WITHSURFACE CLUSTERS=dfs RCUT_SURF=0.3
size2a: CLUSTER_NATOMS CLUSTERS=clust2a CLUSTER=2
PRINT ARG=size2a FILE=size FMT=%8.4f
OUTPUT_CLUSTER CLUSTERS=clust2a CLUSTER=2 FILE=dfs2.dat
```

Glossary of keywords and components

Compulsory keywords

CLUSTERS	the label of the action that does the clustering
RCUT_SURF	you also have the option to find the atoms on the surface of the cluster. An atom must be within this distance of one of the atoms of the cluster in order to be considered a surface atom

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.6.8 COLUMNSUMS

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Sum the columns of a contact matrix

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. This action allows you to calculate the sum of the columns in this adjacency matrix and to then calculate further functions of these quantities.

Examples

The first instruction in the following input file tells PLUMED to compute a 10×10 matrix in which the ij-element tells you whether atoms i and j are within 1.0 nm of each other. The numbers in each of this rows are then added together and the average value is computed. As such the following input provides an alternative method for calculating the coordination numbers of atoms 1 to 10.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COLUMNSUMS.tmp
mat: CONTACT_MATRIX ATOMS=1-10 SWITCH={RATIONAL R_0=1.0}
rsums: COLUMNSUMS MATRIX=mat MEAN
PRINT ARG=rsums.* FILE=colvar
```

The following input demonstrates another way that an average coordination number can be computed. This input calculates the number of atoms with indices between 1 and 5 that are within the first coordination spheres of each of the atoms within indices between 6 and 15. The average coordination number is then calculated from these fifteen coordination numbers and this quantity is output to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COLUMNSUMS.tmp
mat2: CONTACT_MATRIX ATOMSA=1-5 ATOMSB=6-15 SWITCH={RATIONAL R_0=1.0}
rsums: COLUMNSUMS MATRIX=mat2 MEAN
PRINT ARG=rsums.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

Compulsory keywords

MAT↩ RIX	the action that calculates the adjacency matrix vessel we would like to analyze

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...

MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.6.9 DFSCLUSTERING

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Find the connected components of the matrix using the depth first search clustering algorithm.

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are

adjacent or not. As detailed in [32] these matrices provide a representation of a graph and can thus can be analyzed using tools from graph theory. This particular action performs a depth first search clustering to find the connected components of this graph. You can read more about depth first search here:

https://en.wikipedia.org/wiki/Depth-first_search

This action is useful if you are looking at a phenomenon such as nucleation where the aim is to detect the sizes of the crystalline nuclei that have formed in your simulation cell.

Examples

The input below calculates the coordination numbers of atoms 1-100 and then computes the an adjacency matrix whose elements measures whether atoms i and j are within 0.55 nm of each other. The action labelled dfs then treats the elements of this matrix as zero or ones and thus thinks of the matrix as defining a graph. This dfs action then finds the largest connected component in this graph. The sum of the coordination numbers for the atoms in this largest connected component are then computed and this quantity is output to a colvar file. The way this input can be used is described in detail in [32].

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DFSCLUSTERING.tmp
lq: COORDINATIONNUMBER SPECIES=1-100 SWITCH={CUBIC D_0=0.45  D_MAX=0.55} LOWMEM
cm: CONTACT_MATRIX ATOMS=lq  SWITCH={CUBIC D_0=0.45  D_MAX=0.55}
dfs: DFSCLUSTERING MATRIX=cm
clust1: CLUSTER_PROPERTIES CLUSTERS=dfs CLUSTER=1 SUM
PRINT ARG=clust1.* FILE=colvar
```

Glossary of keywords and components

Compulsory keywords

MATRIX	the action that calculates the adjacency matrix vessel we would like to analyze
MAXCONNECT	(default=0) maximum number of connections that can be formed by any given node in the graph. By default this is set equal to zero and the number of connections is set equal to the number of nodes. You only really need to set this if you are working with a very large system and memory is at a premium

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.6.10 ROWSUMS

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Sum the rows of a adjacency matrix.

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. This action allows you to calculate the sum of the rows in this adjacency matrix and to then calculate further functions of these quantities.

Examples

The first instruction in the following input file tells PLUMED to compute a 10×10 matrix in which the ij-element tells you whether atoms i and j are within 1.0 nm of each other. The numbers in each of this rows are then added together and the average value is computed. As such the following input provides an alternative method for calculating the coordination numbers of atoms 1 to 10.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ROWSUMS.tmp
mat: CONTACT_MATRIX ATOMS=1-10 SWITCH={RATIONAL R_0=1.0}
rsums: ROWSUMS MATRIX=mat MEAN
PRINT ARG=rsums.* FILE=colvar
```

The following input demonstrates another way that an average coordination number can be computed. This input calculates the number of atoms with indices between 6 and 15 that are within the first coordination spheres of each of the atoms within indices between 1 and 5. The average coordination number is then calculated from these five coordination numbers and this quantity is output to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ROWSUMS.tmp
mat2: CONTACT_MATRIX ATOMSA=1-5 ATOMSB=6-15 SWITCH={RATIONAL R_0=1.0}
rsums: ROWSUMS MATRIX=mat2 MEAN
PRINT ARG=rsums.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

Compulsory keywords

MAT↩ RIX	the action that calculates the adjacency matrix vessel we would like to analyze

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest

HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEA↩ N2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log \sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta \log \sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPO↩ NENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

5.6.11 SPRINT

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate SPRINT topological variables from an adjacency matrix.

The SPRINT topological variables are calculated from the largest eigenvalue, λ of an $n \times n$ adjacency matrix and its corresponding eigenvector, $\mathbf{V}$, using:

$$s_i = \sqrt{n}\lambda v_i$$

You can use different quantities to measure whether or not two given atoms/molecules are adjacent or not in the adjacency matrix. The simplest measure of adjacency is is whether two atoms/molecules are within some cutoff of each other. Further complexity can be added by insisting that two molecules are adjacent if they are within a certain distance of each other and if they have similar orientations.

Examples

This example input calculates the 7 SPRINT coordinates for a 7 atom cluster of Lennard-Jones atoms and prints their values to a file. In this input the SPRINT coordinates are calculated in the manner described in [34] so two atoms are adjacent if they are within a cutoff:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SPRINT.tmp
DENSITY SPECIES=1-7 LABEL=d1
CONTACT_MATRIX ATOMS=d1 SWITCH={RATIONAL R_0=0.1} LABEL=mat
SPRINT MATRIX=mat LABEL=ss
PRINT ARG=ss.* FILE=colvar
```

This example input calculates the 14 SPRINT coordinates for a molecule composed of 7 hydrogen and 7 carbon atoms. Once again two atoms are adjacent if they are within a cutoff:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SPRINT.tmp
DENSITY SPECIES=1-7 LABEL=c
DENSITY SPECIES=8-14 LABEL=h

CONTACT_MATRIX ...
  ATOMS=c,h
  SWITCH11={RATIONAL R_0=2.6 NN=6 MM=12}
  SWITCH12={RATIONAL R_0=2.2 NN=6 MM=12}
  SWITCH22={RATIONAL R_0=2.2 NN=6 MM=12}
  LABEL=mat
... CONTACT_MATRIX

SPRINT MATRIX=mat LABEL=ss

PRINT ARG=ss.* FILE=colvar
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	all n sprint coordinates are calculated and then stored in increasing order. the smallest sprint coordinate will be labeled *label*.coord-0, the second smallest will be labelled *label*.coord-1 and so on

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

MAT↩ RIX	the action that calculates the adjacency matrix vessel we would like to analyze

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.6.12 CLUSTER_DIAMETER

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Print out the diameter of one of the connected components

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. When analyzing these matrix we can treat them as a graph and find connected components using some clustering algorithm. This action is used in tandem with this form of analysis to output the largest of the distances between the pairs of atoms that are connected together in a particular connected component. It is important to note that the quantity that is output by this action cannot be differentiated. As such it cannot be used as a collective variable in a biased simulation.

Examples

The following input uses PLUMED to calculate a adjacency matrix that connects a pair of atoms if they both have a coordination number that is greater than 2.0 and if they are within 6.0 nm of each other. Depth first search clustering is used to find the connected components in this matrix. The distance between every pair of atoms that are within the largest of the clusters found is then calculated and the largest of these distances is output to a file named colvar.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CLUSTER_DIAMETER.tmp
# Calculate coordination numbers
c1: COORDINATIONNUMBER SPECIES=1-512 SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
# Select coordination numbers that are more than 2.0
cf: MFILTER_MORE DATA=c1 SWITCH={RATIONAL D_0=2.0 R_0=0.1} LOWMEM
# Build a contact matrix
mat: CONTACT_MATRIX ATOMS=cf SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
# Find largest cluster
dfs: DFSCLUSTERING MATRIX=mat
clust1: CLUSTER_PROPERTIES CLUSTERS=dfs CLUSTER=1
dia: CLUSTER_DIAMETER CLUSTERS=dfs CLUSTER=1
PRINT ARG=dia FILE=colvar
```

Glossary of keywords and components

Compulsory keywords

CLUSTERS	the label of the action that does the clustering
CLUSTER	(default=1) which cluster would you like to look at 1 is the largest cluster, 2 is the second largest, 3 is the the third largest and so on.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.6.13 CLUSTER_DISTRIBUTION

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate functions of the distribution of properties in your connected components.

This collective variable was developed for looking at nucleation phenomena, where you are interested in using studying the behavior of atoms in small aggregates or nuclei. In these sorts of problems you might be interested in the distribution of the sizes of the clusters in your system. A detailed description of this CV can be found in [32].

Examples

The input provided below calculates the local q6 Steinhardt parameter on each atom. The coordination number that atoms with a high value for the local q6 Steinhardt parameter have with other atoms that have a high value for the local q6 Steinhardt parameter is then computed. A contact matrix is then computed that measures whether atoms atoms i and j have a high value for this coordination number and if they are within 3.6 nm of each other. The connected components of this matrix are then found using a depth first clustering algorithm on the corresponding graph. The number of components in this graph that contain more than 27 atoms is then computed. As discussed in [32] an input similar to this one was used to analyze the formation of a polycrystal of GeTe from amorphous GeTe.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CLUSTER_DISTRIBUTION.tmp
q6: Q6 SPECIES=1-300 SWITCH={GAUSSIAN D_0=5.29 R_0=0.01 D_MAX=5.3} LOWMEM
lq6: LOCAL_Q6 SPECIES=q6 SWITCH={GAUSSIAN D_0=5.29 R_0=0.01 D_MAX=5.3} LOWMEM
flq6: MFILTER_MORE DATA=lq6 SWITCH={GAUSSIAN D_0=0.19 R_0=0.01 D_MAX=0.2}
cc: COORDINATIONNUMBER SPECIES=flq6 SWITCH={GAUSSIAN D_0=3.59 R_0=0.01 D_MAX=3.6}
fcc: MFILTER_MORE DATA=cc SWITCH={GAUSSIAN D_0=5.99 R_0=0.01 D_MAX=6.0}
mat: CONTACT_MATRIX ATOMS=fcc SWITCH={GAUSSIAN D_0=3.59 R_0=0.01 D_MAX=3.6}
dfs: DFSCLUSTERING MATRIX=mat
nclust: CLUSTER_DISTRIBUTION CLUSTERS=dfs TRANSFORM={GAUSSIAN D_0=5.99 R_0=0.01 D_MAX=6.0} MORE_THAN={GAUSSIAN
PRINT ARG=nclust.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

Compulsory keywords

CLUSTERS	the label of the action that does the clustering
TRANSFORM	(default=none) the switching function to use to convert the crystallinity parameter to a number between zero and one

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
INVERSE_TRANSFORM	(default=off) when TRANSFORM appears alone the input symmetry functions, x are transformed used $1-s(x)$ where $s(x)$ is a switching function. When this option is used you instead transform using $s(x)$ only.
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0-\sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THA↩ N3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRA↩ M3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...

5.6.14 CLUSTER_NATOMS

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Gives the number of atoms in the connected component

As discussed in the section of the manual on Exploiting contact matrices a useful tool for developing complex collective variables is the notion of the so called adjacency matrix. An adjacency matrix is an $N \times N$ matrix in which the ith, jth element tells you whether or not the ith and jth atoms/molecules from a set of N atoms/molecules are adjacent or not. When analyzing these matrix we can treat them as a graph and find connected components using some clustering algorithm. This action is used in tandem with this form of analysis to output the number of atoms that are connected together in a particular connected component. It is important to note that the quantity that is output by this action cannot be differentiated. As such it cannot be used as a collective variable in a biased simulation.

Examples

The following input uses PLUMED to calculate a adjacency matrix that connects a pair of atoms if they both have a coordination number that is greater than 2.0 and if they are within 6.0 nm of each other. Depth first search clustering is used to find the connected components in this matrix and then the number of atoms in the largest cluster is found. This quantity is then output to a file called colvar

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CLUSTER_NATOMS.tmp
# Calculate coordination numbers
c1: COORDINATIONNUMBER SPECIES=1-512 SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
# Select coordination numbers that are more than 2.0
cf: MFILTER_MORE DATA=c1 SWITCH={RATIONAL D_0=2.0 R_0=0.1} LOWMEM
# Build a contact matrix
mat: CONTACT_MATRIX ATOMS=cf SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
# Find largest cluster
dfs: DFSCLUSTERING MATRIX=mat
clust1: CLUSTER_PROPERTIES CLUSTERS=dfs CLUSTER=1
nat: CLUSTER_NATOMS CLUSTERS=dfs CLUSTER=1
PRINT ARG=nat FILE=COLVAR
```

Glossary of keywords and components

Compulsory keywords

CLUSTERS	the label of the action that does the clustering
CLUSTER	(default=1) which cluster would you like to look at 1 is the largest cluster, 2 is the second largest, 3 is the the third largest and so on.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

5.6.15 CLUSTER_PROPERTIES

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate properties of the distribution of some quantities that are part of a connected component

This collective variable was developed for looking at nucleation phenomena, where you are interested in using studying the behavior of atoms in small aggregates or nuclei. In these sorts of problems you might be interested in the degree the atoms in a nucleus have adopted their crystalline structure or (in the case of heterogeneous nucleation of a solute from a solvent) you might be interested in how many atoms are present in the largest cluster [32].

Examples

The input below calculates the coordination numbers of atoms 1-100 and then computes the an adjacency matrix whose elements measures whether atoms i and j are within 0.55 nm of each other. The action labelled dfs then treats the elements of this matrix as zero or ones and thus thinks of the matrix as defining a graph. This dfs action then finds the largest connected component in this graph. The sum of the coordination numbers for the atoms in this largest connected component are then computed and this quantity is output to a colvar file. The way this input can be used is described in detail in [32].

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CLUSTER_PROPERTIES.tmp
lq: COORDINATIONNUMBER SPECIES=1-100 SWITCH={CUBIC D_0=0.45  D_MAX=0.55} LOWMEM
cm: CONTACT_MATRIX ATOMS=lq  SWITCH={CUBIC D_0=0.45  D_MAX=0.55}
dfs: DFSCLUSTERING MATRIX=cm
clust1: CLUSTER_PROPERTIES CLUSTERS=dfs CLUSTER=1 SUM
PRINT ARG=clust1.* FILE=colvar
```

Glossary of keywords and components

Description of components

When the label of this action is used as the input for a second you are not referring to a scalar quantity as you are in regular collective variables. The label is used to reference the full set of quantities calculated by the action. This is usual when using MultiColvar functions. Generally when doing this the previously calculated multicolvar will be referenced using the DATA keyword rather than ARG.

This Action can be used to calculate the following scalar quantities directly. These quantities are calculated by employing the keywords listed below. These quantities can then be referenced elsewhere in the input file by using this Action's label followed by a dot and the name of the quantity. Some of them can be calculated multiple times with different parameters. In this case the quantities calculated can be referenced elsewhere in the input by using the name of the quantity followed by a numerical identifier e.g. *label*.lessthan-1, *label*.lessthan-2 etc. When doing this and, for clarity we have made it so that the user can set a particular label for each of the components. As such by using the LABEL keyword in the description of the keyword input you can customize the component name

Quantity	Keyword	Description
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

CLUSTERS	the label of the action that does the clustering
CLUSTER	(default=1) which cluster would you like to look at 1 is the largest cluster, 2 is the second largest, 3 is the the third largest and so on.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
MEAN	take the mean of these variables. The final value can be referenced using *label*.mean. You can use multiple instances of this keyword i.e. MEAN1, MEAN2, MEAN3... The corresponding values are then referenced using *label*.mean-1, *label*.mean-2, *label*.mean-3...
MORE_THAN	calculate the number of variables more than a certain target value. This quantity is calculated using $\sum_i 1.0 - \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.morethan. You can use multiple instances of this keyword i.e. MORE_THAN1, MORE_THAN2, MORE_THAN3... The corresponding values are then referenced using *label*.morethan-1, *label*.morethan-2, *label*.morethan-3...
LESS_THAN	calculate the number of variables less than a certain target value. This quantity is calculated using $\sum_i \sigma(s_i)$, where $\sigma(s)$ is a switchingfunction. The final value can be referenced using *label*.lessthan. You can use multiple instances of this keyword i.e. LESS_THAN1, LESS_THAN2, LESS_THAN3... The corresponding values are then referenced using *label*.lessthan-1, *label*.lessthan-2, *label*.lessthan-3...
VMEAN	calculate the norm of the mean vector. The final value can be referenced using *label*.vmean. You can use multiple instances of this keyword i.e. VMEAN1, VMEAN2, VMEAN3... The corresponding values are then referenced using *label*.vmean-1, *label*.vmean-2, *label*.vmean-3...
VSUM	calculate the norm of the sum of vectors. The final value can be referenced using *label*.vsum. You can use multiple instances of this keyword i.e. VSUM1, VSUM2, VSUM3... The corresponding values are then referenced using *label*.vsum-1, *label*.vsum-2, *label*.vsum-3...
BETWEEN	calculate the number of values that are within a certain range. These quantities are calculated using kernel density estimation as described on histogrambead. The final value can be referenced using *label*.between. You can use multiple instances of this keyword i.e. BETWEEN1, BETWEEN2, BETWEEN3... The corresponding values are then referenced using *label*.between-1, *label*.between-2, *label*.between-3...
HISTOGRAM	calculate how many of the values fall in each of the bins of a histogram. This shortcut allows you to calculates NBIN quantities like BETWEEN. The final value can be referenced using *label*.histogram. You can use multiple instances of this keyword i.e. HISTOGRAM1, HISTOGRAM2, HISTOGRAM3... The corresponding values are then referenced using *label*.histogram-1, *label*.histogram-2, *label*.histogram-3...
MOMENTS	calculate the moments of the distribution of collective variables. The mth moment of a distribution is calculated using $\frac{1}{N}\sum_{i=1}^{N}(s_i - \overline{s})^m$, where $\overline{s}$ is the average for the distribution. The moments keyword takes a lists of integers as input or a range. Each integer is a value of m. The final calculated values can be referenced using moment- m. You can use the COMPONENT keyword in this action but the syntax is slightly different. If you would like the second and third moments of the third component you would use MOMENTS={COMPONENT=3 MOMENTS=2-3}. The moments would then be referred to using the labels moment-3-2 and moment-3-3. This syntax is also required if you are using numbered MOMENT keywords i.e. MOMENTS1, MOMENTS2...

ALT_MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = -\frac{1}{\beta}\log\sum_i \exp\left(-\beta s_i\right)$ The value of β in this function is specified using (BETA= β). The final value can be referenced using *label*.altmin. You can use multiple instances of this keyword i.e. ALT_MIN1, ALT_MIN2, ALT_MIN3... The corresponding values are then referenced using *label*.altmin-1, *label*.altmin-2, *label*.altmin-3...
MIN	calculate the minimum value. To make this quantity continuous the minimum is calculated using $\min = \frac{\beta}{\log\sum_i \exp\left(\frac{\beta}{s_i}\right)}$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.min. You can use multiple instances of this keyword i.e. MIN1, MIN2, MIN3... The corresponding values are then referenced using *label*.min-1, *label*.min-2, *label*.min-3...
MAX	calculate the maximum value. To make this quantity continuous the maximum is calculated using $\max = \beta\log\sum_i \exp\left(\frac{s_i}{\beta}\right)$ The value of β in this function is specified using (BETA= β) The final value can be referenced using *label*.max. You can use multiple instances of this keyword i.e. MAX1, MAX2, MAX3... The corresponding values are then referenced using *label*.max-1, *label*.max-2, *label*.max-3...
SUM	calculate the sum of all the quantities. The final value can be referenced using *label*.sum. You can use multiple instances of this keyword i.e. SUM1, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...
LOWEST	this flag allows you to recover the lowest of these variables. The final value can be referenced using *label*.lowest
HIGHEST	this flag allows you to recover the highest of these variables. The final value can be referenced using *label*.highest

5.6.16 DUMPGRAPH

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Write out the connectivity of the nodes in the graph in dot format.

Examples

Glossary of keywords and components

Compulsory keywords

MATRIX	the action that calculates the adjacency matrix vessel we would like to analyze
STRIDE	(default=1) the frequency with which you would like to output the graph
FILE	the name of the file on which to output the data
MAXCONNECT	(default=0) maximum number of connections that can be formed by any given node in the graph. By default this is set equal to zero and the number of connections is set equal to the number of nodes. You only really need to set this if you are working with a very large system and memory is at a premium

5.6.17 OUTPUT_CLUSTER

	This is part of the adjmat module
	It is only available if you configure PLUMED with ./configure –enable-modules=adjmat . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Output the indices of the atoms in one of the clusters identified by a clustering object

This action provides one way of getting output from a DFSCLUSTERING calculation. The output in question here is either

- a file that contains a list of the atom indices that form part of one of the clusters that was identified using DFSCLUSTERING
- an xyz file containing the positions of the atoms in one of the the clusters that was identified using DFSCLUSTERING

Notice also that if you choose to output an xyz file you can ask PLUMED to try to reconstruct the cluster taking the periodic boundary conditions into account by using the MAKE_WHOLE flag.

Examples

The input shown below identifies those atoms with a coordination number less than 13 and then constructs a contact matrix that describes the connectivity between the atoms that satisfy this criteria. The DFS algorithm is then used to find the connected components in this matrix and the indices of the atoms in the largest connected component are then output to a file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OUTPUT_CLUSTER.tmp
c1: COORDINATIONNUMBER SPECIES=1-1996 SWITCH={CUBIC D_0=0.34 D_MAX=0.38}
cf: MFILTER_LESS DATA=c1 SWITCH={CUBIC D_0=13 D_MAX=13.5}
mat: CONTACT_MATRIX ATOMS=cf SWITCH={CUBIC D_0=0.34 D_MAX=0.38}
dfs: DFSCLUSTERING MATRIX=mat
OUTPUT_CLUSTER CLUSTERS=dfs CLUSTER=1 FILE=dfs.dat
```

Glossary of keywords and components

Compulsory keywords

CLUSTERS	the action that performed the clustering
CLUSTER	(default=1) which cluster would you like to look at 1 is the largest cluster, 2 is the second largest, 3 is the the third largest and so on
STRIDE	(default=1) the frequency with which you would like to output the atoms in the cluster
FILE	the name of the file on which to output the details of the cluster
MAXDEPTH	(default=6) maximum depth for searches over paths to reconstruct clusters for PBC
MAXGOES	(default=200) number of times to run searches to reconstruct clusters

Options

MAKE_WHOLE	(default=off) reconstruct the clusters and remove all periodic boundary conditions.

Chapter 6

Analysis

PLUMED can be used to analyze trajectories either on the fly during an MD run or via post processing a trajectory using driver. A molecular dynamics trajectory is in essence an ordered set of configurations of atoms. Trajectory analysis algorithms are methods that allow us to extract meaningful information from this extremely high-dimensionality information. In extracting this information much of the information in the trajectory will be discarded and assumed to be irrelevant to the problem at hand. For example, when we calculate a histogram from a trajectory we throw away all information on the order the frames were visited during the trajectory. We instead opt to display a time average that shows the parts of configuration space that were
visited most frequently. There are many situations in which this is a reasonable thing to do as we know that time averages are equivalent to ensemble averages in the long timescale limit and that these average probabilities of being in different parts of configuration space, $P(s)$, are thus related to the underlying free energy, $F(s)$, via:

$$F(s) = -k_B T \ln P(s)$$

About the simplest form of analysis that PLUMED can perform involves printing information to a file. PLUMED can output various different kinds of information to files as described below:

COMMITTOR	Does a committor analysis.
DUMPATOMS	Dump selected atoms on a file.
DUMPDERIVATIVES	Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).
DUMPFORCES	Dump the force acting on one of a values in a file.
DUMPMASSCHARGE	Dump masses and charges on a selected file.
DUMPMULTICOLVAR	Dump atom positions and multicolvar on a file.
DUMPPROJECTIONS	Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).
PRINT	Print quantities to a file.
UPDATE_IF	Conditional update of other actions.

The UPDATE_IF action allows you to do more complex things using the above print commands. As detailed in the documentation for UPDATE_IF when you put any of the above actions within an UPDATE_IF block then data will only be output to the file if colvars are within particular ranges. In other words, the above printing commands, in tandem with UPDATE_IF, allow you to identify the frames in your trajectory that satisfy some particular criteria and output information on those frames only.

Another useful command is the COMMITTOR command. As detailed in the documentation for COMMITTOR this command tells PLUMED (and the underlying MD code) to stop the calculation one some criteria is satisfied, alternatively one can use it to keep track of the number of times a criteria is satisfied.

A number of more complicated forms of analysis can be performed that take a number of frames from the trajectory as input. In all these commands the STRIDE keyword is used to tell PLUMED how frequently to collect data from the trajectory. In all these methods the output from the analysis is a form of ensemble average. If you are running with a bias it is thus likely that you may want to reweight the trajectory frames in order to remove the effect the bias has on the static behavior of the system. The following methods can thus be used to calculate weights for the various trajectory frames so that the final ensemble average is an average for the canonical ensemble at the appropriate temperature.

6.1 Reweighting and Averaging

REWEIGHT_BIAS	Calculate weights for ensemble averages that negate the effect the bias has on the region of phase space explored
REWEIGHT_METAD	Calculate the weights configurations should contribute to the histogram in a simulation in which a metadynamics bias acts upon the system.
REWEIGHT_TEMP_PRESS	Calculate weights for ensemble averages at temperatures and/or pressures different than those used in your original simulation.
REWEIGHT_WHAM	Calculate the weights for configurations using the weighted histogram analysis method.
WHAM_HISTOGRAM	This can be used to output the a histogram using the weighted histogram technique
WHAM_WEIGHTS	Calculate and output weights for configurations using the weighted histogram analysis method.

You can then calculate ensemble averages using the following actions.

AVERAGE	Calculate the ensemble average of a collective variable
HISTOGRAM	Accumulate the average probability density along a few CVs from a trajectory.
MULTICOLVARDENS	Evaluate the average value of a multicolvar on a grid.

For many of the above commands data is accumulated on the grids. These grids can be further analyzed using one of the actions detailed below at some time.

CONVERT_TO_FES	Convert a histogram, H(x), to a free energy surface using F(x) = -k_B T ln H(x).
DUMPCUBE	Output a three dimensional grid using the Gaussian cube file format.
DUMPGRID	Output the function on the grid to a file with the PLUMED grid format.
FIND_CONTOUR	Find an isocontour in a smooth function.
FIND_CONTOUR_SURFACE	Find an isocontour by searching along either the x, y or z direction.
FIND_SPHERICAL_CONTOUR	Find an isocontour in a three dimensional grid by searching over a Fibonacci sphere.
FOURIER_TRANSFORM	Compute the Discrete Fourier Transform (DFT) by means of FFTW of data stored on a 2D grid.
GRID_TO_XYZ	Output the function on the grid to an xyz file
INTEGRATE_GRID	Calculate the total integral of the function on the input grid
INTERPOLATE_GRID	Interpolate a smooth function stored on a grid onto a grid with a smaller grid spacing.

As an example the following set of commands instructs PLUMED to calculate the distance between atoms 1 and 2

for every fifth frame in the trajectory and to accumulate a histogram from this data which will be output every 100 steps (i.e. when 20 distances have been added to the histogram).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AnalysisPP.md
x: DISTANCE ATOMS=1,2
h: HISTOGRAM ARG=x GRID_MIN=0.0 GRID_MAX=3.0 GRID_BIN=100 BANDWIDTH=0.1 STRIDE=5
DUMPGRID GRID=h FILE=histo STRIDE=100
```

It is important to note when using commands such as the above the first frame in the trajectory is assumed to be the initial configuration that was input to the MD code. It is thus ignored. Furthermore, if you are running with driver and you would like to analyze the whole trajectory (without specifying its length) and then print the result you simply call DUMPGRID (or any of the commands above) without a STRIDE keyword as shown in the example below.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AnalysisPP.md
x: DISTANCE ATOMS=1,2
h: HISTOGRAM ARG=x GRID_MIN=0.0 GRID_MAX=3.0 GRID_BIN=100 BANDWIDTH=0.1 STRIDE=5
DUMPGRID GRID=h FILE=histo
```

Please note that even with this calculation the first frame in the trajectory is ignored when computing the histogram.

Notice that all the commands for calculating smooth functions described above calculate some sort of average. There are two ways that you may wish to average the data in your trajectory:

- You might want to calculate block averages in which the first NN frames in your trajectory are averaged separately to the second block of N frames. If this is the case you should use the keyword CLEAR in the input to the action that calculates the smooth function. This keyword is used to specify how frequently you are clearing the stored data.
- You might want to calculate an accumulate an average over the whole trajectory and output the average accumulated at step N, step $2N$... This is what PLUMED does by default so you do not need to use CLEAR in this case.

6.2 Diagnostic tools

PLUMED has a number of diagnostic tools that can be used to check that new Actions are working correctly:

DUMPFORCES	Dump the force acting on one of a values in a file.
DUMPDERIVATIVES	Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).
DUMPMASSCHARGE	Dump masses and charges on a selected file.
DUMPPROJECTIONS	Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).

These commands allow you to test that derivatives and forces are calculated correctly within colvars and functions. One place where this is very useful is when you are testing whether or not you have implemented the derivatives of a new collective variables correctly. So for example if we wanted to do such a test on the distance CV we would employ an input file something like this:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AnalysisPP.md
d1: DISTANCE ATOMS=1,2
d1n: DISTANCE ATOMS=1,2 NUMERICAL_DERIVATIVES
DUMPDERIVATIVES ARG=d1,d1n FILE=derivatives
```

The first of these two distance commands calculates the analytical derivatives of the distance while the second calculates these derivatives numerically. Obviously, if your CV is implemented correctly these two sets of quantities should be nearly identical.

6.3 Storing data for analysis

All the analysis methods described in previous sections accumulate averages or output diagnostic information on the fly. That is to say these methods calculate something given the instantaneous positions of the atoms or the instantaneous values of a set of collective variables. Many methods (e.g. dimensionality reduction and clustering) will not work like this, however, as information from multiple trajectory frames is required at the point when the analysis is performed. In other words the output from these types of analysis cannot be accumulated one frame at time. When using these methods you must therefore store trajectory frames for later analysis. You can do this storing by using the following action:

COLLECT_FRAMES	Collect and store trajectory frames for later analysis with one of the methods detailed below.

6.4 Calculating dissimilarity matrices

One of the simplest things that we can do if we have stored a set of trajectory frames using COLLECT_FRAMES is we can calculate the dissimilarity between every pair of frames we have stored. When using the dimensionality reduction algorithms described in the sections that follow the first step is to calculate this matrix. Consequently, within PLUMED the following command will collect the trajectory data as your simulation progressed and calculate the dissimilarities:

EUCLIDEAN_DISSIMILARITIES	Calculate the matrix of dissimilarities between a trajectory of atomic configurations.

By exploiting the functionality described in Distances from reference configurations you can calculate these dissimilarities in a wide variety of different ways (e.g. you can use RMSD, or you can use a collection of collective variable values see TARGET). If you wish to view this dissimilarity information you can print these quantities to a file using:

PRINT_DISSIMILARITY_MATRIX	Print the matrix of dissimilarities between a trajectory of atomic configurations.

In addition, if PLUMED does not calculate the dissimilarities you need you can read this information from an external file

READ_DISSIMILARITY_MATRIX	Read a matrix of dissimilarities between a trajectory of atomic configurations from a file.

N.B. You can only use the two commands above when you are doing post-processing.

6.5 Landmark Selection

Many of the techniques described in the following sections are very computationally expensive to run on large trajectories. A common strategy is thus to use a landmark selection algorithm to pick a particularly-representative subset of trajectory frames and to only apply the expensive analysis algorithm on these configurations. The various landmark selection algorithms that are available in PLUMED are as follows

LANDMARK_SELECT_FPS	Select a set of landmarks using farthest point sampling.
LANDMARK_SELECT_RANDOM	Select a random set of landmarks from a large set of configurations.
LANDMARK_SELECT_STAGED	Select a set of landmarks using the staged algorithm.
LANDMARK_SELECT_STRIDE	Select every kth landmark from the trajectory.
RESELECT_LANDMARKS	This allows us to use one measure in landmark selection and a different measure in dimensionality reduction

In general most of these landmark selection algorithms must be used in tandem with a dissimilarity matrix object as as follows:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AnalysisPP.md
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=3,4
d3: DISTANCE ATOMS=5,6
data: COLLECT_FRAMES ARG=d1,d2,d3 STRIDE=1
ss1: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=data
ll2: LANDMARK_SELECT_FPS USE_OUTPUT_DATA_FROM=ss1 NLANDMARKS=300
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=ll2 FILE=mylandmarks
```

When landmark selection is performed in this way a weight is ascribed to each of the landmark configurations. This weight is calculated by summing the weights of all the trajectory frames in each of the landmarks Voronoi polyhedron (https://en.wikipedia.org/wiki/Voronoi_diagram). The weight of each trajectory frame is one unless you are reweighting using the formula described in the Reweighting and Averaging to counteract the fact of a simulation bias or an elevated temperature. If you are reweighting using these formula the weight of each of the points is equal to the exponential term in the numerator of these expressions.

6.6 Dimensionality Reduction

Many dimensionality reduction algorithms work in a manner similar to the way we use when we make maps. You start with distances between London, Belfast, Paris and Dublin and then you try to arrange points on a piece of paper so that the (suitably transformed) distances between the points in your map representing each of those cities are related to the true distances between the cities.
Stating this more mathematically MDS endeavors to find an isometry between points distributed in a high-dimensional space and a set of points distributed in a low-dimensional plane.
In other words, if we have M D-dimensional points, $\mathbf{X}$, and we can calculate dissimilarities between pairs them, D_{ij}, we can, with an MDS calculation, try to create M projections, $\mathbf{x}$, of the high dimensionality points in a d-dimensional linear space by trying to arrange the projections so that the Euclidean distances between pairs of them, d_{ij}, resemble the dissimilarities between the high dimensional points. In short we minimize:

$$\chi^2 = \sum_{i \neq j} w_i w_j \left(F(D_{ij}) - f(d_{ij}) \right)^2$$

where $F(D_{ij})$ is some transformation of the distance between point X^i and point X^j and $f(d_{ij})$ is some transformation of the distance between the projection of X^i, x^i, and the projection of X^j, x^j. w_i and w_j are the weights of

configurations X^i and ^{j} respectively. These weights are calculated using the reweighting and Voronoi polyhedron approaches described in previous sections. A tutorial on dimensionality reduction and how it can be used to analyze simulations can be found in the tutorial Lugano tutorial: Dimensionality reduction and in the following short video.

Within PLUMED running an input to run a dimensionality reduction algorithm can be as simple as:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AnalysisPP.md
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=3,4
d3: DISTANCE ATOMS=5,6
data: COLLECT_FRAMES STRIDE=1 ARG=d1,d2,d3
ss1: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=data
mds: CLASSICAL_MDS USE_OUTPUT_DATA_FROM=ss1 NLOW_DIM=2
```

Where we have to use the EUCLIDEAN_DISSIMILARITIES action here in order to calculate the matrix of dissimilarities between trajectory frames. We can even throw some landmark selection into this procedure and perform

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AnalysisPP.md
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=3,4
d3: DISTANCE ATOMS=5,6
data: COLLECT_FRAMES STRIDE=1 ARG=d1,d2,d3
matrix: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=data
ll2: LANDMARK_SELECT_FPS USE_OUTPUT_DATA_FROM=matrix NLANDMARKS=300
mds: CLASSICAL_MDS USE_OUTPUT_DATA_FROM=ll2 NLOW_DIM=2
osample: PROJECT_ALL_ANALYSIS_DATA USE_OUTPUT_DATA_FROM=matrix PROJECTION=mds
```

Notice here that the final command allows us to calculate the projections of all the non-landmark points that were collected by the action with label matrix.

Dimensionality can be more complicated, however, because the stress function that calculates χ^2 has to optimized rather carefully using a number of different algorithms. The various algorithms that can be used to optimize this function are described below

CLASSICAL_MDS	Create a low-dimensional projection of a trajectory using the classical multidimensional scaling algorithm.
OUTPUT_PCA_PROJECTION	This is used to output the projection calculated by principle component analysis
PCA	Perform principal component analysis (PCA) using either the positions of the atoms a large number of collective variables as input.
PROJECT_ALL_ANALYSIS_DATA	Find projections of all non-landmark points using the embedding calculated by a dimensionality reduction optimization calculation.
SKETCHMAP_CONJGRAD	Optimize the sketch-map stress function using conjugate gradients.
SKETCHMAP_POINTWISE	Optimize the sketch-map stress function using a pointwise global optimization algorithm.
SKETCHMAP_READ	Read in a sketch-map projection from an input file
SKETCHMAP_SMACOF	Optimize the sketch-map stress function using the SMACOF algorithm.
SKETCH_MAP	This can be used to output the data that has been stored in an Analysis object.
SMACOF_MDS	Optimize the multidimensional scaling stress function using the SMACOF algorithm.

6.7 Outputting the results from analysis algorithms

The following methods are available for printing the result output by the various analysis algorithms:

OUTPUT_ANALYSIS_DATA_TO_COLVAR	Output the results from an analysis using the PLUMED colvar file format.
OUTPUT_ANALYSIS_DATA_TO_PDB	Output the results from an analysis using the PDB file format.

Using the above commands to output the data from any form of analysis is important as **the STRIDE with which you output the data to a COLVAR or PDB file controls how frequently the analysis is performed on the collected data** . If you specified no stride on the output lines then PLUMED assumes you want to perform analysis on the entire trajectory.

If you use the above commands to output data from one of the Landmark Selection algorithms then only the second will give you information on the atomic positions in your landmark configurations and their associated weights. The first of these commands will give the values of the colvars in the landmark configurations only. If you use the above commands to output data from one of the Dimensionality Reduction algorithms then OUTPUT_ANALYSIS_DATA_TO_COLVAR will give you an output file that contains the projection for each of your input points. OUTPUT_ANALYSIS_DATA_TO_PDB will give you a PDB that contains the position of the input point, the projections and the weight of the configuration.

A nice feature of plumed is that when you use Landmark Selection algorithms or Dimensionality Reduction algorithms the output information is just a vector of variables. As such you can use HISTOGRAM to construct a histogram of the information generated by these algorithms.

6.8 COMMITTOR

	This is part of the analysis module

Does a committor analysis.

Examples

The following input monitors two torsional angles during a simulation, defines two basins (A and B) as a function of the two torsion angles and stops the simulation when it falls in one of the two. In the log file will be shown the latest values for the CVs and the basin reached.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/COMMITTOR.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
COMMITTOR ...
  ARG=r1,r2
  STRIDE=10
  BASIN_LL1=0.15,0.20
  BASIN_UL1=0.25,0.40
  BASIN_LL2=-0.25,-0.40
  BASIN_UL2=-0.15,-0.20
... COMMITTOR
```

Glossary of keywords and components

Compulsory keywords

BASIN_LL	List of lower limits for basin #. You can use multiple instances of this keyword i.e. BASIN_LL1, BASIN_LL2, BASIN_LL3...
BASIN_UL	List of upper limits for basin #. You can use multiple instances of this keyword i.e. BASIN_UL1, BASIN_UL2, BASIN_UL3...
STRIDE	(default=1) the frequency with which the CVs are analyzed

Options

NOSTOP	(default=off) if true do not stop the simulation when reaching a basin but just keep track of it
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
FILE	the name of the file on which to output the reached basin
FMT	the format that should be used to output real numbers

6.9 DUMPATOMS

This is part of the generic module

Dump selected atoms on a file.

This command can be used to output the positions of a particular set of atoms. The atoms required are output in a xyz or gro formatted file. If PLUMED has been compiled with xdrfile support, then also xtc and trr files can be written. To this aim one should install xdrfile library (http://www.gromacs.org/Developer_Zone/Programming_Guide/XTC_Library). If the xdrfile library is installed properly the PLUMED configure script should be able to detect it and enable it. The type of file is automatically detected from the file extension, but can be also enforced with TYPE. Importantly, if your input file contains actions that edit the atoms position (e.g. WHOLEMOLECULES) and the DUMPATOMS command appears after this instruction, then the edited atom positions are output. You can control the buffering of output using the FLUSH keyword on a separate line.

Units of the printed file can be controlled with the UNITS keyword. By default PLUMED units as controlled in the UNITS command are used, but one can override it e.g. with UNITS=A. Notice that gro/xtc/trr files can only contain coordinates in nm.

Examples

The following input instructs plumed to print out the positions of atoms 1-10 together with the position of the center of mass of atoms 11-20 every 10 steps to a file called file.xyz.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPATOMS.tmp
COM ATOMS=11-20 LABEL=c1
DUMPATOMS STRIDE=10 FILE=file.xyz ATOMS=1-10,c1
```

Notice that the coordinates in the xyz file will be expressed in nm, since these are the defaults units in PLUMED. If you want the xyz file to be expressed in A, you should use the following input

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPATOMS.tmp
COM ATOMS=11-20 LABEL=c1
DUMPATOMS STRIDE=10 FILE=file.xyz ATOMS=1-10,c1 UNITS=A
```

As an alternative, you might want to set all the length used by PLUMED to Angstrom using the UNITS action. However, this latter choice will affect all your input and output.

The following input is very similar but dumps a .gro (gromacs) file, which also contains atom and residue names.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPATOMS.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
# this is required to have proper atom names:
MOLINFO STRUCTURE=reference.pdb
# if omitted, atoms will have "X" name...

COM ATOMS=11-20 LABEL=c1
DUMPATOMS STRIDE=10 FILE=file.gro ATOMS=1-10,c1
# notice that last atom is a virtual one and will not have
# a correct name in the resulting gro file
```

The `file.gro` will contain coordinates expressed in nm, since this is the convention for gro files.

In case you have compiled PLUMED with `xdrfile` library, you might even write xtc or trr files as follows

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPATOMS.tmp
COM ATOMS=11-20 LABEL=c1
DUMPATOMS STRIDE=10 FILE=file.xtc ATOMS=1-10,c1
```

Notice that xtc files are significantly smaller than gro and xyz files.

Finally, consider that gro and xtc file store coordinates with limited precision set by the `PRECISION` keyword. Default value is 3, which means "3 digits after dot" in nm (1/1000 of a nm). The following will write a larger xtc file with high resolution coordinates:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPATOMS.tmp
COM ATOMS=11-20 LABEL=c1
DUMPATOMS STRIDE=10 FILE=file.xtc ATOMS=1-10,c1 PRECISION=7
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the atom indices whose positions you would like to print out. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

STRIDE	(default=1) the frequency with which the atoms should be output
FILE	file on which to output coordinates; extension is automatically detected
UNITS	(default=PLUMED) the units in which to print out the coordinates. PLUMED means internal PLU↩ MED units

Options

PRECISION	The number of digits in trajectory file
TYPE	file type, either xyz, gro, xtc, or trr, can override an automatically detected file extension
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UN↩ TIL	Only update this action until this time

6.10 DUMPDERIVATIVES

	This is part of the generic module

Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).

For a CV this line in input instructs plumed to print the derivative of the CV with respect to the atom positions and the cell vectors (virial-like form). In contrast, for a function or bias the derivative with respect to the input "CVs" will be output. This command is most often used to test whether or not analytic derivatives have been implemented correctly. This can be done by outputting the derivatives calculated analytically and numerically. You can control the buffering of output using the FLUSH keyword.

Examples

The following input instructs plumed to write a file called deriv that contains both the analytical and numerical derivatives of the distance between atoms 1 and 2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPDERIVATIVES.tmp
DISTANCE ATOMS=1,2 LABEL=distance
DISTANCE ATOMS=1,2 LABEL=distanceN NUMERICAL_DERIVATIVES
DUMPDERIVATIVES ARG=distance,distanceN STRIDE=1 FILE=deriv
```

(See also DISTANCE)

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the derivatives should be output
FILE	the name of the file on which to output the derivatives
FMT	(default=%15.10f) the format with which the derivatives should be output

Options

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

6.11 DUMPFORCES

This is part of the generic module

Dump the force acting on one of a values in a file.

For a CV this command will dump the force on the CV itself. Be aware that in order to have the forces on the atoms you should multiply the output from this argument by the output from DUMPDERIVATIVES. Furthermore, also note that you can output the forces on multiple quantities simultaneously by specifying more than one argument. You can control the buffering of output using the FLUSH keyword.

Examples

The following input instructs plumed to write a file called forces that contains the force acting on the distance between atoms 1 and 2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPFORCES.tmp
DISTANCE ATOMS=1,2 LABEL=distance
DUMPFORCES ARG=distance STRIDE=1 FILE=forces
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the forces should be output
FILE	the name of the file on which to output the forces
FMT	(default=%15.10f) the format with which the derivatives should be output

Options

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

6.12 DUMPMASSCHARGE

This is part of the generic module

Dump masses and charges on a selected file.

This command dumps a file containing charges and masses. It does so only once in the simulation (at first step). File can be recycled in the driver tool.

Notice that masses and charges are only written once at the beginning of the simulation. In case no atom list is provided, charges and masses for all atoms are written.

Examples

You can add the DUMPMASSCHARGE action at the end of the plumed.dat file that you use during an MD simulations:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPMASSCHARGE.tmp
c1: COM ATOMS=1-10
c2: COM ATOMS=11-20
DUMPATOMS ATOMS=c1,c2 FILE=coms.xyz STRIDE=100

DUMPMASSCHARGE FILE=mcfile
```

In this way, you will be able to use the same masses while processing a trajectory from the driver . To do so, you need to add the –mc flag on the driver command line, e.g.

```
plumed driver --mc mcfile --plumed plumed.dat --ixyz traj.xyz
```

With the following input you can dump only the charges for a specific group:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPMASSCHARGE.tmp
solute_ions: GROUP ATOMS=1-121,200-2012
DUMPATOMS FILE=traj.gro ATOMS=solute_ions STRIDE=100
DUMPMASSCHARGE FILE=mcfile ATOMS=solute_ions
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the atom indices whose charges and masses you would like to print out. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

STRIDE	(default=1) the frequency with which the atoms should be output
FILE	file on which to output charges and masses.

Options

ONLY_MASSES	(default=off) Only output masses to file
ONLY_CHARGES	(default=off) Only output charges to file

6.13 DUMPMULTICOLVAR

This is part of the multicolvar module

Dump atom positions and multicolvar on a file.

Examples

In this examples we calculate the distances between the atoms of the first and the second group and we write them in the file MULTICOLVAR.xyz. For each couple it writes the coordinates of their geometric center and their distance.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPMULTICOLVAR.tmp
pos:   GROUP ATOMS=220,221,235,236,247,248,438,439,450,451,534,535
neg:   GROUP ATOMS=65,68,138,182,185,267,270,291,313,316,489,583,621,711
DISTANCES GROUPA=pos GROUPB=neg LABEL=slt

DUMPMULTICOLVAR DATA=slt FILE=MULTICOLVAR.xyz
```

(see also DISTANCES)

Glossary of keywords and components

The atoms involved can be specified using

ORIGIN	You can use this keyword to specify the position of an atom as an origin. The positions output will then be displayed relative to that origin. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

DATA	certain actions in plumed work by calculating a list of variables and summing over them. This particular action can be used to calculate functions of these base variables or prints them to a file. This keyword thus takes the label of one of those such variables as input.
STRIDE	(default=1) the frequency with which the atoms should be output
FILE	file on which to output coordinates
UNITS	(default=PLUMED) the units in which to print out the coordinates. PLUMED means internal PLU↩ MED units

Options

PRECISION	The number of digits in trajectory file

6.14 DUMPPROJECTIONS

	This is part of the generic module

Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).

Examples

Compute the distance between two groups and write on a file the derivatives of this distance with respect to all the atoms of the two groups

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPPROJECTIONS.tmp
x1: CENTER ATOMS=1-10
x2: CENTER ATOMS=11-20
d: DISTANCE ATOMS=x1,x2
DUMPPROJECTIONS ARG=d FILE=proj STRIDE=20
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the derivatives should be output
FILE	the name of the file on which to output the derivatives
FMT	(default=%15.10f) the format with which the derivatives should be output

Options

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

6.15 PRINT

This is part of the generic module

Print quantities to a file.

This directive can be used multiple times in the input so you can print files with different strides or print different quantities to different files. You can control the buffering of output using the FLUSH keyword. Output file is either appended or backed up depending on the presence of the RESTART action. A per-action `RESTART` keyword can be used as well.

Notice that printing happens in the so-called "update" phase. This implies that printing is affected by the presence of UPDATE_IF actions. In addition, one might decide to start and stop printing at preassigned values of time using the `UPDATE_FROM` and `UPDATE_UNTIL` keywords. Keep into account that even on steps when the action is not updated (and thus the file is not printed) the argument will be activated. In other words, if you use `UPDATE_FROM` to start printing at a given time, the collective variables this PRINT statement depends on will be computed also before that time.

Examples

The following input instructs plumed to print the distance between atoms 3 and 5 on a file called COLVAR every 10 steps, and the distance and total energy on a file called COLVAR_ALL every 1000 steps.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PRINT.tmp
# compute distance:
distance: DISTANCE ATOMS=2,5
# compute total energy (potential)
energy: ENERGY
# print distance on a file
PRINT ARG=distance          STRIDE=10   FILE=COLVAR
# print both variables on another file
PRINT ARG=distance,energy   STRIDE=1000 FILE=COLVAR_ALL
```

Notice that DISTANCE and ENERGY are computed respectively every 10 and 1000 steps, that is only when required.

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the quantities of interest should be output

Options

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
FILE	the name of the file on which to output these quantities
FMT	the format that should be used to output real numbers
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

6.15.1 FLUSH

	This is part of the generic module

This command instructs plumed to flush all the open files with a user specified frequency. Notice that all files are flushed anyway every 10000 steps.

This is useful for preventing data loss that would otherwise arise as a consequence of the code storing data for printing in the buffers. Notice that wherever it is written in the plumed input file, it will flush all the open files.

Examples

A command like this in the input will instruct plumed to flush all the output files every 100 steps

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FLUSH.tmp
d1: DISTANCE ATOMS=1,10
PRINT ARG=d1 STRIDE=5 FILE=colvar1

FLUSH STRIDE=100

d2: DISTANCE ATOMS=2,11
# also this print is flushed every 100 steps:
PRINT ARG=d2 STRIDE=10 FILE=colvar2
```

(see also DISTANCE and PRINT).

Glossary of keywords and components

Compulsory keywords

STRIDE	the frequency with which all the open files should be flushed

6.16 UPDATE_IF

	This is part of the generic module

Conditional update of other actions.

This action can be used to enable and disable the update step for the following actions depending on the value of its arguments. This allows for example to extract snapshots with value of some CVs in a given range.

When called with MORE_THAN and/or LESS_THAN keywords, this action starts an if block. The block is executed if all the arguments are less than all the respective values in the LESS_THAN keyword (if present) and all the arguments are more than all the respective values in the MORE_THAN keyword (if present).

When called with the END flag, this action ends the corresponding IF block. Notice that in this case one should also provide the ARG keyword. It is recommended to use the same ARG keyword that was used to begin the block, so as to make the input more readable.

Of course, blocks can be nested at will.

There are many potential usages for this keyword. One might e.g. decide to analyze some variable only when another variable is within a given range.

Warning

Notice that not all the possible usage make particular sense. For example, conditionally updating a METAD keyword (that is: adding hills only if a variable is within a given range) can lead to unexpected results.

Examples

The following input instructs plumed dump all the snapshots where an atom is in touch with the solute.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/UPDATE_IF.tmp
solute: GROUP ATOMS=1-124
coord: COORDINATION GROUPA=solute GROUPB=500 R_0=0.5

# A coordination number higher than 0.5 indicate that there is at least one
# atom of group `solute` at less than 5 A from atom number 500

UPDATE_IF ARG=coord MORE_THAN=0.5
DUMPATOMS ATOMS=solute,500 FILE=output.xyz
UPDATE_IF ARG=coord END
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the quantities of interest should be output

Options

END	(default=off) end
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
LESS_THAN	upper bound
MORE_THAN	lower bound

6.17 REWEIGHT_BIAS

	This is part of the bias module

Calculate weights for ensemble averages that negate the effect the bias has on the region of phase space explored

If a static or pseudo-static bias $V(x,t')$ is acting on the system we can remove the bias and get the unbiased probability distribution using:

$$\langle P(s',t)\rangle = \frac{\sum_{t'}^{t} \delta(s(x)-s') \exp\left(+\frac{V(x,t')}{k_B T}\right)}{\sum_{t'}^{t} \exp\left(+\frac{V(x,t')}{k_B T}\right)}$$

The weights calculated by this action are equal to $\exp\left(+\frac{V(x,t')}{k_B T}\right)$ these weights can then be used in any action that computes ensemble averages. For example this action can be used in tandem with HISTOGRAM or AVERAGE.

Examples

In the following example there is a fixed restraint on the distance between atoms 1 and 2. Clearly, this restraint will have an effect on the region of phase space that will be sampled when an MD simulation is run using this variable. Consequently, when the histogram as a function of the distance, x, is accumulated, we use reweighting into order to discount the effect of the bias from our final histogram.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/REWEIGHT_BIAS.tmp
x: DISTANCE ATOMS=1,2
RESTRAINT ARG=x SLOPE=1.0 AT=0.0
bias: REWEIGHT_BIAS TEMP=300

HISTOGRAM ...
  ARG=x
  GRID_MIN=0.0
  GRID_MAX=3.0
  GRID_BIN=100
  BANDWIDTH=0.1
  LOGWEIGHTS=bias
  LABEL=hB
... HISTOGRAM

DUMPGRID GRID=hB FILE=histoB STRIDE=1 FMT=%8.4f
```

Glossary of keywords and components

Compulsory keywords

ARG	(default=∗.bias) the biases that must be taken into account when reweighting

Options

TEMP	the system temperature. This is not required if your MD code passes this quantity to PLUMED

6.18 REWEIGHT_METAD

This is part of the bias module

Calculate the weights configurations should contribute to the histogram in a simulation in which a metadynamics bias acts upon the system.

This command allows you to use the reweighting algorithm discussed in [35] when constructing a histogram of the configurations visited during a metadynamics simulation.

Examples

In the following example there is a metadynamics bias acting on the distance between atoms 1 and 2. Clearly, this bias will have an effect on the region of phase space that will be sampled when an MD simulation is run using this variable. Consequently, when the histogram as a function of the angle, a, is accumulated, we use reweighting into order to discount the effect of the bias from our final histogram. We do not use REWEIGHT_BIAS here, however, as the bias changes with time. We thus use the reweighting algorithm for metadynamics instead. Notice also that we have to specify how often we would like to calculate the c(t) reweighting factor and the grid over which we calculate c(t) in the input to the METAD command.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/REWEIGHT_METAD.tmp
a: ANGLE ATOMS=1,2,3
x: DISTANCE ATOMS=1,2
METAD ARG=x PACE=100 SIGMA=0.1 HEIGHT=1.5 BIASFACTOR=5 GRID_MIN=0 GRID_MAX=10 GRID_BIN=100 CALC_RCT RCT_USTRID

bias: REWEIGHT_METAD TEMP=300

HISTOGRAM ...
  ARG=a
  GRID_MIN=0.0
  GRID_MAX=pi
  GRID_BIN=100
  BANDWIDTH=0.1
  LOGWEIGHTS=bias
  LABEL=hB
... HISTOGRAM

DUMPGRID GRID=hB FILE=histoB STRIDE=1 FMT=%8.4f
```

Glossary of keywords and components

Compulsory keywords

ARG	(default=∗.rbias) the biases that must be taken into account when reweighting

Options

TEMP	the system temperature. This is not required if your MD code passes this quantity to PLUMED

6.19 REWEIGHT_TEMP_PRESS

This is part of the bias module

Calculate weights for ensemble averages at temperatures and/or pressures different than those used in your original simulation.

We can use our knowledge of the probability distribution in the canonical (N $\mathcal{V}$T) or the isothermal-isobaric ensemble (NPT) to reweight the data contained in trajectories and obtain ensemble averages at different temperatures and/or pressures.

Consider the ensemble average of an observable $O(\mathbf{R},\mathcal{V})$ that depends on the atomic coordinates $\mathbf{R}$ and the volume $\mathcal{V}$. This observable is in practice any collective variable (CV) calculated by Plumed. The ensemble average of the observable in an ensemble ξ' can be calculated from a simulation performed in an ensemble ξ using:

$$\langle O(\mathbf{R},\mathcal{V})\rangle_{\xi'} = \frac{\langle O(\mathbf{R},\mathcal{V})w(\mathbf{R},\mathcal{V})\rangle_{\xi}}{\langle w(\mathbf{R},\mathcal{V})\rangle_{\xi}}$$

where $\langle\cdot\rangle_{\xi}$ and $\langle\cdot\rangle_{\xi'}$ are mean values in the simulated and targeted ensemble, respectively, $E(\mathbf{R})$ is the potential energy of the system, and $w(\mathbf{R},\mathcal{V})$ are the appropriate weights to take from ξ to ξ'. This action calculates the weights $w(\mathbf{R},\mathcal{V})$ and handles 4 different cases:

1. Change of temperature from T to T' at constant volume. That is to say, from a simulation performed in the N $\mathcal{V}$T (canonical) ensemble, obtain an ensemble average in the N $\mathcal{V}$T' ensemble. The weights in this case are $w(\mathbf{R},\mathcal{V}) = e^{(\beta-\beta')E(\mathbf{R})}$ with β and β' the inverse temperatures.
2. Change of temperature from T to T' at constant pressure. That is to say, from a simulation performed in the NPT (isothermal-isobaric) ensemble, obtain an ensemble average in the NPT' ensemble. The weights in this case are $w(\mathbf{R},\mathcal{V}) = e^{(\beta-\beta')(E(\mathbf{R})+P\mathcal{V})}$.
3. Change of pressure from P to P' at constant temperature. That is to say, from a simulation performed in the NPT (isothermal-isobaric) ensemble, obtain an ensemble average in the NP'T ensemble. The weights in this case are $w(\mathbf{R},\mathcal{V}) = e^{\beta(P-P')\mathcal{V}}$.
4. Change of temperature and pressure from T,P to T',P'. That is to say, from a simulation performed in the NPT (isothermal-isobaric) ensemble, obtain an ensemble average in the NP'T' ensemble. The weights in this case are $w(\mathbf{R},\mathcal{V}) = e^{(\beta-\beta')E(\mathbf{R})+(\beta P-\beta' P')\mathcal{V}}$.

These weights can be used in any action that computes ensemble averages. For example this action can be used in tandem with HISTOGRAM or AVERAGE.

The above equation is often impractical since the overlap between the distributions of energy and volume at different temperatures and pressures is only significant for neighboring temperatures and pressures. For this reason an

unbiased simulation is of little use to reweight at different temperatures and/or pressures. A successful approach has been altering the probability of observing a configuration in order to increase this overlap [36]. This is done through a bias potential $V(\mathbf{s})$ where $\mathbf{s}$ is a set of CVs, that often is the energy (and possibly the volume). In order to calculate ensemble averages, also the effect of this bias must be taken into account. The ensemble average of the observable in the ensemble ξ' can be calculated from a biased simulation performed in the ensemble ξ with bias $V(\mathbf{s})$ using:

$$\langle O(\mathbf{R},\mathcal{V})\rangle_{\xi'} = \frac{\langle O(\mathbf{R},\mathcal{V})w(\mathbf{R},\mathcal{V})e^{\beta V(\mathbf{s})}\rangle_{\xi,V}}{\langle w(\mathbf{R},\mathcal{V})e^{\beta V(\mathbf{s})}\rangle_{\xi,V}}$$

where $\langle\cdot\rangle_{\xi,V}$ is a mean value in the biased ensemble with static bias $V(\mathbf{s})$. Therefore in order to reweight the trajectory at different temperatures and/or pressures one must use the weights calculated by this action $w(\mathbf{R},\mathcal{V})$ together with the weights of REWEIGHT_BIAS (see the examples below).

The bias potential $V(\mathbf{s})$ can be constructed with METAD using ENERGY as a CV [37]. More specialized tools are available, for instance using bespoke target distributions such as TD_MULTICANONICAL and TD_MULTITHERMAL_MULTIBARIC [38] [27] within Variationally Enhanced Sampling (VES code). In the latter algorithms the interval of temperatures and pressures in which the trajectory can be reweighted is chosen explicitly.

Examples

We consider the 4 cases described above.

The following input can be used to postprocess a molecular dynamics trajectory of a system of 1000 particles run at 500 K and constant volume using a static bias potential.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/REWEIGHT_TEMP_PRESS.tmp
energy: READ FILE=COLVAR VALUES=energy  IGNORE_TIME
distance: READ FILE=COLVAR VALUES=distance  IGNORE_TIME
mybias: READ FILE=COLVAR VALUES=mybias.bias  IGNORE_TIME

# Shift energy (to avoid numerical issues)
renergy: COMBINE ARG=energy PARAMETERS=-13250 PERIODIC=NO

# Weights
bias_weights: REWEIGHT_BIAS TEMP=500 ARG=mybias.bias
temp_press_weights: REWEIGHT_TEMP_PRESS TEMP=500 REWEIGHT_TEMP=300 ENERGY=renergy

# Ensemble average of the distance at 300 K
avg_dist: AVERAGE ARG=distance LOGWEIGHTS=bias_weights,temp_press_weights

PRINT ARG=avg_dist FILE=COLVAR_REWEIGHT STRIDE=1
```

Clearly, in performing the analysis above we would read from the potential energy, a distance, and the value of the bias potential from a COLVAR file like the one shown below. We would then be able to calculate the ensemble average of the distance at 300 K.

```
#! FIELDS time energy volume mybias.bias distance
 10000.000000 -13133.769283 7.488921 63.740530 0.10293
 10001.000000 -13200.239722 7.116548 36.691988 0.16253
 10002.000000 -13165.108850 7.202273 44.408815 0.17625
```

The next three inputs can be used to postprocess a molecular dynamics trajectory of a system of 1000 particles run at 500 K and 1 bar using a static bias potential.

We read from a file COLVAR the potential energy, the volume, and the value of the bias potential and calculate the ensemble average of the (particle) density at 300 K and 1 bar (the simulation temperature was 500 K).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/REWEIGHT_TEMP_PRESS.tmp
energy: READ FILE=COLVAR VALUES=energy  IGNORE_TIME
volume: READ FILE=COLVAR VALUES=volume  IGNORE_TIME
mybias: READ FILE=COLVAR VALUES=mybias.bias  IGNORE_TIME

# Shift energy and volume (to avoid numerical issues)
rvol: COMBINE ARG=volume PARAMETERS=7.8 PERIODIC=NO
renergy: COMBINE ARG=energy PARAMETERS=-13250 PERIODIC=NO

# Weights
bias_weights: REWEIGHT_BIAS TEMP=500 ARG=mybias.bias
temp_press_weights: REWEIGHT_TEMP_PRESS TEMP=500 REWEIGHT_TEMP=300 PRESSURE=0.06022140857 ENERGY=renergy VOLUM

# Ensemble average of the volume at 300 K
avg_vol: AVERAGE ARG=volume LOGWEIGHTS=bias_weights,temp_press_weights
# Ensemble average of the density at 300 K
avg_density: CUSTOM ARG=avg_vol FUNC=1000/x PERIODIC=NO

PRINT ARG=avg_density FILE=COLVAR_REWEIGHT STRIDE=1
```

In the next example we calculate the ensemble average of the (particle) density at 500 K and 300 MPa (the simulation pressure was 1 bar).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/REWEIGHT_TEMP_PRESS.tmp
volume: READ FILE=COLVAR VALUES=volume  IGNORE_TIME
mybias: READ FILE=COLVAR VALUES=mybias.bias  IGNORE_TIME

# Shift volume (to avoid numerical issues)
rvol: COMBINE ARG=volume PARAMETERS=7.8 PERIODIC=NO

# Weights
bias_weights: REWEIGHT_BIAS TEMP=500 ARG=mybias.bias
temp_press_weights: REWEIGHT_TEMP_PRESS TEMP=500 PRESSURE=0.06022140857 REWEIGHT_PRESSURE=180.66422571 VOLUME=

# Ensemble average of the volume at 300 K and 300 MPa
avg_vol: AVERAGE ARG=volume LOGWEIGHTS=bias_weights,temp_press_weights
# Ensemble average of the density at 300 K and 300 MPa
avg_density: CUSTOM ARG=avg_vol FUNC=1000/x PERIODIC=NO

PRINT ARG=avg_density FILE=COLVAR_REWEIGHT STRIDE=1
```

In this final example we calculate the ensemble average of the (particle) density at 300 K and 300 MPa (the simulation temperature and pressure were 500 K and 1 bar).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/REWEIGHT_TEMP_PRESS.tmp
energy: READ FILE=COLVAR VALUES=energy  IGNORE_TIME
volume: READ FILE=COLVAR VALUES=volume  IGNORE_TIME
mybias: READ FILE=COLVAR VALUES=mybias.bias  IGNORE_TIME

# Shift energy and volume (to avoid numerical issues)
rvol: COMBINE ARG=volume PARAMETERS=7.8 PERIODIC=NO
renergy: COMBINE ARG=energy PARAMETERS=-13250 PERIODIC=NO

# Weights
bias_weights: REWEIGHT_BIAS TEMP=500 ARG=mybias.bias
temp_press_weights: REWEIGHT_TEMP_PRESS TEMP=500 REWEIGHT_TEMP=300 PRESSURE=0.06022140857 REWEIGHT_PRESSURE=18

# Ensemble average of the volume at 300 K and 300 MPa
avg_vol: AVERAGE ARG=volume LOGWEIGHTS=bias_weights,temp_press_weights
# Ensemble average of the density at 300 K and 300 MPa
avg_density: CUSTOM ARG=avg_vol FUNC=1000/x PERIODIC=NO

PRINT ARG=avg_density FILE=COLVAR_REWEIGHT STRIDE=1
```

Glossary of keywords and components

Options

TEMP	the system temperature. This is not required if your MD code passes this quantity to PLUMED
ENERGY	Energy
VOLUME	Volume
REWEIGHT_PRESSURE	Reweighting pressure
PRESSURE	The system pressure
REWEIGHT_TEMP	Reweighting temperature

6.20 REWEIGHT_WHAM

This is part of the bias module

Calculate the weights for configurations using the weighted histogram analysis method.

Suppose that you have run multiple N trajectories each of which was computed by integrating a different biased Hamiltonian. We can calculate the probability of observing the set of configurations during the N trajectories that we ran using the product of multinomial distributions shown below:

$$P(\vec{T}) \propto \prod_{j=1}^{M} \prod_{k=1}^{N} (c_k w_{kj} p_j)^{t_{kj}}$$

In this expression the second product runs over the biases that were used when calculating the N trajectories. The first product then runs over the M bins in our histogram. The p_j variable that is inside this product is the quantity we wish to extract; namely, the unbiased probability of having a set of CV values that lie within the range for the jth bin.

The quantity that we can easily extract from our simulations, t_{kj}, however, measures the number of frames from trajectory k that are inside the jth bin. To interpret this quantity we must consider the bias that acts on each of the replicas so the w_{kj} term is introduced. This quantity is calculated as:

$$w_{kj} =$$

and is essentially the factor that we have to multiply the unbiased probability of being in the bin by in order to get the probability that we would be inside this same bin in the kth of our biased simulations. Obviously, these w_{kj} values depend on the value that the CVs take and also on the particular trajectory that we are investigating all of which, remember, have different simulation biases. Finally, c_k is a free parameter that ensures that, for each k, the biased probability is normalized.

We can use the equation for the probability that was given above to find a set of values for p_j that maximizes the likelihood of observing the trajectory. This constrained optimization must be performed using a set of Lagrange multipliers, λ_k, that ensure that each of the biased probability distributions are normalized, that is $\sum_j c_k w_{kj} p_j = 1$. Furthermore, as the problem is made easier if the quantity in the equation above is replaced by its logarithm we actually chose to minimize

$$\mathcal{L} = \sum_{j=1}^{M} \sum_{k=1}^{N} t_{kj} \ln c_k w_{kj} p_j + \sum_k \lambda_k \left(\sum_{j=1}^{N} c_k w_{kj} p_j - 1 \right)$$

After some manipulations the following (WHAM) equations emerge:

$$p_j \propto \frac{\sum_{k=1}^{N} t_{kj}}{\sum_k c_k w_{kj}} c_k = \frac{1}{\sum_{j=1}^{M} w_{kj} p_j}$$

which can be solved by computing the p_j values using the first of the two equations above with an initial guess for the c_k values and by then refining these p_j values using the c_k values that are obtained by inserting the p_j values obtained into the second of the two equations above.

Notice that only $\sum_k t_{kj}$, which is the total number of configurations from all the replicas that enter the jth bin, enters the WHAM equations above. There is thus no need to record which replica generated each of the frames. One can thus simply gather the trajectories from all the replicas together at the outset. This observation is important as it is the basis of the binless formulation of WHAM that is implemented within PLUMED.

Examples

Glossary of keywords and components

Compulsory keywords

ARG	(default=∗.bias) the biases that must be taken into account when reweighting
MAXITER	(default=1000) maximum number of iterations for WHAM algorithm
WHAMTOL	(default=1e-10) threshold for convergence of WHAM algorithm

Options

TEMP	the system temperature. This is not required if your MD code passes this quantity to PLUMED

6.21 WHAM_HISTOGRAM

	This is part of the analysis module

This can be used to output the a histogram using the weighted histogram technique

This shortcut action allows you to calculate a histogram using the weighted histogram analysis technique. For more detail on how this the weights for configurations are computed see REWEIGHT_WHAM

Examples

The following input can be used to analyze the output from a series of umbrella sampling calculations. The trajectory from each of the simulations run with the different biases should be concatenated into a single trajectory before running the following analysis script on the concatenated trajectory using PLUMED driver. The umbrella sampling simulations that will be analyzed using the script below applied a harmonic restraint that restrained the torsional angle involving atoms 5, 7, 9 and 15 to particular values. The script below calculates the reweighting weights for each of the trajectories and then applies the binless WHAM algorithm to determine a weight for each configuration in the concatenated trajectory. A histogram is then constructed from the configurations visited and their weights. This histogram is then converted into a free energy surface and output to a file called fes.dat

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WHAM_HISTOGRAM.tmp
#SETTINGS NREPLICAS=4
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
rp: RESTRAINT ARG=phi KAPPA=50.0 ...
  AT=@replicas:{
        -3.0000000000000000000
        -1.4516129032258064516
        .09677419354838709664
        1.6451612903225806496
     }
...

hh: WHAM_HISTOGRAM ARG=phi BIAS=rp.bias TEMP=300 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=50
fes: CONVERT_TO_FES GRID=hh TEMP=300
DUMPGRID GRID=fes FILE=fes.dat
```

The script above must be run with multiple replicas using the following command:

```
mpirun -np 4 plumed driver --mf_xtc alltraj.xtc --multi 4
```

Glossary of keywords and components

Compulsory keywords

ARG	the arguments that you would like to make the histogram for
BIAS	(default=∗.bias) the value of the biases to use when performing WHAM
TEMP	the temperature at which the simulation was run
STRIDE	(default=1) the frequency with which the data should be stored to perform WHAM
GRID_MIN	the minimum to use for the grid
GRID_MAX	the maximum to use for the grid
GRID_BIN	the number of bins to use for the grid

Options

BANDWIDTH	the bandwidth for kernel density estimation

6.22 WHAM_WEIGHTS

This is part of the analysis module

Calculate and output weights for configurations using the weighted histogram analysis method.

This shortcut action allows you to calculate and output weights computed using the weighted histogram analysis technique. For more detail on how this technique works see REWEIGHT_WHAM

Examples

The following input can be used to analyze the output from a series of umbrella sampling calculations. The trajectory from each of the simulations run with the different biases should be concatenated into a single trajectory before running the following analysis script on the concatenated trajectory using PLUMED driver. The umbrella sampling simulations that will be analyzed using the script below applied a harmonic restraint that restrained the torsional angle involving atoms 5, 7, 9 and 15 to particular values. The script below calculates the reweighting weights for each of the trajectories and then applies the binless WHAM algorithm to determine a weight for each configuration in the concatenated trajectory.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/WHAM_WEIGHTS.tmp
#SETTINGS NREPLICAS=4
phi: TORSION ATOMS=5,7,9,15
rp: RESTRAINT ARG=phi KAPPA=50.0 ...
  AT=@replicas:{
        -3.0000000000000000000
        -1.4516129032258064516
        .09677419354838709664
        1.6451612903225806449
     }
...

WHAM_WEIGHTS BIAS=rp.bias TEMP=300 FILE=wham-weights
```

The script above must be run with multiple replicas using the following command:

```
mpirun -np 4 plumed driver --mf_xtc alltraj.xtc --multi 4
```

Glossary of keywords and components

Compulsory keywords

BIAS	(default=∗.bias) the value of the biases to use when performing WHAM
TEMP	the temperature at which the simulation was run
STRIDE	(default=1) the frequency with which the bias should be stored to perform WHAM
FILE	the file on which to output the WHAM weights

Options

FMT	the format to use for the real numbers in the output file

6.23 AVERAGE

This is part of the analysis module

Calculate the ensemble average of a collective variable

The ensemble average for a non-periodic, collective variable, s is given by the following expression:

$$\langle s \rangle = \frac{\sum_{t'=0}^{t} w(t')s(t')}{\sum_{t'=0}^{t} w(t')}$$

Here the sum runs over a the trajectory and $s(t')$ is used to denote the value of the collective variable at time t'. The final quantity evaluated is a weighted average as the weights, $w(t')$, allow us to negate the effect any bias might have on the region of phase space sampled by the system. This is discussed in the section of the manual on Analysis.

When the variable is periodic (e.g. TORSION) and has a value, s, in $a \le s \le b$ the ensemble average is evaluated using:

$$\langle s \rangle = a + \frac{b-a}{2\pi} \arctan \left[\frac{\sum_{t'=0}^{t} w(t') \sin\left(\frac{2\pi[s(t')-a]}{b-a}\right)}{\sum_{t'=0}^{t} w(t') \cos\left(\frac{2\pi[s(t')-a]}{b-a}\right)} \right]$$

Examples

The following example calculates the ensemble average for the distance between atoms 1 and 2 and output this to a file called COLVAR. In this example it is assumed that no bias is acting on the system and that the weights, $w(t')$ in the formulas above can thus all be set equal to one.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AVERAGE.tmp
d1: DISTANCE ATOMS=1,2
d1a: AVERAGE ARG=d1
PRINT ARG=d1a FILE=colvar STRIDE=100
```

The following example calculates the ensemble average for the torsional angle involving atoms 1, 2, 3 and 4. At variance with the previous example this quantity is periodic so the second formula in the above introduction is used to calculate the average. Furthermore, by using the CLEAR keyword we have specified that block averages are to be calculated. Consequently, after 100 steps all the information acquired thus far in the simulation is forgotten and the process of averaging is begun again. The quantities output in the colvar file are thus the block averages taken over the first 100 frames of the trajectory, the block average over the second 100 frames of trajectory and so on.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AVERAGE.tmp
t1: TORSION ATOMS=1,2,3,4
t1a: AVERAGE ARG=t1 CLEAR=100
PRINT ARG=t1a FILE=colvar STRIDE=100
```

This third example incorporates a bias. Notice that the effect the bias has on the ensemble average is removed by taking advantage of the REWEIGHT_BIAS method. The final ensemble averages output to the file are thus block ensemble averages for the unbiased canonical ensemble at a temperature of 300 K.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/AVERAGE.tmp
t1: TORSION ATOMS=1,2,3,4
RESTRAINT ARG=t1 AT=pi KAPPA=100.
ww: REWEIGHT_BIAS TEMP=300
t1a: AVERAGE ARG=t1 LOGWEIGHTS=ww CLEAR=100
PRINT ARG=t1a FILE=colvar STRIDE=100
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM

Options

TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile P↩ LUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

6.24 HISTOGRAM

	This is part of the analysis module

Accumulate the average probability density along a few CVs from a trajectory.

When using this method it is supposed that you have some collective variable ζ that gives a reasonable description of some physical or chemical phenomenon. As an example of what we mean by this suppose you wish to examine the following SN2 reaction:

$$\mathrm{OH}^- + \mathrm{CH}_3Cl \rightarrow \mathrm{CH}_3OH + \mathrm{Cl}^-$$

The distance between the chlorine atom and the carbon is an excellent collective variable, ζ, in this case because this distance is short for the reactant, CH_3Cl, because the carbon and chlorine are chemically bonded, and because it is long for the product state when these two atoms are not chemically bonded. We thus might want to accumulate the probability density, $P(\zeta)$, as a function of this distance as this will provide us with information about the overall likelihood of the reaction. Furthermore, the free energy, $F(\zeta)$, is related to this probability density via:

$$F(\zeta) = -k_B T \ln P(\zeta)$$

Accumulating these probability densities is precisely what this Action can be used to do. Furthermore, the conversion of the histogram to the free energy can be achieved by using the method CONVERT_TO_FES.

We calculate histograms within PLUMED using a method known as kernel density estimation, which you can read more about here:

https://en.wikipedia.org/wiki/Kernel_density_estimation

In PLUMED the value of ζ at each discrete instant in time in the trajectory is accumulated. A kernel, $K(\zeta - \zeta(t'), \sigma)$, centered at the current value, $\zeta(t)$, of this quantity is generated with a bandwidth σ, which is set by the user. These kernels are then used to accumulate the ensemble average for the probability density:

$$\langle P(\zeta) \rangle = \frac{\sum_{t'=0}^{t} w(t') K(\zeta - \zeta(t'), \sigma)}{\sum_{t'=0}^{t} w(t')}$$

Here the sums run over a portion of the trajectory specified by the user. The final quantity evaluated is a weighted average as the weights, $w(t')$, allow us to negate the effect any bias might have on the region of phase space sampled by the system. This is discussed in the section of the manual on Analysis.

A discrete analogue of kernel density estimation can also be used. In this analogue the kernels in the above formula are replaced by Dirac delta functions. When this method is used the final function calculated is no longer a probability density - it is instead a probability mass function as each element of the function tells you the value of an integral between two points on your grid rather than the value of a (continuous) function on a grid.

Additional material and examples can be also found in the tutorials Lugano tutorial: Brief guide to PLUMED syntax and analyzing traje

A note on block averaging and errors

Some particularly important issues related to the convergence of histograms and the estimation of error bars around the ensemble averages you calculate are covered in Trieste tutorial: Averaging, histograms and block analysis. The technique for estimating error bars that is known as block averaging is introduced in this tutorial. The essence of this technique is that the trajectory is split into a set of blocks and separate ensemble averages are calculated from each separate block of data. If $\{A_i\}$ is the set of N block averages that are obtained from this technique then the final error bar is calculated as:

$$\text{error} = \sqrt{\frac{1}{N}\frac{1}{N-1}\sum_{i=1}^{N}(A_i^2 - \langle A\rangle)^2} \qquad \text{where} \qquad \langle A\rangle = \frac{1}{N}\sum_{i=1}^{N} A_i$$

If the simulation is biased and reweighting is performed then life is a little more complex as each of the block averages should be calculated as a weighted average. Furthermore, the weights should be taken into account when the final ensemble and error bars are calculated. As such the error should be:

$$\text{error} = \sqrt{\frac{1}{N}\frac{\sum_{i=1}^{N} W_i}{\sum_{i=1}^{N} W_i - \sum_{i=1}^{N} W_i^2 / \sum_{i=1}^{N} W_i}\sum_{i=1}^{N} W_i(A_i^2 - \langle A\rangle)^2}$$

where W_i is the sum of all the weights for the ith block of data.

If we wish to calculate a normalized histogram we must calculate ensemble averages from our biased simulation using:

$$\langle H(x)\rangle = \frac{\sum_{t=1}^{M} w_t K(x - x_t, \sigma)}{\sum_{t=1}^{M} w_t}$$

where the sums runs over the trajectory, w_t is the weight of the tth trajectory frame, x_t is the value of the CV for the tth trajectory frame and K is a kernel function centered on x_t with bandwidth σ. The quantity that is evaluated is the value of the normalized histogram at point x. The following ensemble average will be calculated if you use the NORMALIZATION=true option in HISTOGRAM. If the ensemble average is calculated in this way we must calculate the associated error bars from our block averages using the second of the expressions above.

A number of works have shown that when biased simulations are performed it is often better to calculate an estimate of the histogram that is not normalized using:

$$\langle H(x)\rangle = \frac{1}{M}\sum_{t=1}^{M} w_t K(x - x_t, \sigma)$$

instead of the expression above. As such this is what is done by default in HISTOGRAM or if the NORMALIZAT↩ ION=ndata option is used. When the histogram is calculated in this second way the first of the two formula above can be used when calculating error bars from block averages.

Examples

The following input monitors two torsional angles during a simulation and outputs a continuous histogram as a function of them at the end of the simulation.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/HISTOGRAM.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  BANDWIDTH=0.05,0.05
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo
```

The following input monitors two torsional angles during a simulation and outputs a discrete histogram as a function of them at the end of the simulation.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/HISTOGRAM.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2
  KERNEL=DISCRETE
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo
```

The following input monitors two torsional angles during a simulation and outputs the histogram accumulated thus far every 100000 steps.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/HISTOGRAM.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  BANDWIDTH=0.05,0.05
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo STRIDE=100000
```

The following input monitors two torsional angles during a simulation and outputs a separate histogram for each 100000 steps worth of trajectory. Notice how the CLEAR keyword is used here and how it is not used in the previous example.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/HISTOGRAM.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2 CLEAR=100000
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  BANDWIDTH=0.05,0.05
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo STRIDE=100000
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
BANDWIDTH	the bandwidths for kernel density estimation
KERNEL	(default=gaussian) the kernel function you are using. More details on the kernels available in plumed plumed can be found in kernelfunctions.
NORMALIZATION	(default=ndata) This controls how the data is normalized it can be set equal to true, false or ndata. See above for an explanation
GRID_MIN	the lower bounds for the grid
GRID_MAX	the upper bounds for the grid

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
DATA	input data from action with vessel and compute histogram
VECTORS	input three dimensional vectors for computing histogram
GRID_BIN	the number of bins for the grid
GRID_SPACING	the approximate grid spacing (to be used as an alternative or together with GRID_BIN)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

6.25 MULTICOLVARDENS

This is part of the multicolvar module

Evaluate the average value of a multicolvar on a grid.

This keyword allows one to construct a phase field representation for a symmetry function from an atomistic description. If each atom has an associated order parameter, ϕ_i then a smooth phase field function $\phi(r)$ can be computed using:

$$\phi(\mathbf{r}) = \frac{\sum_i K(\mathbf{r}-\mathbf{r}_i)\phi_i}{\sum_i K(\mathbf{r}-\mathbf{r}_i)}$$

where $\mathbf{r}_i$ is the position of atom i, the sums run over all the atoms input and $K(\mathbf{r}-\mathbf{r}_i)$ is one of the kernelfunctions implemented in plumed. This action calculates the above function on a grid, which can then be used in the input to further actions.

Examples

The following example shows perhaps the simplest way in which this action can be used. The following input computes the density of atoms at each point on the grid and outputs this quantity to a file. In other words this input instructs plumed to calculate $\rho(\mathbf{r}) = \sum_i K(\mathbf{r}-\mathbf{r}_i)$

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MULTICOLVARDENS.tmp
dens: DENSITY SPECIES=1-100
grid: MULTICOLVARDENS DATA=dens ORIGIN=1 DIR=xyz NBINS=100,100,100 BANDWIDTH=0.05,0.05,0.05 STRIDE=1
DUMPGRID GRID=grid STRIDE=500 FILE=density
```

In the above example density is added to the grid on every step. The PRINT_GRID instruction thus tells PLUMED to output the average density at each point on the grid every 500 steps of simulation. Notice that the that grid output on step 1000 is an average over all 1000 frames of the trajectory. If you would like to analyze these two blocks of data separately you must use the CLEAR flag.

This second example computes an order parameter (in this case FCCUBIC) and constructs a phase field model for this order parameter using the equation above.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MULTICOLVARDENS.tmp
fcc: FCCUBIC SPECIES=1-5184 SWITCH={CUBIC D_0=1.2 D_MAX=1.5} ALPHA=27
dens: MULTICOLVARDENS DATA=fcc ORIGIN=1 DIR=xyz NBINS=14,14,28 BANDWIDTH=1.0,1.0,1.0 STRIDE=1 CLEAR=1
DUMPCUBE GRID=dens STRIDE=1 FILE=dens.cube
```

In this example the phase field model is computed and output to a file on every step of the simulation. Furthermore, because the CLEAR=1 keyword is set on the MULTICOLVARDENS line each Gaussian cube file output is a phase field model for a particular trajectory frame. The average value accumulated thus far is cleared at the start of every single timestep and there is no averaging over trajectory frames in this case.

Glossary of keywords and components

The atoms involved can be specified using

ORIGIN	we will use the position of this atom as the origin. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM
BANDWIDTH	the bandwidths for kernel density estimation
KERNEL	(default=gaussian) the kernel function you are using. More details on the kernels available in plumed plumed can be found in kernelfunctions.
DATA	the multicolvar which you would like to calculate the density profile for
DIR	the direction in which to calculate the density profile

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
FRACTIONAL	(default=off) use fractional coordinates for the various axes
XREDUCED	(default=off) limit the calculation of the density/average to a portion of the z-axis only
YREDUCED	(default=off) limit the calculation of the density/average to a portion of the y-axis only
ZREDUCED	(default=off) limit the calculation of the density/average to a portion of the z-axis only
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
NBINS	the number of bins to use to represent the density profile
SPACING	the approximate grid spacing (to be used as an alternative or together with NBINS)
XLOWER	this is required if you are using XREDUCED. It specifies the lower bound for the region of the x-axis that for which you are calculating the density/average
XUPPER	this is required if you are using XREDUCED. It specifies the upper bound for the region of the x-axis that for which you are calculating the density/average
YLOWER	this is required if you are using YREDUCED. It specifies the lower bound for the region of the y-axis that for which you are calculating the density/average
YUPPER	this is required if you are using YREDUCED. It specifies the upper bound for the region of the y-axis that for which you are calculating the density/average
ZLOWER	this is required if you are using ZREDUCED. It specifies the lower bound for the region of the z-axis that for which you are calculating the density/average
ZUPPER	this is required if you are using ZREDUCED. It specifies the upper bound for the region of the z-axis that for which you are calculating the density/average

6.26 CONVERT_TO_FES

	This is part of the gridtools module

Convert a histogram, H(x), to a free energy surface using F(x) = -k_B T ln H(x).

This action allows you to take a free energy surface that was calculated using the HISTOGRAM action and to convert it to a free energy surface. This transformation performed by doing:

$$F(x) = -k_B T \ln H(x)$$

The free energy calculated on a grid is output by this action and can be printed using DUMPGRID

Examples

This is a typical example showing how CONVERT_TO_FES might be used when post processing a trajectory. The input below calculates the free energy as a function of the distance between atom 1 and atom 2. This is done by accumulating a histogram as a function of this distance using kernel density estimation and the HISTOGRAM action. All the data within this trajectory is used in the construction of this HISTOGRAM. Finally, once all the data has been read in, the histogram is converted to a free energy using the formula above and the free energy is output to a file called fes.dat

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CONVERT_TO_FES.tmp
x: DISTANCE ATOMS=1,2
hA1: HISTOGRAM ARG=x GRID_MIN=0.0 GRID_MAX=3.0 GRID_BIN=100 BANDWIDTH=0.1
ff: CONVERT_TO_FES GRID=hA1 TEMP=300
DUMPGRID GRID=ff FILE=fes.dat
```

Glossary of keywords and components

Compulsory keywords

GRID	the action that creates the input grid you would like to use

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

MINTOZERO	(default=off) set the minimum in the free energy to be equal to zero
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to use
TEMP	the temperature at which you are operating

6.27 DUMPCUBE

This is part of the gridtools module

Output a three dimensional grid using the Gaussian cube file format.

Suppose you have calculated the value of a function on a three dimensional grid. This function might be a HISTOGRAM or it might be a free energy energy surface that was calculated from this histogram by using CONVERT_TO_FES. Alternatively, your function might be a phase-field that was calculated using MULTICOLVARDENS. Whatever the function is, however, you obviously cannot show it using a typical contour plotting program such as gnuplot as you have three input variables.

Tools like VMD have nice features for plotting these types of three dimensional functions but typically you are required to use a Gaussian cube file format to input the data. This action thus allows you to output a function evaluated on a grid to a Gaussian cube file format.

Examples

The input below can be used to post process a trajectory. A histogram as a function of the distance between atoms 1 and 2, the distance between atom 1 and 3 and the angle between the vector connecting atoms 1 and 2 and 1 and 3 is computed using kernel density estimation. Once all the data contained in the trajectory has been read in and all the kernels have been added the resulting histogram is output to a file called histoA1.cube. This file has the Gaussian cube file format. The histogram can thus be visualized using tools such as VMD.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPCUBE.tmp
x1: DISTANCE ATOMS=1,2
x2: DISTANCE ATOMS=1,3
x3: ANGLE ATOMS=1,2,3

hA1: HISTOGRAM ARG=x1,x2,x3 GRID_MIN=0.0,0.0,0.0 GRID_MAX=3.0,3.0,3.0 GRID_BIN=10,10,10 BANDWIDTH=1.0,1.0,1.0
DUMPCUBE GRID=hA1 FILE=histoA1.cube
```

Glossary of keywords and components

Compulsory keywords

GRID	the action that creates the grid you would like to output
STRIDE	(default=0) the frequency with which the grid should be output to the file. The default value of 0 ensures that the grid is only output at the end of the trajectory
FILE	(default=density) the file on which to write the grid.
REPLICA	(default=0) the replica for which you would like to output this information

Options

FMT	the format that should be used to output real numbers
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to output

6.28 DUMPGRID

This is part of the gridtools module

Output the function on the grid to a file with the PLUMED grid format.

PLUMED provides a number of actions that calculate the values of functions on grids. For instance, whenever you calculate a free energy as a function of a collective variable using HISTOGRAM and CONVERT_TO_FES you will generally want to output the value of the free energy at a number of points on a discrete grid that covers the CV space uniformly. Alternatively you may want to calculate what value some symmetry function takes at different points inside your simulation cell using MULTICOLVARDENS.

This action allows you to output these functions calculated on a grid using a format that can be read in using gnuplot and other such plotting programs. The file output using this action will have a header that contains some essential information about the function plotted and that looks something like this:

```
#! FIELDS x y hA1 dhA1_x dhA1_x
#! SET normalisation    2.0000
#! SET min_x 0.0
#! SET max_x 3.0
#! SET nbins_x  100
#! SET periodic_x false
#! SET min_y 0.0
#! SET max_y 3.0
#! SET nbins_y  100
#! SET periodic_y false
```

The header shown here tells us that we have grid showing the values that a function with two arguments x and y takes at various points in our cell. The lines beneath the first line then tell us a little bit about these two input arguments.

The remaining lines of the file give us information on the positions of our grid points and the value the function and its partial derivatives with respect to x and y. If the header is as above a list of values of the function that have x=0 and 100 values of y between 0.0 and 3.0 will be provided. This block of data will be followed with a blank line. There will then be a second block of values which will all have been evaluated the same value of x and all possible values for y. This block is then followed by a blank line again and this pattern continues until all points of the grid have been covered.

Examples

The following input monitors two torsional angles during a simulation and outputs a continuous histogram as a function of them at the end of the simulation.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPGRID.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  BANDWIDTH=0.05,0.05
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo
```

The following input monitors two torsional angles during a simulation and outputs a discrete histogram as a function of them at the end of the simulation.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPGRID.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2
  KERNEL=DISCRETE
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo
```

The following input monitors two torsional angles during a simulation and outputs the histogram accumulated thus far every 100000 steps.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPGRID.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  BANDWIDTH=0.05,0.05
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo STRIDE=100000
```

The following input monitors two torsional angles during a simulation and outputs a separate histogram for each 100000 steps worth of trajectory. Notice how the CLEAR keyword is used here and how it is not used in the previous example.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DUMPGRID.tmp
TORSION ATOMS=1,2,3,4 LABEL=r1
TORSION ATOMS=2,3,4,5 LABEL=r2
HISTOGRAM ...
  ARG=r1,r2 CLEAR=100000
  GRID_MIN=-3.14,-3.14
  GRID_MAX=3.14,3.14
  GRID_BIN=200,200
  BANDWIDTH=0.05,0.05
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo STRIDE=100000
```

Glossary of keywords and components

Compulsory keywords

GRID	the action that creates the grid you would like to output
STRIDE	(default=0) the frequency with which the grid should be output to the file. The default value of 0 ensures that the grid is only output at the end of the trajectory
FILE	(default=density) the file on which to write the grid.
REPLICA	(default=0) the replica for which you would like to output this information

Options

FMT	the format that should be used to output real numbers

6.29 FIND_CONTOUR

	This is part of the gridtools module

Find an isocontour in a smooth function.

As discussed in the part of the manual on Analysis PLUMED contains a number of tools that allow you to calculate a function on a grid. The function on this grid might be a HISTOGRAM as a function of a few collective variables or it might be a phase field that has been calculated using MULTICOLVARDENS. If this function has one or two input arguments it is relatively straightforward to plot the function. If by contrast the data has a three or more dimensions it can be difficult to visualize.

This action provides one tool for visualizing these functions. It can be used to search for a set of points on a contour where the function takes a particular values. In other words, for the function $f(x,y)$ this action would find a set of points $\{x_c, y_c\}$ that have:

$$f(x_c, y_c) - c = 0$$

where c is some constant value that is specified by the user. The points on this contour are detected using a variant on the marching squares or marching cubes algorithm, which you can find information on here:

https://en.wikipedia.org/wiki/Marching_squares https://en.wikipedia.org/wiki/Marching_cubes

As such, and unlike FIND_CONTOUR_SURFACE or FIND_SPHERICAL_CONTOUR, the function input to this action can have any dimension. Furthermore, the topology of the contour will be determined by the algorithm and does not need to be specified by the user.

Examples

The input below allows you to calculate something akin to a Willard-Chandler dividing surface [39]. The simulation cell in this case contains a solid phase and a liquid phase. The Willard-Chandler surface is the surface that separates the parts of the box containing the solid from the parts containing the liquid. To compute the position of this surface the FCCUBIC symmetry function is calculated for each of the atoms in the system from on the geometry of the atoms in the first coordination sphere of each of the atoms. These quantities are then transformed using a switching function. This procedure generates a single number for each atom in the system and this quantity has a value of one for atoms that are in parts of the box that resemble the solid structure and zero for atoms that are in parts of the box that resemble the liquid. The position of a virtual atom is then computed using CENTER_OF_MULTICOLVAR and a phase field model is constructed using MULTICOLVARDENS. These procedure ensures that we have a continuous function that gives a measure of the average degree of solidness at each point in the simulation cell. The Willard-Chandler dividing surface is calculated by finding a a set of points at which the value of this phase field is equal to 0.5. This set of points is output to file called mycontour.dat. A new contour is found on every single step for each frame that is read in.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIND_CONTOUR.tmp
UNITS NATURAL
FCCUBIC ...
  SPECIES=1-96000 SWITCH={CUBIC D_0=1.2 D_MAX=1.5}
  ALPHA=27 PHI=0.0 THETA=-1.5708 PSI=-2.35619 LABEL=fcc
... FCCUBIC

tfcc: MTRANSFORM_MORE DATA=fcc LOWMEM SWITCH={SMAP R_0=0.5 A=8 B=8}
center: CENTER_OF_MULTICOLVAR DATA=tfcc

dens: MULTICOLVARDENS ...
  DATA=tfcc ORIGIN=center DIR=xyz
  NBINS=80,80,80 BANDWIDTH=1.0,1.0,1.0 STRIDE=1 CLEAR=1
...

FIND_CONTOUR GRID=dens CONTOUR=0.5 FILE=mycontour.xyz
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM
GRID	the action that creates the input grid you would like to use
CONTOUR	the value we would like to draw the contour at in the space
BUFFER	(default=0) number of buffer grid points around location where grid was found on last step. If this is zero the full grid is calculated on each step
FILE	file on which to output coordinates
UNITS	(default=PLUMED) the units in which to print out the coordinates. PLUMED means internal PLUMED units

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to use
PRECISION	The number of digits in trajectory file

6.30 FIND_CONTOUR_SURFACE

This is part of the gridtools module

Find an isocontour by searching along either the x, y or z direction.

As discussed in the part of the manual on Analysis PLUMED contains a number of tools that allow you to calculate a function on a grid. The function on this grid might be a HISTOGRAM as a function of a few collective variables or it might be a phase field that has been calculated using MULTICOLVARDENS. If this function has one or two input arguments it is relatively straightforward to plot the function. If by contrast the data has a three dimensions it can be difficult to visualize.

This action provides one tool for visualizing these functions. It can be used to search for a set of points on a contour where the function takes a particular value. In other words, for the function $f(x,y,z)$ this action would find a set of points $\{x_c, y_c, z_c\}$ that have:

$$f(x_c, y_c, z_c) - c = 0$$

where c is some constant value that is specified by the user. The points on this contour are find by searching along lines that run parallel to the x, y or z axis of the simulation cell. The result is, therefore, a two dimensional function evaluated on a grid that gives us the height of the interface as a function of two coordinates.

It is important to note that this action can only be used to detect contours in three dimensional functions. In addition, this action will fail to find the full set of contour points if the contour does not have the same topology as an infinite plane. If you are uncertain that the isocontours in your function have the appropriate topology you should use FIND_CONTOUR in place of FIND_CONTOUR_SURFACE.

Examples

The input shown below was used to analyze the results from a simulation of an interface between solid and molten Lennard Jones. The interface between the solid and the liquid was set up in the plane perpendicular to the z direction of the simulation cell. The input below calculates something akin to a Willard-Chandler dividing surface [39] between the solid phase and the liquid phase. There are two of these interfaces within the simulation box because of the periodic boundary conditions but we were able to determine that one of these two surfaces lies in

a particular part of the simulation box. The input below detects the height profile of one of these two interfaces. It does so by computing a phase field average of the FCCUBIC symmetry function using the MULTICOLVARDENS action. Notice that we use the fact that we know roughly where the interface is when specifying how this phase field is to be calculated and specify the region over the z-axis in which we are going to search for the phase field in the line defining the MULTICOLVARDENS. Once we have calculated the phase field we search for contour points on the lines that run parallel to the z-direction of the cell box using the FIND_CONTOUR_SURFACE command. The final result is a 14×14 grid of values for the height of the interface as a function of the (x, y) position. This grid is then output to a file called contour2.dat.

Notice that the commands below calculate the instantaneous position of the surface separating the solid and liquid and that as such the accumulated average is cleared on every step.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIND_CONTOUR_SURFACE.tmp
UNITS NATURAL
FCCUBIC ...
  SPECIES=1-96000 SWITCH={CUBIC D_0=1.2 D_MAX=1.5}
  ALPHA=27 PHI=0.0 THETA=-1.5708 PSI=-2.35619 LABEL=fcc
... FCCUBIC

dens2: MULTICOLVARDENS DATA=fcc ORIGIN=1 DIR=xyz NBINS=14,14,50 ZREDUCED ZLOWER=6.0 ZUPPER=11.0 BANDWIDTH=1.0,

ss2: FIND_CONTOUR_SURFACE GRID=dens2 CONTOUR=0.42 SEARCHDIR=z STRIDE=1 CLEAR=1
DUMPGRID GRID=ss2 FILE=contour2.dat FMT=%8.4f STRIDE=1
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM
GRID	the action that creates the input grid you would like to use
CONTOUR	the value we would like to draw the contour at in the space
SEARCHDIR	In which directions do you wish to search for the contour.
BUFFER	(default=0) number of buffer grid points around location where grid was found on last step. If this is zero the full grid is calculated on each step

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to use

6.31 FIND_SPHERICAL_CONTOUR

This is part of the gridtools module

Find an isocontour in a three dimensional grid by searching over a Fibonacci sphere.

As discussed in the part of the manual on Analysis PLUMED contains a number of tools that allow you to calculate a function on a grid. The function on this grid might be a HISTOGRAM as a function of a few collective variables or it might be a phase field that has been calculated using MULTICOLVARDENS. If this function has one or two input arguments it is relatively straightforward to plot the function. If by contrast the data has a three dimensions it can be difficult to visualize.

This action provides one tool for visualizing these functions. It can be used to search for a set of points on a contour where the function takes a particular value. In other words, for the function $f(x,y,z)$ this action would find a set of points $\{x_c, y_c, z_c\}$ that have:

$$f(x_c, y_c, z_c) - c = 0$$

where c is some constant value that is specified by the user. The points on this contour are find by searching along a set of equally spaced radii of a sphere that centered at on particular, user-specified atom or virtual atom. To ensure that these search radii are equally spaced on the surface of the sphere the search directions are generated by using a Fibonacci spiral projected on a sphere. In other words, the search directions are given by:

$$\mathbf{r}_i = \left(\sqrt{1-y^2}\cos(\phi)\frac{2i}{n} - 1 + \frac{1}{n}\sqrt{1-y^2}\sin(\phi) \right)$$

where y is the quantity second component of the vector defined above, n is the number of directions to look in and ϕ is

$$\phi = (i + R, n)\pi(3 - \sqrt{5})$$

where R is a random variable between 0 and $n-1$ that is generated during the read in of the input file and that is fixed during the whole calculation.

It is important to note that this action can only be used to detect contours in three dimensional functions. In addition, this action will fail to find the full set of contour points if the contour does not have the same topology as a sphere. If you are uncertain that the isocontours in your function have a spherical topology you should use FIND_CONTOUR in place of FIND_SPHERICAL_CONTOUR.

Examples

The following input demonstrates how this action can be used. The input here is used to study the shape of a droplet that has been formed during the condensation of Lennard Jones from the vapor. The input below achieves this by calculating the coordination numbers of all the atoms within the gas. Obviously, those atoms within the droplet will have a large value for the coordination number while the isolated atoms in the gas will have a low value. As such we can detect the sizes of the droplets by constructing a CONTACT_MATRIX whose ij element tells us whether atom i and atom j have coordination number that is greater that two. The atoms within the various droplets within the system can then be found by performing a DFSCLUSTERING on this matrix to detect the connected components. We can take the largest of these connected components and find the center of the droplet by exploiting the functionality within CENTER_OF_MULTICOLVAR. We can then construct a phase field based on the positions of the atoms in the largest cluster and the values of the coordination numbers of these atoms. The final line in the input then finds the a set of points on the dividing surface that separates the droplet from the surrounding gas. The value of the phase field on this isocontour is equal to 0.75.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FIND_SPHERICAL_CONTOUR.tmp
# Calculate coordination numbers
c1: COORDINATIONNUMBER SPECIES=1-512 SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
# Select coordination numbers that are more than 2.0
cf: MFILTER_MORE DATA=c1 SWITCH={RATIONAL D_0=2.0 R_0=0.1} LOWMEM
# Build a contact matrix
mat: CONTACT_MATRIX ATOMS=cf SWITCH={EXP D_0=4.0 R_0=0.5 D_MAX=6.0}
# Find largest cluster
dfs: DFSCLUSTERING MATRIX=mat LOWMEM
clust1: CLUSTER_PROPERTIES CLUSTERS=dfs CLUSTER=1
# Find center of largest cluster
trans1: MTRANSFORM_MORE DATA=clust1 SWITCH={RATIONAL D_0=2.0 R_0=0.1} LOWMEM
cent: CENTER_OF_MULTICOLVAR DATA=trans1
# Calculate the phase field of the coordination
dens: MULTICOLVARDENS DATA=trans1 ORIGIN=cent DIR=xyz NBINS=30,30,30 BANDWIDTH=2.0,2.0,2.0
# Find the isocontour around the nucleus
sc: FIND_SPHERICAL_CONTOUR GRID=dens CONTOUR=0.85 INNER_RADIUS=10.0 OUTER_RADIUS=40.0 NPOINTS=100
# And print the grid to a file
GRID_TO_XYZ GRID=sc FILE=mysurface.xyz UNITS=A
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM
GRID	the action that creates the input grid you would like to use
CONTOUR	the value we would like to draw the contour at in the space
NPOINTS	the number of points for which we are looking for the contour
INNER_RADIUS	the minimum radius on which to look for the contour
OUTER_RADIUS	the outer radius on which to look for the contour
NBINS	(default=1) the number of discrete sections in which to divide the distance between the inner and outer radius when searching for a contour

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to use

6.32 FOURIER_TRANSFORM

This is part of the gridtools module

Compute the Discrete Fourier Transform (DFT) by means of FFTW of data stored on a 2D grid.

This action can operate on any other action that outputs scalar data on a two-dimensional grid.

Up to now, even if the input data are purely real the action uses a complex DFT.

Just as a quick reference, given a 1D array $\mathbf{X}$ of size n, this action computes the vector $\mathbf{Y}$ given by

$$Y_k = \sum_{j=0}^{n-1} X_j e^{2\pi\, jk\sqrt{-1}/n}.$$

This can be easily extended to more than one dimension. All the other details can be found at http://www.fftw.org/doc/What-FFTW-Really-Computes.html#What-FFTW-Really-Computes.

The keyword "FOURIER_PARAMETERS" deserves just a note on the usage. This keyword specifies how the Fourier transform will be normalized. The keyword takes two numerical parameters (a, b) that define the normalization according to the following expression

$$\frac{1}{n^{(1-a)/2}} \sum_{j=0}^{n-1} X_j e^{2\pi b\, jk\sqrt{-1}/n}$$

The default values of these parameters are: $a = 1$ and $b = 1$.

Examples

The following example tells Plumed to compute the complex 2D 'backward' Discrete Fourier Transform by taking the data saved on a grid called 'density', and normalizing the output by $\frac{1}{\sqrt{N_x\, N_y}}$, where N_x and N_y are the number of data on the grid (it can be the case that $N_x \neq N_y$):

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FOURIER_TRANSFORM.tmp
FOURIER_TRANSFORM STRIDE=1 GRID=density FT_TYPE=complex FOURIER_PARAMETERS=0,-1
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM
GRID	the action that creates the input grid you would like to use
FOURIER_PARAMETERS	(default=default) what kind of normalization is applied to the output and if the Fourier transform in FORWARD or BACKWARD. This keyword takes the form FOURIER_PARAMETERS=A,B, where A and B can be 0, 1 or -1. The default values are A=1 (no normalization at all) and B=1 (forward FFT). Other possible choices for A are: A=-1: normalize by the number of data, A=0: normalize by the square root of the number of data (one forward and followed by backward FFT recover the original data).

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to use
FT_TYPE	choose what kind of data you want as output on the grid. Possible values are: ABS = compute the complex modulus of Fourier coefficients (DEFAULT); NORM = compute the norm (i.e. ABS^2) of Fourier coefficients; COMPLEX = store the FFTW complex output on the grid (as a vector).

6.33 GRID_TO_XYZ

This is part of the gridtools module

Output the function on the grid to an xyz file

Examples

Glossary of keywords and components

Compulsory keywords

GRID	the action that creates the grid you would like to output
STRIDE	(default=0) the frequency with which the grid should be output to the file. The default value of 0 ensures that the grid is only output at the end of the trajectory
FILE	(default=density) the file on which to write the grid.
REPLICA	(default=0) the replica for which you would like to output this information
UNITS	(default=PLUMED) the units in which to print out the coordinates. PLUMED means internal PL↩ UMED units

Options

COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to output
PRECISION	The number of digits in trajectory file

6.34 INTEGRATE_GRID

	This is part of the gridtools module

Calculate the total integral of the function on the input grid

Examples

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM
GRID	the action that creates the input grid you would like to use
CLEAR	(default=1) the frequency with which to clear all the accumulated data.

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to use

6.35 INTERPOLATE_GRID

This is part of the gridtools module

Interpolate a smooth function stored on a grid onto a grid with a smaller grid spacing.

This action takes a function evaluated on a grid as input and can be used to interpolate the values of that function on to a finer grained grid. The interpolation within this algorithm is done using splines.

Examples

The input below can be used to post process a trajectory. It calculates a HISTOGRAM as a function the distance between atoms 1 and 2 using kernel density estimation. During the calculation the values of the kernels are evaluated at 100 points on a uniform grid between 0.0 and 3.0. Prior to outputting this function at the end of the simulation this function is interpolated onto a finer grid of 200 points between 0.0 and 3.0.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INTERPOLATE_GRID.tmp
x: DISTANCE ATOMS=1,2
hA1: HISTOGRAM ARG=x GRID_MIN=0.0 GRID_MAX=3.0 GRID_BIN=100 BANDWIDTH=0.1
ii: INTERPOLATE_GRID GRID=hA1 GRID_BIN=200
DUMPGRID GRID=ii FILE=histo.dat
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the data should be collected and added to the quantity being averaged
CLEAR	(default=0) the frequency with which to clear all the accumulated data. The default value of 0 implies that all the data will be used and that the grid will never be cleared
NORMALIZATION	(default=true) This controls how the data is normalized it can be set equal to true, false or ndata. The differences between these options are explained in the manual page for HISTOGRAM
GRID	the action that creates the input grid you would like to use

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specify the component of the input vector field for which you wish to use
GRID_BIN	the number of bins for the grid
GRID_SPACING	the approximate grid spacing (to be used as an alternative or together with GRID_BIN)

6.36 COLLECT_FRAMES

	This is part of the analysis module

This allows you to convert a trajectory and a dissimilarity matrix into a dissimilarity object

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

ATOMS	the atoms whose positions we are tracking for the purpose of analyzing the data
STRIDE	the frequency with which data should be stored for analysis. By default data is collected on every step

Compulsory keywords

CLEAR	(default=0) the frequency with which data should all be deleted and restarted

Options

LOWMEM	(default=off) lower the memory requirements
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile P↩ LUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
LOGWEIGHTS	list of actions that calculates log weights that should be used to weight configurations when calculating averages

6.37 EUCLIDEAN_DISSIMILARITIES

	This is part of the analysis module

Calculate the matrix of dissimilarities between a trajectory of atomic configurations.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Alternatively data can be collected from the trajectory using

ATOMS	the list of atoms that you are going to use in the measure of distance that you are using. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

METRIC	(default=EUCLIDEAN) the method that you are going to use to measure the distances between points

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

6.38 PRINT_DISSIMILARITY_MATRIX

	This is part of the analysis module

Print the matrix of dissimilarities between a trajectory of atomic configurations.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

FILE	name of file on which to output the data
STRIDE	(default=0) the frequency with which to perform the required analysis and to output the data. The default value of 0 tells plumed to use all the data

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
FMT	the format to use for the output of numbers

6.39 READ_DISSIMILARITY_MATRIX

	This is part of the analysis module

Read a matrix of dissimilarities between a trajectory of atomic configurations from a file.

Examples

Glossary of keywords and components

Compulsory keywords

FILE	an input file containing the matrix of dissimilarities

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object
WFILE	input file containing weights of points

6.40 LANDMARK_SELECT_FPS

	This is part of the analysis module

Select a set of landmarks using farthest point sampling.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

NLANDMARKS	the number of landmarks that you would like to select
SEED	(default=1234) a random number seed

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
NOVORONOI	(default=off) do not do a Voronoi analysis of the data to determine weights of final points
IGNORE_WEIGHTS	(default=off) ignore the weights in the underlying analysis object

6.41 LANDMARK_SELECT_RANDOM

	This is part of the analysis module

Select a random set of landmarks from a large set of configurations.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

NLANDMARKS	the number of landmarks that you would like to select
SEED	(default=1234) a random number seed

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
NOVORONOI	(default=off) do not do a Voronoi analysis of the data to determine weights of final points
IGNORE_WEIGHTS	(default=off) ignore the weights in the underlying analysis object

6.42 LANDMARK_SELECT_STAGED

	This is part of the analysis module

Select a set of landmarks using the staged algorithm.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

NLANDMARKS	the number of landmarks that you would like to select
GAMMA	the gamma parameter to be used in weights
SEED	(default=1234) a random number seed

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
NOVORONOI	(default=off) do not do a Voronoi analysis of the data to determine weights of final points
IGNORE_WEIGHTS	(default=off) ignore the weights in the underlying analysis object

6.43 LANDMARK_SELECT_STRIDE

	This is part of the analysis module

Select every kth landmark from the trajectory.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

NLANDMARKS	the number of landmarks that you would like to select

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
NOVORONOI	(default=off) do not do a Voronoi analysis of the data to determine weights of final points
IGNORE_WEIGHTS	(default=off) ignore the weights in the underlying analysis object

6.44 RESELECT_LANDMARKS

	This is part of the analysis module

This allows us to use one measure in landmark selection and a different measure in dimensionality reduction

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

LANDMARKS	the action that selects the landmarks that you want to reselect using this action

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
NOVORONOI	(default=off) do not do a Voronoi analysis of the data to determine weights of final points
IGNORE_WEIGHTS	(default=off) ignore the weights in the underlying analysis object

6.45 CLASSICAL_MDS

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Create a low-dimensional projection of a trajectory using the classical multidimensional scaling algorithm.

Multidimensional scaling (MDS) is similar to what is done when you make a map. You start with distances between London, Belfast, Paris and Dublin and then you try to arrange points on a piece of paper so that the (suitably scaled) distances between the points in your map representing each of those cities are related to the true distances between the cities. Stating this more mathematically MDS endeavors to find an `isometry` between points distributed in a high-dimensional space and a set of points distributed in a low-dimensional plane. In other words, if we have M D-dimensional points, $\mathbf{X}$, and we can calculate dissimilarities between pairs them, D_{ij}, we can, with an MDS calculation, try to create M projections, $\mathbf{x}$, of the high dimensionality points in a d-dimensional linear space by trying to arrange the projections so that the Euclidean distances between pairs of them, d_{ij}, resemble the dissimilarities between the high dimensional points. In short we minimize:

$$\chi^2 = \sum_{i \neq j} (D_{ij} - d_{ij})^2$$

where D_{ij} is the distance between point X^i and point X^j and d_{ij} is the distance between the projection of X^i, x^i, and the projection of X^j, x^j. A tutorial on this approach can be used to analyze simulations can be found in the tutorial Lugano tutorial: Dimensionality reduction and in the following `short video.`

Examples

The following command instructs plumed to construct a classical multidimensional scaling projection of a trajectory. The RMSD distance between atoms 1-256 have moved is used to measure the distances in the high-dimensional space.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CLASSICAL_MDS.tmp
data: COLLECT_FRAMES ATOMS=1-256
mat: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=data
mds: CLASSICAL_MDS USE_OUTPUT_DATA_FROM=mat NLOW_DIM=2
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=mds FILE=rmsd-embed
```

The following section is for people who are interested in how this method works in detail. A solid understanding of this material is not necessary to use MDS.

6.45.1 Method of optimization

The stress function can be minimized using a standard optimization algorithm such as conjugate gradients or steepest descent. However, it is more common to do this minimization using a technique known as classical scaling. Classical scaling works by recognizing that each of the distances D_{ij} in the above sum can be written as:

$$D_{ij}^2 = \sum_\alpha (X_\alpha^i - X_\alpha^j)^2 = \sum_\alpha (X_\alpha^i)^2 + (X_\alpha^j)^2 - 2X_\alpha^i X_\alpha^j$$

We can use this expression and matrix algebra to calculate multiple distances at once. For instance if we have three points, $\mathbf{X}$, we can write distances between them as:

$$\begin{aligned} D^2(\mathbf{X}) &= \begin{bmatrix} 0 & d_{12}^2 & d_{13}^2 \\ d_{12}^2 & 0 & d_{23}^2 \\ d_{13}^2 & d_{23}^2 & 0 \end{bmatrix} \\ &= \sum_\alpha \begin{bmatrix} (X_\alpha^1)^2 & (X_\alpha^1)^2 & (X_\alpha^1)^2 \\ (X_\alpha^2)^2 & (X_\alpha^2)^2 & (X_\alpha^2)^2 \\ (X_\alpha^3)^2 & (X_\alpha^3)^2 & (X_\alpha^3)^2 \end{bmatrix} + \sum_\alpha \begin{bmatrix} (X_\alpha^1)^2 & (X_\alpha^2)^2 & (X_\alpha^3)^2 \\ (X_\alpha^1)^2 & (X_\alpha^2)^2 & (X_\alpha^3)^2 \\ (X_\alpha^1)^2 & (X_\alpha^2)^2 & (X_\alpha^3)^2 \end{bmatrix} - 2\sum_\alpha \begin{bmatrix} X_\alpha^1 X_\alpha^1 & X_\alpha^1 X_\alpha^2 & X_\alpha^1 X_\alpha^3 \\ X_\alpha^2 X_\alpha^1 & X_\alpha^2 X_\alpha^2 & X_\alpha^2 X_\alpha^3 \\ X_\alpha^1 X_\alpha^3 & X_\alpha^3 X_\alpha^2 & X_\alpha^3 X_\alpha^3 \end{bmatrix} \\ &= \mathbf{c1^T} + \mathbf{1c^T} - 2\sum_\alpha \mathbf{x}_a \mathbf{x}_a^T = \mathbf{c1^T} + \mathbf{1c^T} - 2\mathbf{XX^T} \end{aligned}$$

This last equation can be extended to situations when we have more than three points. In it $\mathbf{X}$ is a matrix that has one high-dimensional point on each of its rows and $\mathbf{X^T}$ is its transpose. $\mathbf{1}$ is an $M \times 1$ vector of ones and $\mathbf{c}$ is a vector with components given by:

$$c_i = \sum_\alpha (x_\alpha^i)^2$$

These quantities are the diagonal elements of $\mathbf{XX^T}$, which is a dot product or Gram Matrix that contains the dot product of the vector X_i with the vector X_j in element i, j.

In classical scaling we introduce a centering matrix $\mathbf{J}$ that is given by:

$$\mathbf{J} = \mathbf{I} - \frac{1}{M}\mathbf{11^T}$$

where $\mathbf{I}$ is the identity. Multiplying the equations above from the front and back by this matrix and a factor of a $-\frac{1}{2}$ gives:

$$
\begin{aligned}
-\frac{1}{2}\mathbf{J}\mathbf{D}^2(\mathbf{X})\mathbf{J} &= -\frac{1}{2}\mathbf{J}(\mathbf{c}\mathbf{1}^\mathbf{T} + \mathbf{1}\mathbf{c}^\mathbf{T} - 2\mathbf{X}\mathbf{X}^\mathbf{T})\mathbf{J} \\
&= -\frac{1}{2}\mathbf{J}\mathbf{c}\mathbf{1}^\mathbf{T}\mathbf{J} - \frac{1}{2}\mathbf{J}\mathbf{1}\mathbf{c}^\mathbf{T}\mathbf{J} + \frac{1}{2}\mathbf{J}(2\mathbf{X}\mathbf{X}^\mathbf{T})\mathbf{J} \\
&= \mathbf{J}\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{J} = \mathbf{X}\mathbf{X}^\mathbf{T}
\end{aligned}
$$

The fist two terms in this expression disappear because $\mathbf{1}^\mathbf{T}\mathbf{J} = \mathbf{J}\mathbf{1} = \mathbf{0}$, where $\mathbf{0}$ is a matrix containing all zeros. In the final step meanwhile we use the fact that the matrix of squared distances will not change when we translate all the points. We can thus assume that the mean value, μ, for each of the components, α:

$$
\mu_\alpha = \frac{1}{M}\sum_{i=1}^{N} \mathbf{X}_\alpha^i
$$

is equal to 0 so the columns of $\mathbf{X}$ add up to 0. This in turn means that each of the columns of $\mathbf{X}\mathbf{X}^\mathbf{T}$ adds up to zero, which is what allows us to write $\mathbf{J}\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{J} = \mathbf{X}\mathbf{X}^\mathbf{T}$.

The matrix of squared distances is symmetric and positive-definite we can thus use the spectral decomposition to decompose it as:

$$
\Phi = \mathbf{V}\Lambda\mathbf{V}^T
$$

Furthermore, because the matrix we are diagonalizing, $\mathbf{X}\mathbf{X}^\mathbf{T}$, is the product of a matrix and its transpose we can use this decomposition to write:

$$
\mathbf{X} = \mathbf{V}\Lambda^{\frac{1}{2}}
$$

Much as in PCA there are generally a small number of large eigenvalues in Λ and many small eigenvalues. We can safely use only the large eigenvalues and their corresponding eigenvectors to express the relationship between the coordinates $\mathbf{X}$. This gives us our set of low-dimensional projections.

This derivation makes a number of assumptions about the how the low dimensional points should best be arranged to minimize the stress. If you use an interactive optimization algorithm such as SMACOF you may thus be able to find a better (lower-stress) projection of the points. For more details on the assumptions made see this website.

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

NLOW_DIM	number of low-dimensional coordinates required

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

6.46 OUTPUT_PCA_PROJECTION

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

This is used to output the projection calculated by principle component analysis

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

FILE	the name of the file to output to
STRIDE	(default=0) the frequency with which to perform the required analysis and to output the data. The default value of 0 tells plumed to use all the data

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
FMT	the format to use in the output file

6.47 PCA

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Perform principal component analysis (PCA) using either the positions of the atoms a large number of collective variables as input.

Principal component analysis is a statistical technique that uses an orthogonal transformation to convert a set of observations of poorly correlated variables into a set of linearly uncorrelated variables. You can read more about the specifics of this technique here: https://en.wikipedia.org/wiki/Principal_component_analysis

When used with molecular dynamics simulations a set of frames taken from the trajectory, $\{X_i\}$, or the values of a number of collective variables which are calculated from the trajectory frames are used as input. In this second instance your input to the PCA analysis algorithm is thus a set of high-dimensional vectors of collective variables. However, if collective variables are calculated from the positions of the atoms or if the positions are used directly the assumption is that this input trajectory is a set of poorly correlated (high-dimensional) vectors. After principal component analysis has been performed the output is a set of orthogonal vectors that describe the directions in which the largest motions have been seen. In other words, principal component analysis provides a method for lowering the dimensionality of the data contained in a trajectory. These output directions are some linear combination of the x, y and z positions if the positions were used as input or some linear combination of the input collective variables if a high-dimensional vector of collective variables was used as input.

As explained on the Wikipedia page you must calculate the average and covariance for each of the input coordinates. In other words, you must calculate the average structure and the amount the system fluctuates around this average structure. The problem in doing so when the x, y and z coordinates of a molecule are used as input is that the majority of the changes in the positions of the atoms comes from the translational and rotational degrees of freedom of the molecule. The first six principal components will thus, most likely, be uninteresting. Consequently, to remedy this problem PLUMED provides the functionality to perform an RMSD alignment of the all the structures to be analyzed to the first frame in the trajectory. This can be used to effectively remove translational and/or rotational motions from consideration. The resulting principal components thus describe vibrational motions of the molecule.

If you wish to calculate the projection of a trajectory on a set of principal components calculated from this PCA action then the output can be used as input for the PCAVARS action.

Examples

The following input instructs PLUMED to perform a principal component analysis in which the covariance matrix is calculated from changes in the positions of the first 22 atoms. The TYPE=OPTIMAL instruction ensures that translational and rotational degrees of freedom are removed from consideration. The first two principal components will be output to a file called PCA-comp.pdb. Trajectory frames will be collected on every step and the PCA calculation will be performed at the end of the simulation.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PCA.tmp
ff: COLLECT_FRAMES ATOMS=1-22 STRIDE=1
pca: PCA USE_OUTPUT_DATA_FROM=ff METRIC=OPTIMAL NLOW_DIM=2
OUTPUT_PCA_PROJECTION USE_OUTPUT_DATA_FROM=pca FILE=PCA-comp.pdb
```

The following input instructs PLUMED to perform a principal component analysis in which the covariance matrix is calculated from changes in the six distances seen in the previous lines. Notice that here the TYPE=EUCLID↩ EAN keyword is used to indicate that no alignment has to be done when calculating the various elements of the covariance matrix from the input vectors. In this calculation the first two principal components will be output to a file called PCA-comp.pdb. Trajectory frames will be collected every five steps and the PCA calculation is performed every 1000 steps. Consequently, if you run a 2000 step simulation the PCA analysis will be performed twice. The REWEIGHT_BIAS action in this input tells PLUMED that rather that ascribing a weight of one to each of the frames when calculating averages and covariance matrices a reweighting should be performed based and each frames' weight in these calculations should be determined based on the current value of the instantaneous bias (see REWEIGHT_BIAS).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PCA.tmp
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=1,3
d3: DISTANCE ATOMS=1,4
d4: DISTANCE ATOMS=2,3
d5: DISTANCE ATOMS=2,4
d6: DISTANCE ATOMS=3,4
rr: RESTRAINT ARG=d1 AT=0.1 KAPPA=10
rbias: REWEIGHT_BIAS TEMP=300

ff: COLLECT_FRAMES ARG=d1,d2,d3,d4,d5,d6 LOGWEIGHTS=rbias STRIDE=5
pca: PCA USE_OUTPUT_DATA_FROM=ff METRIC=EUCLIDEAN NLOW_DIM=2
OUTPUT_PCA_PROJECTION USE_OUTPUT_DATA_FROM=pca STRIDE=100 FILE=PCA-comp.pdb
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.

between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Alternatively data can be collected from the trajectory using

ATOMS	the list of atoms that you are going to use in the measure of distance that you are using. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NLOW_DIM	number of low-dimensional coordinates required
METRIC	(default=EUCLIDEAN) the method that you are going to use to measure the distances between points

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

6.48 PROJECT_ALL_ANALYSIS_DATA

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Find projections of all non-landmark points using the embedding calculated by a dimensionality reduction optimization calculation.

Examples

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

PROJECTION	the projection that you wish to generate out-of-sample projections with
CGTOL	(default=1E-6) the tolerance for the conjugate gradient optimization

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

6.49 SKETCHMAP_CONJGRAD

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Optimize the sketch-map stress function using conjugate gradients.

Examples

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry

zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

HIGH_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the high dimensional space
LOW_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the low dimensional space
MIXPARAM	(default=0.0) the amount of the pure distances to mix into the stress function
CGTOL	(default=1E-6) the tolerance for the conjugate gradient minimization

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

6.50 SKETCHMAP_POINTWISE

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Optimize the sketch-map stress function using a pointwise global optimization algorithm.

Examples

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.

lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

HIGH_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the high dimensional space
LOW_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the low dimensional space
MIXPARAM	(default=0.0) the amount of the pure distances to mix into the stress function
NCYCLES	(default=5) the number of cycles of global optimization to attempt
CGTOL	(default=1E-6) the tolerance for the conjugate gradient minimization
BUFFER	(default=1.1) grid extent for search is (max projection - minimum projection) multiplied by this value
CGRID_SIZE	(default=10) number of points to use in each grid direction
FGRID_SIZE	(default=0) interpolate the grid onto this number of points – only works in 2D

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

6.51 SKETCHMAP_READ

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Read in a sketch-map projection from an input file

Examples

Glossary of keywords and components

Description of components

The names of the components in this action can be customized by the user in the actions input file. However, in addition to the components that can be customized the following quantities will always be output

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action

max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

Compulsory keywords

HIGH_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the high dimensional space
LOW_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the low dimensional space
MIXPARAM	(default=0.0) the amount of the pure distances to mix into the stress function
TYPE	(default=OPTIMAL-FAST) the manner in which distances are calculated. More information on the different metrics that are available in PLUMED can be found in the section of the manual on Distances from reference configurations
REFERENCE	the file containing the sketch-map projection
PROPERTY	the property to be used in the index. This should be in the REMARK of the reference

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
DISABLE_CHECKS	(default=off) disable checks on reference input structures.

6.52 SKETCHMAP_SMACOF

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Optimize the sketch-map stress function using the SMACOF algorithm.

Examples

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.

morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

HIGH_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the high dimensional space
LOW_DIM_FUNCTION	(default=as in input action) the parameters of the switching function in the low dimensional space
MIXPARAM	(default=0.0) the amount of the pure distances to mix into the stress function
SMACOF_TOL	(default=1E-4) the tolerance for each SMACOF cycle
SMACOF_MAXCYC	(default=1000) maximum number of optimization cycles for SMACOF algorithm
SMAP_TOL	(default=1E-4) the tolerance for sketch-map
SMAP_MAXCYC	(default=100) maximum number of optimization cycles for iterative sketch-map algorithm
REGULARISE_PARAM	(default=0.001) this is used to ensure that we don't divide by zero when updating weights

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

6.53 SKETCH_MAP

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

This can be used to output the data that has been stored in an Analysis object.

Examples

Glossary of keywords and components

Compulsory keywords

NLOW_DIM	the dimension of the low dimensional space in which the projections will be constructed
MATRIX	the matrix of distances between points that you want to reproduce in your sketch-map projection
HIGH_DIM_FUNCTION	the parameters of the switching function in the high dimensional space
LOW_DIM_FUNCTION	the parameters of the switching function in the low dimensional space
ANNEAL_RATE	(default=0.5) the rate at which to do the annealing
ANNEAL_STEPS	(default=10) the number of steps of annealing to do
CGTOL	(default=1E-6) the tolerance for the conjugate gradient minimization
NCYCLES	(default=5) the number of cycles of global optimization to attempt
BUFFER	(default=1.1) grid extent for search is (max projection - minimum projection) multiplied by this value
CGRID_SIZE	(default=10) number of points to use in each grid direction
FGRID_SIZE	(default=0) interpolate the grid onto this number of points – only works in 2D

6.54 SMACOF_MDS

	This is part of the dimred module
	It is only available if you configure PLUMED with ./configure –enable-modules=dimred . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Optimize the multidimensional scaling stress function using the SMACOF algorithm.

Examples

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
coord	the low-dimensional projections of the various input configurations

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradient	**GRADIENT**	the gradient
vmean	**VMEAN**	the norm of the mean vector. The output component can be referred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be referred to elsewhere in the input file by using the label.vsum
spath	**SPATH**	the position on the path
gspath	**GPATH**	the position on the path calculated using trigonometry
gzpath	**GPATH**	the distance from the path calculated using trigonometry
zpath	**ZPATH**	the distance from the path
altmin	**ALT_MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
between	**BETWEEN**	the number/fraction of values within a certain range. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
highest	**HIGHEST**	the highest of the quantities calculated by this action
lessthan	**LESS_THAN**	the number of values less than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
lowest	**LOWEST**	the lowest of the quantities calculated by this action
max	**MAX**	the maximum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
mean	**MEAN**	the mean value. The output component can be referred to elsewhere in the input file by using the label.mean
min	**MIN**	the minimum value. This is calculated using the formula described in the description of the keyword so as to make it continuous.
moment	**MOMENTS**	the central moments of the distribution of values. The second moment would be referenced elsewhere in the input file using *label*.moment-2, the third as *label*.moment-3, etc.
morethan	**MORE_THAN**	the number of values more than a target value. This is calculated using one of the formula described in the description of the keyword so as to make it continuous. You can calculate this quantity multiple times using different parameters.
sum	**SUM**	the sum of values

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

SMACOF_TOL	(default=1E-4) tolerance for the SMACOF optimization algorithm
SMACOF_MAXCYC	(default=1000) maximum number of optimization cycles for SMACOF algorithm

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements

6.55 OUTPUT_ANALYSIS_DATA_TO_COLVAR

	This is part of the analysis module

This can be used to output the data that has been stored in an Analysis object.

The most useful application of this method is to output all projections of all the points that were stored in an analysis object that performs some form of dimensionality reduction. If you use the USE_DIMRED_DATA_FROM option below projections of all the stored points will be output to a file. The positions of these projections will be calculated using that dimensionality reduction algorithms out-of-sample extension algorithm.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

FILE	the name of the file to output to
REPLICA	(default=0) the replicas for which you would like to output this information
STRIDE	(default=0) the frequency with which to perform the required analysis and to output the data. The default value of 0 tells plumed to use all the data

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
FMT	the format to output the data using

6.56 OUTPUT_ANALYSIS_DATA_TO_PDB

This is part of the analysis module

This can be used to output the data that has been stored in an Analysis object.

Examples

Glossary of keywords and components

The data to analyze can be the output from another analysis algorithm

USE_OUTPUT_DATA_FROM	use the output of the analysis performed by this object as input to your new analysis object

Compulsory keywords

FILE	the name of the file to output to
STRIDE	(default=0) the frequency with which to perform the required analysis and to output the data. The default value of 0 tells plumed to use all the data

Options

SERIAL	(default=off) do the calculation in serial. Do not use MPI
LOWMEM	(default=off) lower the memory requirements
FMT	the format to use in the output file

Chapter 7

Bias

PLUMED allows you to run a number of enhanced sampling algorithms. The list of enhanced sampling algorithms contained in PLUMED is as follows:

ABMD	Adds a ratchet-and-pawl like restraint on one or more variables.
BIASVALUE	Takes the value of one variable and use it as a bias
EXTENDED_LAGRANGIAN	Add extended Lagrangian.
EXTERNAL	Calculate a restraint that is defined on a grid that is read during start up
LOWER_WALLS	Defines a wall for the value of one or more collective variables, which limits the region of the phase space accessible during the simulation.
MAXENT	Add a linear biasing potential on one or more variables that satisfies a maximum entropy principle.
METAD	Used to performed metadynamics on one or more collective variables.
MOVINGRESTRAINT	Add a time-dependent, harmonic restraint on one or more variables.
PBMETAD	Used to performed Parallel Bias metadynamics.
RESTRAINT	Adds harmonic and/or linear restraints on one or more variables.
UPPER_WALLS	Defines a wall for the value of one or more collective variables, which limits the region of the phase space accessible during the simulation.

In addition to the keywords above, by enabling optional modules you can access to the following keywords:

CALIBER	(from PLUMED-ISDB module) Add a time-dependent, harmonic restraint on one or more variables.
DRR	(from Extended-System Adaptive Biasing Force module) Used to performed extended-system adaptive biasing force(eABF)
EDS	(from Experiment Directed Simulation module) Add a linear bias on a set of observables.
FISST	(from FISST (Infinite Switch Simulated Tempering in Force) module) Compute and apply the optimal linear force on an observable to enhance sampling of conformational distributions over a range of applied forces.
FUNNEL	(from Funnel-Metadynamics (FM) module) Calculate a funnel-shape restraint potential that is defined on a grid that is read during the setup.
LOGMFD	(from Logarithmic Mean Force Dynamics module) Used to perform LogMFD, LogPD, and TAMD/d-AFED.
MAZE_OPTIMIZER_BIAS	(from MAZE module)
METAINFERENCE	(from PLUMED-ISDB module) Calculates the Metainference energy for a set of experimental data.

OPES_EXPANDED	(from OPES (On-the-fly Probability Enhanced Sampling) module) On-the-fly probability enhanced sampling with expanded ensembles for the target distribution.
OPES_METAD	(from OPES (On-the-fly Probability Enhanced Sampling) module) On-the-fly probability enhanced sampling with metadynamics-like target distribution.
OPES_METAD_EXPLORE	(from OPES (On-the-fly Probability Enhanced Sampling) module) On-the-fly probability enhanced sampling with well-tempered target distribution in exploreation mode.
RESCALE	(from PLUMED-ISDB module) Scales the value of an another action, being a Collective Variable or a Bias.
VES_DELTA_F	(from Variationally Enhanced Sampling (VES code) module) Implementation of VES Delta F method
VES_LINEAR_EXPANSION	(from Variationally Enhanced Sampling (VES code) module) Linear basis set expansion bias.

Methods, such as METAD or PBMETAD, that work by introducing a history dependent bias can be restarted using the RESTART keyword

7.1 ABMD

This is part of the bias module

Adds a ratchet-and-pawl like restraint on one or more variables.

This action can be used to evolve a system towards a target value in CV space using an harmonic potential moving with the thermal fluctuations of the CV [40] [41] [42]. The biasing potential in this method is as follows:

$$V(\rho(t)) = \begin{cases} \frac{K}{2}\left(\rho(t) - \rho_m(t)\right)^2, & \rho(t) > \rho_m(t) \\ 0, & \rho(t) \leq \rho_m(t), \end{cases}$$

where

$$\rho(t) = \left(CV(t) - TO\right)^2$$

and

$$\rho_m(t) = \min_{0 \leq \tau \leq t} \rho(\tau) + \eta(t).$$

The method is based on the introduction of a biasing potential which is zero when the system is moving towards the desired arrival point and which damps the fluctuations when the system attempts to move in the opposite direction. As in the case of the ratchet and pawl system, propelled by thermal motion of the solvent molecules, the biasing potential does not exert work on the system. $\eta(t)$ is an additional white noise acting on the minimum position of the bias.

Examples

The following input sets up two biases, one on the distance between atoms 3 and 5 and another on the distance between atoms 2 and 4. The two target values are defined using TO and the two strength using KAPPA. The total energy of the bias is printed.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ABMD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
ABMD ARG=d1,d2 TO=1.0,1.5 KAPPA=5.0,5.0 LABEL=abmd
PRINT ARG=abmd.bias,abmd.d1_min,abmd.d2_min
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	the instantaneous value of the squared force due to this bias potential
_min	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will be named with the arguments of the bias followed by the character string _min. These quantities tell the user the minimum value assumed by rho_m(t).

Compulsory keywords

TO	The array of target values
KAPPA	The array of force constants.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
MIN	Array of starting values for the bias (set rho_m(t), otherwise it is set using the current value of ARG)
NOISE	Array of white noise intensities (add a temperature to the ABMD)
SEED	Array of seeds for the white noise (add a temperature to the ABMD)

7.2 BIASVALUE

This is part of the bias module

Takes the value of one variable and use it as a bias

This is the simplest possible bias: the bias potential is equal to a collective variable. It is useful to create custom biasing potential, e.g. applying a function (see Functions) to some collective variable then using the value of this function directly as a bias.

Examples

The following input tells plumed to use the value of the distance between atoms 3 and 5 and the value of the distance between atoms 2 and 4 as biases. It then tells plumed to print the energy of the restraint

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BIASVALUE.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=3,6 LABEL=d2
BIASVALUE ARG=d1,d2 LABEL=b
PRINT ARG=d1,d2,b.d1_bias,b.d2_bias
```

Another thing one can do is asking one system to follow a circle in sin/cos according a time dependence

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BIASVALUE.tmp
t: TIME
# this just print cos and sin of time
cos: MATHEVAL ARG=t VAR=t FUNC=cos(t) PERIODIC=NO
sin: MATHEVAL ARG=t VAR=t FUNC=sin(t) PERIODIC=NO
c1: COM ATOMS=1,2
c2: COM ATOMS=3,4
d: DISTANCE COMPONENTS ATOMS=c1,c2
PRINT ARG=t,cos,sin,d.x,d.y,d.z STRIDE=1 FILE=colvar FMT=%8.4f
# this calculates sine and cosine of a projected component of distance
mycos:  MATHEVAL ARG=d.x,d.y  VAR=x,y   FUNC=x/sqrt(x*x+y*y) PERIODIC=NO
mysin:  MATHEVAL ARG=d.x,d.y  VAR=x,y   FUNC=y/sqrt(x*x+y*y) PERIODIC=NO
# this creates a moving spring so that the system follows a circle-like dynamics
# but it is not a bias, it is a simple value now
vv1:  MATHEVAL ARG=mycos,mysin,cos,sin VAR=mc,ms,c,s  FUNC=100*((mc-c)^2+(ms-s)^2) PERIODIC=NO
# this takes the value calculated with matheval and uses as a bias
cc: BIASVALUE ARG=vv1
# some printout
PRINT ARG=t,cos,sin,d.x,d.y,d.z,mycos,mysin,cc.vv1_bias STRIDE=1 FILE=colvar FMT=%8.4f
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
_bias	one or multiple instances of this quantity can be referenced elsewhere in the input file. these quantities will named with the arguments of the bias followed by the character string _bias. These quantities tell the user how much the bias is due to each of the colvars.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

7.3 EXTENDED_LAGRANGIAN

Add extended Lagrangian.

This action can be used to create fictitious collective variables coupled to the real ones. Given x_i the ith argument of this bias potential, potential and kinetic contributions are added to the energy of the system as

$$V = \sum_i \frac{k_i}{2}(x_i - s_i)^2 + \sum_i \frac{\dot{s}_i^2}{2m_i}$$

.

The resulting potential is thus similar to a RESTRAINT, but the restraint center moved with time following Hamiltonian dynamics with mass m_i.

This bias potential accepts thus vectorial keywords (one element per argument) to define the coupling constant (KAPPA) and a relaxation time tau (TAU). The mass is them computed as $m = k(\frac{\tau}{2\pi})^2$.

Notice that this action creates several components. The ones named XX_fict are the fictitious coordinates. It is possible to add further forces on them by means of other bias potential, e.g. to obtain an indirect METAD as in [43] . Also notice that the velocities of the fictitious coordinates are reported (XX_vfict). However, printed velocities are the ones at the previous step.

It is also possible to provide a non-zero friction (one value per component). This is then used to implement a Langevin thermostat, so as to implement TAMD/dAFED method [44] [45] . Notice that here a massive Langevin thermostat is used, whereas usually TAMD employs an overamped Langevin dynamics and dAFED a Gaussian thermostat.

Warning

The bias potential is reported in the component bias. Notice that this bias potential, although formally compatible with replica exchange framework, probably does not work as expected in that case. Indeed, since fictitious coordinates are not swapped upon exchange, acceptace can be expected to be extremely low unless (by chance) two neighboring replicas have the fictitious variables located properly in space.

RESTART is not properly supported by this action. Indeed, at every start the position of the fictitious variable is reset to the value of the real variable, and its velocity is set to zero. This is not expected to introduce big errors, but certainly is introducing a small inconsistency between a single long run and many shorter runs.

Examples

The following input tells plumed to perform a metadynamics with an extended Lagrangian on two torsional angles.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EXTENDED_LAGRANGIAN.tmp
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
ex: EXTENDED_LAGRANGIAN ARG=phi,psi KAPPA=20,20.0 TAU=0.1,0.1
METAD ARG=ex.phi_fict,ex.psi_fict PACE=100 SIGMA=0.35,0.35 HEIGHT=0.1
# monitor the two variables
PRINT STRIDE=10 ARG=phi,psi,ex.phi_fict,ex.psi_fict FILE=COLVAR
```

The following input tells plumed to perform a TAMD (or dAFED) calculation on two torsional angles, keeping the two variables at a fictitious temperature of 3000K with a Langevin thermostat with friction 10

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EXTENDED_LAGRANGIAN.tmp
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
ex: EXTENDED_LAGRANGIAN ARG=phi,psi KAPPA=20,20.0 TAU=0.1,0.1 FRICTION=10,10 TEMP=3000
# monitor the two variables
PRINT STRIDE=10 ARG=phi,psi,ex.phi_fict,ex.psi_fict FILE=COLVAR
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
_fict	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will named with the arguments of the bias followed by the character string _tilde. It is possible to add forces on these variable.
_vfict	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will named with the arguments of the bias followed by the character string _tilde. It is NOT possible to add forces on these variable.

Compulsory keywords

KAPPA	specifies that the restraint is harmonic and what the values of the force constants on each of the variables are
TAU	specifies that the restraint is harmonic and what the values of the force constants on each of the variables are
FRICTION	(default=0.0) add a friction to the variable

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
TEMP	the system temperature - needed when FRICTION is present. If not provided will be taken from MD code (if available)

7.4 EXTERNAL

This is part of the bias module

Calculate a restraint that is defined on a grid that is read during start up

Examples

The following is an input for a calculation with an external potential that is defined in the file bias.dat and that acts on the distance between atoms 3 and 5.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EXTERNAL.tmp
DISTANCE ATOMS=3,5 LABEL=d1
EXTERNAL ARG=d1 FILE=bias.grid LABEL=external
```

The bias.grid will then look something like this:

```
#! FIELDS d1 external.bias der_d1
#! SET min_d1 1.14
#! SET max_d1 1.32
#! SET nbins_d1 6
#! SET periodic_d1 false
   1.1400   0.0031   0.1101
   1.1700   0.0086   0.2842
   1.2000   0.0222   0.6648
   1.2300   0.0521   1.4068
   1.2600   0.1120   2.6873
   1.2900   0.2199   4.6183
   1.3200   0.3948   7.1055
```

This should then be followed by the value of the potential and its derivative at 100 equally spaced points along the distance between 0 and 1.

You can also include grids that are a function of more than one collective variable. For instance the following would be the input for an external potential acting on two torsional angles:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EXTERNAL.tmp
TORSION ATOMS=4,5,6,7 LABEL=t1
TORSION ATOMS=6,7,8,9 LABEL=t2
EXTERNAL ARG=t1,t2 FILE=bias2.grid LABEL=ext
```

The file bias2.grid for this calculation would need to look something like this:

```
#! FIELDS t1 t2 ext.bias der_t1 der_t2
#! SET min_t1 -pi
#! SET max_t1 pi
#! SET nbins_t1 3
#! SET periodic_t1 true
#! SET min_t2 -pi
#! SET max_t2 pi
#! SET nbins_t2 3
#! SET periodic_t2 true
 -3.141593 -3.141593 0.000000 -0.000000 -0.000000
 -1.047198 -3.141593 0.000000 0.000000 -0.000000
 1.047198 -3.141593 0.000000 -0.000000 -0.000000

 -3.141593 -1.047198 0.000000 -0.000000 0.000000
 -1.047198 -1.047198 0.007922 0.033185 0.033185
 1.047198 -1.047198 0.007922 -0.033185 0.033185

 -3.141593 1.047198 0.000000 -0.000000 -0.000000
 -1.047198 1.047198 0.007922 0.033185 -0.033185
 1.047198 1.047198 0.007922 -0.033185 -0.033185
```

This would be then followed by 100 blocks of data. In the first block of data the value of t1 (the value in the first column) is kept fixed and the value of the function is given at 100 equally spaced values for t2 between $-pi$ and $+pi$. In the second block of data t1 is fixed at $-pi + \frac{2pi}{100}$ and the value of the function is given at 100 equally spaced values for t2 between $-pi$ and $+pi$. In the third block of data the same is done but t1 is fixed at $-pi + \frac{4pi}{100}$ and so on until you get to the one hundredth block of data where t1 is fixed at $+pi$.

Please note the order that the order of arguments in the plumed.dat file must be the same as the order of arguments in the header of the grid file.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential

Compulsory keywords

FILE	the name of the file containing the external potential.
SCALE	(default=1.0) a factor that multiplies the external potential, useful to invert free energies

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOSPLINE	(default=off) specifies that no spline interpolation is to be used when calculating the energy and forces due to the external potential
SPARSE	(default=off) specifies that the external potential uses a sparse grid
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

7.5 LOWER_WALLS

	This is part of the bias module

Defines a wall for the value of one or more collective variables, which limits the region of the phase space accessible during the simulation.

The restraining potential starts acting on the system when the value of the CV is greater (in the case of UPPER↩ _WALLS) or lower (in the case of LOWER_WALLS) than a certain limit a_i (AT) minus an offset o_i (OFFSET). The expression for the bias due to the wall is given by:

$\sum_i k_i((x_i - a_i + o_i)/s_i)^e_i$

k_i (KAPPA) is an energy constant in internal unit of the code, s_i (EPS) a rescaling factor and e_i (EXP) the exponent determining the power law. By default: EXP = 2, EPS = 1.0, OFFSET = 0.

Examples

The following input tells plumed to add both a lower and an upper walls on the distance between atoms 3 and 5 and the distance between atoms 2 and 4. The lower and upper limits are defined at different values. The strength of the walls is the same for the four cases. It also tells plumed to print the energy of the walls.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOWER_WALLS.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
UPPER_WALLS ARG=d1,d2 AT=1.0,1.5 KAPPA=150.0,150.0 EXP=2,2 EPS=1,1 OFFSET=0,0 LABEL=uwall
LOWER_WALLS ARG=d1,d2 AT=0.0,1.0 KAPPA=150.0,150.0 EXP=2,2 EPS=1,1 OFFSET=0,0 LABEL=lwall
PRINT ARG=uwall.bias,lwall.bias
```

(See also DISTANCE and PRINT).

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	the instantaneous value of the squared force due to this bias potential

Compulsory keywords

AT	the positions of the wall. The a_i in the expression for a wall.
KAPPA	the force constant for the wall. The k_i in the expression for a wall.
OFFSET	(default=0.0) the offset for the start of the wall. The o_i in the expression for a wall.
EXP	(default=2.0) the powers for the walls. The e_i in the expression for a wall.
EPS	(default=1.0) the values for s_i in the expression for a wall

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

7.6 MAXENT

This is part of the bias module

Add a linear biasing potential on one or more variables that satisfies a maximum entropy principle.

Add a linear biasing potential on one or more variables $f_i\left(x\right)$ satisfying the maximum entropy principle as proposed in Ref. [46] .

Warning

Notice that syntax is still under revision and might change

The resulting biasing potential is given by:

$$V_{BIAS}(x,t) = K_B T \sum_{i=1}^{\#arguments} f_i(x,t)\lambda_i(t)$$

Lagrangian multipliers λ_i are updated, every PACE steps, according to the following update rule:

$$\lambda_i = \lambda_i + \frac{k_i}{1+\frac{t}{\tau_i}}\left(f_{exp,i} + \xi_i\lambda_i - f_i(x)\right)$$

k set the initial value of the learning rate and its units are $[observable]^{-2}ps^{-1}$. This can be set with the keyword KAPPA. The number of components for any KAPPA vector must be equal to the number of arguments of the action.

Variable $\xi_i(\lambda)$ is related to the chosen prior to model experimental errors. If a GAUSSIAN prior is used then:

$$\xi_i(\lambda) = -\lambda_i\sigma^2$$

where σ is the typical expected error on the observable f_i. For a LAPLACE prior:

$$\xi_i(\lambda) = -\frac{\lambda_i\sigma^2}{1-\frac{\lambda^2\sigma^2}{2}}$$

The value of $\xi(\lambda,t)$ is written in output as a component named: argument name followed by the string _error. Setting $\sigma = 0$ is equivalent to enforce a pure Maximum Entropy restraint without any noise modelling. This method can be also used to enforce inequality restraint as shown in following examples.

Notice that a similar method is available as EDS, although with different features and using a different optimization algorithm.

Examples

The following input tells plumed to restrain the distance between atoms 7 and 15 and the distance between atoms 2 and 19, at different equilibrium values, and to print the energy of the restraint. Lagrangian multiplier will be printed on a file called restraint.LAGMULT with a stride set by the variable PACE to 200ps. Moreover plumed will compute the average of each Lagrangian multiplier in the window [TSTART,TEND] and use that to continue the simulations with fixed Lagrangian multipliers.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MAXENT.tmp
DISTANCE ATOMS=7,15 LABEL=d1
DISTANCE ATOMS=2,19 LABEL=d2
MAXENT ...
ARG=d1,d2
TYPE=EQUAL
AT=0.2,0.5
KAPPA=35000.0,35000.0
TAU=0.02,0.02
PACE=200
TSTART=100
TEND=500
LABEL=restraint
... MAXENT
PRINT ARG=restraint.bias
```

Lagrangian multipliers will be printed on a file called restraint.bias The following input tells plumed to restrain the distance between atoms 7 and 15 to be greater than 0.2 and to print the energy of the restraint

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MAXENT.tmp
DISTANCE ATOMS=7,15 LABEL=d
MAXENT ARG=d TYPE=INEQUAL> AT=0.02 KAPPA=35000.0 TAU=3 LABEL=restraint
PRINT ARG=restraint.bias
```

(See also DISTANCE and PRINT).

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	the instantaneous value of the squared force due to this bias potential
work	the instantaneous value of the work done by the biasing force
_work	the instantaneous value of the work done by the biasing force for each argument. These quantities will named with the arguments of the bias followed by the character string _work.
_error	Instantaneous values of the discrepancy between the observable and the restraint center
_coupling	Instantaneous values of Lagrangian multipliers. They are also written by default in a separate output file.

Compulsory keywords

KAPPA	(default=0.0) specifies the initial value for the learning rate
TAU	Specify the dumping time for the learning rate.
TYPE	specify the restraint type. EQUAL to restrain the variable at a given equilibrium value INEQUAL< to restrain the variable to be smaller than a given value INEQUAL> to restrain the variable to be greater than a given value
AT	the position of the restraint

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
REWEIGHT	(default=off) to be used with plumed driver in order to reweight a trajectory a posteriori
NO_BROADCAST	(default=off) If active will avoid Lagrangian multipliers to be communicated to other replicas.
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
ERROR_TYPE	specify the prior on the error to use.GAUSSIAN: use a Gaussian prior LAPL↩ ACE: use a Laplace prior
TSTART	time from where to start averaging the Lagrangian multiplier. By default no average is computed, hence lambda is updated every PACE steps
TEND	time in ps where to stop to compute the average of Lagrangian multiplier. From this time until the end of the simulation Lagrangian multipliers are kept fix to the average computed between TSTART and TEND;
ALPHA	default=1.0; To be used with LAPLACE KEYWORD, allows to choose a prior function proportional to a Gaussian times an exponential function. ALPHA=1 correspond to the LAPLACE prior.
SIGMA	The typical errors expected on observable
FILE	Lagrangian multipliers output file. The default name is: label name followed by the string .LAGMULT
LEARN_REPLICA	In a multiple replica environment specify which is the reference replica. By default replica 0 will be used.
APPLY_WEIGHTS	Vector of weights containing 1 in correspondence of each replica that will receive the Lagrangian multiplier from the current one.
PACE	the frequency for Lagrangian multipliers update

PRINT_STRIDE	stride of Lagrangian multipliers output file. If no STRIDE is passed they are written every time they are updated (PACE).
FMT	specify format for Lagrangian multipliers files (useful to decrease the number of digits in regtests)
TEMP	the system temperature. This is required if you are reweighting.
RESTART	allows per-action setting of restart (YES/NO/AUTO)

7.7 METAD

This is part of the bias module

Used to performed metadynamics on one or more collective variables.

In a metadynamics simulations a history dependent bias composed of intermittently added Gaussian functions is added to the potential [47].

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i^{(0)}(k\tau))^2}{2\sigma_i^2}\right).$$

This potential forces the system away from the kinetic traps in the potential energy surface and out into the unexplored parts of the energy landscape. Information on the Gaussian functions from which this potential is composed is output to a file called HILLS, which is used both the restart the calculation and to reconstruct the free energy as a function of the CVs. The free energy can be reconstructed from a metadynamics calculation because the final bias is given by:

$$V(\vec{s}) = -F(\vec{s})$$

During post processing the free energy can be calculated in this way using the sum_hills utility.

In the simplest possible implementation of a metadynamics calculation the expense of a metadynamics calculation increases with the length of the simulation as one has to, at every step, evaluate the values of a larger and larger number of Gaussian kernels. To avoid this issue you can store the bias on a grid. This approach is similar to that proposed in [48] but has the advantage that the grid spacing is independent on the Gaussian width. Notice that you should provide the grid boundaries (GRID_MIN and GRID_MAX) and either the number of bins for every collective variable (GRID_BIN) or the desired grid spacing (GRID_SPACING). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither GRID_BIN nor GRID_SPACING) PLUMED will use 1/5 of the Gaussian width (SIGMA) as grid spacing if the width is fixed or 1/5 of the minimum Gaussian width (SIGMA_MIN) if the width is variable. This default choice should be reasonable for most applications.

Alternatively to the use of grids, it is possible to use a neighbor list to decrease the cost of evaluating the bias, this can be enabled using NLIST. NLIST can be beneficial with more than 2 collective variables, where GRID becomes expensive and memory consuming. The neighbor list will be updated everytime the CVs go farther than a cut-off value from the position they were at last neighbor list update. Gaussians are added to the neigbhor list if their center is within 6.∗DP2CUTOFF∗sigma∗sigma. While the list is updated if the CVs are farther from the center than 0.5 of the standard deviation of the Gaussian center distribution of the list. These parameters (6 and 0.5) can be modified using NLIST_PARAMETERS. Note that the use of neighbor list does not provide the exact bias.

Metadynamics can be restarted either from a HILLS file as well as from a GRID, in this second case one can first save a GRID using GRID_WFILE (and GRID_WSTRIDE) and at a later stage read it using GRID_RFILE.

The work performed by the METAD bias can be calculated using CALC_WORK, note that this is expensive when not using grids.

Another option that is available in plumed is well-tempered metadynamics [49]. In this variant of metadynamics the heights of the Gaussian hills are scaled at each step so the bias is now given by:

$$V(s,t) = \sum_{t'=0,\tau_G,2\tau_G,\ldots}^{t'<t} We^{-V(s(q(t')),t')/\Delta T} \exp\left(-\sum_{i=1}^{d} \frac{(s_i(q) - s_i(q(t'))^2}{2\sigma_i^2}\right),$$

This method ensures that the bias converges more smoothly. It should be noted that, in the case of well-tempered metadynamics, in the output printed the Gaussian height is re-scaled using the bias factor. Also notice that with well-tempered metadynamics the HILLS file does not contain the bias, but the negative of the free-energy estimate. This choice has the advantage that one can restart a simulation using a different value for the ΔT. The applied bias will be scaled accordingly.

Note that you can use here also the flexible Gaussian approach [50] in which you can adapt the Gaussian to the extent of Cartesian space covered by a variable or to the space in collective variable covered in a given time. In this case the width of the deposited Gaussian potential is denoted by one value only that is a Cartesian space (ADA↩ PTIVE=GEOM) or a time (ADAPTIVE=DIFF). Note that a specific integration technique for the deposited Gaussian kernels should be used in this case. Check the documentation for utility sum_hills.

With the keyword INTERVAL one changes the metadynamics algorithm setting the bias force equal to zero outside boundary [51]. If, for example, metadynamics is performed on a CV s and one is interested only to the free energy for s $>$ boundary, the history dependent potential is still updated according to the above equations but the metadynamics force is set to zero for s $<$ boundary. Notice that Gaussian kernels are added also if s $<$ boundary, as the tails of these Gaussian kernels influence VG in the relevant region s $>$ boundary. In this way, the force on the system in the region s $>$ boundary comes from both metadynamics and the force field, in the region s $<$ boundary only from the latter. This approach allows obtaining a history-dependent bias potential VG that fluctuates around a stable estimator, equal to the negative of the free energy far enough from the boundaries. Note that:

- It works only for one-dimensional biases;
- It works both with and without GRID;
- The interval limit boundary in a region where the free energy derivative is not large;
- If in the region outside the limit boundary the system has a free energy minimum, the INTERVAL keyword should be used together with a UPPER_WALLS or LOWER_WALLS at boundary.

As a final note, since version 2.0.2 when the system is outside of the selected interval the force is set to zero and the bias value to the value at the corresponding boundary. This allows acceptances for replica exchange methods to be computed correctly.

Multiple walkers [52] can also be used. See below the examples.

The $c(t)$ reweighting factor can also be calculated on the fly using the equations presented in [3]. The expression used to calculate $c(t)$ follows directly from Eq. 3 in [3], where $F(\vec{s}) = -\gamma/(\gamma-1)V(\vec{s})$. This gives smoother results than equivalent Eqs. 13 and Eqs. 14 in that paper. The $c(t)$ is given by the rct component while the bias normalized by $c(t)$ is given by the rbias component (rbias=bias-rct) which can be used to obtain a reweighted histogram. The calculation of $c(t)$ is enabled by using the keyword CALC_RCT. By default $c(t)$ is updated every time the bias changes, but if this slows down the simulation the keyword RCT_USTRIDE can be set to a value higher than 1. This option requires that a grid is used.

Additional material and examples can be also found in the tutorials:

- Lugano tutorial: Metadynamics simulations with PLUMED

Concurrent metadynamics as done e.g. in Ref. [53] . This indeed can be obtained by using the METAD action multiple times in the same input file.

Examples

The following input is for a standard metadynamics calculation using as collective variables the distance between atoms 3 and 5 and the distance between atoms 2 and 4. The value of the CVs and the metadynamics bias potential are written to the COLVAR file every 100 steps.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
METAD ARG=d1,d2 SIGMA=0.2,0.2 HEIGHT=0.3 PACE=500 LABEL=restraint
PRINT ARG=d1,d2,restraint.bias STRIDE=100  FILE=COLVAR
```

(See also DISTANCE PRINT).

If you use adaptive Gaussian kernels, with diffusion scheme where you use a Gaussian that should cover the space of 20 time steps in collective variables. Note that in this case the histogram correction is needed when summing up hills.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
METAD ARG=d1,d2 SIGMA=20 HEIGHT=0.3 PACE=500 LABEL=restraint ADAPTIVE=DIFF
PRINT ARG=d1,d2,restraint.bias STRIDE=100  FILE=COLVAR
```

If you use adaptive Gaussian kernels, with geometrical scheme where you use a Gaussian that should cover the space of 0.05 nm in Cartesian space. Note that in this case the histogram correction is needed when summing up hills.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
METAD ARG=d1,d2 SIGMA=0.05 HEIGHT=0.3 PACE=500 LABEL=restraint ADAPTIVE=GEOM
PRINT ARG=d1,d2,restraint.bias STRIDE=100  FILE=COLVAR
```

When using adaptive Gaussian kernels you might want to limit how the hills width can change. You can use SIGMA_MIN and SIGMA_MAX keywords. The sigmas should specified in terms of CV so you should use the CV units. Note that if you use a negative number, this means that the limit is not set. Note also that in this case the histogram correction is needed when summing up hills.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
METAD ...
  ARG=d1,d2 SIGMA=0.05 HEIGHT=0.3 PACE=500 LABEL=restraint ADAPTIVE=GEOM
  SIGMA_MIN=0.2,0.1 SIGMA_MAX=0.5,1.0
... METAD
PRINT ARG=d1,d2,restraint.bias STRIDE=100  FILE=COLVAR
```

Multiple walkers can be also use as in [52] These are enabled by setting the number of walker used, the id of the current walker which interprets the input file, the directory where the hills containing files resides, and the frequency to read the other walkers. Here is an example

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
METAD ...
   ARG=d1 SIGMA=0.05 HEIGHT=0.3 PACE=500 LABEL=restraint
   WALKERS_N=10
   WALKERS_ID=3
   WALKERS_DIR=../
   WALKERS_RSTRIDE=100
... METAD
```

where WALKERS_N is the total number of walkers, WALKERS_ID is the id of the present walker (starting from 0) and the WALKERS_DIR is the directory where all the walkers are located. WALKERS_RSTRIDE is the number of step between one update and the other. Since version 2.2.5, hills files are automatically flushed every WALKERS_RSTRIDE steps.

The $c(t)$ reweighting factor can be calculated on the fly using the equations presented in [3] as described above. This is enabled by using the keyword CALC_RCT, and can be done only if the bias is defined on a grid.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
phi: TORSION ATOMS=1,2,3,4
psi: TORSION ATOMS=5,6,7,8

METAD ...
 LABEL=metad
 ARG=phi,psi SIGMA=0.20,0.20 HEIGHT=1.20 BIASFACTOR=5 TEMP=300.0 PACE=500
 GRID_MIN=-pi,-pi GRID_MAX=pi,pi GRID_BIN=150,150
 CALC_RCT
 RCT_USTRIDE=10
... METAD
```

Here we have asked that the calculation is performed every 10 hills deposition by using RCT_USTRIDE keyword. If this keyword is not given, the calculation will by default be performed every time the bias changes. The $c(t)$ reweighting factor will be given in the rct component while the instantaneous value of the bias potential normalized using the $c(t)$ reweighting factor is given in the rbias component [rbias=bias-rct] which can be used to obtain a reweighted histogram or free energy surface using the HISTOGRAM analysis.

The kinetics of the transitions between basins can also be analyzed on the fly as in [54]. The flag ACC↩ ELERATION turn on accumulation of the acceleration factor that can then be used to determine the rate. This method can be used together with COMMITTOR analysis to stop the simulation when the system get to the target basin. It must be used together with Well-Tempered Metadynamics. If restarting from a previous metadynamics you need to use the ACCELERATION_RFILE keyword to give the name of the data file from which the previous value of the acceleration factor should be read, otherwise the calculation of the acceleration factor will be wrong.

By using the flag FREQUENCY_ADAPTIVE the frequency adaptive scheme introduced in [55] is turned on. The frequency for hill addition then changes dynamically based on the acceleration factor according to the following equation

$$\tau_{\mathrm{dep}}(t) = \min\left[\tau_0 \cdot \max\left[\frac{\alpha(t)}{\theta}, 1\right], \tau_c\right]$$

where τ_0 is the initial hill addition frequency given by the PACE keyword, τ_c is the maximum allowed frequency given by the FA_MAX_PACE keyword, $\alpha(t)$ is the instantaneous acceleration factor at time t, and θ is a threshold value that acceleration factor has to reach before triggering a change in the hill addition frequency

given by the FA_MIN_ACCELERATION keyword. The frequency for updating the hill addition frequency according to this equation is given by the FA_UPDATE_FREQUENCY keyword, by default it is the same as the value given in PACE. The hill hill addition frequency increase monotonously such that if the instantaneous acceleration factor is lower than in the previous updating step the previous τ_{dep} is kept rather than updating it to a lower value. The instantaneous hill addition frequency $\tau_{\mathrm{dep}}(t)$ is outputted to pace component. Note that if restarting from a previous metadynamics run you need to use the ACCELERATION_RFILE keyword to read in the acceleration factors from the previous run, otherwise the hill addition frequency will start from the initial frequency.

You can also provide a target distribution using the keyword TARGET [56] [57] [58] The TARGET should be a grid containing a free-energy (i.e. the - k_BT$*$log of the desired target distribution). Gaussian kernels will then be scaled by a factor

$$e^{\beta(\tilde{F}(s)-\tilde{F}_{max})}$$

Here $\tilde{F}(s)$ is the free energy defined on the grid and $\tilde{F}_{max}$ its maximum value. Notice that we here used the maximum value as in ref [58] This choice allows to avoid exceedingly large Gaussian kernels to be added. However, it could make the Gaussian too small. You should always choose carefully the HEIGHT parameter in this case. The grid file should be similar to other PLUMED grid files in that it should contain both the target free-energy and its derivatives.

Notice that if you wish your simulation to converge to the target free energy you should use the DAMPFACTOR command to provide a global tempering [59] Alternatively, if you use a BIASFACTOR your simulation will converge to a free energy that is a linear combination of the target free energy and of the intrinsic free energy determined by the original force field.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
METAD ...
 LABEL=t1
 ARG=d1 SIGMA=0.05 TAU=200 DAMPFACTOR=100 PACE=250
 GRID_MIN=1.14 GRID_MAX=1.32 GRID_BIN=6
 TARGET=dist.grid
... METAD

PRINT ARG=d1,t1.bias STRIDE=100 FILE=COLVAR
```

The file dist.dat for this calculation would read:

```
#! FIELDS d1 t1.target der_d1
#! SET min_d1 1.14
#! SET max_d1 1.32
#! SET nbins_d1  6
#! SET periodic_d1 false
   1.1400   0.0031   0.1101
   1.1700   0.0086   0.2842
   1.2000   0.0222   0.6648
   1.2300   0.0521   1.4068
   1.2600   0.1120   2.6873
   1.2900   0.2199   4.6183
   1.3200   0.3948   7.1055
```

Notice that BIASFACTOR can also be chosen as equal to 1. In this case one will perform unbiased sampling. Instead of using HEIGHT, one should provide the TAU parameter.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
d: DISTANCE ATOMS=3,5
METAD ARG=d SIGMA=0.1 TAU=4.0 TEMP=300 PACE=100 BIASFACTOR=1.0
```

The HILLS file obtained will still work with `plumed sum_hills` so as to plot a free-energy. The case where this makes sense is probably that of RECT simulations.

Regarding RECT simulations, you can also use the RECT keyword so as to avoid using multiple input files. For instance, a single input file will be

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAD.tmp
d: DISTANCE ATOMS=3,5
METAD ARG=d SIGMA=0.1 TAU=4.0 TEMP=300 PACE=100 RECT=1.0,1.5,2.0,3.0
```

The number of elements in the RECT array should be equal to the number of replicas.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
rbias	**CALC_RCT**	the instantaneous value of the bias normalized using the c(t) reweighting factor [rbias=bias-rct].This component can be used to obtain a reweighted histogram.
rct	**CALC_RCT**	the reweighting factor $c(t)$.
work	**CALC_WORK**	accumulator for work
acc	**ACCELERATION**	the metadynamics acceleration factor
maxbias	**CALC_MAX_BIAS**	the maximum of the metadynamics V(s, t)
transbias	**CALC_TRANSITION_BIAS**	the metadynamics transition bias V$*$(t)
pace	**FREQUENCY_ADAPTIVE**	the hill addition frequency when employing frequency adaptive metadynamics
nlker	**NLIST**	number of hills in the neighbor list
nlsteps	**NLIST**	number of steps from last neighbor list update

Compulsory keywords

SIGMA	the widths of the Gaussian hills
PACE	the frequency for hill addition
FILE	(default=HILLS) a file in which the list of added hills is stored

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
CALC_WORK	(default=off) calculate the total accumulated work done by the bias since last restart
CALC_RCT	(default=off) calculate the c(t) reweighting factor and use that to obtain the normalized bias [rbias=bias-rct].This method is not compatible with metadynamics not on a grid.
GRID_SPARSE	(default=off) use a sparse grid to store hills
GRID_NOSPLINE	(default=off) don't use spline interpolation with grids
STORE_GRIDS	(default=off) store all the grid files the calculation generates. They will be deleted if this keyword is not present
NLIST	(default=off) Use neighbor list for kernels summation, faster but experimental
WALKERS_MPI	(default=off) Switch on MPI version of multiple walkers - not compatible with WALKERS_∗ options other than WALKERS_DIR
FLYING_GAUSSIAN	(default=off) Switch on flying Gaussian method, must be used with WALKE↩ RS_MPI
ACCELERATION	(default=off) Set to TRUE if you want to compute the metadynamics acceleration factor.
CALC_MAX_BIAS	(default=off) Set to TRUE if you want to compute the maximum of the metadynamics V(s, t)
CALC_TRANSITION_BIAS	(default=off) Set to TRUE if you want to compute a metadynamics transition bias V∗(t)
FREQUENCY_ADAPTIVE	(default=off) Set to TRUE if you want to enable frequency adaptive metadynamics such that the frequency for hill addition to change dynamically based on the acceleration factor.
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If ∗ or ∗.∗ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.∗.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
HEIGHT	the heights of the Gaussian hills. Compulsory unless TAU and either BIASF↩ ACTOR or DAMPFACTOR are given
FMT	specify format for HILLS files (useful for decrease the number of digits in regtests)
BIASFACTOR	use well tempered metadynamics and use this bias factor. Please note you must also specify temp
RECT	list of bias factors for all the replicas
DAMPFACTOR	damp hills with exp(-max(V)/(kT∗DAMPFACTOR)
TTBIASFACTOR	use transition tempered metadynamics with this bias factor. Please note you must also specify temp
TTBIASTHRESHOLD	use transition tempered metadynamics with this bias threshold. Please note you must also specify TTBIASFACTOR

TTALPHA	use transition tempered metadynamics with this hill size decay exponent parameter. Please note you must also specify TTBIASFACTOR
TARGET	target to a predefined distribution
TEMP	the system temperature - this is only needed if you are doing well-tempered metadynamics
TAU	in well tempered metadynamics, sets height to (k_B Delta T$*$pace$*$timestep)/tau
RCT_USTRIDE	the update stride for calculating the $c(t)$ reweighting factor.The default 1, so $c(t)$ is updated every time the bias is updated.
GRID_MIN	the lower bounds for the grid
GRID_MAX	the upper bounds for the grid
GRID_BIN	the number of bins for the grid
GRID_SPACING	the approximate grid spacing (to be used as an alternative or together with GRID_BIN)
GRID_WSTRIDE	write the grid to a file every N steps
GRID_WFILE	the file on which to write the grid
GRID_RFILE	a grid file from which the bias should be read at the initial step of the simulation
NLIST_PARAMETERS	(default=6.,0.5) the two cutoff parameters for the Gaussians neighbor list
ADAPTIVE	use a geometric (=GEOM) or diffusion (=DIFF) based hills width scheme. Sigma is one number that has distance units or time step dimensions
SIGMA_MAX	the upper bounds for the sigmas (in CV units) when using adaptive hills. Negative number means no bounds
SIGMA_MIN	the lower bounds for the sigmas (in CV units) when using adaptive hills. Negative number means no bounds
WALKERS_ID	walker id
WALKERS_N	number of walkers
WALKERS_DIR	shared directory with the hills files from all the walkers
WALKERS_RSTRIDE	stride for reading hills files
INTERVAL	one dimensional lower and upper limits, outside the limits the system will not feel the biasing force.
ACCELERATION_RFILE	a data file from which the acceleration should be read at the initial step of the simulation
TRANSITIONWELL	This keyword appears multiple times as TRANSITIONWELL followed by an integer. Each specifies the coordinates for one well as in transition-tempered metadynamics. At least one must be provided.. You can use multiple instances of this keyword i.e. TRANSITIONWELL1, TRANSITIONWELL2, TRANSITIO↩ NWELL3...
FA_UPDATE_FREQUENCY	the frequency for updating the hill addition pace in frequency adaptive metadynamics, by default this is equal to the value given in PACE
FA_MAX_PACE	the maximum hill addition frequency allowed in frequency adaptive metadynamics. By default there is no maximum value.
FA_MIN_ACCELERATION	only update the hill addition pace in frequency adaptive metadynamics after reaching the minimum acceleration factor given here. By default it is 1.0.
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

7.8 MOVINGRESTRAINT

This is part of the bias module

Add a time-dependent, harmonic restraint on one or more variables.

This form of bias can be used to performed steered MD [60] and Jarzynski sampling [61].

The harmonic restraint on your system is given by:

$$V(\vec{s},t) = \frac{1}{2}\kappa(t)(\vec{s} - \vec{s}_0(t))^2$$

The time dependence of κ and $\vec{s}_0$ are specified by a list of STEP, KAPPA and AT keywords. These keywords tell plumed what values κ and $\vec{s}_0$ should have at the time specified by the corresponding STEP keyword. In between these times the values of κ and $\vec{s}_0$ are linearly interpolated.

Additional material and examples can be also found in the tutorial Belfast tutorial: Out of equilibrium dynamics

Examples

The following input is dragging the distance between atoms 2 and 4 from 1 to 2 in the first 1000 steps, then back in the next 1000 steps. In the following 500 steps the restraint is progressively switched off.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MOVINGRESTRAINT.tmp
DISTANCE ATOMS=2,4 LABEL=d
MOVINGRESTRAINT ...
  ARG=d
  STEP0=0    AT0=1.0 KAPPA0=100.0
  STEP1=1000 AT1=2.0
  STEP2=2000 AT2=1.0
  STEP3=2500         KAPPA3=0.0
... MOVINGRESTRAINT
```

The following input is progressively building restraints distances between atoms 1 and 5 and between atoms 2 and 4 in the first 1000 steps. Afterwards, the restraint is kept static.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MOVINGRESTRAINT.tmp
DISTANCE ATOMS=1,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
MOVINGRESTRAINT ...
  ARG=d1,d2
  STEP0=0    AT0=1.0,1.5 KAPPA0=0.0,0.0
  STEP1=1000 AT1=1.0,1.5 KAPPA1=1.0,1.0
... MOVINGRESTRAINT
```

The following input is progressively bringing atoms 1 and 2 close to each other with an upper wall

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MOVINGRESTRAINT.tmp
DISTANCE ATOMS=1,2 LABEL=d1
MOVINGRESTRAINT ...
  ARG=d1
  VERSE=U
  STEP0=0    AT0=1.0 KAPPA0=10.0
  STEP1=1000 AT1=0.0
... MOVINGRESTRAINT
```

By default the Action is issuing some values which are the work on each degree of freedom, the center of the harmonic potential, the total bias deposited

(See also DISTANCE).

Attention

Work is not computed properly when KAPPA is time dependent.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
work	the total work performed changing this restraint
force2	the instantaneous value of the squared force due to this bias potential
_cntr	one or multiple instances of this quantity can be referenced elsewhere in the input file. these quantities will named with the arguments of the bias followed by the character string _cntr. These quantities give the instantaneous position of the center of the harmonic potential.
_work	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will named with the arguments of the bias followed by the character string _work. These quantities tell the user how much work has been done by the potential in dragging the system along the various colvar axis.
_kappa	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will named with the arguments of the bias followed by the character string _kappa. These quantities tell the user the time dependent value of kappa.

Compulsory keywords

VERSE	(default=B) Tells plumed whether the restraint is only acting for CV larger (U) or smaller (L) than the restraint or whether it is acting on both sides (B)
STEP	This keyword appears multiple times as STEP x with x=0,1,2,...,n. Each value given represents the MD step at which the restraint parameters take the values KAPPA x and AT x.. You can use multiple instances of this keyword i.e. STEP1, STEP2, STEP3...
AT	AT x is equal to the position of the restraint at time STEP x. For intermediate times this parameter is linearly interpolated. If no AT x is specified for STEP x then the values of AT are kept constant during the interval of time between STEP $x-1$ and STEP x.. You can use multiple instances of this keyword i.e. AT1, AT2, AT3...
KAPPA	KAPPA x is equal to the value of the force constants at time STEP x. For intermediate times this parameter is linearly interpolated. If no KAPPA x is specified for STEP x then the values of KAPPA x are kept constant during the interval of time between STEP $x-1$ and STEP x.. You can use multiple instances of this keyword i.e. KAPPA1, KAPPA2, KAPPA3...

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

7.9 PBMETAD

This is part of the bias module

Used to performed Parallel Bias metadynamics.

This action activate Parallel Bias Metadynamics (PBMetaD) [62], a version of metadynamics [47] in which multiple low-dimensional bias potentials are applied in parallel. In the current implementation, these have the form of mono-dimensional metadynamics bias potentials:

$$V(s_1,t),...,V(s_N,t)$$

where:

$$V(s_i,t) = \sum_{k\tau<t} W_i(k\tau) \exp\left(-\frac{(s_i - s_i^{(0)}(k\tau))^2}{2\sigma_i^2}\right).$$

To ensure the convergence of each mono-dimensional bias potential to the corresponding free energy, at each deposition step the Gaussian heights are multiplied by the so-called conditional term:

$$W_i(k\tau) = W_0 \frac{\exp\left(-\frac{V(s_i,k\tau)}{k_B T}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{V(s_i,k\tau)}{k_B T}\right)}$$

where W_0 is the initial Gaussian height.

The PBMetaD bias potential is defined by:

$$V_{PB}(\vec{s},t) = -k_B T \log \sum_{i=1}^{N} \exp\left(-\frac{V(s_i,t)}{k_B T}\right).$$

Information on the Gaussian functions that build each bias potential are printed to multiple HILLS files, which are used both to restart the calculation and to reconstruct the mono-dimensional free energies as a function of the corresponding CVs. These can be reconstructed using the sum_hills utility because the final bias is given by:

$$V(s_i) = -F(s_i)$$

Currently, only a subset of the METAD options are available in PBMetaD.

The bias potentials can be stored on a grid to increase performances of long PBMetaD simulations. You should provide either the number of bins for every collective variable (GRID_BIN) or the desired grid spacing (GRID_SP↩ ACING). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither GRID_BIN nor GRID_SPACING) and if Gaussian width is fixed PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.

Another option that is available is well-tempered metadynamics [49]. In this variant of PBMetaD the heights of the Gaussian hills are scaled at each step by the additional well-tempered metadynamics term. This ensures that each bias converges more smoothly. It should be noted that, in the case of well-tempered metadynamics, in the output printed the Gaussian height is re-scaled using the bias factor. Also notice that with well-tempered metadynamics the HILLS files do not contain the bias, but the negative of the free-energy estimate. This choice has the advantage that one can restart a simulation using a different value for the ΔT. The applied bias will be scaled accordingly.

Note that you can use here also the flexible Gaussian approach [50] in which you can adapt the Gaussian to the extent of Cartesian space covered by a variable or to the space in collective variable covered in a given time. In this case the width of the deposited Gaussian potential is denoted by one value only that is a Cartesian space (ADA↩ PTIVE=GEOM) or a time (ADAPTIVE=DIFF). Note that a specific integration technique for the deposited Gaussian kernels should be used in this case. Check the documentation for utility sum_hills.

With the keyword INTERVAL one changes the metadynamics algorithm setting the bias force equal to zero outside boundary [51]. If, for example, metadynamics is performed on a CV s and one is interested only to the free energy for s $>$ boundary, the history dependent potential is still updated according to the above equations but the metadynamics force is set to zero for s $<$ boundary. Notice that Gaussian kernels are added also if s $<$ boundary, as the tails of these Gaussian kernels influence VG in the relevant region s $>$ boundary. In this way, the force on the system in the region s $>$ boundary comes from both metadynamics and the force field, in the region s $<$ boundary only from the latter. This approach allows obtaining a history-dependent bias potential VG that fluctuates around a stable estimator, equal to the negative of the free energy far enough from the boundaries. Note that:

- It works only for one-dimensional biases;
- It works both with and without GRID;
- The interval limit boundary in a region where the free energy derivative is not large;
- If in the region outside the limit boundary the system has a free energy minimum, the INTERVAL keyword should be used together with a UPPER_WALLS or LOWER_WALLS at boundary.

For systems with multiple CVs that share identical properties, PBMetaD with partitioned families can be used to group them under one bias potential that each contributes to [63]. This is done with a list of PF keywords, where each PF$*$ argument contains the list of CVs from ARG to be placed in that family. Once invoked, each CV in ARG must be placed in exactly one PF, even if it results in families containing only one CV. Additionally, in cases where each of SIGMA or GRID entry would correspond to each ARG entry, they now correspond to each PF and must be adjusted accordingly.

Multiple walkers [52] can also be used. See below the examples.

Examples

The following input is for PBMetaD calculation using as collective variables the distance between atoms 3 and 5 and the distance between atoms 2 and 4. The value of the CVs and the PBMetaD bias potential are written to the COLVAR file every 100 steps.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PBMETAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
PBMETAD ARG=d1,d2 SIGMA=0.2,0.2 HEIGHT=0.3 PACE=500 LABEL=pb FILE=HILLS_d1,HILLS_d2
PRINT ARG=d1,d2,pb.bias STRIDE=100 FILE=COLVAR
```

(See also DISTANCE and PRINT).

If you use well-tempered metadynamics, you should specify a single bias factor and initial Gaussian height.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PBMETAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
PBMETAD ...
ARG=d1,d2 SIGMA=0.2,0.2 HEIGHT=0.3
PACE=500 BIASFACTOR=8 LABEL=pb
FILE=HILLS_d1,HILLS_d2
... PBMETAD
PRINT ARG=d1,d2,pb.bias STRIDE=100 FILE=COLVAR
```

Using partitioned families, each CV in ARG must be in exactly one family. Here, the distance between atoms 1,2 is degenerate with 2,4, but not with the distance between 3,5. Note that two SIGMA are provided to match the two PF.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PBMETAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
DISTANCE ATOMS=1,2 LABEL=d3
PBMETAD ...
ARG=d1,d2,d3 SIGMA=0.2,0.2 HEIGHT=0.3
PF0=d1 PF1=d2,d3
PACE=500 BIASFACTOR=8 LABEL=pb
FILE=HILLS_d1,HILLS_d2
... PBMETAD
PRINT ARG=d1,d2,d3,pb.bias STRIDE=100 FILE=COLVAR
```

The following input enables the MPI version of multiple-walkers.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PBMETAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
PBMETAD ...
ARG=d1,d2 SIGMA=0.2,0.2 HEIGHT=0.3
PACE=500 BIASFACTOR=8 LABEL=pb
FILE=HILLS_d1,HILLS_d2
WALKERS_MPI
... PBMETAD
PRINT ARG=d1,d2,pb.bias STRIDE=100 FILE=COLVAR
```

The disk version of multiple-walkers can be enabled by setting the number of walker used, the id of the current walker which interprets the input file, the directory where the hills containing files resides, and the frequency to read the other walkers. Here is an example

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PBMETAD.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
PBMETAD ...
ARG=d1,d2 SIGMA=0.2,0.2 HEIGHT=0.3
PACE=500 BIASFACTOR=8 LABEL=pb
FILE=HILLS_d1,HILLS_d2
WALKERS_N=10
WALKERS_ID=3
WALKERS_DIR=../
WALKERS_RSTRIDE=100
... PBMETAD
PRINT ARG=d1,d2,pb.bias STRIDE=100 FILE=COLVAR
```

where WALKERS_N is the total number of walkers, WALKERS_ID is the id of the present walker (starting from 0) and the WALKERS_DIR is the directory where all the walkers are located. WALKERS_RSTRIDE is the number of step between one update and the other.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential

Compulsory keywords

SIGMA	the widths of the Gaussian hills
PACE	the frequency for hill addition, one for all biases

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
GRID_SPARSE	(default=off) use a sparse grid to store hills
GRID_NOSPLINE	(default=off) don't use spline interpolation with grids
WALKERS_MPI	(default=off) Switch on MPI version of multiple walkers - not compatible with WALKERS_$*$ options other than WALKERS_DIR

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If ∗ or ∗.∗ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.∗.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
FILE	files in which the lists of added hills are stored, default names are assigned using arguments if FILE is not found
HEIGHT	the height of the Gaussian hills, one for all biases. Compulsory unless TAU, TEMP and BIASFACTOR are given
FMT	specify format for HILLS files (useful for decrease the number of digits in regtests)
BIASFACTOR	use well tempered metadynamics with this bias factor, one for all biases. Please note you must also specify temp
TEMP	the system temperature - this is only needed if you are doing well-tempered metadynamics
TAU	in well tempered metadynamics, sets height to (k_B Delta T∗pace∗timestep)/tau
GRID_MIN	the lower bounds for the grid
GRID_MAX	the upper bounds for the grid
GRID_BIN	the number of bins for the grid
GRID_SPACING	the approximate grid spacing (to be used as an alternative or together with GRID_BIN)
GRID_WSTRIDE	frequency for dumping the grid
GRID_WFILES	dump grid for the bias, default names are used if GRID_WSTRIDE is used without GRID_WFILES.
GRID_RFILES	read grid for the bias
ADAPTIVE	use a geometric (=GEOM) or diffusion (=DIFF) based hills width scheme. Sigma is one number that has distance units or timestep dimensions
SIGMA_MAX	the upper bounds for the sigmas (in CV units) when using adaptive hills. Negative number means no bounds
SIGMA_MIN	the lower bounds for the sigmas (in CV units) when using adaptive hills. Negative number means no bounds
PF	specify which CVs belong in a partitioned family. Once a PF is specified, all CVs in ARG must be placed in a PF even if there is one CV per PF". You can use multiple instances of this keyword i.e. PF1, PF2, PF3...
SELECTOR	add forces and do update based on the value of SELECTOR
SELECTOR_ID	value of SELECTOR
WALKERS_ID	walker id
WALKERS_N	number of walkers
WALKERS_DIR	shared directory with the hills files from all the walkers
WALKERS_RSTRIDE	stride for reading hills files
INTERVAL_MIN	one dimensional lower limits, outside the limits the system will not feel the biasing force.

INTERVAL_MAX	one dimensional upper limits, outside the limits the system will not feel the biasing force.
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

7.10 RESTRAINT

This is part of the bias module

Adds harmonic and/or linear restraints on one or more variables.

Either or both of SLOPE and KAPPA must be present to specify the linear and harmonic force constants respectively. The resulting potential is given by:

$$\sum_i \frac{k_i}{2}(x_i - a_i)^2 + m_i * (x_i - a_i)$$

.

The number of components for any vector of force constants must be equal to the number of arguments to the action.

Additional material and examples can be also found in the tutorial Lugano tutorial: Using restraints

Examples

The following input tells plumed to restrain the distance between atoms 3 and 5 and the distance between atoms 2 and 4, at different equilibrium values, and to print the energy of the restraint

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESTRAINT.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
RESTRAINT ARG=d1,d2 AT=1.0,1.5 KAPPA=150.0,150.0 LABEL=restraint
PRINT ARG=restraint.bias
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	the instantaneous value of the squared force due to this bias potential

Compulsory keywords

SLOPE	(default=0.0) specifies that the restraint is linear and what the values of the force constants on each of the variables are
KAPPA	(default=0.0) specifies that the restraint is harmonic and what the values of the force constants on each of the variables are
AT	the position of the restraint

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

7.11 UPPER_WALLS

This is part of the bias module

Defines a wall for the value of one or more collective variables, which limits the region of the phase space accessible during the simulation.

The restraining potential starts acting on the system when the value of the CV is greater (in the case of UPPER↩ _WALLS) or lower (in the case of LOWER_WALLS) than a certain limit a_i (AT) minus an offset o_i (OFFSET). The expression for the bias due to the wall is given by:

$\sum_i k_i((x_i - a_i + o_i)/s_i)_i^e$

k_i (KAPPA) is an energy constant in internal unit of the code, s_i (EPS) a rescaling factor and e_i (EXP) the exponent determining the power law. By default: EXP = 2, EPS = 1.0, OFFSET = 0.

Examples

The following input tells plumed to add both a lower and an upper walls on the distance between atoms 3 and 5 and the distance between atoms 2 and 4. The lower and upper limits are defined at different values. The strength of the walls is the same for the four cases. It also tells plumed to print the energy of the walls.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/UPPER_WALLS.tmp
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=2,4 LABEL=d2
UPPER_WALLS ARG=d1,d2 AT=1.0,1.5 KAPPA=150.0,150.0 EXP=2,2 EPS=1,1 OFFSET=0,0 LABEL=uwall
LOWER_WALLS ARG=d1,d2 AT=0.0,1.0 KAPPA=150.0,150.0 EXP=2,2 EPS=1,1 OFFSET=0,0 LABEL=lwall
PRINT ARG=uwall.bias,lwall.bias
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	the instantaneous value of the squared force due to this bias potential

Compulsory keywords

AT	the positions of the wall. The a_i in the expression for a wall.
KAPPA	the force constant for the wall. The k_i in the expression for a wall.
OFFSET	(default=0.0) the offset for the start of the wall. The o_i in the expression for a wall.
EXP	(default=2.0) the powers for the walls. The e_i in the expression for a wall.
EPS	(default=1.0) the values for s_i in the expression for a wall

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

7.12 RESTART

This is part of the setup module

Activate restart.

This is a Setup directive and, as such, should appear at the beginning of the input file. It influences the way PLUMED treat files open for writing (see also Files).

Notice that it is also possible to enable or disable restart on a per-action basis using the RESTART keyword on a single action. In this case, the keyword should be assigned a value. RESTART=AUTO means that global settings are used, RESTART=YES or RESTART=NO respectively enable and disable restart for that single action.

Attention

This directive can have also other side effects, e.g. on METAD and PBMETAD and on some analysis action.

Examples

Using the following input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESTART.tmp
d: DISTANCE ATOMS=1,2
PRINT ARG=d FILE=out
```

a new 'out' file will be created. If an old one is on the way, it will be automatically backed up.

On the other hand, using the following input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESTART.tmp
RESTART
d: DISTANCE ATOMS=1,2
PRINT ARG=d FILE=out
```

the file 'out' will be appended.

In the following case, file out1 will be backed up and file out2 will be concatenated

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESTART.tmp
RESTART
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=1,2
PRINT ARG=d1 FILE=out1 RESTART=NO
PRINT ARG=d2 FILE=out2
```

In the following case, file out will backed up even if the MD code thinks that we are restarting. Notice that not all the MD code send to PLUMED information about restarts. If you are not sure, always put `RESTART` when you are restarting and nothing when you aren't

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESTART.tmp
RESTART NO
d1: DISTANCE ATOMS=1,2
PRINT ARG=d1 FILE=out1
```

Glossary of keywords and components

Options

NO	(default=off) switch off restart - can be used to override the behavior of the MD engine

Chapter 8

Additional Modules

Here is collected the documentation for the additional modules contributed to PLUMED. For information on how to enable modules see List of modules.

Module	Description	Authors	References
ANN (Artificial Neural Network) module	ANN (Artificial Neural Network) function	Wei Chen and Andrew Ferguson	
FISST (Infinite Switch Simulated Tempering in Force)	Infinite Switch Simulated Tempering in Force (FIS↩ ST)	Glen Hocky	[64]
PLUMED-ISDB	Integrative Structural and Dynamical Biology with PLUMED	Max Bonomi and Carlo Camilloni	[65]
PYTORCH (Machine Learning Collective Variables)	Machine Learning Collective Variables with PyTorch (pytorch)	Luigi Bonati	[66]
Logarithmic Mean Force Dynamics	Method for enhanced sampling and for free energy calculations along collective variables	Tetsuya Morishita, Naoki Watanabe	[67] [68] [69]
Experiment Directed Simulation	Methods for incorporating additional information about CVs into MD simulations by adaptively determined linear bias parameters	Glen Hocky, Andrew White	[26] [70] [71]
Extended-System Adaptive Biasing Force	Methods for performing eABF or DRR method to calculate PMF along CVs	Haochuan Chen, Haohao Fu	[72] [73] [74] [75]
Variationally Enhanced Sampling (VES module)	Module that implements enhanced sampling methods based on Variationally Enhanced Sampling	Omar Valsson	[76]
MAZE	Module that implements enhanced sampling methods for ligand unbinding from protein tunnels	Jakub Rydzewski	[77]
OPES (On-the-fly Probability Enhanced Sampling)	On-the-fly Probability Enhanced Sampling (OPES)	Michele Invernizzi	[78] [79]

PIV collective variable	Permutation invariant collective variable (PIV)	S. Pipolo, F. Pietrucci	[80] [81]
S2 contact model collective variable	S2 contact model collective variable (S2CM)	Omar Valsson	[82]
SASA collective variable	Solvent Accessible Surface Area collective variable (SASA)	Andrea Arsiccio	[83] [84] [85] [86] [87]
Funnel-Metadynamics (FM)	a collective variable and a bias action necessary to perform Funnel-↩ Metadynamics on Molecular Dynamics simulations	Stefano Raniolo, Vittorio Limongelli	[88] [89]
Membrane Fusion	a set of collective variables that induces different steps in the MF process.	Ary Lautaro Di Bartolo, Diego Masone	[90] [91] [92] [93]

8.1 ANN (Artificial Neural Network) function

8.1.1 Overview

This is plumed ANN function (annfunc) module. It implements `ANN` class, which is a subclass of `Function` class. `ANN` class takes multi-dimensional arrays as inputs for a fully-connected feedforward neural network with specified neural network weights and generates corresponding outputs. The `ANN` outputs can be used as collective variables, inputs for other collective variables, or inputs for data analysis tools.

8.1.2 Installation

This module is not installed by default. Add '--enable-modules=annfunc' to your './configure' command when building PLUMED to enable these features.

8.1.3 Usage

Currently, all features of the ANNfunc module are included in a single ANNfunc collective variable: ANN

8.1.4 Contents

8.1.5 Functions Documentation

The following list contains descriptions of functions developed for the PLUMED-ANNfunc module. They can be used in combination with other actions outside of the ANNfunc module.

ANN	Calculates the ANN-function.

8.1.5.1 ANN

	This is part of the annfunc module
	It is only available if you configure PLUMED with ./configure –enable-modules=annfunc . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculates the ANN-function.

This module implements ANN class, which is a subclass of Function class. ANN class takes multi-dimensional arrays as inputs for a fully-connected feedforward neural network with specified neural network weights and generates corresponding outputs. The ANN outputs can be used as collective variables, inputs for other collective variables, or inputs for data analysis tools.

Examples

Assume we have an ANN with numbers of nodes being [2, 3, 1], and weights connecting layer 0 and 1 are

[[1,2], [3,4], [5,6]]

weights connecting layer 1 and 2 are

[[7,8,9]]

Bias for layer 1 and 2 are [10, 11, 12] and [13], respectively.

All activation functions are Tanh.

Then if input variables are l_0_out_0, l_0_out_1, the corresponding ANN function object can be defined using following plumed script:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/ANN.tmp
ANN ...
LABEL=ann
ARG=l_0_out_0,l_0_out_1
NUM_LAYERS=3
NUM_NODES=2,3,1
ACTIVATIONS=Tanh,Tanh
WEIGHTS0=1,2,3,4,5,6
WEIGHTS1=7,8,9
BIASES0=10,11,12
BIASES1=13
... ANN
```

To access its components, we use "ann.node-0", "ann.node-1", ..., which represents the components of neural network outputs.

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
node	components of ANN outputs

Compulsory keywords

PERIODIC	if the output of your function is periodic then you should specify the periodicity of the function. If the output is not periodic you must state this using PERIODIC=NO
NUM_LAYERS	number of layers of the neural network
NUM_NODES	numbers of nodes in each layer of the neural network
ACTIVATIONS	activation functions for the neural network

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
WEIGHTS	flattened weight arrays connecting adjacent layers, WEIGHTS0 represents flattened weight array connecting layer 0 and layer 1, WEIGHTS1 represents flattened weight array connecting layer 1 and layer 2, ... You can use multiple instances of this keyword i.e. WEIGHTS1, WEIGHTS2, WEIGHTS3...
BIASES	bias array for each layer of the neural network, BIASES0 represents bias array for layer 1, BIASES1 represents bias array for layer 2, ... You can use multiple instances of this keyword i.e. BIASES1, BIASES2, BIASES3...

8.2 FISST (Infinite Switch Simulated Tempering in Force)

8.2.1 Overview

This FISST module contains methods for adaptively determining weight parameters to construct a bias function that represents the Infinite Switch limit of Simulated Tempering for a linear bias coefficient of a CV, as described in [64].

8.2.2 Installation

This module is not installed by default. Add '--enable-modules=fisst' to your './configure' command when building PLUMED to enable these features.

8.2.3 Usage

Currently, all features of the FISST module are included in a single FISST bias function: FISST

8.2.4 Contents

8.2.5 Biases Documentation

The following list contains descriptions of biases developed for the PLUMED-FISST module. They can be used in combination with other biases outside of the FISST module.

FISST	Compute and apply the optimal linear force on an observable to enhance sampling of conformational distributions over a range of applied forces.

8.2.5.1 FISST

	This is part of the fisst module
	It is only available if you configure PLUMED with ./configure –enable-modules=fisst . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Compute and apply the optimal linear force on an observable to enhance sampling of conformational distributions over a range of applied forces.

This method is described in [64]

If the system's Hamiltonian is given by:

$$H(\vec{p},\vec{q}) = \sum_j \frac{p_j^2}{2m_j} + U(\vec{q}),$$

This bias modifies the Hamiltonian to be:

$$H'(\vec{p},\vec{q}) = H(\vec{p},\vec{q}) - \bar{F}Q$$

where for CV Q, a coupling constant $\bar{F}$ is determined adaptively according to the FISST algorithm.

Specifically,

$$\bar{F}(Q) = \frac{\int_{F_{min}}^{F_{max}} e^{\beta F Q(\vec{q})}\omega(F)F dF}{\int_{F_{min}}^{F_{max}} e^{\beta F Q(\vec{q})}\omega(F) dF},$$

where $\vec{q}$ are the molecular coordinates of the system, and $w(F)$ is a weighting function that is learned on the fly for each force by the FISST algorithm (starting from an initial weight distribution, uniform by default).

The target for $w(F) = 1/Z_q(F)$, where

$$Z_q(F) \equiv \int d\vec{q} e^{-\beta U(\vec{q}) + \beta F Q(\vec{q})}.$$

FISST also computes and writes Observable Weights $W_F(\vec{q}_t)$ for a molecular configuration at time t, so that averages of other quantities $A(\vec{q})$ can be reconstructed later at different force values (over a trajectory with T samples):

$$\langle A \rangle_F = \frac{1}{T} \sum_t W_F(\vec{q}_t) A(\vec{q}_t).$$

Examples

In the following example, an adaptive restraint is learned to bias the distance between two atoms in a system, for a force range of 0-100 pN.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FISST.tmp
UNITS LENGTH=A TIME=fs ENERGY=kcal/mol

b1: GROUP ATOMS=1
b2: GROUP ATOMS=12

dend: DISTANCE ATOMS=b1,b2

#The conversion factor is 69.4786 pN = 1 kcal/mol/Angstrom

#0 pN to 100 pN
f: FISST MIN_FORCE=0 MAX_FORCE=1.44 PERIOD=100 NINTERPOLATE=31 ARG=dend OUT_RESTART=pull.restart.txt OUT_OBSER

PRINT ARG=dend,f.dend_fbar,f.bias,f.force2 FILE=pull.colvar.txt STRIDE=1000
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	squared value of force from the bias.
_fbar	For each named CV biased, there will be a corresponding output CV_fbar storing the current linear bias prefactor.

Compulsory keywords

PERIOD	Steps corresponding to the learning rate
NINTERPOLATE	Number of grid points on which to do interpolation.
MIN_FORCE	Minimum force (per CV) to use for sampling. Units: [Energy]/[CV] (can be negative).
MAX_FORCE	Maximum force (per CV) to use for sampling.
CENTER	(default=0) The CV value at which the applied bias energy will be zero

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
FREEZE	(default=off) Fix bias weights at current level (only used for restarting).
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
RESET_PERIOD	Reset the learning statistics every time this number of steps comes around.
KBT	The system temperature in units of KB$*$T. If not provided will be taken from MD code (if available)
INITIAL_WEIGHT_DIST	Starting distribution for the force weights (options: UNIFORM, EXP, GAUSS).
INITIAL_WEIGHT_RATE	Rate of decay for exponential and gaussian distributions. W(F)$\sim$exp(-r $\vert$F$\vert\hat{}$d).
RESTART_FMT	the format that should be used to output real numbers in FISST restarts.
OUT_RESTART	Output file for all information needed to continue FISST simulation.If you have the RESTART directive set (global or for FISST), this file will be appended to.$\hookleftarrow$ Note that the header will be printed again if appending.
IN_RESTART	Read this file to continue an FISST simulation. If same as OUT_RESTART and you have not set the RESTART directive, the file will be backed-up and overwritten with new output.If you do have the RESTART flag set and it is the same name as OUT_RESTART, this file will be appended.
OUT_OBSERVABLE	Output file putting weights needed to compute observables at different force values.If you have the RESTART directive set (global or for FISST), this file will be appended to. Note that the header will be printed again if appending.
OBSERVABLE_FREQ	How often to write out observable weights (default=period).
RESTART	allows per-action setting of restart (YES/NO/AUTO)

8.3 PLUMED-ISDB

Here are listed the collective variables, functions and biases originally developed for the Integrative Structural and Dynamical Biology module of PLUMED. They are related but not limited to the interpretation and modelling of experimental data in molecular modelling.

- CVs Documentation
- Functions Documentation
- General Actions Documentation
- Biases Documentation

Additional tutorials focused on the ISDB module are included in the following and are meant as advanced tutorials.

- Tutorials

8.3.1 CVs Documentation

The following list contains descriptions of a number of the colvars that are currently implemented in the PLU↩ MED-ISDB module. These collective variables are related to the definitions of models to interpret experimental observables. They can be used in combination with any other collective variable, function or bias also outside the ISDB module.

CS2BACKBONE	Calculates the backbone chemical shifts for a protein.
EMMI	Calculate the fit of a structure or ensemble of structures with a cryo-EM density map.
FRET	Calculates the FRET efficiency between a pair of atoms.The efficiency is calculated using the Forster relation:
JCOUPLING	Calculates ^3J coupling constants for a dihedral angle.
NOE	Calculates NOE intensities as sums of 1/r^6, also averaging over multiple equivalent atoms or ambiguous NOE.
PCS	Calculates the Pseudo-contact shift of a nucleus determined by the presence of a metal ion susceptible to anisotropic magnetization.
PRE	Calculates the Paramagnetic Resonance Enhancement intensity ratio between a spin label atom and a list of atoms .
RDC	Calculates the (Residual) Dipolar Coupling between two atoms.
SANS	Calculates SANS intensity.
SAXS	Calculates SAXS intensity.

8.3.1.1 CS2BACKBONE

	This is part of the isdb module

Calculates the backbone chemical shifts for a protein.

The functional form is that of CamShift [94]. The chemical shift of the selected nuclei can be saved as components. Alternatively one can calculate either the CAMSHIFT score (useful as a collective variable [95] or as a scoring

function [96]) or a METAINFERENCE score (using DOSCORE). For these two latter cases experimental chemical shifts must be provided.

CS2BACKBONE calculation can be relatively heavy because it often uses a large number of atoms, it can be run in parallel using MPI and OpenMP.

As a general rule, when using CS2BACKBONE or other experimental restraints it may be better to increase the accuracy of the constraint algorithm due to the increased strain on the bonded structure. In the case of GROMACS it is safer to use lincs-iter=2 and lincs-order=6.

In general the system for which chemical shifts are calculated must be completely included in ATOMS and a T↩ EMPLATE pdb file for the same atoms should be provided as well in the folder DATADIR. The system is made automatically whole unless NOPBC is used, in particular if the system is made by multiple chains it is usually better to use NOPBC and make the molecule whole WHOLEMOLECULES selecting an appropriate order of the atoms. The pdb file is needed to the generate a simple topology of the protein. For histidine residues in protonation states different from D the HIE/HSE HIP/HSP name should be used. GLH and ASH can be used for the alternative protonation of GLU and ASP. Non-standard amino acids and other molecules are not yet supported, but in principle they can be named UNK. If multiple chains are present the chain identifier must be in the standard PDB format, together with the TER keyword at the end of each chain. Termini groups like ACE or NME should be removed from the TEMPLATE pdb because they are not recognized by CS2BACKBONE.

Atoms indices in the TEMPLATE file should be numbered from 1 to N where N is the number of atoms used in A↩ TOMS. This is not a problem for simple cases where atoms goes from 1 to N but is instead something to be carefull in case that a terminal group is removed from the PDB file.

In addition to a pdb file one needs to provide a list of chemical shifts to be calculated using one file per nucleus type (CAshifts.dat, CBshifts.dat, Cshifts.dat, Hshifts.dat, HAshifts.dat, Nshifts.dat), add only the files for the nuclei you need, but each file should include all protein residues. A chemical shift for a nucleus is calculated if a value greater than 0 is provided. For practical purposes the value can correspond to the experimental value. Residues numbers should match that used in the pdb file, but must be positive, so double check the pdb. The first and last residue of each chain should be preceded by a # character.

```
CAshifts.dat:
#1 0.0
2 55.5
3 58.4
.
.
#last 0.0
#first of second chain
.
#last of second chain
```

The default behavior is to store the values for the active nuclei in components (ca-#, cb-#, co-#, ha-#, hn-#, nh-# and expca-#, expcb-#, expco-#, expha-#, exphn-#, exp-nh#) with NOEXP it is possible to only store the back-calculated values, where # includes a chain and residue number.

One additional file is always needed in the folder DATADIR: camshift.db. This file includes all the parameters needed to calculate the chemical shifts and can be found in regtest/isdb/rt-cs2backbone/data/ .

Additional material and examples can be also found in the tutorial ISDB: setting up a Metadynamics Metainference simulation as well as in the cs2backbone regtests in the isdb folder.

Examples

In this first example the chemical shifts are used to calculate a collective variable to be used in NMR driven Metadynamics [95] :

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CS2BACKBONE.tmp
#SETTINGS AUXFOLDER=regtest/isdb/rt-cs2backbone/data
whole: GROUP ATOMS=2612-2514:-1,961-1:-1,2466-962:-1,2513-2467:-1
WHOLEMOLECULES ENTITY0=whole
cs: CS2BACKBONE ATOMS=1-2612 DATADIR=data/ TEMPLATE=template.pdb CAMSHIFT NOPBC
metad: METAD ARG=cs HEIGHT=0.5 SIGMA=0.1 PACE=200 BIASFACTOR=10
PRINT ARG=cs,metad.bias FILE=COLVAR STRIDE=100
```

In this second example the chemical shifts are used as replica-averaged restrained as in [97] [98] .

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CS2BACKBONE.tmp
#SETTINGS AUXFOLDER=regtest/isdb/rt-cs2backbone/data NREPLICAS=2
cs: CS2BACKBONE ATOMS=1-174 DATADIR=data/
encs: ENSEMBLE ARG=(cs\.hn-.*),(cs\.nh-.*)
stcs: STATS ARG=encs.* SQDEVSUM PARARG=(cs\.exphn-.*),(cs\.expnh-.*)
RESTRAINT ARG=stcs.sqdevsum AT=0 KAPPA=0 SLOPE=24

PRINT ARG=(cs\.hn-.*),(cs\.nh-.*) FILE=RESTRAINT STRIDE=100
```

This third example show how to use chemical shifts to calculate a METAINFERENCE score .

```
BEGIN_PLUMED_FILE working DATADIR=example-check/CS2BACKBONE.tmp
#SETTINGS AUXFOLDER=regtest/isdb/rt-cs2backbone/data
cs: CS2BACKBONE ATOMS=1-174 DATADIR=data/ SIGMA_MEAN0=1.0 DOSCORE
csbias: BIASVALUE ARG=cs.score

PRINT ARG=(cs\.hn-.*),(cs\.nh-.*) FILE=CS.dat STRIDE=1000
PRINT ARG=cs.score FILE=BIAS STRIDE=100
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
ha	the calculated Ha hydrogen chemical shifts
hn	the calculated H hydrogen chemical shifts
nh	the calculated N nitrogen chemical shifts

ca	the calculated Ca carbon chemical shifts
cb	the calculated Cb carbon chemical shifts
co	the calculated C' carbon chemical shifts
expha	the experimental Ha hydrogen chemical shifts
exphn	the experimental H hydrogen chemical shifts
expnh	the experimental N nitrogen chemical shifts
expca	the experimental Ca carbon chemical shifts
expcb	the experimental Cb carbon chemical shifts
expco	the experimental C' carbon chemical shifts

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator

The atoms involved can be specified using

ATOMS	The atoms to be included in the calculation, e.g. the whole protein.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUT↩ LIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation
DATADIR	(default=data/) The folder with the experimental chemical shifts.

TEMPLATE	(default=template.pdb) A PDB file of the protein system.
NEIGH_FREQ	(default=20) Period in step for neighbor list update.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging
REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
CAMSHIFT	(default=off) Set to TRUE if you to calculate a single CamShift score.
NOEXP	(default=off) Set to TRUE if you don't want to have fixed components with the experimental values.
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps

MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)

8.3.1.2 EMMI

This is part of the isdb module

Calculate the fit of a structure or ensemble of structures with a cryo-EM density map.

This action implements the multi-scale Bayesian approach to cryo-EM data fitting introduced in Ref. [99] . This method allows efficient and accurate structural modeling of cryo-electron microscopy density maps at multiple scales, from coarse-grained to atomistic resolution, by addressing the presence of random and systematic errors in the data, sample heterogeneity, data correlation, and noise correlation.

The experimental density map is fit by a Gaussian Mixture Model (GMM), which is provided as an external file specified by the keyword GMM_FILE. We are currently working on a web server to perform this operation. In the meantime, the user can request a stand-alone version of the GMM code at massimiliano.bonomi_AT_gmail.com.

When run in single-replica mode, this action allows atomistic, flexible refinement of an individual structure into a density map. Combined with a multi-replica framework (such as the -multi option in GROMACS), the user can model an ensemble of structures using the Metainference approach [100] .

Warning

To use EMMI, the user should always add a MOLINFO line and specify a pdb file of the system.

Note

To enhance sampling in single-structure refinement, one can use a Replica Exchange Method, such as Parallel Tempering. In this case, the user should add the NO_AVER flag to the input line. To use a replica-based enhanced sampling scheme such as Parallel-Bias Metadynamics (PBMETAD), one should use the REWEI↩ GHT flag and pass the Metadynamics bias using the ARG keyword.

EMMI can be used in combination with periodic and non-periodic systems. In the latter case, one should add the NOPBC flag to the input line

Examples

In this example, we perform a single-structure refinement based on an experimental cryo-EM map. The map is fit with a GMM, whose parameters are listed in the file GMM_fit.dat. This file contains one line per GMM component in the following format:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/EMMI.tmp
#! FIELDS Id Weight Mean_0 Mean_1 Mean_2 Cov_00 Cov_01 Cov_02 Cov_11 Cov_12 Cov_22 Beta
     0  2.9993805e+01   6.54628 10.37820 -0.92988  2.078920e-02 1.216254e-03 5.990827e-04 2.556246e-02 8.41183
     1  2.3468312e+01   6.56095 10.34790 -0.87808  1.879859e-02 6.636049e-03 3.682865e-04 3.194490e-02 1.75052
     ...
```

To accelerate the computation of the Bayesian score, one can:

- use neighbor lists, specified by the keywords NL_CUTOFF and NL_STRIDE;
- calculate the restraint every other step (or more).

All the heavy atoms of the system are used to calculate the density map. This list can conveniently be provided using a GROMACS index file.

The input file looks as follows:

```
BEGIN_PLUMED_FILE
# include pdb info
MOLINFO STRUCTURE=prot.pdb

#  all heavy atoms
protein-h: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein-H

# create EMMI score
gmm: EMMI NOPBC SIGMA_MIN=0.01 TEMP=300.0 NL_STRIDE=100 NL_CUTOFF=0.01 GMM_FILE=GMM_fit.dat ATOMS=protein-h

# translate into bias - apply every 2 steps
emr: BIASVALUE ARG=gmm.scoreb STRIDE=2

PRINT ARG=emr.* FILE=COLVAR STRIDE=500 FMT=%20.10f
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
scoreb	Bayesian score
neff	effective number of replicas

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acc	**NOISETYPE**	MC acceptance for uncertainty
scale	**REGRESSION**	scale factor
accscale	**REGRESSION**	MC acceptance for scale regression
enescale	**REGRESSION**	MC energy for scale regression
anneal	**ANNEAL**	annealing factor
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
sigma	**NOISETYPE**	uncertainty in the forward models and experiment

The atoms involved can be specified using

ATOMS	atoms for which we calculate the density map, typically all heavy atoms. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

GMM_FILE	file with the parameters of the GMM components
NL_CUTOFF	The cutoff in overlap for the neighbor list
NL_STRIDE	The frequency with which we are updating the neighbor list
SIGMA_MIN	minimum uncertainty
RESOLUTION	Cryo-EM map resolution
NOISETYPE	functional form of the noise (GAUSS, OUTLIERS, MARGINAL)

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
NO_AVER	(default=off) don't do ensemble averaging in multi-replica mode
REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
SIGMA0	initial value of the uncertainty
DSIGMA	MC step for uncertainties
MC_STRIDE	Monte Carlo stride
ERR_FILE	file with experimental or GMM fit errors
OV_FILE	file with experimental overlaps
NORM_DENSITY	integral of the experimental density
STATUS_FILE	write a file with all the data useful for restart
WRITE_STRIDE	write the status to a file every N steps, this can be used for restart
REGRESSION	regression stride

REG_SCALE_MIN	regression minimum scale
REG_SCALE_MAX	regression maximum scale
REG_DSCALE	regression maximum scale MC move
SCALE	scale factor
ANNEAL	Length of annealing cycle
ANNEAL_FACT	Annealing temperature factor
TEMP	temperature
PRIOR	exponent of uncertainty prior
WRITE_OV_STRIDE	write model overlaps every N steps
WRITE_OV	write a file with model overlaps
AVERAGING	Averaging window for weights

8.3.1.3 FRET

This is part of the isdb module

Calculates the FRET efficiency between a pair of atoms. The efficiency is calculated using the Forster relation:

$$E = \frac{1}{1 + (R/R_0)^6}$$

where R is the distance and R_0 is the Forster radius.

By default the distance is computed taking into account periodic boundary conditions. This behavior can be changed with the NOPBC flag.

Examples

The following input tells plumed to print the FRET efficiencies calculated as a function of the distance between atoms 3 and 5 and the distance between atoms 2 and 4.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FRET.tmp
fe1:  FRET ATOMS=3,5 R0=5.5
fe2:  FRET ATOMS=2,4 R0=5.5
PRINT ARG=fe1,fe2
```

The following input computes the FRET efficiency calculated on the terminal atoms of a polymer of 100 atoms and keeps it at a value around 0.5.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FRET.tmp
WHOLEMOLECULES ENTITY0=1-100
fe: FRET ATOMS=1,100 R0=5.5 NOPBC
RESTRAINT ARG=fe KAPPA=100 AT=0.5
```

Notice that NOPBC is used to be sure that if the distance is larger than half the simulation box the distance is compute properly. Also notice that, since many MD codes break molecules across cell boundary, it might be necessary to use the WHOLEMOLECULES keyword (also notice that it should be *before* FRET). Just be sure that the ordered list provide to WHOLEMOLECULES has the following properties:

- Consecutive atoms should be closer than half-cell throughout the entire simulation.
- Atoms required later for the distance (e.g. 1 and 100) should be included in the list

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the pair of atom that we are calculating the distance between. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

R0	The value of the Forster radius.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances

8.3.1.4 JCOUPLING

	This is part of the isdb module

Calculates ${}^3J$ coupling constants for a dihedral angle.

The J-coupling between two atoms is given by the Karplus relation:

$$ {}^3J(\theta) = A\cos^2(\theta + \Delta\theta) + B\cos(\theta + \Delta\theta) + C $$

where A, B and C are the Karplus parameters and $\Delta\theta$ is an additional constant added on to the dihedral angle θ. The Karplus parameters are determined empirically and are dependent on the type of J-coupling.

This collective variable computes the J-couplings for a set of atoms defining a dihedral angle. You can specify the atoms involved using the MOLINFO notation. You can also specify the experimental couplings using the COUPLING keywords. These will be included in the output. You must choose the type of coupling using the type keyword, you can also supply custom Karplus parameters using TYPE=CUSTOM and the A, B, C and SHIFT keywords. You will need to make sure you are using the correct dihedral angle:

- Ha-N: ψ
- Ha-HN: ϕ

- N-C γ: χ_1
- CO-C γ: χ_1

J-couplings can be used to calculate a Metainference score using the internal keyword DOSCORE and all the options of METAINFERENCE .

Examples

In the following example we calculate the Ha-N J-coupling from a set of atoms involved in dihedral ψ angles in the peptide backbone. We also add the experimental data points and compute the correlation and other measures and finally print the results.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/JCOUPLING.tmp
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO MOLTYPE=protein STRUCTURE=peptide.pdb
WHOLEMOLECULES ENTITY0=1-111

JCOUPLING ...
    TYPE=HAN
    ATOMS1=@psi-2 COUPLING1=-0.49
    ATOMS2=@psi-4 COUPLING2=-0.54
    ATOMS3=@psi-5 COUPLING3=-0.53
    ATOMS4=@psi-7 COUPLING4=-0.39
    ATOMS5=@psi-8 COUPLING5=-0.39
    LABEL=jhan
... JCOUPLING

jhanst: STATS ARG=(jhan\.j-.*) PARARG=(jhan\.exp-.*)

PRINT ARG=jhanst.*,jhan.* FILE=COLVAR STRIDE=100
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
j	the calculated J-coupling

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**COUPLING**	the experimental J-coupling

The atoms involved can be specified using

ATOMS	the 4 atoms involved in each of the bonds for which you wish to calculate the J-coupling. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one J-coupling will be calculated for each ATOMS keyword you specify.. You can use multiple instances of this keyword i.e. ATOMS1, ATO↩ MS2, ATOMS3...

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUT↩ LIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation
TYPE	Type of J-coupling to compute (HAN,HAHN,CCG,NCG,CUSTOM)

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging

REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)
A	Karplus parameter A
B	Karplus parameter B
C	Karplus parameter C
SHIFT	Angle shift in radians
COUPLING	Add an experimental value for each coupling. You can use multiple instances of this keyword i.e. COUPLING1, COUPLING2, COUPLING3...

8.3.1.5 NOE

This is part of the isdb module

Calculates NOE intensities as sums of 1/r^6, also averaging over multiple equivalent atoms or ambiguous NOE.

Each NOE is defined by two groups containing the same number of atoms, distances are calculated in pairs, transformed in 1/r^6, summed and saved as components.

$$NOE() = (\frac{1}{N_{eq}} \sum_{j}^{N_{eq}} (\frac{1}{r_j^6}))$$

NOE can be used to calculate a Metainference score over one or more replicas using the intrinsic implementation of METAINFERENCE that is activated by DOSCORE.

Examples

In the following examples three noes are defined, the first is calculated based on the distances of atom 1-2 and 3-2; the second is defined by the distance 5-7 and the third by the distances 4-15,4-16,8-15,8-16. METAINFERENCE is activated using DOSCORE.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/NOE.tmp
NOE ...
GROUPA1=1,3 GROUPB1=2,2 NOEDIST1=0.6
GROUPA2=5 GROUPB2=7 NOEDIST2=0.6
GROUPA3=4,4,8,8 GROUPB3=15,16,15,16 NOEDIST3=0.6
DOSCORE
SIGMA_MEAN0=1
LABEL=noes
... NOE

PRINT ARG=noes.* FILE=colvar
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
noe	the # NOE

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value

acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**NOEDIST**	the # NOE experimental distance

The atoms involved can be specified using

GROUPA	the atoms involved in each of the contacts you wish to calculate. Keywords like GROUPA1, GR↩ OUPA2, GROUPA3,... should be listed and one contact will be calculated for each ATOM keyword you specify.. You can use multiple instances of this keyword i.e. GROUPA1, GROUPA2, GROU↩ PA3...
GROUPB	the atoms involved in each of the contacts you wish to calculate. Keywords like GROUPB1, GR↩ OUPB2, GROUPB3,... should be listed and one contact will be calculated for each ATOM keyword you specify.. You can use multiple instances of this keyword i.e. GROUPB1, GROUPB2, GROU↩ PB3...

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUT↩ LIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging

REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)
NOEDIST	Add an experimental value for each NOE.. You can use multiple instances of this keyword i.e. NOEDIST1, NOEDIST2, NOEDIST3...

8.3.1.6 PCS

This is part of the isdb module

Calculates the Pseudo-contact shift of a nucleus determined by the presence of a metal ion susceptible to anisotropic magnetization.

The PCS of an atomic nucleus depends on the θ angle between the vector from the spin-label to the nucleus and the external magnetic field and the module of the vector itself [101] . While in principle the averaging resulting from the tumbling should remove the pseudo-contact shift, in presence of the NMR magnetic field the magnetically anisotropic molecule bound to system will break the rotational symmetry does resulting in measurable values for the PCS and RDC.

PCS values can also be calculated using a Single Value Decomposition approach, in this case the code rely on the a set of function from the GNU Scientific Library (GSL). (With SVD forces are not currently implemented).

Replica-Averaged simulations can be performed using PCS values, ENSEMBLE, STATS and RESTRAINT . Metainference simulations can be performed with this CV and METAINFERENCE .

Examples

In the following example five PCS values are defined and their correlation with respect to a set of experimental data is calculated and restrained. In addition, and only for analysis purposes, the same PCS values are calculated using a Single Value Decomposition algorithm.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PCS.tmp
PCS ...
ATOMS1=20,21
ATOMS2=20,38
ATOMS3=20,57
ATOMS4=20,77
ATOMS5=20,93
LABEL=nh
... PCS

enh: ENSEMBLE ARG=nh.*

st: STATS ARG=enh.* PARAMETERS=8.17,-8.271,-10.489,-9.871,-9.152

pcse: RESTRAINT ARG=st.corr KAPPA=0. SLOPE=-25000.0 AT=1.

PRINT ARG=st.corr,pcse.bias FILE=colvar
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
rdc	the calculated # RDC

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**SVD/COUPLING**	the experimental # RDC
Sxx	**SVD**	Tensor component
Syy	**SVD**	Tensor component
Szz	**SVD**	Tensor component
Sxy	**SVD**	Tensor component
Sxz	**SVD**	Tensor component
Syz	**SVD**	Tensor component

The atoms involved can be specified using

ATOMS	the couple of atoms involved in each of the bonds for which you wish to calculate the RDC. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one dipolar coupling will be calculated for each ATOMS keyword you specify.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUT↩ LIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation
GYROM	(default=1.) Add the product of the gyromagnetic constants for the bond.
SCALE	(default=1.) Add the scaling factor to take into account concentration and other effects.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging
REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SVD	(default=off) Set to TRUE if you want to back calculate using Single Value Decomposition (need GSL at compilation time).
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)
COUPLING	Add an experimental value for each coupling (needed by SVD and useful for STATS).. You can use multiple instances of this keyword i.e. COUPLING1, COUPLING2, COUPLING3...

8.3.1.7 PRE

	This is part of the isdb module

Calculates the Paramagnetic Resonance Enhancement intensity ratio between a spin label atom and a list of atoms .

The reference atom for the spin label is added with SPINLABEL, the affected atom(s) are give as numbered GR↩ OUPA1, GROUPA2, ... The additional parameters needed for the calculation are given as INEPT, the inept time, TAUC the correlation time, OMEGA, the Larmor frequency and RTWO for the relaxation time.

METAINFERENCE can be activated using DOSCORE and the other relevant keywords.

Examples

In the following example five PRE intensities are calculated using the distance between the oxygen of the spin label and the backbone hydrogen atoms. Omega is the NMR frequency, RTWO the R2 for the hydrogen atoms, INEPT of 8 ms for the experiment and a TAUC of 1.21 ns

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PRE.tmp
PRE ...
LABEL=HN_pre
INEPT=8
TAUC=1.21
OMEGA=900
SPINLABEL=1818
GROUPA1=86  RTWO1=0.0120272827
GROUPA2=177 RTWO2=0.0263953158
GROUPA3=285 RTWO3=0.0058899829
GROUPA4=335 RTWO4=0.0102072646
GROUPA5=451 RTWO5=0.0086341843
... PRE

PRINT ARG=HN_pre.* FILE=PRE.dat STRIDE=1
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
pre	the # PRE

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**PREINT**	the # PRE experimental intensity

The atoms involved can be specified using

SPINLABEL	The atom to be used as the paramagnetic center.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
GROUPA	the atoms involved in each of the contacts you wish to calculate. Keywords like GROUPA1, GROUPA2, GROUPA3,... should be listed and one contact will be calculated for each ATOM keyword you specify.. You can use multiple instances of this keyword i.e. GROUPA1, GROU↩ PA2, GROUPA3...

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUT↩ LIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation
INEPT	is the INEPT time (in ms).
TAUC	is the correlation time (in ns) for this electron-nuclear interaction.
OMEGA	is the Larmor frequency of the nuclear spin (in MHz).

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging
REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
NORATIO	(default=off) Set to TRUE if you want to compute PRE without Intensity Ratio
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)
RTWO	The relaxation of the atom/atoms in the corresponding GROUPA of atoms. Keywords like RTWO1, RTWO2, RTWO3,... should be listed.. You can use multiple instances of this keyword i.e. RTWO1, RTWO2, RTWO3...
PREINT	Add an experimental value for each PRE.. You can use multiple instances of this keyword i.e. PREINT1, PREINT2, PREINT3...

8.3.1.8 RDC

	This is part of the isdb module

Calculates the (Residual) Dipolar Coupling between two atoms.

The Dipolar Coupling between two nuclei depends on the θ angle between the inter-nuclear vector and the external magnetic field.

$$D = D_{max} 0.5(3\cos^2(\theta) - 1)$$

where

$$D_{max} = -\mu_0 \gamma_1 \gamma_2 h/(8\pi^3 r^3)$$

that is the maximal value of the dipolar coupling for the two nuclear spins with gyromagnetic ratio γ. μ is the magnetic constant and h is the Planck constant.

Common Gyromagnetic Ratios (C.G.S)

- H(1) 26.7513
- C(13) 6.7261
- N(15) -2.7116 and their products (this is what is given in input using the keyword GYROM)
- N-H -72.5388
- C-H 179.9319
- C-N -18.2385
- C-C 45.2404

In isotropic media DCs average to zero because of the rotational averaging, but when the rotational symmetry is broken, either through the introduction of an alignment medium or for molecules with highly anisotropic paramagnetic susceptibility, then the average of the DCs is not zero and it is possible to measure a Residual Dipolar Coupling (RDCs).

This collective variable calculates the Dipolar Coupling for a set of couple of atoms using the above definition.

In a standard MD simulation the average over time of the DC should then be zero. If one wants to model the meaning of a set of measured RDCs it is possible to try to solve the following problem: "what is the distribution of structures and orientations that reproduce the measured RDCs".

This collective variable can then be use to break the rotational symmetry of a simulation by imposing that the average of the DCs over the conformational ensemble must be equal to the measured RDCs [102] . Since measured RDCs are also a function of the fraction of aligned molecules in the sample it is better to compare them modulo a constant or looking at the correlation.

Alternatively if the molecule is rigid it is possible to use the experimental data to calculate the alignment tensor and the use that to back calculate the RDCs, this is what is usually call the Single Value Decomposition approach. In this case the code rely on the a set of function from the GNU Scientific Library (GSL). (With SVD forces are not currently implemented).

Replica-Averaged simulations can be performed using RDCs, ENSEMBLE, STATS and RESTRAINT . METAINFERENCE can be activated using DOSCORE and the other relevant keywords.

Additional material and examples can be also found in the tutorial ISDB: setting up a Metadynamics Metainference simulation

Examples

In the following example five N-H RDCs are defined and averaged over multiple replicas, their correlation is then calculated with respect to a set of experimental data and restrained. In addition, and only for analysis purposes, the same RDCs each single conformation are calculated using a Single Value Decomposition algorithm, then averaged and again compared with the experimental data.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RDC.tmp
#SETTINGS NREPLICAS=2
RDC ...
GYROM=-72.5388
SCALE=0.001
ATOMS1=20,21
ATOMS2=37,38
ATOMS3=56,57
ATOMS4=76,77
ATOMS5=92,93
LABEL=nh
... RDC

erdc: ENSEMBLE ARG=nh.*

st: STATS ARG=erdc.* PARAMETERS=8.17,-8.271,-10.489,-9.871,-9.152

rdce: RESTRAINT ARG=st.corr KAPPA=0. SLOPE=-25000.0 AT=1.

RDC ...
GYROM=-72.5388
SVD
ATOMS1=20,21 COUPLING1=8.17
ATOMS2=37,38 COUPLING2=-8.271
ATOMS3=56,57 COUPLING3=-10.489
ATOMS4=76,77 COUPLING4=-9.871
ATOMS5=92,93 COUPLING5=-9.152
LABEL=svd
... RDC

esvd: ENSEMBLE ARG=(svd\.rdc-.*)

st_svd: STATS ARG=esvd.* PARAMETERS=8.17,-8.271,-10.489,-9.871,-9.152

PRINT ARG=st.corr,st_svd.corr,rdce.bias FILE=colvar
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
rdc	the calculated # RDC

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**SVD/COUPLING**	the experimental # RDC
Sxx	**SVD**	Tensor component
Syy	**SVD**	Tensor component
Szz	**SVD**	Tensor component
Sxy	**SVD**	Tensor component
Sxz	**SVD**	Tensor component
Syz	**SVD**	Tensor component

The atoms involved can be specified using

ATOMS	the couple of atoms involved in each of the bonds for which you wish to calculate the RDC. Keywords like ATOMS1, ATOMS2, ATOMS3,... should be listed and one dipolar coupling will be calculated for each ATOMS keyword you specify.. You can use multiple instances of this keyword i.e. ATOMS1, ATOMS2, ATOMS3...

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUT↩ LIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation
GYROM	(default=1.) Add the product of the gyromagnetic constants for the bond.
SCALE	(default=1.) Add the scaling factor to take into account concentration and other effects.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging
REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SVD	(default=off) Set to TRUE if you want to back calculate using Single Value Decomposition (need GSL at compilation time).
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)
COUPLING	Add an experimental value for each coupling (needed by SVD and useful for STATS).. You can use multiple instances of this keyword i.e. COUPLING1, COUPLING2, COUPLING3...

8.3.1.9 SANS

	This is part of the isdb module

Calculates SANS intensity.

SANS intensities are calculated for a set of scattering vectors using QVALUE keywords numbered from 1. Form factors are automatically assigned to atoms using the ATOMISTIC flag by reading a PDB file or, alternatively, a ONEBEAD coarse-grained implementation is available.

Both for ATOMISTIC and ONEBEAD the user must provide an all-atom PDB file via MOLINFO before the SANS instruction.

ONEBEAD scheme consists in a single-bead per amino acid residue or three-bead for nucleic acid residue (one for the phosphate group, one for the pentose sugar, one for the nucleobase). PLUMED creates a virtual bead on which the SANS calculations are performed, centred on the COM of all atoms belonging to the bead. It is possible to account for the contribution of the solvation layer to the SAXS intensity by adding a correction term for the solvent accessible beads only: the form factors of the amino acids / phosphate groups / pentose sugars / nucleobases with a SASA (computed via LCPO algorithm) greater than a threshold are corrected according to an electron density term. Both the surface cut-off threshold and the electron density term can be set by the user with the SASA_↩ CUTOFF and SOLVATION_CORRECTION keywords. Moreover, SASA stride calculation can be modified using SOLVATION_STRIDE, which is set to 100 steps by default. The deuteration of the solvent-exposed residues is chosen with a probability equal to the deuterium concentration in the buffer. The deuterated residues are updated with a stride equal to SOLVATION_STRIDE. The fraction of deuterated water can be set with DEUTER_CONC, the default value is 0. ONEBEAD requires an additional PDB file to perform mapping conversion, which must be provided via TEMPLATE keyword. This PDB file should only include the atoms for which the SANS intensity will be computed. The AMBER OL3 (RNA) and OL15 (DNA) naming is required for nucleic acids. Two additional bead types are available for DNA and RNA besides phosphate group, pentose sugar, and nucleobase:

- 5'-end pentose sugar capped with an hydroxyl moiety at C5' (the residue name in the PDB must be followed by "5", e.g., DC5 or C5 for cytosine in DNA and RNA, respectively);
- 3'-end pentose sugar capped with an hydroxyl moiety at C3' (the residue name in the PDB must be followed by "3", e.g., DC3 or C3 for cytosine in DNA and RNA, respectively).

PLEASE NOTE: at the moment, we DO NOT explicitly take into account deuterated residues in the ATOMISTIC representation, but we correct the solvent contribution via the DEUTER_CONC keyword.

Experimental reference intensities can be added using the EXPINT keywords. All these values must be normalised to the SANS intensity at q = 0. To facilitate this operation, the SCALE_EXPINT keyword can be used to provide the intensity at q = 0. Each EXPINT is divided by SCALE_EXPINT. The maximum QVALUE for ONEBEAD is set to 0.3 inverse angstroms. The solvent density, that by default is set to 0.334 electrons per cubic angstrom (bulk water), can be modified using the SOLVDENS keyword.

By default SANS is calculated using Debye on CPU, by adding the GPU flag it is possible to solve the equation on a GPU if the ARRAYFIRE libraries are installed and correctly linked. METAINFERENCE can be activated using DOSCORE and the other relevant keywords.

Examples

in the following example the SANS intensities are calculated at atomistic resolution. The form factors are assigned according to the PDB file specified in the MOLINFO. Each experimental intensity is divided by 1.↩ 4002, which is the corresponding theoretical intensity value at q = 0. The deuterated water fraction is set to 48%.

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=template_AA.pdb

SANS ...
LABEL=SANS
ATOMS=1-355
ATOMISTIC
SCALE_EXPINT=1.4002
DEUTER_CONC=0.48
QVALUE1=0.03 EXPINT1=1.0902
QVALUE2=0.06 EXPINT2=0.790632
QVALUE3=0.09 EXPINT3=0.453808
QVALUE4=0.12 EXPINT4=0.254737
QVALUE5=0.15 EXPINT5=0.154928
QVALUE6=0.18 EXPINT6=0.0921503
QVALUE7=0.21 EXPINT7=0.052633
QVALUE8=0.24 EXPINT8=0.0276557
QVALUE9=0.27 EXPINT9=0.0122775
QVALUE10=0.30 EXPINT10=0.00880634
... SANS

PRINT ARG=(SANS\.q-.*),(SANS\.exp-.*) FILE=sansdata STRIDE=1
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
q	the # SAXS of q

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter

offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**EXPINT**	the # experimental intensity

The atoms involved can be specified using

ATOMS	The atoms to be included in the calculation, e.g. the whole protein. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUTLIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation
DEVICEID	(default=-1) Identifier of the GPU to be used
TEMPLATE	(default=template.pdb) A PDB file is required for ONEBEAD mapping
DEUTER_CONC	(default=0.) Fraction of deuterated solvent
SOLVDENS	(default=0.334) Density of the solvent to be used for the correction of atomistic form factors
SOLVATION_CORRECTION	(default=0.0) Hydration layer electron density correction (ONEBEAD only)
SASA_CUTOFF	(default=1.0) SASA value to consider a residue as exposed to the solvent (ONEBEAD only)
SOLVATION_STRIDE	(default=100) Number of steps between every new residues solvation estimation via LCPO (ONEBEAD only)
SCALE_EXPINT	(default=1.0) Scaling value for experimental data normalization

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging
REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
GPU	(default=off) calculate SAXS using ARRAYFIRE on an accelerator device
ATOMISTIC	(default=off) calculate SAXS for an atomistic model
MARTINI	(default=off) calculate SAXS for a Martini model
ONEBEAD	(default=off) calculate SAXS for a single bead model
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)

QVALUE	Selected scattering lengths in inverse angstroms are given as QVALUE1, Q↩ VALUE2, ... You can use multiple instances of this keyword i.e. QVALUE1, QVALUE2, QVALUE3...
PARAMETERS	Used parameter Keywords like PARAMETERS1, PARAMETERS2. These are used to calculate the form factor for the ith atom/bead. You can use multiple instances of this keyword i.e. PARAMETERS1, PARAMETERS2, PARAME↩ TERS3...
EXPINT	Add an experimental value for each q value. You can use multiple instances of this keyword i.e. EXPINT1, EXPINT2, EXPINT3...

8.3.1.10 SAXS

This is part of the isdb module

Calculates SAXS intensity.

SAXS intensities are calculated for a set of scattering vectors using QVALUE keywords numbered from 1. Form factors can be assigned either by polynomial expansion of any order by using the PARAMETERS keywords, or automatically matched to atoms using the ATOMISTIC flag by reading a PDB file. Alternatively to the atomistic representation, two types of coarse-grained mapping are available:

- MARTINI.
- ONEBEAD.

Whether for PARAMETERS, ATOMISTIC, and ONEBEAD the user must provide an all-atom PDB file via MOLIN↩ FO before the SAXS instruction. MARTINI requires a mapping scheme consisting of a PDB file that contains both the all-atom and MARTINI representations, and a bead position file (e.g., bead1: CENTER ATOMS=1,5,7,11,12 WEIGHTS=14,12,12, 12,16).

ONEBEAD scheme consists in a single-bead per amino acid residue or three-bead for nucleic acid residue (one for the phosphate group, one for the pentose sugar, one for the nucleobase). PLUMED creates a virtual bead on which the SAXS calculations are performed, centred on the COM of all atoms belonging to the bead. It is possible to account for the contribution of the solvation layer to the SAXS intensity by adding a correction term for the solvent accessible beads only: the form factors of the amino acids / phosphate groups / pentose sugars / nucleobases with a SASA (computed via LCPO algorithm) greater than a threshold are corrected according to an electron density term. Both the surface cut-off threshold and the electron density term can be set by the user with the SASA_↩ CUTOFF and SOLVATION_CORRECTION keywords. Moreover, SASA stride calculation can be modified using SOLVATION_STRIDE, which is set to 100 steps by default. ONEBEAD requires an additional PDB file to perform mapping conversion, which must be provided via TEMPLATE keyword. This PDB file should only include the atoms for which the SAXS intensity will be computed. The AMBER OL3 (RNA) and OL15 (DNA) naming is required for nucleic acids. Two additional bead types are available for DNA and RNA besides phosphate group, pentose sugar, and nucleobase:

- 5'-end pentose sugar capped with an hydroxyl moiety at C5' (the residue name in the PDB must be followed by "5", e.g., DC5 or C5 for cytosine in DNA and RNA, respectively);
- 3'-end pentose sugar capped with an hydroxyl moiety at C3' (the residue name in the PDB must be followed by "3", e.g., DC3 or C3 for cytosine in DNA and RNA, respectively).

Experimental reference intensities can be added using the EXPINT keywords. All these values must be normalised to the SAXS intensity at q = 0. To facilitate this operation, the SCALE_EXPINT keyword can be used to provide the intensity at q = 0. Each EXPINT is divided by SCALE_EXPINT. The maximum QVALUE for ONEBEAD is set to 0.3 inverse angstroms. The solvent density, that by default is set to 0.334 electrons per cubic angstrom (bulk water), can be modified using the SOLVDENS keyword.

By default SAXS is calculated using Debye on CPU, by adding the GPU flag it is possible to solve the equation on a GPU if the ARRAYFIRE libraries are installed and correctly linked. METAINFERENCE can be activated using DOSCORE and the other relevant keywords.

Examples

in the following example the SAXS intensities are calculated using single-bead per residue approximation, with a SASA threshold of 1 square nanometer and a solvation term of 0.04. Each experimental intensity is divided by 1.4002, which is the corresponding theoretical intensity value at q = 0. The form factors are selected according to the PDB file specified by TEMPLATE keyword.

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=template_AA.pdb

SAXS ...
LABEL=SAXS
ATOMS=1-355
ONEBEAD
TEMPLATE=template_AA.pdb
SOLVDENS=0.334
SOLVATION_CORRECTION=0.04
SOLVATION_STRIDE=1
SASA_CUTOFF=1.0
SCALE_EXPINT=1.4002
QVALUE1=0.03 EXPINT1=1.0902
QVALUE2=0.06 EXPINT2=0.790632
QVALUE3=0.09 EXPINT3=0.453808
QVALUE4=0.12 EXPINT4=0.254737
QVALUE5=0.15 EXPINT5=0.154928
QVALUE6=0.18 EXPINT6=0.0921503
QVALUE7=0.21 EXPINT7=0.052633
QVALUE8=0.24 EXPINT8=0.0276557
QVALUE9=0.27 EXPINT9=0.0122775
QVALUE10=0.30 EXPINT10=0.00880634
... SAXS

PRINT ARG=(SAXS\.q-.*),(SAXS\.exp-.*) FILE=saxsdata STRIDE=1
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
score	the Metainference score
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values
q	the # SAXS of q

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value

weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**EXPINT**	the # experimental intensity

The atoms involved can be specified using

ATOMS	The atoms to be included in the calculation, e.g. the whole protein. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUTLIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation
DEVICEID	(default=-1) Identifier of the GPU to be used
TEMPLATE	(default=template.pdb) A PDB file is required for ONEBEAD mapping
DEUTER_CONC	(default=0.) Fraction of deuterated solvent
SOLVDENS	(default=0.334) Density of the solvent to be used for the correction of atomistic form factors
SOLVATION_CORRECTION	(default=0.0) Hydration layer electron density correction (ONEBEAD only)
SASA_CUTOFF	(default=1.0) SASA value to consider a residue as exposed to the solvent (ONEBEAD only)
SOLVATION_STRIDE	(default=100) Number of steps between every new residues solvation estimation via LCPO (ONEBEAD only)
SCALE_EXPINT	(default=1.0) Scaling value for experimental data normalization

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DOSCORE	(default=off) activate metainference
NOENSEMBLE	(default=off) don't perform any replica-averaging
REWEIGHT	(default=off) simple REWEIGHT using the ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
GPU	(default=off) calculate SAXS using ARRAYFIRE on an accelerator device
ATOMISTIC	(default=off) calculate SAXS for an atomistic model
MARTINI	(default=off) calculate SAXS for a Martini model
ONEBEAD	(default=off) calculate SAXS for a single bead model
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]

RESTART	allows per-action setting of restart (YES/NO/AUTO)
QVALUE	Selected scattering lengths in inverse angstroms are given as QVALUE1, Q↩ VALUE2, ... You can use multiple instances of this keyword i.e. QVALUE1, QVALUE2, QVALUE3...
PARAMETERS	Used parameter Keywords like PARAMETERS1, PARAMETERS2. These are used to calculate the form factor for the ith atom/bead. You can use multiple instances of this keyword i.e. PARAMETERS1, PARAMETERS2, PARAME↩ TERS3...
EXPINT	Add an experimental value for each q value. You can use multiple instances of this keyword i.e. EXPINT1, EXPINT2, EXPINT3...

8.3.2 Functions Documentation

The following list contains descriptions of functions originally developed for the PLUMED-ISDB module. They can be used in combination with any other collective variable, function or bias also outside the ISDB module.

SELECT	Selects an argument based on the value of a SELECTOR.

8.3.2.1 SELECT

This is part of the isdb module

Selects an argument based on the value of a SELECTOR.

Examples

In this example we use a simulated-tempering like approach activated by the RESCALE action. For each value of the scale parameter, we perform an independent Parallel Bias Metadynamics simulation (see PBMETAD). At each moment of the simulation, only one of the PBMETAD actions is activated, based on the current value of the associated SELECTOR. The SELECT action can then be used to print out the value of the (active) PBMETAD bias potential.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SELECT.tmp
ene:  ENERGY
d: DISTANCE ATOMS=1,2

SELECTOR NAME=GAMMA VALUE=0

pbmetad0: PBMETAD ARG=d SELECTOR=GAMMA SELECTOR_ID=0 SIGMA=0.1 PACE=500 HEIGHT=1 BIASFACTOR=8 FILE=HILLS.0
pbmetad1: PBMETAD ARG=d SELECTOR=GAMMA SELECTOR_ID=1 SIGMA=0.1 PACE=500 HEIGHT=1 BIASFACTOR=8 FILE=HILLS.1

RESCALE ...
LABEL=res ARG=ene,pbmetad0.bias,pbmetad1.bias TEMP=300
SELECTOR=GAMMA MAX_RESCALE=1.2 NOT_RESCALED=2 NBIN=2
W0=1000 BIASFACTOR=100.0 BSTRIDE=2000 BFILE=bias.dat
...

pbactive: SELECT ARG=pbmetad0.bias,pbmetad1.bias SELECTOR=GAMMA

PRINT ARG=pbactive STRIDE=100 FILE=COLVAR
```

Glossary of keywords and components

Compulsory keywords

SELECTOR	name of the variable used to select

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

8.3.3 General Actions Documentation

The following list contains descriptions of actions originally developed for the PLUMED-ISDB module. They can be used in combination with any other collective variable, function or bias also outside the ISDB module.

Using SELECTOR it is possible to define a variable inside the PLUMED code that can be used and modified by other actions. For example, a SELECTOR can be used in combination with RESCALE to activate a simulated-tempering like approach.

SELECTOR	Defines a variable (of the type double) inside the PLUMED code that can be used and modified by other actions.

8.3.3.1 SELECTOR

	This is part of the isdb module

Defines a variable (of the type double) inside the PLUMED code that can be used and modified by other actions.

A SELECTOR can be used for example to activate or modify a bias based on its current value.

Examples

A typical example is the simulated-tempering like approach activated by RESCALE. In this example the total potential energy of the system is scaled by a parameter defined on a grid of dimension NBIN in the range from 1 to MAX_RESCALE. The value of the scaling parameter is determined by the current value of the SELECTOR GAMMA. The value of the SELECTOR is updated by a MC protocol inside the RESCALE class. A well-tempered metadynamics potential is used to enhance sampling in the SELECTOR space.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SELECTOR.tmp
ene:  ENERGY

SELECTOR NAME=GAMMA VALUE=0

RESCALE ...
LABEL=res ARG=ene TEMP=300
SELECTOR=GAMMA MAX_RESCALE=1.2 NBIN=2
W0=1000 BIASFACTOR=100.0 BSTRIDE=2000 BFILE=bias.dat
...

PRINT FILE=COLVAR ARG=* STRIDE=100
```

Glossary of keywords and components

Compulsory keywords

NAME	name of the SELECTOR
VALUE	set (initial) value of the SELECTOR

8.3.4 Biases Documentation

The following list contains descriptions of biases originally developed for the PLUMED-ISDB module. They can be used in combination with any other collective variable, function or bias also outside the ISDB module.

CALIBER	Add a time-dependent, harmonic restraint on one or more variables.
METAINFERENCE	Calculates the Metainference energy for a set of experimental data.
RESCALE	Scales the value of an another action, being a Collective Variable or a Bias.

8.3.4.1 CALIBER

	This is part of the isdb module

Add a time-dependent, harmonic restraint on one or more variables.

This allows implementing a maximum caliber restraint on one or more experimental time series by replica-averaged restrained simulations. See [103] .

The time resolved experiments are read from a text file and intermediate values are obtained by splines.

Examples

In the following example a restraint is applied on the time evolution of a saxs spectrum

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=first.pdb

# Define saxs variable
SAXS ...
LABEL=saxs
ATOMISTIC
ATOMS=1-436
QVALUE1=0.02 # Q-value at which calculate the scattering
QVALUE2=0.0808
QVALUE3=0.1264
QVALUE4=0.1568
QVALUE5=0.172
QVALUE6=0.1872
QVALUE7=0.2176
QVALUE8=0.2328
QVALUE9=0.248
QVALUE10=0.2632
QVALUE11=0.2936
QVALUE12=0.3088
QVALUE13=0.324
QVALUE14=0.3544
QVALUE15=0.4
... SAXS


#define the caliber restraint
CALIBER ...
  ARG=(saxs\.q_.*)
  FILE=expsaxs.dat
  KAPPA=10
  LABEL=cal0
  STRIDE=10
  REGRES_ZERO=200
  AVERAGING=200
... CALIBER
```

In particular the file expsaxs.dat contains the time traces for the 15 intensities at the selected scattering lengths, organized as time, q_1, etc. The strength of the bias is automatically evaluated from the standard error of the mean over AVERAGING steps and multiplied by KAPPA. This is useful when working with multiple experimental data Because SAXS is usually defined in a manner that is irrespective of a scaling factor the scaling is evaluated from a linear fit every REGRES_ZERO step. Alternatively it can be given as a fixed constant as SCALE. The bias is here applied every tenth step.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
x0	the instantaneous value of the center of the potential
mean	the current average value of the calculated observable
kappa	the current force constant

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
scale	**REGRES_ZERO**	the current scaling constant

Compulsory keywords

FILE	the name of the file containing the time-resolved values
KAPPA	a force constant, this can be use to scale a constant estimated on-the-fly using AVERAGING
TSCALE	(default=1.0) Apply a time scaling on the experimental time scale
SCALE	(default=1.0) Apply a constant scaling on the data provided as arguments

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOENSEMBLE	(default=off) don't perform any replica-averaging
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
AVERAGING	Stride for calculation of the optimum kappa, if 0 only KAPPA is used.
REGRES_ZERO	stride for regression with zero offset

8.3.4.2 METAINFERENCE

This is part of the isdb module

Calculates the Metainference energy for a set of experimental data.

Metainference [100] is a Bayesian framework to model heterogeneous systems by integrating prior information with noisy, ensemble-averaged data. Metainference models a system and quantifies the level of noise in the data by considering a set of replicas of the system.

Calculated experimental data are given in input as ARG while reference experimental values can be given either from fixed components of other actions using PARARG or as numbers using PARAMETERS. The default behavior

is that of averaging the data over the available replicas, if this is not wanted the keyword NOENSEMBLE prevent this averaging.

Metadynamics Metainference [104] or more in general biased Metainference requires the knowledge of biasing potential in order to calculate the weighted average. In this case the value of the bias can be provided as the last argument in ARG and adding the keyword REWEIGHT. To avoid the noise resulting from the instantaneous value of the bias the weight of each replica can be averaged over a give time using the keyword AVERAGING.

The data can be averaged by using multiple replicas and weighted for a bias if present. The functional form of Metainference can be chosen among four variants selected with NOISE=GAUSS,MGAUSS,OUTLIERS,MOUTL↩ IERS,GENERIC which correspond to modelling the noise for the arguments as a single gaussian common to all the data points, a gaussian per data point, a single long-tailed gaussian common to all the data points, a log-tailed gaussian per data point or using two distinct noises as for the most general formulation of Metainference. In this latter case the noise of the replica-averaging is gaussian (one per data point) and the noise for the comparison with the experimental data can chosen using the keyword LIKELIHOOD between gaussian or log-normal (one per data point), furthermore the evolution of the estimated average over an infinite number of replicas is driven by DFTILDE.

As for Metainference theory there are two sigma values: SIGMA_MEAN0 represent the error of calculating an average quantity using a finite set of replica and should be set as small as possible following the guidelines for replica-averaged simulations in the framework of the Maximum Entropy Principle. Alternatively, this can be obtained automatically using the internal sigma mean optimization as introduced in [105] (OPTSIGMAMEAN=SEM), in this second case sigma_mean is estimated from the maximum standard error of the mean either over the simulation or over a defined time using the keyword AVERAGING. SIGMA_BIAS is an uncertainty parameter, sampled by a MC algorithm in the bounded interval defined by SIGMA_MIN and SIGMA_MAX. The initial value is set at SIGMA0. The MC move is a random displacement of maximum value equal to DSIGMA. If the number of data point is too large and the acceptance rate drops it is possible to make the MC move over mutually exclusive, random subset of size MC↩ _CHUNKSIZE and run more than one move setting MC_STEPS in such a way that MC_CHUNKSIZE∗MC_STEPS will cover all the data points.

Calculated and experimental data can be compared modulo a scaling factor and/or an offset using SCALEDATA and/or ADDOFFSET, the sampling is obtained by a MC algorithm either using a flat or a gaussian prior setting it with SCALE_PRIOR or OFFSET_PRIOR.

Examples

In the following example we calculate a set of RDC, take the replica-average of them and comparing them with a set of experimental values. RDCs are compared with the experimental data but for a multiplication factor SCALE that is also sampled by MC on-the-fly

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAINFERENCE.tmp
RDC ...
LABEL=rdc
SCALE=0.0001
GYROM=-72.5388
ATOMS1=22,23
ATOMS2=25,27
ATOMS3=29,31
ATOMS4=33,34
... RDC

METAINFERENCE ...
ARG=rdc.*
NOISETYPE=MGAUSS
PARAMETERS=1.9190,2.9190,3.9190,4.9190
SCALEDATA SCALE0=1 SCALE_MIN=0.1 SCALE_MAX=3 DSCALE=0.01
SIGMA0=0.01 SIGMA_MIN=0.00001 SIGMA_MAX=3 DSIGMA=0.01
SIGMA_MEAN0=0.001
LABEL=spe
... METAINFERENCE

PRINT ARG=spe.bias FILE=BIAS STRIDE=1
```

in the following example instead of using one uncertainty parameter per data point we use a single uncertainty value in a long-tailed gaussian to take into account for outliers, furthermore the data are weighted for the bias applied to other variables of the system.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/METAINFERENCE.tmp
RDC ...
LABEL=rdc
SCALE=0.0001
GYROM=-72.5388
ATOMS1=22,23
ATOMS2=25,27
ATOMS3=29,31
ATOMS4=33,34
... RDC

cv1: TORSION ATOMS=1,2,3,4
cv2: TORSION ATOMS=2,3,4,5
mm: METAD ARG=cv1,cv2 HEIGHT=0.5 SIGMA=0.3,0.3 PACE=200 BIASFACTOR=8 WALKERS_MPI

METAINFERENCE ...
#SETTINGS NREPLICAS=2
ARG=rdc.*,mm.bias
REWEIGHT
NOISETYPE=OUTLIERS
PARAMETERS=1.9190,2.9190,3.9190,4.9190
SCALEDATA SCALE0=1 SCALE_MIN=0.1 SCALE_MAX=3 DSCALE=0.01
SIGMA0=0.01 SIGMA_MIN=0.00001 SIGMA_MAX=3 DSIGMA=0.01
SIGMA_MEAN0=0.001
LABEL=spe
... METAINFERENCE
```

(See also RDC, PBMETAD).

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
neff	effective number of replicas
acceptSigma	MC acceptance for sigma values

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance for scale value
acceptFT	**GENERIC**	MC acceptance for general metainference f tilde value
weight	**REWEIGHT**	weights of the weighted average

biasDer	**REWEIGHT**	derivatives with respect to the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator

Compulsory keywords

NOISETYPE	(default=MGAUSS) functional form of the noise (GAUSS,MGAUSS,OUTLIERS,MOUT↩ LIERS,GENERIC)
LIKELIHOOD	(default=GAUSS) the likelihood for the GENERIC metainference model, GAUSS or LOGN
DFTILDE	(default=0.1) fraction of sigma_mean used to evolve ftilde
SCALE0	(default=1.0) initial value of the scaling factor
SCALE_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
OFFSET0	(default=0.0) initial value of the offset
OFFSET_PRIOR	(default=FLAT) either FLAT or GAUSSIAN
SIGMA0	(default=1.0) initial value of the uncertainty parameter
SIGMA_MIN	(default=0.0) minimum value of the uncertainty parameter
SIGMA_MAX	(default=10.) maximum value of the uncertainty parameter
OPTSIGMAMEAN	(default=NONE) Set to NONE/SEM to manually set sigma mean, or to estimate it on the fly
WRITE_STRIDE	(default=10000) write the status to a file every N steps, this can be used for restart/continuation

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOENSEMBLE	(default=off) don't perform any replica-averaging
REWEIGHT	(default=off) simple REWEIGHT using the latest ARG as energy
SCALEDATA	(default=off) Set to TRUE if you want to sample a scaling factor common to all values and replicas
ADDOFFSET	(default=off) Set to TRUE if you want to sample an offset common to all values and replicas
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

PARARG	reference values for the experimental data, these can be provided as arguments without derivatives
PARAMETERS	reference values for the experimental data
AVERAGING	Stride for calculation of averaged weights and sigma_mean
SCALE_MIN	minimum value of the scaling factor
SCALE_MAX	maximum value of the scaling factor
DSCALE	maximum MC move of the scaling factor
OFFSET_MIN	minimum value of the offset
OFFSET_MAX	maximum value of the offset
DOFFSET	maximum MC move of the offset
REGRES_ZERO	stride for regression with zero offset
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
SIGMA_MAX_STEPS	Number of steps used to optimise SIGMA_MAX, before that the SIGMA_MAX value is used
TEMP	the system temperature - this is only needed if code doesn't pass the temperature to plumed
MC_STEPS	number of MC steps
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data useful for restart/continuation of Metainference
SELECTOR	name of selector
NSELECT	range of values for selector [0, N-1]
RESTART	allows per-action setting of restart (YES/NO/AUTO)

8.3.4.3 RESCALE

	This is part of the isdb module

Scales the value of an another action, being a Collective Variable or a Bias.

The rescaling factor is determined by a parameter defined on a logarithmic grid of dimension NBIN in the range from 1 to MAX_RESCALE. The current value of the rescaling parameter is stored and shared across other actions using a SELECTOR. A Monte Carlo procedure is used to update the value of the rescaling factor every MC_STRIDE steps of molecular dynamics. Well-tempered metadynamics, defined by the parameters W0 and BIASFACTOR, is used to enhance the sampling in the space of the rescaling factor. The well-tempered metadynamics bias potential is written to the file BFILE every BSTRIDE steps and read when restarting the simulation using the directive RESTART.

Note

Additional arguments not to be scaled, one for each bin in the rescaling parameter ladder, can be provided at the end of the ARG list. The number of such arguments is specified by the option NOT_RESCALED. These arguments will be not be scaled, but they will be considered as bias potentials and used in the computation of the Metropolis acceptance probability when proposing a move in the rescaling parameter. See example below.

If PLUMED is running in a multiple-replica framework (for example using the -multi option in GROMACS), the arguments will be summed across replicas, unless the NOT_SHARED option is used. Also, the value of the SELECTOR will be shared and thus will be the same in all replicas.

Examples

In this example we use RESCALE to implement a simulated-tempering like approach. The total potential energy of the system is scaled by a parameter defined on a logarithmic grid of 5 bins in the range from 1 to 1.5. A well-tempered metadynamics bias potential is used to ensure diffusion in the space of the rescaling parameter.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESCALE.tmp
ene: ENERGY

SELECTOR NAME=GAMMA VALUE=0

RESCALE ...
LABEL=res ARG=ene TEMP=300
SELECTOR=GAMMA MAX_RESCALE=1.5 NBIN=5
W0=1000 BIASFACTOR=100.0 BSTRIDE=2000 BFILE=bias.dat
...

PRINT FILE=COLVAR ARG=* STRIDE=100
```

In this second example, we add to the simulated-tempering approach introduced above one Parallel Bias metadynamics simulation (see PBMETAD) for each value of the rescaling parameter. At each moment of the simulation, only one of the PBMETAD actions is activated, based on the current value of the associated SELECTOR. The PBMETAD bias potentials are not scaled, but just used in the calculation of the Metropolis acceptance probability when proposing a move in the rescaling parameter.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RESCALE.tmp
ene: ENERGY
d: DISTANCE ATOMS=1,2

SELECTOR NAME=GAMMA VALUE=0

pbmetad0: PBMETAD ARG=d SELECTOR=GAMMA SELECTOR_ID=0 SIGMA=0.1 PACE=500 HEIGHT=1 BIASFACTOR=8 FILE=HILLS.0
pbmetad1: PBMETAD ARG=d SELECTOR=GAMMA SELECTOR_ID=1 SIGMA=0.1 PACE=500 HEIGHT=1 BIASFACTOR=8 FILE=HILLS.1
pbmetad2: PBMETAD ARG=d SELECTOR=GAMMA SELECTOR_ID=2 SIGMA=0.1 PACE=500 HEIGHT=1 BIASFACTOR=8 FILE=HILLS.2
pbmetad3: PBMETAD ARG=d SELECTOR=GAMMA SELECTOR_ID=3 SIGMA=0.1 PACE=500 HEIGHT=1 BIASFACTOR=8 FILE=HILLS.3
pbmetad4: PBMETAD ARG=d SELECTOR=GAMMA SELECTOR_ID=4 SIGMA=0.1 PACE=500 HEIGHT=1 BIASFACTOR=8 FILE=HILLS.4

RESCALE ...
LABEL=res TEMP=300
ARG=ene,pbmetad0.bias,pbmetad1.bias,pbmetad2.bias,pbmetad3.bias,pbmetad4.bias
SELECTOR=GAMMA MAX_RESCALE=1.5 NOT_RESCALED=5 NBIN=5
W0=1000 BIASFACTOR=100.0 BSTRIDE=2000 BFILE=bias.dat
...

PRINT FILE=COLVAR ARG=* STRIDE=100
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
igamma	gamma parameter
accgamma	MC acceptance for gamma
wtbias	well-tempered bias

Compulsory keywords

TEMP	temperature
SELECTOR	name of the SELECTOR used for rescaling
MAX_RESCALE	maximum values for rescaling
NBIN	number of bins for gamma grid
W0	initial bias height
BIASFACTOR	bias factor
BSTRIDE	stride for writing bias
BFILE	file name for bias

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
NOT_SHARED	list of arguments (from 1 to N) not summed across replicas
NOT_RESCALED	these last N arguments will not be scaled
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
PACE	Pace for adding bias, in MC stride unit

8.3.5 Tutorials

The following are tutorials meant to learn how to use the different methods implemented in the ISDB module.

ISDB: setting up a Metadynamics Metainference simulation	This tutorial show an example on how to use PLUMED-ISDB to run Metadynamics Metainference
ISDB: setting up a SAXS post processing and refinement calculation	This tutorial show an example on how to use PLUMED-ISDB to run SAXS analysis and refinement calculation

8.3.5.1 ISDB: setting up a Metadynamics Metainference simulation

8.3.5.1.1 Aims The aim of this tutorial is to introduce the users to the ISDB module and in particular to Metadynamics Metainference [100] [104] ensemble determination. We will reproduce the setup of the simulation for a simple system [105] . For a general overview of the problem of ensembles determination please read [106] .

8.3.5.1.2 Objectives Once this tutorial is completed students will be able to:

- Setup their own PLUMED-ISDB simulation.

8.3.5.1.3 Resources The TARBALL for this project contains the following files:

- charmm36-eef1sb.ff: the force-field files for gromacs (not needed)
- system: a folder with reference files for gromacs (not needed)
- reference-impl: a folder to perform a simple implicit solvent simulation
- reference-impl-pbmetad: a folder to perform a pbmetad implicit solvent simulation
- m_and_m: a folder to perform a metadynamics metainference simulation

8.3.5.1.4 Introduction Molecular dynamics simulations are the ideal tool to determine at atomistic resolution the behavior of complex molecules. This great resolution power comes at the cost of approximations that affects the agreement with actual experimental observables. At the same time experimental data alone are generally speaking not enough to determine a structural ensemble due the inverse nature of the problem, that is to go from few observables to many atoms in many different configurations. Furthermore, experimental data are affected by errors of multiple nature, from noise, systematic errors and errors in their atomistic interpretation. Most important experimental data are the result of the averaging over the ensemble of structure so it is not trivial to deconvolve this signal. One possibility is that of employing MD simulations together with experimental data to generate simulations already corrected for the data themselves. With METAINFERENCE this is done on-the-fly by adding an additional energy to the system that takes into account the agreement with the experimental data considering the multiple sources of errors.

8.3.5.1.5 Run a reference simulation The system we use is the EGAAWAASS peptide used in ref. [105] . First of all we will run a simulation in implicit solvent using the EEF1-SB CHARMM36 force field. EEF1-SB includes a correction to the standard backbone torsion potential of CHARMM36, an electrostatic interaction with a distance dependent dielectric constant and a simple gaussian form for the solvation energy. The first two terms are implemented in the force field and using table potentials while the latter is implemented as a collective variable in PLUMED, EEFSOLV .

```
BEGIN_PLUMED_FILE working DATADIR=example-check/isdb-1.txt
# this is optional and tell to VIM that this is a PLUMED file
# vim: ft=plumed
# see comments just below this input file
#SETTINGS MOLFILE=user-doc/tutorials/others/isdb-1/reference-impl/egaawaass.pdb
MOLINFO MOLTYPE=protein STRUCTURE=egaawaass.pdb
WHOLEMOLECULES ENTITY0=1-111

# EEF1SB Implicit solvation
#SETTINGS AUXFILE=user-doc/tutorials/others/isdb-1/reference-impl/index.ndx
protein-h: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein-H
solv: EEFSOLV ATOMS=protein-h NOPBC NL_STRIDE=20 NL_BUFFER=0.1
bias: BIASVALUE ARG=solv
```

This can be run using gromacs (unfortunately recent versions of gromacs do not support Verlet groups with table potentials, so performances are currently sub-optimal on the gromacs side)

```
gmx_mpi mdrun -s run.tpr -table table.xvg -tablep table.xvg -plumed plumed-eef1.dat -v
```

In order to have a converged sampling for this reference ensemble calculation it is useful to setup a Metadynamics calculation. In particular we will use PBMETAD because it is then a natural choice for Metadynamics Metainference later.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/isdb-1.txt
# this is optional and tell to VIM that this is a PLUMED file
# vim: ft=plumed
# see comments just below this input file
#SETTINGS MOLFILE=user-doc/tutorials/others/isdb-1/reference-impl/egaawaass.pdb
MOLINFO MOLTYPE=protein STRUCTURE=egaawaass.pdb
WHOLEMOLECULES ENTITY0=1-111

# EEF1SB Implicit solvation
#SETTINGS AUXFILE=user-doc/tutorials/others/isdb-1/reference-impl/index.ndx
protein-h: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein-H
solv: EEFSOLV ATOMS=protein-h NOPBC NL_STRIDE=20 NL_BUFFER=0.1
bias: BIASVALUE ARG=solv

# CVs, Psi9, Phi1 are not defined
psi1: TORSION ATOMS=@psi-1 NOPBC
psi2: TORSION ATOMS=@psi-2 NOPBC
psi3: TORSION ATOMS=@psi-3 NOPBC
psi4: TORSION ATOMS=@psi-4 NOPBC
psi5: TORSION ATOMS=@psi-5 NOPBC
psi6: TORSION ATOMS=@psi-6 NOPBC
psi7: TORSION ATOMS=@psi-7 NOPBC
psi8: TORSION ATOMS=@psi-8 NOPBC

phi2: TORSION ATOMS=@phi-2 NOPBC
phi3: TORSION ATOMS=@phi-3 NOPBC
phi4: TORSION ATOMS=@phi-4 NOPBC
phi5: TORSION ATOMS=@phi-5 NOPBC
phi6: TORSION ATOMS=@phi-6 NOPBC
phi7: TORSION ATOMS=@phi-7 NOPBC
phi8: TORSION ATOMS=@phi-8 NOPBC
phi9: TORSION ATOMS=@phi-9 NOPBC

ahc:  ALPHARMSD RESIDUES=all TYPE=OPTIMAL LESS_THAN={RATIONAL R_0=0.12}

# Bulky Trp residue dihedral
dihtrp_cacb: TORSION ATOMS=67,47,49,52 NOPBC
dihtrp_cbcg: TORSION ATOMS=47,49,52,53 NOPBC

protein-ca: GROUP NDX_FILE=index.ndx NDX_GROUP=C-alpha
gyr: GYRATION TYPE=RADIUS ATOMS=protein-ca NOPBC

# PBMetaD
PBMETAD ...
    LABEL=pb
    ARG=phi2,phi3,phi4,phi5,phi6,phi7,phi8,phi9,psi1,psi2,psi3,psi4,psi5,psi6,psi7,psi8,dihtrp_cacb,dihtrp_cbc
    SIGMA=0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.01
    HEIGHT=0.5
    PACE=400
    BIASFACTOR=20
    GRID_MIN=-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,0
    GRID_MAX=pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,5
    GRID_WSTRIDE=5000
    WALKERS_MPI
... PBMETAD

PRINT FILE=COLVAR ARG=phi2,phi3,phi4,phi5,phi6,phi7,phi8,phi9,psi1,psi2,psi3,psi4,psi5,psi6,psi7,psi8,dihtrp_c
PRINT FILE=ENERGY ARG=bias.bias,pb.bias STRIDE=200
```

In this case we are running a multiple-replica simulation where the sampling is used to parallelize the Metadynamics time-dependent potential through the use of multiple walkers.

```
mpiexec -np 14 gmx_mpi mdrun -s topolnew -multi 14 -plumed plumed-eef1-pbmetad.dat -table table.xvg -tablep ta
```

8.3.5.1.6 Metadynamics Metainference The former simulations should provide a converged (check for this) ensemble for the peptide. As shown in [105] the agreement with the multiple available NMR experimental data is not perfect. In order to generate an ensemble compatible with most of the available experimental data it is possible to include them in the simulation using METAINFERENCE . To do so the forward models for the data sets should be defined in the input file. In this case we have backbone chemical shifts, CS2BACKBONE ; residual dipolar couplings for two bonds, RDC ; and J-couplings for multiple atoms, JCOUPLING. Once the forward models are defined for the data sets, the calculated data together with the corresponding experimental values can be used to calculate the metainference score. The metainference score is additive so it can be split into multiple METAINFERENCE entries. In this case we are using two metainference entries for the two sets of RDCs because these are compared with the experimental data modulo a constant that should be unique each data set. Then we use one metainference for all the jcouplings and another one for the chemical shifts.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/isdb-1.txt
# this is optional and tell to VIM that this is a PLUMED file
# vim: ft=plumed
# see comments just below this input file
#SETTINGS NREPLICAS=2
#SETTINGS MOLFILE=user-doc/tutorials/others/isdb-1/reference-impl/egaawaass.pdb
MOLINFO MOLTYPE=protein STRUCTURE=egaawaass.pdb
WHOLEMOLECULES ENTITY0=1-111

# EEF1SB Implicit solvation
#SETTINGS AUXFILE=user-doc/tutorials/others/isdb-1/reference-impl/index.ndx
protein-h: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein-H
solv: EEFSOLV ATOMS=protein-h NOPBC NL_STRIDE=20 NL_BUFFER=0.1
bias: BIASVALUE ARG=solv

# CVs, Psi9, Phi1 are not defined
psi1: TORSION ATOMS=@psi-1 NOPBC
psi2: TORSION ATOMS=@psi-2 NOPBC
psi3: TORSION ATOMS=@psi-3 NOPBC
psi4: TORSION ATOMS=@psi-4 NOPBC
psi5: TORSION ATOMS=@psi-5 NOPBC
psi6: TORSION ATOMS=@psi-6 NOPBC
psi7: TORSION ATOMS=@psi-7 NOPBC
psi8: TORSION ATOMS=@psi-8 NOPBC

phi2: TORSION ATOMS=@phi-2 NOPBC
phi3: TORSION ATOMS=@phi-3 NOPBC
phi4: TORSION ATOMS=@phi-4 NOPBC
phi5: TORSION ATOMS=@phi-5 NOPBC
phi6: TORSION ATOMS=@phi-6 NOPBC
phi7: TORSION ATOMS=@phi-7 NOPBC
phi8: TORSION ATOMS=@phi-8 NOPBC
phi9: TORSION ATOMS=@phi-9 NOPBC

ahc:  ALPHARMSD RESIDUES=all TYPE=OPTIMAL LESS_THAN={RATIONAL R_0=0.12}

# Bulky Trp residue dihedral
dihtrp_cacb: TORSION ATOMS=67,47,49,52 NOPBC
dihtrp_cbcg: TORSION ATOMS=47,49,52,53 NOPBC

protein-ca: GROUP NDX_FILE=index.ndx NDX_GROUP=C-alpha
gyr: GYRATION TYPE=RADIUS ATOMS=protein-ca NOPBC

# PBMetaD
PBMETAD ...
    LABEL=pb
    ARG=phi2,phi3,phi4,phi5,phi6,phi7,phi8,phi9,psi1,psi2,psi3,psi4,psi5,psi6,psi7,psi8,dihtrp_cacb,dihtrp_cbc
    SIGMA=0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.01
    HEIGHT=0.5
    PACE=400
    BIASFACTOR=20
    GRID_MIN=-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,-pi,0
    GRID_MAX=pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,pi,5
    GRID_WSTRIDE=5000
    WALKERS_MPI
... PBMETAD

PRINT FILE=COLVAR ARG=phi2,phi3,phi4,phi5,phi6,phi7,phi8,phi9,psi1,psi2,psi3,psi4,psi5,psi6,psi7,psi8,dihtrp_c
```

```
PRINT FILE=ENERGY ARG=bias.bias,pb.bias STRIDE=200

# EXPERIMENTAL DATA SECTION

# RDCs (Grzesiek et al.)
# xGAAWAASS
RDC ...
    GYROM=-72.5388
    SCALE=0.0001
    NOPBC
    ATOMS1=18,19 COUPLING1=-5.4
    ATOMS2=25,26 COUPLING2=-1.26
    ATOMS3=35,36 COUPLING3=-5.22
    ATOMS4=45,46 COUPLING4=-0.91
    ATOMS5=69,70 COUPLING5=2.33
    ATOMS6=79,80 COUPLING6=-2.88
    ATOMS7=89,90 COUPLING7=-8.37
    ATOMS8=100,101 COUPLING8=-3.78
    LABEL=nh
... RDC

# ExAAWAASx
RDC ...
    GYROM=179.9319
    SCALE=0.0001
    NOPBC
    ATOMS1=5,6 COUPLING1=12.95
    ATOMS2=27,28 COUPLING2=11.5
    ATOMS3=37,38 COUPLING3=21.42
    ATOMS4=47,48 COUPLING4=-9.37
    ATOMS5=71,72 COUPLING5=10.01
    ATOMS6=81,82 COUPLING6=15.01
    ATOMS7=91,92 COUPLING7=15.73
    LABEL=caha
... RDC

#RDCS
METAINFERENCE ...
    ARG=(nh\.rdc-.*),pb.bias
    PARARG=(nh\.exp-.*)
    REWEIGHT
    NOISETYPE=MGAUSS
    OPTSIGMAMEAN=SEM_MAX AVERAGING=400
    SCALEDATA SCALE_PRIOR=GAUSSIAN SCALE0=8.0 DSCALE=0.5
    SIGMA0=5.0 SIGMA_MIN=0.0001 SIGMA_MAX=15.0 DSIGMA=0.1
    WRITE_STRIDE=10000
    LABEL=byrdcnh
... METAINFERENCE

#RDCS
METAINFERENCE ...
    ARG=(caha\.rdc-.*),pb.bias
    PARARG=(caha\.exp-.*)
    REWEIGHT
    NOISETYPE=MGAUSS
    OPTSIGMAMEAN=SEM_MAX AVERAGING=400
    SCALEDATA SCALE_PRIOR=GAUSSIAN SCALE0=9.0 DSCALE=0.5
    SIGMA0=5.0 SIGMA_MIN=0.0001 SIGMA_MAX=15.0 DSIGMA=0.1
    WRITE_STRIDE=10000
    LABEL=byrdccaha
... METAINFERENCE

# xGxAWxASx
JCOUPLING ...
    TYPE=HAN
    NOPBC
    ATOMS1=@psi-2 COUPLING1=-0.49
    ATOMS2=@psi-4 COUPLING2=-0.54
    ATOMS3=@psi-5 COUPLING3=-0.53
    ATOMS4=@psi-7 COUPLING4=-0.39
    ATOMS5=@psi-8 COUPLING5=-0.39
    LABEL=jhan
... JCOUPLING
```

```
# xxAAWAASS
JCOUPLING ...
    TYPE=HAHN
    NOPBC
    ATOMS1=@phi-2 COUPLING1=6.05
    ATOMS2=@phi-3 COUPLING2=5.95
    ATOMS3=@phi-4 COUPLING3=6.44
    ATOMS4=@phi-5 COUPLING4=6.53
    ATOMS5=@phi-6 COUPLING5=5.93
    ATOMS6=@phi-7 COUPLING6=6.98
    ATOMS7=@phi-8 COUPLING7=7.16
    LABEL=jhahn
... JCOUPLING

# xxxxWxxxx
JCOUPLING ...
    TYPE=CCG
    NOPBC
    ATOMS1=@chi1-5 COUPLING1=1.59
    LABEL=jccg
... JCOUPLING

# xxxxWxxxx
JCOUPLING ...
    TYPE=NCG
    NOPBC
    ATOMS1=@chi1-5 COUPLING1=1.21
    LABEL=jncg
... JCOUPLING

#JC METAINFERENCE
METAINFERENCE ...
    ARG=(jhan\.j-.*),(jhahn\.j-.*),(jccg\.j.*),(jncg\.j.*),pb.bias
    PARARG=(jhan\.exp-.*),(jhahn\.exp-.*),(jccg\.exp.*),(jncg\.exp.*)
    REWEIGHT
    NOISETYPE=MGAUSS
    OPTSIGMAMEAN=SEM_MAX AVERAGING=400
    SIGMA0=5.0 SIGMA_MIN=0.0001 SIGMA_MAX=15.0 DSIGMA=0.1
    WRITE_STRIDE=10000
    LABEL=byj
... METAINFERENCE

# Chemical shifts
#SETTINGS AUXFOLDER=user-doc/tutorials/others/isdb-1/m_and_m/data
# Chemical shifts
cs: CS2BACKBONE ...
NOPBC ATOMS=1-111 DATADIR=data TEMPLATE=egaawaass.pdb DOSCORE ARG=pb.bias
NOISETYPE=MGAUSS REWEIGHT OPTSIGMAMEAN=SEM_MAX AVERAGING=400
SIGMA0=4.0
SIGMA_MIN=0.0001
SIGMA_MAX=5.00
WRITE_STRIDE=10000
...

mcs: BIASVALUE ARG=cs.score


# output from METAINFERENCE

PRINT ARG=byrdcnh.*   STRIDE=200 FILE=BAYES.RDC.NH
PRINT ARG=byrdccaha.* STRIDE=200 FILE=BAYES.RDC.CAHA
PRINT ARG=byj.*       STRIDE=200 FILE=BAYES.J
PRINT ARG=cs.*        STRIDE=200 FILE=BAYES.CS

# the following are useful for the analysis on-the-fly of the quality of the agreement with the experimentl da
ENSEMBLE ...
    ARG=(nh\.rdc-.*),(caha\.rdc-.*),(jhan\.j-.*),(jhahn\.j-.*),(jccg\.j-.*),(jncg\.j-.*),(cs\...-.*),pb.bias R
    LABEL=ens
... ENSEMBLE

STATS ...
    ARG=(ens\.nh\.rdc-.*) PARARG=(nh\.exp-.*)
```

```
    LABEL=nhst
... STATS

STATS ...
    ARG=(ens\.caha\.rdc-.*) PARARG=(caha\.exp-.*)
    LABEL=cahast
... STATS

STATS ...
    ARG=(ens\.cs\...-.*) PARARG=(cs\.exp.*)
    LABEL=csst
... STATS

STATS ...
    ARG=(ens\.jhan\.j-.*) PARARG=(jhan\.exp-.*)
    LABEL=jhanst
... STATS

STATS ...
    ARG=(ens\.jhahn\.j-.*) PARARG=(jhahn\.exp-.*)
    LABEL=jhahnst
... STATS

STATS ...
    ARG=(ens\.jccg\.j.*),(ens\.jccg\.j.*) PARARG=(jccg\.exp-.*),(jccg\.exp-.*)
    SQDEVSUM
    LABEL=jw5ccyst
... STATS

STATS ...
    ARG=(ens\.jncg\.j.*),(ens\.jncg\.j.*) PARARG=(jncg\.exp-.*),(jncg\.exp-.*)
    SQDEVSUM
    LABEL=jw5ncyst
... STATS

#output from STATS
PRINT ARG=nhst.*      STRIDE=2000 FILE=ST.RDC.NH
PRINT ARG=cahast.*    STRIDE=2000 FILE=ST.RDC.CAHA
PRINT ARG=csst.*      STRIDE=2000 FILE=ST.CS
PRINT ARG=jhanst.*,jhahnst.*,jw5ccyst.*,jw5ncyst.* STRIDE=2000 FILE=ST.J
```

As for the former case we are running a multiple-replica simulation where in addition to multiple-walker metadynamics we are also coupling the replicas through Metainference. The use of multiple-walkers metadynamics is here key in order to have the same bias defined for all the replicas. This allows us to calculate a weighted average of the experimental observables where the weights are defined univocally from the bias [104] .

```
mpiexec -np 14 gmx_mpi mdrun -s topolnew -multi 14 -plumed plumed-eef1-pbmetad-m_m.dat -table table.xvg -table
```

8.3.5.2 ISDB: setting up a SAXS post processing and refinement calculation using MARTINI form factors

Authors

Cristina Paissoni

8.3.5.2.1 Aims This tutorial is thought to illustrate how it is possible to compute SAXS intensities from single PDB files or trajectories using PLUMED. In particular, we will show how to compute scattering intensities with the hybrid coarse-grained/atomistic approach that is described in [107] . The tutorial will provide basic instructions to prepare files for the back-calculation of SAXS intensities using the hybrid approach (this process is simplified by the possibility to use an ad-hoc python script). Further, it is explained how to adjust the plumed input file for specific purposes, e.g. to use SAXS data as restraints in MD simulations or to compare SAXS intensities computed with both the atomistic and the hybrid approach.

8.3.5.2.2 Objectives Once this tutorial is completed users will be able to:

- Calculate SAXS intensities using PLUMED from PDB files or trajectories.
- Setup a Metainference simulations using SAXS intensities as restraints (to this aim, ISDB: setting up a Metadynamics Metainfere is required as a prior condition)

8.3.5.2.3 Resources The TARBALL for this project contains the following files:

- martiniFormFactor_p3.py: a python script to be used with python 3. This is based on the martinize.py script (http://cgmartini.nl/index.php/tools2/proteins-and-bilayers/204-martinize)
- start.pdb and samplextc.xtc: PDB and trajectory files on which you can perform the calculations;
- SASDAB7.dat: a file containing SAXS intensities, downloaded from the SASDB database [108] .

This tutorial has been tested on PLUMED version 2.5.1

8.3.5.2.4 Introduction Calculations of Small-angle X-ray scattering (SAXS) intensities from a structure of N atoms could be extremely demanding from a computational perspective as it is an $O(N^2)$ problem. This issue has limited the applicability of SAXS in numerous situations, including their use as restraints in combination with MD simulations. A strategy to reduce the cost of computing SAXS from atomic structure consists in using a coarse grain representation of the structure, represented as a collection of M beads with M<N, each comprising a variable number of atoms. Niebling et al [109] have previously derived the Martini beads form factors for proteins, showing how this approach can be almost 50 times faster than the standard SAXS calculation. More recently, Martini beads form factors for nucleic acids have been also derived and, together with the ones for proteins, have been implemented in PLUMED. Importantly, it has been shown that the computation of scattering intensities using these Martini form factors achieves a good accuracy, as compared to the atomistic ones, for values up to 0.45 Å-1.

The Martini form factors can be exploited within a hybrid multi-resolution strategy to speed up SAXS profiles calculations, both for the back-calculation of scattering intensities from atomistic PDB or trajectory files and for the inclusion of SAXS data as restraints in experimental-driven all-atom simulations (e.g. METAINFERENCE). This can be achieved using PLUMED, which computes on the fly the virtual position of the Martini beads from the atomistic 3D-structure and then uses the centres of mass of these beads, along with Martini form factors, to quickly back-calculate SAXS intensities.

8.3.5.2.5 Computing SAXS intensities with the hybrid coarse-grained/atomistic approach Given a PDB file, PLUMED is able to compute SAXS intensities for the molecule in the PDB and to compare these intensities with the experimental ones stored in a data file. It is possible to apply the same procedure to all the conformations visited during a MD trajectory. This is achieved by using the PLUMED driver utility and the SAXS variable of the ISDB module. While computing scattering intensities with a full atomistic representation is quite easy (see the SAXS keyword and later in this tutorial), in order to adopt the hybrid coarse-grained/atomistic approach a more elaborated procedure is needed, requiring the generation of few specific files to be used as input by driver. To facilitate this step, it is possible to use this script (martiniFormFactor_p3.py for python 3) and type in the bash shell:

```
python martiniFormFactor_p3.py -f start.pdb -dat SASDAB7.dat [-unit Ang/nm -nq 15]
```

The input files are:

- filepdb.pdb: a PDB file containing the atomic coordinates of the molecule; in our case it is start.pdb. Note that only one model should be present in the PDB and an ENDMDL statement is expected to appear only at the end of the file. Further, if different chains are present, they are expected to be separated by a TER statement or named differently (this is true also for chain breaks, but it is generally not recommended to use broken molecules for these calculations). If the aim is to analyse a trr/xtc trajectories, the PDB should contain the same atoms of the trajectories and with the same order, including water molecules and ions (you can simply save a sample PDB from the xtc/trr and use it to generate the necessary files).
- filedat.dat: a file containing the momentum transfer in the first column and the experimental SAXS intensities in the second, with the two columns separated by blanks or commas; in our case it is SASDAB7.dat. Further columns are accepted but they will not be considered. The momentum transfer is expected to be expressed in inverse Angstrom, if it is expressed in inverse nm you can use the option "-unit nm" (note that this is needed in our case!). By default, the python script will select 15 equally separated q values, and the relative intensities, between the first and the last values of the file. If you want to change this default behaviour of the python script you can use the option "-nq" to indicate the number of q values to consider. These values will be used by PLUMED to compute SAXS intensities for the selected momentum transfers and to compare them with the corresponding experimental values provided.

The files generated in this step, to be used later as input for driver, are the following:

- aacg_template.pdb: a PDB file, which PLUMED uses as a template, in which the atomistic model provided (with the atoms renumbered in sequential order) is concatenated to the coarse-grained representation.
- plumed_beads.dat: a file that instructs PLUMED about how to compute the centre of mass of each bead and that define a group of atoms (called "martini") containing all the identified beads. These atoms are the ones that will be used later for the calculation of SAXS intensities (the atom group "martini" is indeed recalled in the SAXS keyword of the plumed.dat file). We suggest to always double-check this file, verifying that the lists of atoms in the last beads correspond to the actual Martini mapping.
- plumed.dat: a file that tells PLUMED to compute the CENTER of each bead (achieved reading plumed_↩ beads.dat), compute scattering intensities for the selected q-values using the hybrid coarse-grained/atomistic approach and, lastly, print some output files that could be useful for further statistics. We will see later how to modify this file to achieve different goals.

The default files generated should be sufficient to post-process single PDB files or trajectories, by typing:

```
plumed driver --plumed plumed.dat --mf_pdb start.pdb
```

or

```
plumed driver --plumed plumed.dat --mf_xtc samplextc.xtc
```

The output files generated by this step are:

- SAXSINT, containing the computed SAXS intensities. Each line of this file contains in the first column a time and in the following columns the scattering intensities for each of the selected q-values. If the file post-processed is a PDB there will be only one line with time 0. Otherwise, each line will correspond to a frame of the input trajectory (by default the time unit is 1.0, but you can set the desired time step using the option "–timestep" of PLUMED driver).
- ST.SAXS, containing in the first column a time (as above) and in the following column the correlation between experimental and back-calculated SAXS intensities for the selected q-values.

Let's see now the meaning of each line in plumed.dat and how it is possible to modify it for other purposes, e.g. to compare atomistic/coarse-grained scattering intensities and to perform Metainference simulations where SAXS data are used as restraints. Here is a sample of plumed.dat produced by the python script martiniFormFactor_p3.py:

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=aacg_template.pdb

# BEADS DEFINITION
INCLUDE FILE=plumed_beads.dat

# SAXS
SAXS ...

        LABEL=saxsdata
        ATOMS=martini
        MARTINI

        # You can use SCALEINT keyword to set appropriate scaling factor.
        # SCALEINT is expected to correspond to the intensity in q=0
        # SCALEINT=

        QVALUE1=0.0111721000    EXPINT1=0.0527250000
        QVALUE2=0.0368675000    EXPINT2=0.0327126000
        QVALUE3=0.0625615000    EXPINT3=0.0128316000
        QVALUE4=0.0882529000    EXPINT4=0.0045545200
        QVALUE5=0.1139410000    EXPINT5=0.0022799600
        QVALUE6=0.1396240000    EXPINT6=0.0013048600
        QVALUE7=0.1653020000    EXPINT7=0.0007215740
        QVALUE8=0.1909730000    EXPINT8=0.0004340930
        QVALUE9=0.2166360000    EXPINT9=0.0002717150
        QVALUE10=0.2422910000   EXPINT10=0.0002574160
        QVALUE11=0.2679360000   EXPINT11=0.0001878030
        QVALUE12=0.2935700000   EXPINT12=0.0001592670
        QVALUE13=0.3191930000   EXPINT13=0.0000811279
        QVALUE14=0.3448030000   EXPINT14=0.0001110630
        QVALUE15=0.3703990000   EXPINT15=0.0001264680

        # METAINFERENCE
        # Uncomment the following keywords and adjust parameters to activate METAINFERENCE
        # DOSCORE NOENSEMBLE SIGMA_MEAN0=0
        # REGRES_ZERO=500
        # SIGMA0=5 SIGMA_MIN=0.001 SIGMA_MAX=5.00
        # NOISETYPE=MGAUSS

... SAXS

# METAINFERENCE
# Uncomment the following keyword to activate METAINF
# saxsbias: BIASVALUE ARG=(saxsdata\.score) STRIDE=10

# STATISTICS
statcg: STATS ARG=(saxsdata\.q_.*) PARARG=(saxsdata\.exp_.*)

# PRINT
# Uncomment the following line to print METAINFERENCE output
# PRINT ARG=(saxsdata\.score),(saxsdata\.biasDer),(saxsdata\.weight),(saxsdata\.scale),  (saxsdata\.offset),(s
# change stride if you are using METAINFERENCE
PRINT ARG=(saxsdata\.q_.*) STRIDE=1 FILE=SAXSINT
PRINT ARG=statcg.corr STRIDE=1 FILE=ST.SAXSCG
```

The first two lines tell PLUMED to use aacg_template.pdb as a template and to read the file plumed_beads.↩ dat (which rules how to compute the CENTER of the beads and defines the “martini” group, containing all the beads). Note that you do not need to prepare these two files, since they both are produced by the python script martiniFormFactor_p[2,3].py. These lines are followed by the SAXS keyword, labelled “saxsdata”. Here are defined the atoms to be used for SAXS calculations (in our case these are all the beads within the group “martini”) and the structure factors to adopt: in this case we are using the Martini form factors (flag MARTINI), alternatively it is possible to use the atomistic ones (flag ATOMISTIC) or define custom form factors using a polynomial expansion to any order (flag PARAMETERS). By default, PLUMED computes a scaling factor to fit experimental and back-calculated intensities, however it is possible to set manually this scaling factor using the flag SCALEINT, which is expected to correspond to the intensity in q=0. The following lines indicate the q-values at which the calculation of scattering intensities is required (QVALUE) and the corresponding intensities (EXPINT). These are the ones selected by default by the python script, but it is possible to manually adjust them and/or to add new values. Lastly, within the SAXS keyword, there are few lines that are needed to activate METAINFERENCE. These are commented

since they are not necessary for the back-calculation of SAXS intensities, however you have to uncomment and adjust them if you want to perform Metainference simulations in which SAXS data are used as restraints. The same is true for the line containing the keyword BIASVALUE. In the last part of the file, we ask PLUMED to compute some statistics comparing experimental and back-calculated intensities; finally, we print the computed intensities and statistics into the SAXSINT and ST.SAXSCG files, respectively.

It is easy to modify the plumed.dat for your own purposes, for instance:

1. it could be used to perform Metainference simulations: to this aim it is sufficient to uncomment the lines indicated in the plumed.dat above and adjust the parameters to make them suitable for the investigated system (see the tutorial ISDB: setting up a Metadynamics Metainference simulation and the SAXS keyword). We further suggest adding (if needed) WHOLEMOLECULES and NOPBC flags.

2. it could be exploited to compute SAXS intensities with both the atomistic and the hybrid approach to compare them later. A sample plumed.dat to achieve this could be:

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=aacg_template.pdb
# BEADS DEFINITION
INCLUDE FILE=plumed_beads.dat

# SAXS
SAXS ...
LABEL=saxscg
ATOMS=martini
MARTINI
QVALUE1=0.010
QVALUE2=0.020
# etc..
... SAXS

SAXS ...
LABEL=saxsaa
ATOMS=1-11104
      ATOMISTIC
QVALUE1=0.010
QVALUE2=0.020
# etc..
... SAXS

#PRINT
PRINT ARG=(saxscg\.q_.*) FILE=QVAL_CG
PRINT ARG=(saxsaa\.q_.*) FILE=QVAL_AA
```

8.4 PYTORCH (Machine Learning Collective Variables)

8.4.1 Overview

The PYTORCH module is an interface between PyTorch machine learning library and PLUMED. It implements the PYTORCH_MODEL class, which is a subclass of `Function` class. `PYTORCH_MODEL` provide the ability to load models defined in Pytorch and compiled with `TorchScript`.

For instance, this allows one to use the outputs of a neural network as collective variables, as done in [66] and in [110]. Furthermore, the PYTORCH_MODEL outputs can also be used as inputs for other collective variables and for data analysis tools.

8.4.2 Installation

This module is not installed by default. It requires the PyTorch C++ APIs (LibTorch) to be linked against PLUMED.

Warning

Note that Libtorch APIs are still in beta phase regarding stability, so there might be breaking changes in newer versions. Currently, versions between 1.8.∗ and 1.13.∗ are supported. Please note that if you want to link a different version it might be necessary to manually specify the required libraries within LIBS in configure. Furthermore, it is advised to use the same version of PyTorch and LibTorch to avoid compatibility issues.

Download LibTorch C++ API library

You can download the pre-built LibTorch library from their website. For example, the following script downloads the 1.13.1 version (CPU, with C++11 ABI compatibility).

```
wget https://download.pytorch.org/libtorch/cpu/libtorch-cxx11-abi-shared-with-deps-1.13.1%2Bcpu.zip
unzip libtorch-cxx11-abi-shared-with-deps-1.13.1+cpu.zip ;
LIBTORCH=${PWD}/libtorch
```

The location of the include and library files need to be exported in the environment. For convenience, we can save them in a file sourceme.sh inside the libtorch folder:

```
> echo "export CPATH=${LIBTORCH}/include/torch/csrc/api/include/:${LIBTORCH}/include/:${LIBTORCH}/include/torc
> echo "export INCLUDE=${LIBTORCH}/include/torch/csrc/api/include/:${LIBTORCH}/include/:${LIBTORCH}/include/to
> echo "export LIBRARY_PATH=${LIBTORCH}/lib:$LIBRARY_PATH" >> ${LIBTORCH}/sourceme.sh
> echo "export LD_LIBRARY_PATH=${LIBTORCH}/lib:$LD_LIBRARY_PATH" >> ${LIBTORCH}/sourceme.sh
> . ${LIBTORCH}/sourceme.sh
```

Remember to source the sourceme.sh file in your ∼/.bashrc or ∼/.bash_profile file.

Configure PLUMED

In order to install the PYTORCH module when compiling PLUMED we need to (1) specify to look for libtorch (--enable-libtorch) and (2) enable the related module (--enable-modules=pytorch or also --enable-modules=all):

```
> ./configure --enable-libtorch --enable-modules=pytorch
```

Attention

To verify that the linking of LibTorch is succesful, one should look at the output of the configure command, which should report one of the following lines: checking libtorch without extra libs.. yes or checking libtorch with -ltorch_cpu -lc10... yes. If not, configure will display a warning (and not an error!) that says: configure: WARNING: cannot enable __PLUMED_↩ HAS_LIBTORCH. In this case, it is recommended to examine the output of the above two commands in the config.log file to understand the reason (e.g. it cannot find the required libraries).

Additional notes

- A compiler with C++14 support is required.
- If you want to use the pre-cxx11 ABI LibTorch binaries the following flag should be added to the configure: CXXFLAGS="-D_GLIBCXX_USE_CXX11_ABI=0".

8.4.3 Usage

Currently, all features of the PYTORCH module are included in a single function: PYTORCH_MODEL

8.4.4 CVs with PyTorch: the mlcvs package

`mlcvs` is a Python package (under development) that can be used to optimize different kinds of neural-networks based CVs, e.g. that discriminate between states [66] or that approximate the slow dynamical modes of the system [110]. The CVs are optimized in Python and the resulting model is compiled with TorchScript, in order to allowed the models to be employed without Python dependencies.

8.4.5 Contents

8.4.6 Functions Documentation

The following list contains descriptions of functions developed for the PYTORCH module. They can be used in combination with other actions outside of the PYTORCH module.

PYTORCH_MODEL	Load a PyTorch model compiled with TorchScript.

8.4.6.1 PYTORCH_MODEL

	This is part of the pytorch module
	It is only available if you configure PLUMED with ./configure –enable-modules=pytorch . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Load a PyTorch model compiled with TorchScript.

This can be a function defined in Python or a more complex model, such as a neural network optimized on a set of data. In both cases the derivatives of the outputs with respect to the inputs are computed using the automatic differentiation (autograd) feature of Pytorch.

By default it is assumed that the model is saved as: `model.ptc`, unless otherwise indicated by the `FILE` keyword. The function automatically checks for the number of output dimensions and creates a component for each of them. The outputs are called node-i with i between 0 and N-1 for N outputs.

Note that this function is active only if LibTorch is correctly linked against PLUMED. Please check the instructions in the PYTORCH (Machine Learning Collective Variables) page.

Examples

Load a model called `torch_model.ptc` that takes as input two dihedral angles and returns two outputs.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PYTORCH_MODEL.tmp
#SETTINGS AUXFILE=regtest/pytorch/rt-pytorch_model_2d/torch_model.ptc
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
model: PYTORCH_MODEL FILE=torch_model.ptc ARG=phi,psi
PRINT FILE=COLVAR ARG=model.node-0,model.node-1
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
node	Model outputs

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
FILE	Filename of the PyTorch compiled model

8.5 Logarithmic Mean Force Dynamics

8.5.1 Overview

The LOGMFD module contains the LogMFD/LogPD method for enhanced sampling in a CV space and for on-the-fly free energy reconstruction along the CVs. This module implements the multiple-replica algorithm (LogPD [68]) as well as the single-replica algorithm (LogMFD [67]), the former invoking the Crooks-Jarzynski non-equilibrium work relation. In addition, TAMD/d-AFED [45] can also be implemented by this module.

8.5.2 Installation

This module is not installed by default. Add '--enable-modules=logmfd' to your './configure' command when building PLUMED to enable these features.

8.5.3 Usage

Currently, all features of the LOGMFD module are included in a single LOGMFD bias function: LOGMFD

8.5.4 Contents

8.5.5 Biases Documentation

The following list contains descriptions of biases developed for the PLUMED-LOGMFD module. They can be used in combination with other biases outside of the LOGMFD module.

LOGMFD	Used to perform LogMFD, LogPD, and TAMD/d-AFED.

8.5.5.1 LOGMFD

	This is part of the logmfd module
	It is only available if you configure PLUMED with ./configure –enable-modules=logmfd . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Used to perform LogMFD, LogPD, and TAMD/d-AFED.

8.5.5.1.1 LogMFD Consider a physical system of N_q particles, for which the Hamiltonian is given as

$$H_{\mathrm{MD}}\left(\mathbf{\Gamma}\right)=\sum_{j=1}^{N_q}\frac{\mathbf{p}_j^2}{2m_j}+\Phi\left(\mathbf{q}\right)$$

where $\mathbf{q}_j$, $\mathbf{p}_j$ ($\mathbf{\Gamma}=\mathbf{q},\mathbf{p}$), and m_j are the position, momentum, and mass of particle j respectively, and Φ is the potential energy function for $\mathbf{q}$. The free energy $F(\mathbf{X})$ as a function of a set of N collective variables (CVs) is given as

$$\begin{aligned}F\left(\mathbf{X}\right) &= -k_BT\log\int\exp\left[-\beta H_{\mathrm{MD}}\right]\prod_{i=1}^{N}\delta\left(s_i\left(\mathbf{q}\right)-X_i\right)d\mathbf{\Gamma}\\ &\simeq -k_BT\log\int\exp\left[-\beta\left\{H_{\mathrm{MD}}+\sum_i^N\frac{K_i}{2}\left(s_i\left(\mathbf{q}\right)-X_i\right)^2\right\}\right]d\mathbf{\Gamma}\end{aligned}$$

where $\mathbf{s}$ are the CVs, k_B is Boltzmann constant, $\beta=1/k_BT$, and K_i is the spring constant which is large enough to invoke

$$\delta\left(x\right)=\lim_{k\to\infty}\sqrt{\beta k/2\pi}\exp\left(-\beta kx^2/2\right)$$

In mean-force dynamics, $\mathbf{X}$ are treated as fictitious dynamical variables, which are associated with the following Hamiltonian,

$$H_{\log} = \sum_{i=1}^{N} \frac{P_{X_i}^2}{2M_i} + \Psi\left(\mathbf{X}\right)$$

where P_{X_i} and M_i are the momentum and mass of X_i, respectively, and Ψ is the potential function for $\mathbf{X}$. We assume that Ψ is a functional of $F(\mathbf{X})$; $[F(\mathbf{X})]$. The simplest form of Ψ is $\Psi = F(\mathbf{X})$, which corresponds to TAMD/d-AFED [45], [44] (or the extended Lagrangian dynamics in the limit of the adiabatic decoupling between $\mathbf{q}$ and $\mathbf{X}$). In the case of LogMFD, the following form of Ψ is introduced [67];

$$\Psi_{\log}\left(\mathbf{X}\right) = \gamma \log\left[\alpha F\left(\mathbf{X}\right) + 1\right]$$

where α (ALPHA) and γ (GAMMA) are positive parameters. The logarithmic form of $\Psi_{\log}$ ensures the dynamics of $\mathbf{X}$ on a much smoother energy surface [i.e., $\Psi_{\log}(\mathbf{X})$] than $F(\mathbf{X})$, thus enhancing the sampling in the $\mathbf{X}$-space. The parameters α and γ determine the degree of flatness of $\Psi_{\log}$, but adjusting only α is normally sufficient to have a relatively flat surface (with keeping the relation $\gamma = 1/\alpha$).

The equation of motion for X_i in LogMFD (no thermostat) is

$$M_i \ddot{X}_i = -\left(\frac{\alpha\gamma}{\alpha F + 1}\right)\frac{\partial F}{\partial X_i}$$

where $-\partial F/\partial X_i$ is evaluated as a canonical average under the condition that $\mathbf{X}$ is fixed;

$$\begin{aligned} -\frac{\partial F}{\partial X_i} &\simeq \frac{1}{Z}\int K_i\left(s_i\left(\mathbf{q}\right) - X_i\right)\exp\left[-\beta\left\{H_{\mathrm{MD}} + \sum_i^N \frac{K_i}{2}\left(s_i\left(\mathbf{q}\right) - X_i\right)^2\right\}\right] d\mathbf{\Gamma} \\ &\equiv \left\langle K_i\left(s_i\left(\mathbf{q}\right) - X_i\right)\right\rangle_{\mathbf{X}} \end{aligned}$$

where

$$Z = \int \exp\left[-\beta\left\{H_{\mathrm{MD}} + \sum_i^N \frac{K_i}{2}\left(s_i\left(\mathbf{q}\right) - X_i\right)^2\right\}\right] d\mathbf{\Gamma}$$

The mean-force (MF) is practically evaluated by performing a shot-time canonical MD run of N_m steps each time $\mathbf{X}$ is updated according to the equation of motion for $\mathbf{X}$.

If the canonical average for the MF is effectively converged, the dynamical variables $\mathbf{q}$ and $\mathbf{X}$ are decoupled and they evolve adiabatically, which can be exploited for the on-the-fly evaluation of $F(\mathbf{X})$. I.e., $H_{\log}$ should be a constant of motion in this case, thus $F(\mathbf{X})$ can be evaluated each time $\mathbf{X}$ is updated as

$$F\left(\mathbf{X}\left(t\right)\right) = \frac{1}{\alpha}\left[\exp\frac{1}{\gamma}\left\{\left(H_{\log} - \sum_i \frac{P_{X_i}^2}{2M_i}\right)\right\} - 1\right]$$

This means that $F(\mathbf{X})$ can be constructed without post processing (on-the-fly free energy reconstruction). Note that the on-the-fly free energy reconstruction is also possible in TAMD/d-AFED if the Hamiltonian-like conserved quantity is available (e.g., the Nose-Hoover type dynamics).

8.5.5.1.2 LogPD The accuracy in the MF is critical to the on-the-fly free energy reconstruction. To improve the evaluation of the MF, parallel-dynamics (PD) is incorporated into LogMFD, leading to logarithmic parallel-dynamics (LogPD) [68].

In PD, the MF is evaluated by a non-equilibrium path-ensemble based on the Crooks-Jarzynski non-equilibrium work relation. To this end, multiple replicas of the MD system which run in parallel are introduced. The CVs [$\mathbf{s}(\mathbf{q})$] in each replica is restrained to the same value of $\mathbf{X}(t)$. A canonical MD run with N_m steps is performed in each replica, then the MF on X_i is evaluated using the MD trajectories from all replicas. The MF is practically calculated as

$$-\frac{\partial F}{\partial X_i} = \sum_{k=1}^{N_r} W_k \sum_{n=1}^{N_m} \frac{1}{N_m} K_i \left[s_i \left(\mathbf{q}_n^k \right) - X_i \right]$$

where $\mathbf{q}_n^k$ indicates the $\mathbf{q}$-configuration at time step n in the N_m-step shot-time MD run in the kth replica among N_r replicas. W_k is given as

$$W_k = \frac{e^{-\beta w_k(t)}}{\sum\limits_{k=1}^{N_r} e^{-\beta w_k(t)}}$$

where

$$w_k\left(t\right) = \int_{X_0}^{X(t)} \sum_{i=1}^{N} \frac{\partial H_{\mathrm{MD}}^k}{\partial X_i} dX_i$$

$$H_{\mathrm{MD}}^k\left(\mathbf{\Gamma}, \mathbf{X}\right) = H_{\mathrm{MD}}\left(\mathbf{\Gamma}^k\right) + \sum_{i=1}^{N} \frac{K_i}{2} \left(s_i^k - X_i\right)^2$$

and s_i^k is the ith CV in the kth replica.

W_k comes from the Crooks-Jarzynski non-equilibrium work relation by which we can evaluate an equilibrium ensemble average from a set of non-equilibrium trajectories. Note that, to avoid possible numerical errors in the exponential function, the following form of W_k is instead used in PLUMED,

$$W_k\left(t\right) = \frac{\exp\left[-\beta\left\{w_k\left(t\right) - w_{\min}\left(t\right)\right\}\right]}{\sum_k \exp\left[-\beta\left\{w_k\left(t\right) - w_{\min}\left(t\right)\right\}\right]}$$

where

$$w_{\min}\left(t\right) = \mathrm{Min}_k\left[w_k\left(t\right)\right]$$

With the MF evaluated using the PD approach, free energy profiles can be reconstructed more efficiently (requiring less elapsed computing time) in LogPD than with a single MD system in LogMFD. In the case that there exist more than one stable state separated by high energy barriers in the hidden subspace orthogonal to the CV-subspace, LogPD is particularly of use to incorporate all the contributions from such hidden states with appropriate weights (in the limit $N_r \to \infty$).

Note that LogPD calculations should always be initiated with an equilibrium $\mathbf{q}$-configuration in each replica, because the Crooks-Jarzynski non-equilibrium work relation is invoked. Also note that LogPD is currently available only with Gromacs, while LogMFD can be performed with LAMMPS, Gromacs, Quantum Espresso, NAMD, and so on.

8.5.5.1.3 Using LogMFD/PD with a thermostat Introducing a thermostat on $\mathbf{X}$ is often recommended in Log↩ MFD/PD to maintain the adiabatic decoupling between $\mathbf{q}$ and $\mathbf{X}$. In the framework of the LogMFD approach, the Nose-Hoover type thermostat and the Gaussian isokinetic (velocity scaling) thermostat can be used to control the kinetic energy of $\mathbf{X}$.

8.5.5.1.3.1 Nose-Hoover LogMFD/PD The equation of motion for X_i coupled to a Nose-Hoover thermostat variable η (single heat bath) is

$$M_i \ddot{X}_i = -\left(\frac{\alpha\gamma}{\alpha F + 1}\right)\frac{\partial F}{\partial X_i} - M_i \dot{X}_i \dot{\eta}$$

The equation of motion for η is

$$Q\ddot{\eta} = \sum_{i=1}^{N} \frac{P_{X_i}^2}{M_i} - N k_B T$$

where N is the number of the CVs. Since the following pseudo-Hamiltonian is a constant of motion in Nose-Hoover LogMFD/PD,

$$H_{\log}^{\mathrm{NH}} = \sum_{i=1}^{N} \frac{P_{X_i}^2}{2M_i} + \gamma \log\left[\alpha F\left(\mathbf{X}\right) + 1\right] + \frac{1}{2} Q\dot{\eta}^2 + N k_B T \eta$$

$F(\mathbf{X}(t))$ is obtained at each MFD step as

$$F\left(\mathbf{X}\left(t\right)\right) = \frac{1}{\alpha}\left[\exp\left\{\frac{1}{\gamma}\left(H_{\log}^{\mathrm{NH}} - \sum_i \frac{P_{X_i}^2}{2M_i} - \frac{1}{2}Q\dot{\eta}^2 - N k_B T \eta\right)\right\} - 1\right]$$

8.5.5.1.3.2 Velocity scaling LogMFD/PD The velocity scaling algorithm (which is equivalent to the Gaussian isokinetic dynamics in the limit $\Delta t \to 0$) can also be employed to control the velocity of $\mathbf{X}$, $\mathbf{V_x}$.

The following algorithm is introduced to perform isokinetic LogMFD calculations [69];

$$\begin{aligned}
V_{X_i}\left(t_{n+1}\right) &= V'_{X_i}\left(t_n\right) + \Delta t\left[-\left(\frac{\alpha\gamma}{\alpha F\left(t_n\right)+1}\right)\frac{\partial F\left(t_n\right)}{\partial X_i}\right] \\
S\left(t_{n+1}\right) &= \sqrt{\frac{N k_B T}{\sum_i M_i V_{X_i}^2\left(t_{n+1}\right)}} \\
{V_{X_i}}'\left(t_{n+1}\right) &= S\left(t_{n+1}\right) V_{X_i}\left(t_{n+1}\right) \\
X_i\left(t_{n+1}\right) &= X_i\left(t_n\right) + \Delta t V'_{X_i}\left(t_{n+1}\right) \\
\Psi_{\log}\left(t_{n+1}\right) &= N k_B T \log S\left(t_{n+1}\right) + \Psi_{\log}\left(t_n\right) \\
F\left(t_{n+1}\right) &= \frac{1}{\alpha}\left[\exp\left\{\Psi_{\log}\left(t_{n+1}\right)/\gamma\right\} - 1\right]
\end{aligned}$$

Note that $V'_{X_i}\left(t_0\right)$ is assumed to be initially given, which meets the following relation,

$$\sum_{i=1}^{N} M_i V'^2_{X_i}\left(t_0\right) = N k_B T$$

It should be stressed that, in the same way as in the NVE and Nose-Hoover LogMFD/PD, $F(\mathbf{X}(t))$ can be evaluated at each MFD step (on-the-fly free energy reconstruction) in Velocity Scaling LogMFD/PD.

Examples

8.5.5.1.4 Examples

8.5.5.1.4.1 Example of LogMFD The following input file tells plumed to restrain collective variables to the fictitious dynamical variables in LogMFD/PD.

plumed.dat

```
BEGIN_PLUMED_FILE working DATADIR=example-check/LOGMFD.tmp
UNITS TIME=fs LENGTH=1.0 ENERGY=kcal/mol MASS=1.0 CHARGE=1.0
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

# LogMFD
LOGMFD ...
LABEL=logmfd
ARG=phi,psi
KAPPA=1000.0,1000.0
DELTA_T=1.0
INTERVAL=200
TEMP=300.0
FLOG=2.0
MFICT=5000000,5000000
VFICT=3.5e-4,3.5e-4
ALPHA=4.0
THERMOSTAT=NVE
FICT_MAX=3.15,3.15
FICT_MIN=-3.15,-3.15
... LOGMFD
```

To submit this simulation with Gromacs, use the following command line to execute a LogMFD run with Gromacs-MD. Here topol.tpr is the input file which contains atomic coordinates and Gromacs parameters.

```
gmx_mpi mdrun -s topol.tpr -plumed
```

This command will output files named logmfd.out and replica.out.

The output file logmfd.out records free energy and all fictitious dynamical variables at every MFD step.

logmfd.out

```
# LogMFD
# CVs : phi psi
# Mass for CV particles : 5000000.000000 5000000.000000
# Mass for thermostat   :   11923.224809
# 1:iter_mfd, 2:Flog, 3:2*Ekin/gkb[K], 4:eta, 5:Veta,
# 6:phi_fict(t), 7:phi_vfict(t), 8:phi_force(t),
# 9:psi_fict(t), 10:psi_vfict(t), 11:psi_force(t),
       0       2.000000     308.221983       0.000000       0.000000      -2.442363       0.000350       5.522
       1       1.995466     308.475775       0.000000       0.000000      -2.442013       0.000350      -4.406
       2       1.992970     308.615664       0.000000       0.000000      -2.441663       0.000350      -3.346
       3       1.988619     308.859946       0.000000       0.000000      -2.441313       0.000350       6.463
...
```

The output file replica.out records all collective variables at every MFD step.

replica.out

```
 Replica No. 0 of 1.
# 1:iter_mfd, 2:work, 3:weight,
# 4:phi(q)
# 5:psi(q)
       0    0.000000e+00     1.000000e+00       -2.436841       2.434093
       1   -4.539972e-03     1.000000e+00       -2.446420       2.438531
       2   -7.038516e-03     1.000000e+00       -2.445010       2.443114
       3   -1.139672e-02     1.000000e+00       -2.434850       2.434677
...
```

8.5.5.1.4.2 Example of LogPD Use the following command line to execute a LogPD run using two MD replicas (note that only Gromacs is currently available for LogPD). Here 0/topol.tpr and 1/topol.tpr are the input files, each containing the atomic coordinates for the corresponding replica and Gromacs parameters. All the directories, 0/ and 1/, should contain the same plumed.dat.

```
mpirun -np 2 gmx_mpi mdrun -s topol -plumed -multidir 0 1
```

This command will output files named logmfd.out in 0/, and replica.out.0 and replica.out.1 in 0/ and 1/, respectively.

The output file logmfd.out records free energy and all fictitious dynamical variables at every MFD step.

logmfd.out

```
# LogPD, replica parallel of LogMFD
# number of replica : 2
# CVs : phi psi
# Mass for CV particles : 5000000.000000 5000000.000000
# Mass for thermostat   :   11923.224809
# 1:iter_mfd, 2:Flog, 3:2*Ekin/gkb[K], 4:eta, 5:Veta,
# 6:phi_fict(t), 7:phi_vfict(t), 8:phi_force(t),
# 9:psi_fict(t), 10:psi_vfict(t), 11:psi_force(t),
       0       2.000000     308.221983       0.000000       0.000000      -2.417903       0.000350       4.930
       1       1.999367     308.257404       0.000000       0.000000      -2.417552       0.000350       0.851
       2       1.999612     308.243683       0.000000       0.000000      -2.417202       0.000350      -1.588
       3       1.999608     308.243922       0.000000       0.000000      -2.416852       0.000350       4.267
...
```

The output file replica.out.0 records all collective variables and the Jarzynski weight of replica No.0 at every MFD step.

replica.out.0

```
# Replica No. 0 of 2.
# 1:iter_mfd, 2:work, 3:weight,
# 4:phi(q)
# 5:psi(q)
       0    0.000000e+00     5.000000e-01       -2.412607       2.446191
       1   -4.649101e-06     4.994723e-01       -2.421403       2.451318
       2    1.520985e-03     4.983996e-01       -2.420269       2.455049
       3    1.588855e-03     4.983392e-01       -2.411081       2.447394
...
```

The output file replica.out.1 records all collective variables and the Jarzynski weight of replica No.1 at every MFD step.

replica.out.1

```
# Replica No. 1 of 2.
# 1:iter_mfd, 2:work, 3:weight,
# 4:phi(q)
# 5:psi(q)
       0    0.000000e+00     5.000000e-01       -2.413336       2.450516
       1   -1.263077e-03     5.005277e-01       -2.412009       2.449229
       2   -2.295444e-03     5.016004e-01       -2.417322       2.452512
       3   -2.371507e-03     5.016608e-01       -2.414078       2.448521
...
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
_fict	For example, the fictitious collective variable for LogMFD is specified as ARG=dist12 and LAB↩ EL=logmfd in LOGMFD section in Plumed input file, the associated fictitious dynamical variable can be specified as PRINT ARG=dist12,logmfd.dist12_fict FILE=COLVAR
_vfict	For example, the fictitious collective variable for LogMFD is specified as ARG=dist12 and LAB↩ EL=logmfd in LOGMFD section in Plumed input file, the velocity of the associated fictitious dynamical variable can be specified as PRINT ARG=dist12,logmfd.dist12_vfict FILE=COLVAR

Compulsory keywords

INTERVAL	Period of MD steps (N_m) to update fictitious dynamical variables.
DELTA_T	Time step for the fictitious dynamical variables (DELTA_T=1 often works).
THERMOSTAT	Type of thermostat for the fictitious dynamical variables. NVE, NVT, VS are available.
KAPPA	Spring constant of the harmonic restraining potential.
FICT_MAX	Maximum values reachable for the fictitious dynamical variables. The variables will elastically bounce back at the boundary (mirror boundary).
FICT_MIN	Minimum values reachable for the fictitious dynamical variables. The variables will elastically bounce back at the boundary (mirror boundary).
FLOG	The initial free energy value in the LogMFD/PD run.The origin of the free energy profile is adjusted by FLOG to realize $F(\mathbf{X}(t)) > 0$ at any $\mathbf{X}(t)$, resulting in enhanced barrier-crossing. (The value of $H_{\log}$ is automatically set according to FLOG). In fact, $F(\mathbf{X}(t))$ is correctly estimated in PLUMED even when $F(\mathbf{X}(t)) < 0$ in the LogMFD/PD run.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
TEMP	Target temperature for the fictitious dynamical variables in LogMFD/PD. It is recommended to set TEMP to be the same as the temperature of the MD system in any thermostated LogMFD/PD run. If not provided, it will be taken from the temperature of the MD system (Gromacs only).

TAMD	When TAMD=1, TAMD/d-AFED calculations can be performed instead of LogMFD. Otherwise, the LogMFD protocol is switched on (default).
ALPHA	Alpha parameter for LogMFD. If not provided or provided as 0, it will be taken as 1/gamma. If gamma is also not provided, Alpha is set as 4, which is a sensible value when the unit of kcal/mol is used.
GAMMA	Gamma parameter for LogMFD. If not provided or provided as 0, it will be taken as 1/alpha. If alpha is also not provided, Gamma is set as 0.25, which is a sensible value when the unit of kcal/mol is used.
FICT	The initial values of the fictitious dynamical variables. If not provided, they are set equal to their corresponding CVs for the initial atomic configuration.
VFICT	The initial velocities of the fictitious dynamical variables. If not provided, they will be taken as 0. If THERMOSTAT=VS, they are instead automatically adjusted according to TEMP.
MFICT	Masses of each fictitious dynamical variable. If not provided, they will be taken as 10000.
XETA	The initial eta variable of the Nose-Hoover thermostat for the fictitious dynamical variables. If not provided, it will be taken as 0.
VETA	The initial velocity of eta variable. If not provided, it will be taken as 0.
META	Mass of eta variable. If not provided, it will be taken as $N * kb * T * 100 * 100$.
WORK	The initial value of the work done by fictitious dynamical variables in each replica. If not provided, it will be taken as 0.
TEMPPD	Temperature of the Boltzmann factor in the Jarzynski weight in LogPD (Gromacs only). TEMPPD should be the same as the temperature of the MD system, while TEMP may be (in principle) different from it. If not provided, TEM↩ PPD is set to be the same value as TEMP.

8.6 Experiment Directed Simulation

8.6.1 Overview

This Experiment Directed Simulation module contains methods for adaptively determining linear bias parameters such that each biased CV samples a new target mean value. This module implements the stochastic gradient descent algorithm in the original EDS paper [26] as well as additional minimization algorithms for Coarse-Grained Directed Simulation [70]. There is a recent review on the method and its applications here: [71].

Notice that a similar method is available as MAXENT, although with different features and using a different optimization algorithm.

8.6.2 Installation

This module is not installed by default. Add '--enable-modules=eds' to your './configure' command when building PLUMED to enable these features.

8.6.3 Usage

Currently, all features of the EDS module are included in a single EDS bias function: EDS

A tutorial using EDS specifically for biasing coordination number can be found on Andrew White's webpage.

8.6.4 Contents

8.6.5 Biases Documentation

The following list contains descriptions of biases developed for the PLUMED-EDS module. They can be used in combination with other biases outside of the EDS module.

EDS	Add a linear bias on a set of observables.

8.6.5.1 EDS

	This is part of the eds module
	It is only available if you configure PLUMED with ./configure –enable-modules=eds . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Add a linear bias on a set of observables.

This force is the same as the linear part of the bias in RESTRAINT, but this bias has the ability to compute the prefactors adaptively using the scheme of White and Voth [26] in order to match target observable values for a set of CVs. Further updates to the algorithm are described in [70] and you can read a review on the method and its applications here: [71].

You can see a tutorial on EDS specifically for biasing coordination number at `Andrew White's webpage`.

The addition to the potential is of the form

$$\sum_i \frac{\alpha_i}{s_i} x_i$$

where for CV x_i, a coupling constant α_i is determined adaptively or set by the user to match a target value for x_i. s_i is a scale parameter, which by default is set to the target value. It may also be set separately.

Warning

It is not possible to set the target value of the observable to zero with the default value of s_i as this will cause a divide-by-zero error. Instead, set $s_i = 1$ or modify the CV so the desired target value is no longer zero.

Notice that a similar method is available as MAXENT, although with different features and using a different optimization algorithm.

Virial

The bias forces modify the virial and this can change your simulation density if the bias is used in an NPT simulation. One way to avoid changing the virial contribution from the bias is to add the keyword VIRIAL=1.0, which changes how the bias is computed to minimize its contribution to the virial. This can also lead to smaller magnitude biases that are more robust if transferred to other systems. VIRIAL=1.0 can be a reasonable starting point and increasing the value changes the balance between matching the set-points and minimizing the virial. See [71] for details on the equations. Since the coupling constants are unique with a single CV, VIRIAL is not applicable with a single CV. When used with multiple CVs, the CVs should be correlated which is almost always the case.

Weighting

EDS computes means and variances as part of its algorithm. If you are also using a biasing method like metadynamics, you may wish to remove the effect of this bias in your EDS computations so that EDS works on the canonical values (reweighted to be unbiased). For example, you may be using metadynamics to bias a dihedral angle to enhance sampling and be using EDS to set the average distance between two particular atoms. Specifically:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/EDS.tmp
# set-up metadynamics
t: TORSION ATOMS=1,2,3,4
md: METAD ARG=d SIGMA=0.2 HEIGHT=0.3 PACE=500 TEMP=300
# compute bias weights
bias: REWEIGHT_METAD TEMP=300
# now do EDS on distance while removing effect of metadynamics
d: DISTANCE ATOMS=4,7
eds: EDS ARG=d CENTER=3.0 PERIOD=100 TEMP=300 LOGWEIGHTS=bias
```

This is an approximation though because EDS uses a finite samples while running to get means/variances. At the end of a run, you should ensure this approach worked and indeed your reweighted CV matches the target value.

Examples

The following input for a harmonic oscillator of two beads will adaptively find a linear bias to change the mean and variance to the target values. The PRINT line shows how to access the value of the coupling constants.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EDS.tmp
dist: DISTANCE ATOMS=1,2
# this is the squared of the distance
dist2: COMBINE ARG=dist POWERS=2 PERIODIC=NO

# bias mean and variance
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 PERIOD=100 TEMP=1.0
PRINT ARG=dist,dist2,eds.dist_coupling,eds.dist2_coupling,eds.bias,eds.force2 FILE=colvars.dat STRIDE=100
```

Rather than trying to find the coupling constants adaptively, one can ramp up to a constant value.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EDS.tmp
dist: DISTANCE ATOMS=1,2
dist2: COMBINE ARG=dist POWERS=2 PERIODIC=NO

# ramp couplings from 0,0 to -1,1 over 50000 steps
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 FIXED=-1,1 RAMP PERIOD=50000 TEMP=1.0

# same as above, except starting at -0.5,0.5 rather than default of 0,0
eds2: EDS ARG=dist,dist2 CENTER=2.0,1.0 FIXED=-1,1 INIT=-0.5,0.5 RAMP PERIOD=50000 TEMP=1.0
```

A restart file can be added to dump information needed to restart/continue simulation using these parameters every PERIOD.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EDS.tmp
dist: DISTANCE ATOMS=1,2
dist2: COMBINE ARG=dist POWERS=2 PERIODIC=NO

# add the option to write to a restart file
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 PERIOD=100 TEMP=1.0 OUT_RESTART=checkpoint.eds
```

The first few lines of the restart file that is output if we run a calculation with one CV will look something like this:

```
#! FIELDS time d1_center d1_set d1_target d1_coupling d1_maxrange d1_maxgrad d1_accum d1_mean d1_std
#! SET adaptive  1
#! SET update_period  1
#! SET seed  0
#! SET kbt    2.4943
   0.0000   1.0000   0.0000   0.0000   0.0000   7.4830   0.1497   0.0000   0.0000   0.0000
   1.0000   1.0000   0.0000   0.0000   0.0000   7.4830   0.1497   0.0000   0.0000   0.0000
   2.0000   1.0000  -7.4830   0.0000   0.0000   7.4830   0.1497   0.0224   0.0000   0.0000
   3.0000   1.0000  -7.4830   0.0000  -7.4830   7.4830   0.1497   0.0224   0.0000   0.0000
   4.0000   1.0000  -7.4830   0.0000  -7.4830   7.4830   0.1497   0.0224   0.0000   0.0000
```

Read in a previous restart file. Adding RESTART flag makes output append

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EDS.tmp
d1: DISTANCE ATOMS=1,2

eds: EDS ARG=d1 CENTER=2.0 PERIOD=100 TEMP=1.0 IN_RESTART=restart.eds RESTART=YES
```

Read in a previous restart file and freeze the bias at the final level from the previous simulation

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EDS.tmp
d1: DISTANCE ATOMS=1,2

eds: EDS ARG=d1 CENTER=2.0 TEMP=1.0 IN_RESTART=restart.eds FREEZE
```

Read in a previous restart file and freeze the bias at the mean from the previous simulation

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EDS.tmp
d1: DISTANCE ATOMS=1,2

eds: EDS ARG=d1 CENTER=2.0 TEMP=1.0 IN_RESTART=restart.eds FREEZE MEAN
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	squared value of force from the bias
pressure	If using virial keyword, this is the current sum of virial terms. It is in units of pressure (energy / vol^3)
_coupling	For each named CV biased, there will be a corresponding output CV_coupling storing the current linear bias prefactor.

Compulsory keywords

RANGE	(default=25.0) The (starting) maximum increase in coupling constant per PERIOD (in k_BT/[BIAS_SCALE unit]) for each CV biased
SEED	(default=0) Seed for random order of changing bias
INIT	(default=0) Starting value for coupling constant
FIXED	(default=0) Fixed target values for coupling constant. Non-adaptive.
LM_MIXING	(default=1) Initial mixing parameter when using Levenberg-Marquadt minimization.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
LM	(default=off) Use Levenberg-Marquadt algorithm along with simultaneous keyword. Otherwise use gradient descent.
RAMP	(default=off) Slowly increase bias constant to a fixed value
COVAR	(default=off) Utilize the covariance matrix when updating the bias. Default Off, but may be enabled due to other options
FREEZE	(default=off) Fix bias at current level (only used for restarting).
MEAN	(default=off) Instead of using final bias level from restart, use average. Can only be used in conjunction with FREEZE
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
CENTER	The desired centers (equilibrium values) which will be sought during the adaptive linear biasing. This is for fixed centers
CENTER_ARG	The desired centers (equilibrium values) which will be sought during the adaptive linear biasing. CENTER_ARG is for calculated centers, e.g. from a CV or analysis.
PERIOD	Steps over which to adjust bias for adaptive or ramping
BIAS_SCALE	A divisor to set the units of the bias. If not set, this will be the CENTER value by default (as is done in White and Voth 2014).
TEMP	The system temperature. If not provided will be taken from MD code (if available)
MULTI_PROP	What proportion of dimensions to update at each step. Must be in interval [1,0), where 1 indicates all and any other indicates a stochastic update. If not set, default is 1 / N, where N is the number of CVs.
VIRIAL	Add an update penalty for having non-zero virial contributions. Only makes sense with multiple correlated CVs.
LOGWEIGHTS	Add weights to use for computing statistics. For example, if biasing with metadynamics.
RESTART_FMT	the format that should be used to output real numbers in EDS restarts
OUT_RESTART	Output file for all information needed to continue EDS simulation. If you have the RESTART directive set (global or for EDS), this file will be appended to. Note that the header will be printed again if appending.
IN_RESTART	Read this file to continue an EDS simulation. If same as OUT_RESTART and you have not set the RESTART directive, the file will be backed-up and overwritten with new output. If you do have the RESTART flag set and it is the same name as OUT_RESTART, this file will be appended.
RESTART	allows per-action setting of restart (YES/NO/AUTO)

8.7 Extended-System Adaptive Biasing Force

8.7.1 Overview

This module contains the eABF/DRR method to do free energy calculation or enhance sampling along CVs.

8.7.2 Installation

This module is not installed by default and depends on the boost serialization module. Please make sure the boost serialization library is compiled and installed in your system before trying to compile this module. Add '--enable-modules=drr --enable-boost_serialization' to your './configure' command when building PLUMED to enable these features.

8.7.3 Usage

Please read drr_tool and DRR for more information.

8.7.4 Contents

8.7.5 Biases Documentation

The following list contains descriptions of biases developed for the eABF module. They can be used in combination with other biases outside of the eABF module.

DRR	Used to performed extended-system adaptive biasing force(eABF)

8.7.5.1 DRR

	This is part of the drr module
	It is only available if you configure PLUMED with ./configure –enable-modules=drr . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Used to performed extended-system adaptive biasing force(eABF)
This method was introduced in [73]. It is used on one or more collective variables. This method is also called dynamic reference restraining(DRR) [111] . A detailed description of this module can be found at [72] .
For each collective variable ξ_i, a fictitious variable λ_i is attached through a spring. The fictitious variable λ_i undergoes overdamped Langevin dynamics just like EXTENDED_LAGRANGIAN. The ABF algorithm applies bias force on λ_i. The bias force acts on λ_i is the negative average spring force on λ_i, which enhances the sampling of λ_i.

$$F_{bias}(\lambda_i) = k(\lambda_i - \langle \xi_i \rangle_{\lambda_i})$$

If spring force constant k is large enough, then ξ_i synchronizes with λ_i. The naive(ABF) estimator is just the negative average spring force of λ_i.
The naive(ABF) estimator is biased. There are unbiased estimators such as CZAR(Corrected z-averaged restraint) [74] and UI(Umbrella Integration). The CZAR estimates the gradients as:

$$\frac{\partial A}{\partial \xi_i}(\xi) = -\frac{1}{\beta}\frac{\partial \ln \tilde{\rho}(\xi)}{\partial \xi_i} + k\left(\langle \lambda_i \rangle_\xi - \xi_i\right)$$

The UI estimates the gradients as:

$$A'(\xi^*) = \frac{\sum_\lambda N(\xi^*, \lambda)\left[\frac{\xi^* - \langle \xi \rangle_\lambda}{\beta \sigma_\lambda^2} - k(\xi^* - \lambda)\right]}{\sum_\lambda N(\xi^*, \lambda)}$$

The code performing UI(colvar_UIestimator.h) is contributed by Haohao Fu [75] . It may be slow. I only change the Boltzmann constant and output precision in it. For new version and issues, please see: https://github.com/fhh2626/colvars

After running eABF/DRR, the drr_tool utility can be used to extract the gradients and counts files from .drrstate. Naive(ABF) estimator's result is in .abf.grad and .abf.count files and CZAR estimator's result is in .czar.grad and .czar.count files. The additional .zcount and .zgrad files contain the number of samples of ξ, and the negative of ξ-averaged spring forces, respectively, which are mainly for inspecting and debugging purpose. To get PMF, the abf_integrate(https://github.com/Colvars/colvars/tree/master/colvartools) is useful for numerically integrating the .czar.grad file.

Examples

The following input tells plumed to perform a eABF/DRR simulation on two torsional angles.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DRR.tmp
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

DRR ...
LABEL=eabf
ARG=phi,psi
FULLSAMPLES=500
GRID_MIN=-pi,-pi
GRID_MAX=pi,pi
GRID_BIN=180,180
FRICTION=8.0,8.0
TAU=0.5,0.5
OUTPUTFREQ=50000
HISTORYFREQ=500000
... DRR

# monitor the two variables, their fictitious variables and applied forces.
PRINT STRIDE=10 ARG=phi,psi,eabf.phi_fict,eabf.psi_fict,eabf.phi_biasforce,eabf.psi_biasforce FILE=COLVAR
```

The following input tells plumed to perform a eABF/DRR simulation on the distance of atom 10 and 92. The distance is restraint by LOWER_WALLS and UPPER_WALLS.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DRR.tmp
dist1: DISTANCE ATOMS=10,92
eabf_winall: DRR ARG=dist1 FULLSAMPLES=2000 GRID_MIN=1.20 GRID_MAX=3.20 GRID_BIN=200 FRICTION=8.0 TAU=0.5 OUTP
uwall: UPPER_WALLS ARG=eabf_winall.dist1_fict AT=3.2 KAPPA=418.4
lwall: LOWER_WALLS ARG=eabf_winall.dist1_fict AT=1.2 KAPPA=418.4
PRINT STRIDE=10 ARG=dist1,eabf_winall.dist1_fict,eabf_winall.dist1_biasforce FILE=COLVAR
```

It's also possible to run extended generalized adaptive biasing force (egABF) described in [112] . An egABF example:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DRR.tmp
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

DRR ...
LABEL=gabf_phi
ARG=phi
FULLSAMPLES=500
GRID_MIN=-pi
GRID_MAX=pi
GRID_BIN=180
FRICTION=8.0
TAU=0.5
OUTPUTFREQ=50000
HISTORYFREQ=500000
... DRR

DRR ...
LABEL=gabf_psi
ARG=psi
FULLSAMPLES=500
GRID_MIN=-pi
GRID_MAX=pi
GRID_BIN=180
FRICTION=8.0
TAU=0.5
OUTPUTFREQ=50000
```

```
HISTORYFREQ=500000
... DRR

DRR ...
LABEL=gabf_2d
ARG=phi,psi
EXTERNAL_FORCE=gabf_phi.phi_springforce,gabf_psi.psi_springforce
EXTERNAL_FICT=gabf_phi.phi_fictNoPBC,gabf_psi.psi_fictNoPBC
GRID_MIN=-pi,-pi
GRID_MAX=pi,pi
GRID_BIN=180,180
NOBIAS
OUTPUTFREQ=50000
HISTORYFREQ=500000
... DRR

PRINT STRIDE=10 ARG=phi,psi FILE=COLVAR
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
_fict	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will named with the arguments of the bias followed by the character string _tilde. It is possible to add forces on these variable.
_vfict	one or multiple instances of this quantity can be referenced elsewhere in the input file. These quantities will named with the arguments of the bias followed by the character string _tilde. It is NOT possible to add forces on these variable.
_biasforce	The bias force from eABF/DRR of the fictitious particle.
_springforce	Spring force between real CVs and extended CVs
_fictNoPBC	the positions of fictitious particles (without PBC).

Compulsory keywords

TAU	(default=0.5) specifies relaxation time on each of variables are, similar to extended Time Constant in Colvars
FRICTION	(default=8.0) add a friction to the variable, similar to extended Langevin Damping in Colvars
GRID_MIN	the lower bounds for the grid (GRID_BIN or GRID_SPACING should be specified)
GRID_MAX	the upper bounds for the grid (GRID_BIN or GRID_SPACING should be specified)
REFLECTINGWALL	(default=0) whether add reflecting walls for each CV at GRID_MIN and GRID_MAX. Setting non-zero values will enable this feature
FULLSAMPLES	(default=500) number of samples in a bin prior to application of the ABF
MAXFACTOR	(default=1.0) maximum scaling factor of biasing force
OUTPUTFREQ	write results to a file every N steps

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOCZAR	(default=off) disable the CZAR estimator
UI	(default=off) enable the umbrella integration estimator
NOBIAS	(default=off) DO NOT apply bias forces.
TEXTOUTPUT	(default=off) use text output for grad and count files instead of boost↩ ::serialization binary output
MERGEHISTORYFILES	(default=off) output all historic results to a single file rather than multiple .drrstate files. This option is effective only when textOutput is on.
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
KAPPA	specifies that the restraint is harmonic and what the values of the force constants on each of the variables are (default to k_BT/(GRID_SPACING)^2)
GRID_BIN	the number of bins for the grid
GRID_SPACING	the approximate grid spacing (to be used as an alternative or together with GRID_BIN)
ZGRID_MIN	the lower bounds for the grid (ZGRID_BIN or ZGRID_SPACING should be specified)
ZGRID_MAX	the upper bounds for the grid (ZGRID_BIN or ZGRID_SPACING should be specified)
ZGRID_BIN	the number of bins for the grid
ZGRID_SPACING	the approximate grid spacing (to be used as an alternative or together with ZGRID_BIN)
EXTERNAL_FORCE	use forces from other action instead of internal spring force, this disable the extended system!
EXTERNAL_FICT	position of external fictitious particles, useful for UIESTIMATOR
HISTORYFREQ	save history to a file every N steps
UIRESTARTPREFIX	specify the restart files for umbrella integration
OUTPUTPREFIX	specify the output prefix (default to the label name)
TEMP	the system temperature - needed when FRICTION is present. If not provided will be taken from MD code (if available)
EXTTEMP	the temperature of extended variables (default to system temperature)
DRR_RFILE	specifies the restart file (.drrstate file)

8.7.6 Command Line Tools

The following list contains the command line tools available in the eABF module.

drr_tool	- Extract .grad and .count files from the binary output .drrstate - Merge windows

8.7.6.1 drr_tool

	This is part of the drr module
	It is only available if you configure PLUMED with ./configure –enable-modules=drr . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

- Extract .grad and .count files from the binary output .drrstate
- Merge windows

Examples

The following command will extract .grad and .count files.

```
plumed drr_tool --extract eabf.drrstate
```

The following command will merge windows of two .drrstate file, and output the .grad and .count files.

```
plumed drr_tool --merge win1.drrstate,win2.drrstate
```

After getting the .grad and .count file, you can do numerical integration by using abf_integrate tool from https://github.com/Colvars/colvars/tree/master/colvartools

```
abf_integrate eabf.czar.grad
```

Note

The abf_integrate in colvartools is in kcal/mol, so it may be better to use –units kcal/mol when running drr_tool

Glossary of keywords and components

Compulsory keywords

--units	(default=kj/mol) the units of energy can be kj/mol, kcal/mol, j/mol, eV or the conversion factor from kj/mol

Options

--help/-h	(default=off) print this help
-v	(default=off) Verbose output
--extract	Extract drrstate file(s)
--merge	Merge eABF windows
--merge_output	The output filename of the merged result
--divergence	Calculate divergence of gradient field (experimental)

8.8 Variationally Enhanced Sampling (VES code)

The VES code is a module for PLUMED that implements enhanced sampling methods based on *Variationally Enhanced Sampling* (VES) [76]. The VES code is developed by Omar Valsson, see the homepage of the VES code for further information.
The VES code is an optional module that needs to be enabled when configuring the compilation of PLUMED by using the '--enable-modules=ves' (or '--enable-modules=all') flag when running the 'configure' script.
In the tutorials you can learn how to use the methods implemented in the VES code.
The various components of the VES code module are listed and described in the following sections

- Biases
- Basis functions
- Target Distributions
- Optimizers
- Utilities
- Command Line Tools
- Tutorials

8.8.1 Biases

The following list contains the biases available in the VES code.

VES_DELTA_F	Implementation of VES Delta F method
VES_LINEAR_EXPANSION	Linear basis set expansion bias.

8.8.1.1 VES_DELTA_F

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Implementation of VES Delta F method
Implementation of VES ΔF method [113] (step two only).

Warning

Notice that this is a stand-alone bias Action, it does not need any of the other VES module components

First you should create some estimate of the local free energy basins of your system, using e.g. multiple METAD short runs, and combining them with the sum_hills utility. Once you have them, you can use this bias Action to perform the VES optimization part of the method.
These $N+1$ local basins are used to model the global free energy. In particular, given the conditional probabilities $P(\mathbf{s}|i) \propto e^{-\beta F_i(\mathbf{s})}$ and the probabilities of being in a given basin P_i, we can write:

$$e^{-\beta F(\mathbf{s})} \propto P(\mathbf{s}) = \sum_{i=0}^{N} P(\mathbf{s}|i) P_i \, .$$

We use this free energy model and the chosen bias factor γ to build the bias potential: $V(\mathbf{s}) = -(1-1/\gamma)F(\mathbf{s})$.
Or, more explicitly:

$$V(\mathbf{s}) = (1-1/\gamma)\frac{1}{\beta}\log\left[e^{-\beta F_0(\mathbf{s})} + \sum_{i=1}^{N} e^{-\beta F_i(\mathbf{s})} e^{-\beta \alpha_i}\right] ,$$

where the parameters α are the N free energy differences (see below) from the F_0 basin.
By default the $F_i(\mathbf{s})$ are shifted so that $\min[F_i(\mathbf{s})] = 0$ for all $i = \{0, ..., N\}$. In this case the optimization parameters α_i are the difference in height between the minima of the basins. Using the keyword NORMALIZE, you

can also decide to normalize the local free energies so that $\int ds\, e^{-\beta F_i(\mathbf{s})} = 1$. In this case the parameters will represent not the difference in height (which depends on the chosen CVs), but the actual free energy difference, $\alpha_i = \Delta F_i$.
However, as discussed in Ref. [113], a better estimate of ΔF_i should be obtained through the reweighting procedure.

Examples

The following performs the optimization of the free energy difference between two metastable basins:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/VES_DELTA_F.tmp
cv: DISTANCE ATOMS=1,2
ves: VES_DELTA_F ...
  ARG=cv
  TEMP=300
  FILE_F0=fesA.data
  FILE_F1=fesB.data
  BIASFACTOR=10.0
  M_STEP=0.1
  AV_STRIDE=500
  PRINT_STRIDE=100
...
PRINT FMT=%g STRIDE=500 FILE=Colvar.data ARG=cv,ves.bias,ves.rct
```

The local FES files can be obtained as described in Sec. 4.2 of Ref. [113], i.e. for example:

- run 4 independent metad runs, all starting from basin A, and kill them as soon as they make the first transition (see e.g. COMMITTOR)
- `cat HILLS* > all_HILLS`
- `plumed sum_hills --hills all_HILLS --outfile all_fesA.dat --mintozero --min 0 --max 1 --bin 100`
- `awk -v n_rep=4 '{if($1!="#!" && $1!="") {for(i=1+(NF-1)/2; i<=NF; i++) $i/=n_rep;} print $0}' all_fesA.da`

The header of both FES files must be identical, and should be similar to the following:

```
#! FIELDS cv file.free der_cv
#! SET min_cv 0
#! SET max_cv 1
#! SET nbins_cv  100
#! SET periodic_cv false
0 0 0

#! FIELDS cv file.free der_cv
#! SET min_cv 0
#! SET max_cv 1
#! SET nbins_cv  100
#! SET periodic_cv false
0 0 0
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
rct	the reweighting factor c(t)
work	the work done by the bias in one AV_STRIDE

Compulsory keywords

FILE_F	names of files containing local free energies and derivatives. The first one, FILE_F0, is used as reference for all the free energy differences.. You can use multiple instances of this keyword i.e. FILE_F1, FILE_F2, FILE_F3...
BIASFACTOR	(default=0) the gamma bias factor used for well-tempered target p(s). Set to 0 for non-tempered flat target
M_STEP	(default=1.0) the mu step used for the Omega functional minimization
AV_STRIDE	(default=500) number of simulation steps between alpha updates
ALPHA_FILE	(default=ALPHA) file name for output minimization parameters

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NORMALIZE	(default=off) normalize all local free energies so that alpha will be (approx) Delta F
NO_MINTOZERO	(default=off) leave local free energies as provided, without shifting them to zero min
DAMPING_OFF	(default=off) do not use an AdaGrad-like term to rescale M_STEP
SERIAL	(default=off) perform the calculation in serial even if multiple tasks are available
MULTIPLE_WALKERS	(default=off) use multiple walkers connected via MPI for the optimization
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
TEMP	temperature is compulsory, but it can be sometimes fetched from the MD engine
TG_STRIDE	(default=1) number of AV_STRIDE between updates of target p(s) and reweighing factor c(t)
TAU_MEAN	exponentially decaying average for alpha (rescaled using AV_STRIDE). Should be used only in very specific cases
INITIAL_ALPHA	(default=0) an initial guess for the bias potential parameter alpha
PRINT_STRIDE	(default=10) stride for printing to ALPHA_FILE
FMT	specify format for ALPHA_FILE
RESTART	allows per-action setting of restart (YES/NO/AUTO)

8.8.1.2 VES_LINEAR_EXPANSION

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Linear basis set expansion bias.
This VES bias action takes the bias potential to be a linear expansion in some basis set that is written as a product of one-dimensional basis functions. For example, for one CV the bias would be written as

$$V(s_1;\alpha) = \sum_{i_1} \alpha_{i_1} f_{i_1}(s_1),$$

while for two CVs it is written as

$$V(s_1,s_2;\alpha) = \sum_{i_1,i_2} \alpha_{i_1,i_2} f_{i_1}(s_1) f_{i_2}(s_2)$$

where α is the set of expansion coefficients that are optimized within VES. With an appropriate choice of the basis functions it is possible to represent any generic free energy surface. The relationship between the bias and the free energy surface is given by

$$V(\mathbf{s}) = -F(\mathbf{s}) - \frac{1}{\beta}\log p(\mathbf{s}).$$

where $p(\mathbf{s})$ is the target distribution that is employed in the VES simulation.

Basis Functions

Various one-dimensional basis functions are available in the VES code, see the complete list here. At the current moment we recommend to use Legendre polynomials (BF_LEGENDRE) for non-periodic CVs and Fourier basis functions (BF_FOURIER) for periodic CV (e.g. dihedral angles).
To use basis functions within VES_LINEAR_EXPANSION you first need to define them in the input file before the VES_LINEAR_EXPANSION action and then give their labels using the BASIS_FUNCTIONS keyword.

Target Distributions

Various target distributions $p(\mathbf{s})$ are available in the VES code, see the complete list here.
To use a target distribution within VES_LINEAR_EXPANSION you first need to define it in the input file before the VES_LINEAR_EXPANSION action and then give its label using the TARGET_DISTRIBUTION keyword. The default behavior if no TARGET_DISTRIBUTION is given is to employ a uniform target distribution.
Some target distribution, like the well-tempered one (TD_WELLTEMPERED), are dynamic and need to be iteratively updated during the optimization.

Optimizer

In order to optimize the coefficients you will need to use VES_LINEAR_EXPANSION in combination with an optimizer, see the list of optimizers available in the VES code here. At the current moment we recommend to use the averaged stochastic gradient decent optimizer (OPT_AVERAGED_SGD).
The optimizer should be defined after the VES_LINEAR_EXPANSION action.

Grid

Internally the code uses grids to calculate the basis set averages over the target distribution that is needed for the gradient. The same grid is also used for the output files (see next section). The size of the grid is determined by the GRID_BINS keyword. By default it has 100 grid points in each dimension, and generally this value should be sufficient.

Outputting Free Energy Surfaces and Other Files

It is possible to output on-the-fly during the simulation the free energy surface estimated from the bias potential. How often this is done is specified within the optimizer by using the FES_OUTPUT keyword. The filename is specified by the FES_FILE keyword, but by default is it fes.LABEL.data, with an added suffix indicating the iteration number (iter-#).
For multi-dimensional case is it possible to also output projections of the free energy surfaces. The arguments for which to do these projections is specified using the numbered PROJ_ARG keywords. For these files a suffix indicating the projection (proj-#) will be added to the filenames. You will also need to specify the frequency of the output by using the FES_PROJ_OUTPUT keyword within the optimizer.
It is also possible to output the bias potential itself, for this the relevant keyword is BIAS_OUTPUT within the optimizer. The filename is specified by the BIAS_FILE keyword, but by default is it bias.LABEL.data, with an added suffix indicating the iteration number (iter-#).
Furthermore is it possible to output the target distribution, and its projections (i.e. marginal distributions). The filenames of these files are specified with the TARGETDIST_FILE, but by default is it targetdist.LABEL.data. The logarithm of the target distribution will also be outputted to file that has the added suffix log. For static target distribution these files will be outputted in the beginning of the simulation while for dynamic ones you will need to specify the frequency of the output by using the TARGETDIST_OUTPUT and TARGETDIST_PROJ_OUTPUT keywords within the optimizer.
It is also possible to output free energy surfaces and bias in post processing by using the VES_OUTPUT_FES action. However, be aware that this action does does not support dynamic target distribution (e.g. well-tempered).

Static Bias

It is also possible to use VES_LINEAR_EXPANSION as a static bias that uses previously obtained coefficients. In this case the coefficients should be read in from the coefficient file given in the COEFFS keyword.

Bias Cutoff

It is possible to impose a cutoff on the bias potential using the procedure introduced in [114] such that the free energy surface is only flooded up to a certain value. The bias that results from this procedure can then be used as a static bias for obtaining kinetic rates. The value of the cutoff is given by the BIAS_CUTOFF keyword. To impose the cutoff the code uses a Fermi switching function $1/(1+e^{\lambda x})$ where the parameter λ controls how sharply the switchingfunction goes to zero. The default value is $\lambda = 10$ but this can be changed by using the BIAS_CUTOFF↩
_FERMI_LAMBDA keyword.

Examples

In the following example we run a VES_LINEAR_EXPANSION for one CV using a Legendre basis functions (BF_LEGENDRE) and a uniform target distribution as no target distribution is specified. The coefficients are optimized using averaged stochastic gradient descent optimizer (OPT_AVERAGED_SGD). Within the optimizer we specify that the FES should be outputted to file every 500 coefficients iterations (the FES_OUTPUT keyword). Parameters that are very specific to the problem at hand, like the order of the basis functions, the interval on which the basis functions are defined, and the step size used in the optimizer, are left unfilled.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/VES_LINEAR_EXPANSION.tmp
bf1: BF_LEGENDRE ORDER=__FILL__ MINIMUM=__FILL__ MAXIMUM=__FILL__

VES_LINEAR_EXPANSION ...
 ARG=d1
 BASIS_FUNCTIONS=bf1
 TEMP=__FILL__
 GRID_BINS=200
 LABEL=b1
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
 BIAS=b1
 STRIDE=1000
```

```
 LABEL=o1
 STEPSIZE=__FILL__
 FES_OUTPUT=500
 COEFFS_OUTPUT=10
... OPT_AVERAGED_SGD
```

In the following example we employ VES_LINEAR_EXPANSION for two CVs, The first CV is periodic and therefore we employ a Fourier basis functions (BF_LEGENDRE) while the second CV is non-periodic so we employ a Legendre polynomials as in the previous example. For the target distribution we employ a well-tempered target distribution (TD_WELLTEMPERED), which is dynamic and needs to be iteratively updated with a stride that is given using the TARGETDIST_STRIDE within the optimizer.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/VES_LINEAR_EXPANSION.tmp
bf1: BF_FOURIER  ORDER=__FILL__ MINIMUM=__FILL__ MAXIMUM=__FILL__
bf2: BF_LEGENDRE ORDER=__FILL__ MINIMUM=__FILL__ MAXIMUM=__FILL__

td_wt: TD_WELLTEMPERED BIASFACTOR=10.0

VES_LINEAR_EXPANSION ...
 ARG=cv1,cv2
 BASIS_FUNCTIONS=bf1,bf2
 TEMP=__FILL__
 GRID_BINS=100
 LABEL=b1
 TARGET_DISTRIBUTION=td_wt
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
 BIAS=b1
 STRIDE=1000
 LABEL=o1
 STEPSIZE=__FILL__
 FES_OUTPUT=500
 COEFFS_OUTPUT=10
 TARGETDIST_STRIDE=500
... OPT_AVERAGED_SGD
```

In the following example we employ a bias cutoff such that the bias only fills the free energy landscape up a certain level. In this case the target distribution is also dynamic and needs to iteratively updated.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/VES_LINEAR_EXPANSION.tmp
bf1: BF_LEGENDRE ORDER=__FILL__ MINIMUM=__FILL__ MAXIMUM=__FILL__
bf2: BF_LEGENDRE ORDER=__FILL__ MINIMUM=__FILL__ MAXIMUM=__FILL__

VES_LINEAR_EXPANSION ...
 ARG=cv1,cv2
 BASIS_FUNCTIONS=bf1,bf2
 TEMP=__FILL__
 GRID_BINS=100
 LABEL=b1
 BIAS_CUTOFF=20.0
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
 BIAS=b1
 STRIDE=1000
 LABEL=o1
 STEPSIZE=__FILL__
 FES_OUTPUT=500
 COEFFS_OUTPUT=10
 TARGETDIST_STRIDE=500
... OPT_AVERAGED_SGD
```

The optimized bias potential can then be used as a static bias for obtaining kinetics. For this you need read in the final coefficients from file (e.g. coeffs_final.data in this case) by using the COEFFS keyword (also, no optimizer should be defined in the input)

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/VES_LINEAR_EXPANSION.tmp
bf1: BF_LEGENDRE ORDER=__FILL__ MINIMUM=__FILL__ MAXIMUM=__FILL__
bf2: BF_LEGENDRE ORDER=__FILL__ MINIMUM=__FILL__ MAXIMUM=__FILL__

VES_LINEAR_EXPANSION ...
```

```
 ARG=cv1,cv2
 BASIS_FUNCTIONS=bf1,bf2
 TEMP=__FILL__
 GRID_BINS=100
 LABEL=b1
 BIAS_CUTOFF=20.0
 COEFFS=coeffs_final.data
... VES_LINEAR_EXPANSION
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	the instantaneous value of the squared force due to this bias potential.

Compulsory keywords

BASIS_FUNCTIONS	the label of the one dimensional basis functions that should be used.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
TEMP	the system temperature - this is needed if the MD code does not pass the temperature to PLUMED.
BIAS_FILE	filename of the file on which the bias should be written out. By default it is bias.LABEL.data. Note that suffixes indicating the iteration number (iter-#) are added to the filename when optimizing coefficients.
FES_FILE	filename of the file on which the FES should be written out. By default it is fes.LABEL.data. Note that suffixes indicating the iteration number (iter-#) are added to the filename when optimizing coefficients.
TARGETDIST_FILE	filename of the file on which the target distribution should be written out. By default it is targetdist.LABEL.data. Note that suffixes indicating the iteration number (iter-#) are added to the filename when optimizing coefficients and the target distribution is dynamic.
OPTIMIZATION_THRESHOLD	Threshold value to turn off optimization of localized basis functions.
COEFFS	read in the coefficients from files.
TARGET_DISTRIBUTION	the label of the target distribution to be used.
BIAS_CUTOFF	cutoff the bias such that it only fills the free energy surface up to certain level F_cutoff, here you should give the value of the F_cutoff.
BIAS_CUTOFF_FERMI_LAMBDA	the lambda value used in the Fermi switching function for the bias cutoff (BIAS_CUTOFF), the default value is 10.0.
GRID_BINS	the number of bins used for the grid. The default value is 100 bins per dimension.

PROJ_ARG	arguments for doing projections of the FES or the target distribution.. You can use multiple instances of this keyword i.e. PROJ_ARG1, PROJ_A↩ RG2, PROJ_ARG3...
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

8.8.2 Basis functions

The following list contains the one-dimensional basis functions available in the VES code.

BF_CHEBYSHEV	Chebyshev polynomial basis functions.
BF_COMBINED	Combining other basis functions types
BF_COSINE	Fourier cosine basis functions.
BF_CUBIC_B_SPLINES	Cubic B spline basis functions.
BF_CUSTOM	Basis functions given by arbitrary mathematical expressions.
BF_FOURIER	Fourier basis functions.
BF_GAUSSIANS	Gaussian basis functions.
BF_LEGENDRE	Legendre polynomials basis functions.
BF_POWERS	Polynomial power basis functions.
BF_SINE	Fourier sine basis functions.
BF_WAVELETS	Daubechies Wavelets basis functions.

8.8.2.1 BF_CHEBYSHEV

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Chebyshev polynomial basis functions.
Use as basis functions `Chebyshev polynomials` of the first kind $T_n(x)$ defined on a bounded interval. You need to provide the interval $[a,b]$ on which the basis functions are to be used, and the order of the expansion N (i.e. the highest order polynomial used). The total number of basis functions is $N+1$ as the constant $T_0(x)=1$ is also included. These basis functions should not be used for periodic CVs.
Intrinsically the Chebyshev polynomials are defined on the interval $[-1,1]$. A variable t in the interval $[a,b]$ is transformed to a variable x in the intrinsic interval $[-1,1]$ by using the transform function

$$x(t) = \frac{t-(a+b)/2}{(b-a)/2}$$

The Chebyshev polynomials are given by the recurrence relation $\mathrm{T}_0(x)=1$
$T_1(x)=x$

$T_{n+1}(x) = 2\,x\,T_n(x) - T_{n-1}(x)$
The first 6 polynomials are shown below
The Chebyshev polynomial are orthogonal over the interval $[-1,1]$ with respect to the weight $\frac{1}{\sqrt{1-x^2}}$

$$\int_{-1}^{1} dx\, T_n(x)\, T_m(x)\, \frac{1}{\sqrt{1-x^2}} = \{\, 0\; n \neq m \pi n = m = 0 \pi/2 n = m \neq 0$$

For further mathematical properties of the Chebyshev polynomials see for example the Wikipedia page.

Examples

Here we employ a Chebyshev expansion of order 20 over the interval 0.0 to 10.0. This results in a total number of 21 basis functions. The label used to identify the basis function action can then be referenced later on in the input file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_CHEBYSHEV.tmp
bfC: BF_CHEBYSHEV MINIMUM=0.0 MAXIMUM=10.0 ORDER=20
```

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
NUMERICAL_INTEGRALS	(default=off) Calculate basis function integral for the uniform distribution numerically. Useful for debugging.

8.8.2.2 BF_COMBINED

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Combining other basis functions types

Examples

Here we define both Fourier cosine and sine expansions of order 10, each with 11 basis functions, which are combined. This results in a total number of 21 basis functions as only the constant from is bf_cos is used.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_COMBINED.tmp
bf_cos: BF_COSINE MINIMUM=-pi MAXIMUM=+pi ORDER=10
bf_sin: BF_SINE   MINIMUM=-pi MAXIMUM=+pi ORDER=10
bf_comb: BF_COMBINED BASIS_FUNCTIONS=bf_cos,bf_sin
```

In principle this is the same as using BF_FOURIER with ORDER=10 but with different ordering of the basis functions. Note that the order used in BASIS_FUNCTIONS matters for the ordering of the basis functions, using BAS↩ IS_FUNCTIONS=bf_sin,bf_cos would results in a different order of the basis functions. This should be kept in mind when restarting from previous coefficients.

Glossary of keywords and components

Compulsory keywords

BASIS_FUNCTIONS	Labels of the basis functions that should be combined. Note that the order used matters for the ordering of the basis functions. This needs to be kept in mind when restarting from previous coefficients.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.

8.8.2.3 BF_COSINE

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Fourier cosine basis functions.
Use as basis functions Fourier cosine series defined on a periodic interval. You need to provide the periodic interval $[a,b]$ on which the basis functions are to be used, and the order of the expansion N (i.e. the highest Fourier cosine mode used). The total number of basis functions is $N+1$ as the constant $f_0(x)=1$ is also included. These basis functions should only be used for periodic CVs. They can be useful if the periodic function being expanded is an even function, i.e. $F(-x)=F(x)$.
The Fourier cosine basis functions are given by $f_0(x)=1$
$f_1(x)=cos(\frac{2\pi}{P}x)$
$f_2(x)=cos(2\cdot\frac{2\pi}{P}x)$
$f_3(x)=cos(3\cdot\frac{2\pi}{P}x)$
$\vdots$
$f_n(x)=cos(n\cdot\frac{2\pi}{P}x)$
$\vdots$
$f_N(x)=cos(N\cdot\frac{2\pi}{P}x)$
where $P=(b-a)$ is the periodicity of the interval. They are orthogonal over the interval $[a,b]$

$$\int_a^b dx\, f_n(x)\, f_m(x) = \{\, 0\ n\neq m(b-a)n=m=0(b-a)/2n=m\neq 0.$$

Examples

Here we employ a Fourier cosine expansion of order 10 over the periodic interval $-\pi$ to $+\pi$. This results in a total number of 11 basis functions. The label used to identify the basis function action can then be referenced later on in the input file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_COSINE.tmp
BF_COSINE MINIMUM=-pi MAXIMUM=+pi ORDER=10 LABEL=bf1
```

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
NUMERICAL_INTEGRALS	(default=off) Calculate basis function integral for the uniform distribution numerically. Useful for debugging.

8.8.2.4 BF_CUBIC_B_SPLINES

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Cubic B spline basis functions.

Attention

These basis functions do not form orthogonal bases. We recommend using wavelets (BF_WAVELETS) instead that do for orthogonal bases.

A basis using cubic B spline functions according to [115]. See [116] for full details.
The mathematical expression of the individual splines is given by $\mathrm{h}(x) = \{ \quad (2-x^3, 1 \leq x \leq 2$
$4-6x^2+3x^3, \qquad x \leq 1$
$0, elsewhere.$
The full basis consists of equidistant splines at positions μ_i which are optimized in their height: $\mathrm{f}_i(x) = h\left(\frac{x-\mu_i}{\sigma}\right)$
Note that the distance between individual splines cannot be chosen freely but is equal to the width: $\mu_{i+1} = \mu_i + \sigma$.
The ORDER keyword of the basis set determines the number of equally sized sub-intervalls to be used. On the borders of each of these sub-intervalls the mean μ_i of a spline function is placed.
The total number of basis functions is $ORDER + 4$ as the constant $f_0(x) = 1$, as well as the two splines with means just outside the interval are also included.
As an example two adjacent basis functions can be seen below. The full basis consists of shifted splines in the full specified interval.
When the splines are used for a periodic CV (with the PERIODIC keyword), the sub-intervals are chosen in the same way, but only $ORDER + 1$ functions are required to fill it (the ones at the boundary coincide and the ones outside can be omitted).
To avoid 'blind' optimization of the basis functions outside the currently sampled area, it is often beneficial to use the OPTIMIZATION_THRESHOLD keyword of the VES_LINEAR_EXPANSION (set it to a small value, e.g. 1e-6)

Examples

The bias is expanded with cubic B splines in the intervall from 0.0 to 10.0 specifying an order of 20. This results in 24 basis functions.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_CUBIC_B_SPLINES.tmp
bf: BF_CUBIC_B_SPLINES MINIMUM=0.0 MAXIMUM=10.0 ORDER=20
```

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
PERIODIC	(default=off) Use periodic version of basis set.
NORMALIZATION	the normalization factor that is used to normalize the basis functions by dividing the values. By default it is 2.

8.8.2.5 BF_CUSTOM

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Basis functions given by arbitrary mathematical expressions.
This allows you to define basis functions using arbitrary mathematical expressions that are parsed using the lepton library. The basis functions $f_i(x)$ are given in mathematical expressions with *x* as a variable using the numbered FUNC keywords that start from FUNC1. Consistent with other basis functions is $f_0(x) = 1$ defined as the constant.
The interval on which the basis functions are defined is given using the MINIMUM and MAXIMUM keywords.
Using the TRANSFORM keyword it is possible to define a function $x(t)$ that is used to transform the argument before calculating the basis functions values. The variables *min* and *max* can be used to indicate the minimum and the maximum of the interval. By default the arguments are not transformed, i.e. $x(t) = t$.
For periodic basis functions you should use the PERIODIC flag to indicate that they are periodic.
The basis functions $f_i(x)$ and the transform function $x(t)$ need to be well behaved in the interval on which the basis functions are defined, e.g. not result in a not a number (nan) or infinity (inf). The code will not perform checks to make sure that this is the case unless the flag CHECK_NAN_INF is enabled.

Examples

Defining Legendre polynomial basis functions of order 6 using BF_CUSTOM where the appropriate transform function is given by the TRANSFORM keyword. This is just an example of what can be done, in practice you should use BF_LEGENDRE for Legendre polynomial basis functions.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_CUSTOM.tmp
BF_CUSTOM ...
```

```
 TRANSFORM=(t-(min+max)/2)/((max-min)/2)
 FUNC1=x
 FUNC2=(1/2)*(3*x^2-1)
 FUNC3=(1/2)*(5*x^3-3*x)
 FUNC4=(1/8)*(35*x^4-30*x^2+3)
 FUNC5=(1/8)*(63*x^5-70*x^3+15*x)
 FUNC6=(1/16)*(231*x^6-315*x^4+105*x^2-5)
 MINIMUM=-4.0
 MAXIMUM=4.0
 LABEL=bf1
... BF_CUSTOM
```

Defining Fourier basis functions of order 3 using BF_CUSTOM where the periodicity is indicated using the PERIODIC flag. This is just an example of what can be done, in practice you should use BF_FOURIER for Fourier basis functions.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_CUSTOM.tmp
BF_CUSTOM ...
 FUNC1=cos(x)
 FUNC2=sin(x)
 FUNC3=cos(2*x)
 FUNC4=sin(2*x)
 FUNC5=cos(3*x)
 FUNC6=sin(3*x)
 MINIMUM=-pi
 MAXIMUM=+pi
 LABEL=bf1
 PERIODIC
... BF_CUSTOM
```

Glossary of keywords and components

Compulsory keywords

MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
PERIODIC	(default=off) Indicate that the basis functions are periodic.
CHECK_NAN_INF	(default=off) Check that the basis functions do not result in a not a number (nan) or infinity (inf).
FUNC	The basis functions f_i(x) given in mathematical expressions using *x* as a variable.. You can use multiple instances of this keyword i.e. FUNC1, FUNC2, FUNC3...
TRANSFORM	An optional function that can be used to transform the argument before calculating the basis function values. You should use *t* as a variable. You can use the variables *min* and *max* to give the minimum and the maximum of the interval.

8.8.2.6 BF_FOURIER

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Fourier basis functions.
Use as basis functions Fourier series defined on a periodic interval. You need to provide the periodic interval $[a, b]$ on which the basis functions are to be used, and the order of the expansion N (i.e. the highest Fourier mode used). The total number of basis functions is $2N + 1$ as for each Fourier mode there is both the cosine and sine term, and the constant $f_0(x) = 1$ is also included. These basis functions should only be used for periodic CVs.
The Fourier series basis functions are given by $f_0(x) = 1$
$f_1(x) = cos(\frac{2\pi}{P}x)$
$f_2(x) = sin(\frac{2\pi}{P}x)$
$f_3(x) = cos(2 \cdot \frac{2\pi}{P}x)$
$f_4(x) = sin(2 \cdot \frac{2\pi}{P}x)$
$\vdots$
$f_{2k-1}(x) = cos(k \cdot \frac{2\pi}{P}x)$
$f_{2k}(x) = sin(k \cdot \frac{2\pi}{P}x)$
$\vdots$
$f_{2N-1}(x) = cos(N \cdot \frac{2\pi}{P}x)$
$f_{2N}(x) = sin(N \cdot \frac{2\pi}{P}x)$
where $P = (b - a)$ is the periodicity of the interval. They are orthogonal over the interval $[a, b]$

$$\int_a^b dx\, f_n(x)\, f_m(x) = \{\, 0\ n \neq m(b-a)n = m = 0(b-a)/2n = m \neq 0.$$

Examples

Here we employ a Fourier expansion of order 10 over the periodic interval $-\pi$ to $+\pi$. This results in a total number of 21 basis functions. The label used to identify the basis function action can then be referenced later on in the input file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_FOURIER.tmp
BF_FOURIER MINIMUM=-pi MAXIMUM=+pi ORDER=10 LABEL=bf_fourier
```

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
NUMERICAL_INTEGRALS	(default=off) Calculate basis function integral for the uniform distribution numerically. Useful for debugging.

8.8.2.7 BF_GAUSSIANS

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Gaussian basis functions.

Attention

> **These basis functions do not form orthogonal bases. We recommend using wavelets (BF_WAVELETS) instead that do form orthogonal bases**.

Basis functions given by Gaussian distributions with shifted centers defined on a bounded interval. See [116] for full details.
You need to provide the interval $[a, b]$ on which the bias is to be expanded. The ORDER keyword of the basis set N determines the number of equally sized sub-intervalls to be used. On the borders of each of these sub-intervalls the mean μ of a Gaussian basis function is placed: $\mu_i = a + (i-1)\frac{b-a}{N}$
The total number of basis functions is $N + 4$ as the constant $f_0(x) = 1$, as well as two additional Gaussians at the Boundaries are also included.
The basis functions are given by $\mathrm{f}_0(x) = 1$
$f_i(x) = \exp\left(-\frac{(x-\mu_i)^2}{2\sigma^2}\right)$
When the Gaussians are used for a periodic CV (with the PERIODIC keyword), the sub-intervals are chosen in the same way, but only $N + 1$ functions are required to fill it (the ones at the boundary coincide and the ones outside can be omitted).
It is possible to specify the width σ (i.e. the standard deviation) of the Gaussians using the WIDTH keyword. By default it is set to the sub-intervall length. It was found that performance can be typically improved with a smaller value (around 75 % of the sub-interval length), although a too small overlap will prevent the basis set from working correctly at all.
The optimization procedure then adjusts the heigths of the individual Gaussians. To avoid 'blind' optimization of the basis functions outside the currently sampled area, it is often beneficial to use the OPTIMIZATION_THRESHOLD keyword of the VES_LINEAR_EXPANSION (set it to a small value, e.g. 1e-6)
As an example two adjacent basis functions (with the mentioned width choice of 75% of the sub-interval length) can be seen below. The full basis consists of shifted Gaussians in the full specified interval.

Examples

The bias is expanded with Gaussian functions in the intervall from 0.0 to 10.0 using order 20. This results in 24 basis functions.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_GAUSSIANS.tmp
bfG: BF_GAUSSIANS MINIMUM=0.0 MAXIMUM=10.0 ORDER=20
```

Because it was not specified, the width of the Gaussians is by default set to the sub-intervall length, i.e. $\sigma = 0.5$. To e.g. enhance the overlap between neighbouring basis functions, it can be specified explicitely:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_GAUSSIANS.tmp
bfG: BF_GAUSSIANS MINIMUM=0.0 MAXIMUM=10.0 ORDER=20 WIDTH=0.7
```

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
PERIODIC	(default=off) Use periodic version of basis set.
WIDTH	The width (i.e. standart deviation) of the Gaussian functions. By default it is equal to the sub-intervall size.

8.8.2.8 BF_LEGENDRE

Legendre polynomials basis functions.
Use as basis functions Legendre polynomials $P_n(x)$ defined on a bounded interval. You need to provide the interval $[a,b]$ on which the basis functions are to be used, and the order of the expansion N (i.e. the highest order polynomial used). The total number of basis functions is $N+1$ as the constant $P_0(x)=1$ is also included. These basis functions should not be used for periodic CVs.
Intrinsically the Legendre polynomials are defined on the interval $[-1,1]$. A variable t in the interval $[a,b]$ is transformed to a variable x in the intrinsic interval $[-1,1]$ by using the transform function

$$x(t) = \frac{t-(a+b)/2}{(b-a)/2}$$

The Legendre polynomials are given by the recurrence relation $\mathsf{P}_0(x) = 1$
$P_1(x) = x$
$P_{n+1}(x) = \frac{2n+1}{n+1}\,x\,P_n(x) - \frac{n}{n+1}\,P_{n-1}(x)$
The first 6 polynomials are shown below
The Legendre polynomial are orthogonal over the interval $[-1,1]$

$$\int_{-1}^{1} dx\,P_n(x)\,P_m(x) = \frac{2}{2n+1}\delta_{n,m}$$

By using the SCALED keyword the polynomials are scaled by a factor of $\sqrt{\frac{2n+1}{2}}$ such that they are orthonormal to 1.
From the above equation it follows that integral of the basis functions over the uniform target distribution $p_{\mathrm{u}}(x)$ are given by

$$\int_{-1}^{1} dx\,P_n(x)p_{\mathrm{u}}(x) = \delta_{n,0},$$

and thus always zero except for the constant $P_0(x)=1$.
For further mathematical properties of the Legendre polynomials see for example the Wikipedia page.

Examples

Here we employ a Legendre expansion of order 20 over the interval -4.0 to 8.0. This results in a total number of 21 basis functions. The label used to identify the basis function action can then be referenced later on in the input file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_LEGENDRE.tmp
bf_leg: BF_LEGENDRE MINIMUM=-4.0 MAXIMUM=8.0 ORDER=20
```

Examples

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
NUMERICAL_INTEGRALS	(default=off) Calculate basis function integral for the uniform distribution numerically. Useful for debugging.
SCALED	(default=off) Scale the polynomials such that they are orthonormal to 1.

8.8.2.9 BF_POWERS

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Polynomial power basis functions.

Attention

> **These basis functions should not be used in conventional biasing simulations**. Instead you should use orthogonal basis functions like Legendre or Chebyshev polynomials. They are only included for usage in ves_md_linearexpansion and some special cases.

Basis functions given by polynomial powers defined on a bounded interval. You need to provide the interval $[a, b]$ on which the basis functions are to be used, and the order of the expansion N (i.e. the highest power used). The total number of basis functions is $N + 1$ as the constant $f_0(x) = 1$ is also included. These basis functions should not be used for periodic CVs.
The basis functions are given by $\mathrm{f}_0(x) = 1$
$f_1(x) = x$
$f_2(x) = x^2$
$\vdots$
$f_n(x) = x^n$
$\vdots$
$f_N(x) = x^N$

Note that these basis functions are **not** orthogonal. In fact the integral over the uniform target distribution blows up as the interval is increased. Therefore they should not be used in conventional biasing simulations. However, they can be useful for usage with ves_md_linearexpansion.

Examples

Here we employ a polynomial power expansion of order 5 over the interval -2.0 to 2.0. This results in a total number of 6 basis functions. The label used to identify the basis function action can then be referenced later on in the input file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_POWERS.tmp
BF_POWERS MINIMUM=-2.0 MAXIMUM=2.0 ORDER=5 LABEL=bf_pow
```

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
NORMALIZATION	The normalization factor that is used to normalize the basis functions. By default it is 1.0.

8.8.2.10 BF_SINE

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Fourier sine basis functions.
Use as basis functions Fourier sine series defined on a periodic interval. You need to provide the periodic interval $[a, b]$ on which the basis functions are to be used, and the order of the expansion N (i.e. the highest Fourier sine mode used). The total number of basis functions is $N + 1$ as the constant $f_0(x) = 1$ is also included. These basis functions should only be used for periodic CVs. They can be useful if the periodic function being expanded is an odd function, i.e. $F(-x) = -F(x)$.
The Fourier sine basis functions are given by $f_0(x) = 1$
$f_1(x) = sin(\frac{2\pi}{P}x)$
$f_2(x) = sin(2 \cdot \frac{2\pi}{P}x)$
$f_3(x) = sin(3 \cdot \frac{2\pi}{P}x)$
$\vdots$
$f_n(x) = sin(n \cdot \frac{2\pi}{P}x)$
$\vdots$
$f_N(x) = sin(N \cdot \frac{2\pi}{P}x)$
where $P = (b - a)$ is the periodicity of the interval. They are orthogonal over the interval $[a, b]$

$$\int_a^b dx\, f_n(x)\, f_m(x) = \{\, 0\ n \neq m(b-a)n = m = 0(b-a)/2n = m \neq 0.$$

Examples

Here we employ a Fourier sine expansion of order 10 over the periodic interval $-\pi$ to $+\pi$. This results in a total number of 11 basis functions. The label used to identify the basis function action can then be referenced later on in the input file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_SINE.tmp
BF_SINE MINIMUM=-pi MAXIMUM=+pi ORDER=10 LABEL=bfS
```

Examples

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
NUMERICAL_INTEGRALS	(default=off) Calculate basis function integral for the uniform distribution numerically. Useful for debugging.

8.8.2.11 BF_WAVELETS

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Daubechies Wavelets basis functions.
Note: at the moment only bases with a single level of scaling functions are usable, as multiscale optimization is not yet implemented.
This basis set uses Daubechies Wavelets [117] to construct a complete and orthogonal basis. See [116] for full details.
The basis set is based on using a pair of functions, the scaling function (or father wavelet) ϕ and the wavelet function (or mother wavelet) ψ. They are defined via the two-scale relations for scale j and shift k:
$\phi_k^j(x) = 2^{-j/2}\phi\left(2^{-j}x - k\right)$
$\psi_k^j(x) = 2^{-j/2}\psi\left(2^{-j}x - k\right)$
The exact properties are set by choosing filter coefficients, e.g. choosing h_k for the father wavelet:

$$\phi(x) = \sqrt{2}\sum_k h_k\,\phi(2x-k)$$

The filter coefficients by Daubechies result in an orthonormal basis of all integer shifted functions:

$$\int \phi(x+i)\phi(x+j)\,\mathrm{d}x = \delta_{ij} \quad for \quad i,j \in \mathbb{Z}$$

Because no analytic formula for these wavelets exist, they are instead constructed iteratively on a grid. The method of construction is close to the "Vector cascade algorithm" described in [118] . The needed filter coefficients of the scaling function are hardcoded, and were previously generated via a python script. Currently the "maximum phase" type (Db) and the "least asymmetric" (Sym) type are implemented. We recommend to use Symlets.
As an example two adjacent basis functions of both Sym8 (ORDER=8, TYPE=SYMLET) and Db8 (ORDER=8, TYPE=DAUBECHIES) is shown in the figure. The full basis consists of shifted wavelets in the full specified interval.

Specify the wavelet type

The TYPE keyword sets the type of Wavelet, at the moment "DAUBECHIES" and "SYMLETS" are available. The specified ORDER of the basis corresponds to the number of vanishing moments of the wavelet, i.e. if TYPE was specified as "DAUBECHIES" an order of 8 results in Db8 wavelets.

Specify the number of functions

The resulting basis set consists of integer shifts of the wavelet with some scaling j,

$$V(x) = \sum_i \alpha_i * \phi_i(x) = \sum_i \alpha_i * \phi(\frac{x+i}{j})$$

with the variational parameters α. Additionally a constant basis function is included.
There are two different ways to specify the number of used basis functions implemented. You can either specify the scale or alternatively a fixed number of basis function.
Coming from the multiresolution aspect of wavelets, you can set the scale of the father wavelets, i.e. the largest scale used for approximation. This can be done with the FUNCTION_LENGTH keyword. It should be given in the same units as the used CV and specifies the length (of the domain interval) of the individual father wavelet functions.
Alternatively a fixed number of basis functions for the bias expansion can be specified with the NUM_BF keyword, which will set the scale automatically to match the desired number of functions. Note that this also includes the constant function.
If you do not specify anything, it is assumed that the range of the bias should match the scale of the wavelet functions. More precise, the basis functions are scaled to match the specified size of the CV space (MINIMUM and MAXIMUM keywords). This has so far been a good initial choice.
If the wavelets are scaled to match the CV range exactly there would be $4 * ORDER - 3$ basis functions whose domain is at least partially in this region. This number is adjusted if FUNCTION_LENGTH or NUM_BF is specified.
Additionally, some of the shifted basis functions will not have significant contributions because of their function values being close to zero over the full range of the bias. These 'tail wavelets' can be omitted by using the TAILS_↩
THRESHOLD keyword. This omits all shifted functions that have only function values smaller than a fraction of their maximum value inside the bias range. Using a value of e.g. 0.01 will already reduce the number of basis functions significantly. The default setting will not omit any tail wavelets (i.e. TAILS_THRESHOLD=0).
The number of basis functions is then not easily determinable a priori but will be given in the logfile. Additionally the starting point (leftmost defined point) of the individual basis functions is printed.
With the PERIODIC keyword the basis set can also be used to bias periodic CVs. Then the shift between the functions will be chosen such that the function at the left border and right border coincide. If the FUNCTION_LE↩
NGTH keyword is used together with PERIODIC, a smaller length might be chosen to satisfy this requirement.

Grid

The values of the wavelet function are generated on a grid. Using the cascade algorithm results in doubling the grid values for each iteration. This means that the grid size will always be a power of two multiplied by the number of coefficients ($2 * ORDER - 1$) for the specified wavelet. Using the MIN_GRID_SIZE keyword a lower bound for the number of grid points can be specified. By default at least 1,000 grid points are used. Function values in between grid points are calculated by linear interpolation.

Optimization notes

To avoid 'blind' optimization of the basis functions outside the currently sampled area, it is often beneficial to use the OPTIMIZATION_THRESHOLD keyword of the VES_LINEAR_EXPANSION (set it to a small value, e.g. 1e-6)

Examples

First a very simple example that relies on the default values. We want to bias some CV in the range of 0 to 4. The wavelets will therefore be scaled to match that range. Using Db8 wavelets this results in 30 basis functions (including the constant one), with their starting points given by $-14 * \frac{4}{15}, -13 * \frac{4}{15}, \cdots, 0, \cdots, 13 * \frac{4}{15}, 14 * \frac{4}{15}$.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_WAVELETS.tmp
BF_WAVELETS ...
 ORDER=8
 TYPE=DAUBECHIES
 MINIMUM=0.0
 MAXIMUM=4.0
 LABEL=bf
... BF_WAVELETS
```

By omitting wavelets with only insignificant parts, we can reduce the number of basis functions. Using a threshold of 0.01 will in this example remove the 8 leftmost shifts, which we can check in the logfile.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_WAVELETS.tmp
BF_WAVELETS ...
 ORDER=8
 TYPE=DAUBECHIES
 MINIMUM=0.0
 MAXIMUM=4.0
 TAILS_THRESHOLD=0.01
 LABEL=bf
... BF_WAVELETS
```

The length of the individual basis functions can also be adjusted to fit the specific problem. If for example the wavelets are instead scaled to length 3, there will be 35 basis functions, with leftmost points at $-14 * \frac{3}{15}, -13 * \frac{3}{15}, \cdots, 0, \cdots, 18 * \frac{3}{15}, 19 * \frac{3}{15}$.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_WAVELETS.tmp
BF_WAVELETS ...
 ORDER=8
 TYPE=DAUBECHIES
 MINIMUM=0.0
 MAXIMUM=4.0
 FUNCTION_LENGTH=3
 LABEL=bf
... BF_WAVELETS
```

Alternatively you can also specify the number of basis functions. Here we specify the usage of 40 Sym10 wavelet functions. We also used a custom minimum size for the grid and want it to be printed to a file with a specific numerical format.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/BF_WAVELETS.tmp
BF_WAVELETS ...
 ORDER=10
 TYPE=SYMLETS
 MINIMUM=0.0
 MAXIMUM=4.0
 NUM_BF=40
 MIN_GRID_SIZE=500
 DUMP_WAVELET_GRID
 WAVELET_FILE_FMT=%11.4f
 LABEL=bf
... BF_WAVELETS
```

Glossary of keywords and components

Compulsory keywords

ORDER	The order of the basis function expansion.
MINIMUM	The minimum of the interval on which the basis functions are defined.
MAXIMUM	The maximum of the interval on which the basis functions are defined.
TYPE	Specify the wavelet type. Currently available are DAUBECHIES Wavelets with minimum phase and the more symmetric SYMLETS

Options

DEBUG_INFO	(default=off) Print out more detailed information about the basis set. Useful for debugging.
MOTHER_WAVELET	(default=off) If this flag is set mother wavelets will be used instead of the scaling function (father wavelet). Makes only sense for multiresolution, which is at the moment not usable.
DUMP_WAVELET_GRID	(default=off) If this flag is set the grid with the wavelet values will be written to a file. This file is called wavelet_grid.data.
PERIODIC	(default=off) Use periodic version of basis set.
FUNCTION_LENGTH	The domain size of the individual basis functions. (length) This is used to alter the scaling of the basis functions. By default it is set to the total size of the interval. This also influences the number of actually used basis functions, as all shifted functions that are partially supported in the CV space are used.
NUM_BF	The number of basis functions that should be used. Includes the constant one and N-1 shifted wavelets within the specified range. Cannot be used together with FUNCTION_LENGTH.
TAILS_THRESHOLD	The threshold for cutting off tail wavelets as a fraction of the maximum value. All shifted wavelet functions that only have values smaller than the threshold in the bias range will be excluded from the basis set. Defaults to 0 (include all).
MIN_GRID_SIZE	The minimal number of grid bins of the Wavelet function. The true number depends also on the used wavelet type and will probably be larger. Defaults to 1000.
WAVELET_FILE_FMT	The number format of the wavelet grid values and derivatives written to file. By default it is %15.8f.

8.8.3 Target Distributions

The following list contains the target distributions available in the VES code.

TD_CHI	Chi distribution (static).
TD_CHISQUARED	Chi-squared distribution (static).
TD_CUSTOM	Target distribution given by an arbitrary mathematical expression (static or dynamic).
TD_EXPONENTIAL	Exponential distribution (static).
TD_EXPONENTIALLY_MODIFIED_GAUSSIAN	Target distribution given by a sum of exponentially modified Gaussian distributions (static).
TD_GAUSSIAN	Target distribution given by a sum of Gaussian kernels (static).
TD_GENERALIZED_EXTREME_VALUE	Generalized extreme value distribution (static).
TD_GENERALIZED_NORMAL	Target distribution given by a sum of generalized normal distributions (static).
TD_GRID	Target distribution from an external grid file (static).
TD_LINEAR_COMBINATION	Target distribution given by linear combination of distributions (static or dynamic).
TD_MULTICANONICAL	Multicanonical target distribution (dynamic).
TD_MULTITHERMAL_MULTIBARIC	Multithermal-multibaric target distribution (dynamic).
TD_PRODUCT_COMBINATION	Target distribution given by product combination of distributions (static or dynamic).
TD_PRODUCT_DISTRIBUTION	Target distribution given by a separable product of one-dimensional distributions (static or dynamic).
TD_UNIFORM	Uniform target distribution (static).

TD_VONMISES	Target distribution given by a sum of Von Mises distributions (static).
TD_WELLTEMPERED	Well-tempered target distribution (dynamic).

8.8.3.1 TD_CHI

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Chi distribution (static).
Employ a target distribution given by a chi distribution that is defined as

$$p(s) = \frac{2^{1-\frac{k}{2}}}{\sigma\,\Gamma\left(\frac{k}{2}\right)} \left(\frac{s-a}{\sigma}\right)^{k-1} \exp\left(-\frac{1}{2}\left(\frac{s-a}{\sigma}\right)^{2}\right),$$

where a is the minimum of the distribution that is defined on the interval $[a,\infty)$, the parameter k (given as a positive integer larger than 1) determines how far the peak of the distribution is from the minimum (known as the "degrees of freedom"), and the parameter $\sigma > 0$ determines the broadness of the distribution.
The minimum a is given using the MINIMUM keyword, the parameter k is given using the KAPPA keyword, and the parameter σ is given using the SIGMA keyword.
This target distribution action is only defined for one dimension, for multiple dimensions it should be used in combination with the TD_PRODUCT_DISTRIBUTION action.

Examples

Chi distribution with $a = 10.0$, $\sigma = 2.0$, and $k = 2$

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CHI.tmp
td: TD_CHI  MINIMUM=10.0  SIGMA=2.0  KAPPA=2
```

The Chi distribution is only defined for one dimension so for multiple dimensions we have to use it in combination with the TD_PRODUCT_DISTRIBUTION action as shown in the following example where we have a uniform distribution for argument 1 and a Chi distribution for argument 1

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CHI.tmp
td_uni: TD_UNIFORM

td_chi: TD_CHI  MINIMUM=-10.0  SIGMA=2.0  KAPPA=2

td_pd: TD_PRODUCT_DISTRIBUTION DISTRIBUTIONS=td_uni,td_chi
```

Glossary of keywords and components

Compulsory keywords

MINIMUM	The minimum of the chi distribution.
SIGMA	The sigma parameter of the chi distribution given as a positive number.
KAPPA	The k parameter of the chi distribution given as positive integer larger than 1.

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.2 TD_CHISQUARED

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Chi-squared distribution (static).
Employ a target distribution given by a chi-squared distribution that is defined as

$$p(s) = \frac{1}{\sigma\, 2^{\frac{k}{2}}\, \Gamma\left(\frac{k}{2}\right)} \left(\frac{s-a}{\sigma}\right)^{\frac{k}{2}-1} \exp\left(-\frac{1}{2}\left(\frac{s-a}{\sigma}\right)\right),$$

where a is the minimum of the distribution that is defined on the interval $[a, \infty)$, the parameter k (given as a positive integer larger than 2) determines how far the peak of the distribution is from the minimum (known as the "degrees of freedom"), and the parameter $\sigma > 0$ determines the broadness of the distribution.
The minimum a is given using the MINIMUM keyword, the parameter k is given using the KAPPA keyword, and the parameter σ is given using the SIGMA keyword.
This target distribution action is only defined for one dimension, for multiple dimensions it should be used in combination with the TD_PRODUCT_DISTRIBUTION action.

Examples

Chi-squared distribution with $a = -10.0$, $\sigma = 2.0$, and $k = 2$

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CHISQUARED.tmp
td: TD_CHISQUARED  MINIMUM=-10.0  SIGMA=2.0  KAPPA=2
```

The Chi-squared distribution is only defined for one dimension so for multiple dimensions we have to use it in combination with the TD_PRODUCT_DISTRIBUTION action as shown in the following example where we have a Chi-squared distribution for argument 1 and uniform distribution for argument 2

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CHISQUARED.tmp
td_chisq: TD_CHISQUARED  MINIMUM=10.0  SIGMA=2.0  KAPPA=2

td_uni: TD_UNIFORM

td_pd: TD_PRODUCT_DISTRIBUTION DISTRIBUTIONS=td_chisq,td_uni
```

Glossary of keywords and components

Compulsory keywords

MINIMUM	The minimum of the chi-squared distribution.
SIGMA	The sigma parameter of the chi-squared distribution given as a positive number.
KAPPA	The k parameter of the chi-squared distribution given as positive integer larger than 2.

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.3 TD_CUSTOM

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by an arbitrary mathematical expression (static or dynamic).
Use as a target distribution the distribution defined by

$$p(\mathbf{s}) = \frac{f(\mathbf{s})}{\int d\mathbf{s}\, f(\mathbf{s})}$$

where $f(\mathbf{s})$ is some arbitrary mathematical function that is parsed by the lepton library.
The function $f(\mathbf{s})$ is given by the FUNCTION keywords by using *s1*,*s2*,..., as variables for the arguments $\mathbf{s} = (s_1, s_2, \ldots, s_d)$. If one variable is not given the target distribution will be taken as uniform in that argument.
It is also possible to include the free energy surface $F(\mathbf{s})$ in the target distribution by using the *FE* variable. In this case the target distribution is dynamic and needs to be updated with current best estimate of $F(\mathbf{s})$, similarly as for the well-tempered target distribution. Furthermore, the inverse temperature $\beta = (k_{\mathrm{B}}T)^{-1}$ and the thermal energy $k_{\mathrm{B}}T$ can be included by using the *beta* and k_BT variables.
The target distribution will be automatically normalized over the region on which it is defined on. Therefore, the function given in FUNCTION needs to be non-negative and it must be possible to normalize the function. The code will perform checks to make sure that this is indeed the case.

Examples

Here we use as shifted Maxwell–Boltzmann distribution as a target distribution in one-dimension. Note that it is not need to include the normalization factor as the distribution will be automatically normalized.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CUSTOM.tmp
TD_CUSTOM ...
 FUNCTION=(s1+20)^2*exp(-(s1+20)^2/(2*10.0^2))
 LABEL=td
... TD_CUSTOM
```

Here we have a two dimensional target distribution where we use a generalized normal distribution for argument s_2 while the distribution for s_1 is taken as uniform as the variable *s1* is not included in the function.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CUSTOM.tmp
TD_CUSTOM ...
 FUNCTION=exp(-(abs(s2-20.0)/5.0)^4.0)
 LABEL=td
... TD_CUSTOM
```

By using the *FE* variable the target distribution can depend on the free energy surface $F(\mathrm{s})$. For example, the following input is identical to using TD_WELLTEMPERED with a bias factor of 10.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CUSTOM.tmp
TD_CUSTOM ...
 FUNCTION=exp(-(beta/10.0)*FE)
 LABEL=td
... TD_CUSTOM
```

Here the inverse temperature is automatically obtained by using the *beta* variable. It is also possible to use the k_BT variable. The following syntax will give the exact same results as the syntax above

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_CUSTOM.tmp
TD_CUSTOM ...
 FUNCTION=exp(-(1.0/(kBT*10.0))*FE)
 LABEL=td
... TD_CUSTOM
```

Glossary of keywords and components

Compulsory keywords

FUNCTION	The function you wish to use for the target distribution where you should use the variables *s1*,*s2*,... for the arguments. You can also use the current estimate of the FES by using the variable *FE* and the temperature by using the k_BT and *beta* variables.

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.4 TD_EXPONENTIAL

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Exponential distribution (static).
Employ a target distribution given by an exponential distribution that is defined as

$$p(s) = \lambda e^{-\lambda(s-a)}$$

where a is the minimum of the distribution that is defined on the interval $[a, \infty)$, and $\lambda > 0$ is the so-called rate parameter.

The minimum a is given using the MINIMUM keyword, and the rate parameter λ is given using the LAMBDA keyword. This target distribution action is only defined for one dimension, for multiple dimensions it should be used in combination with TD_PRODUCT_DISTRIBUTION action.

Examples

Exponential distribution with $a = 10.0$ and $\lambda = 0.5$

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_EXPONENTIAL.tmp
td: TD_EXPONENTIAL  MINIMUM=-10.0  LAMBDA=0.5
```

The exponential distribution is only defined for one dimension so for multiple dimensions we have to use it in combination with the TD_PRODUCT_DISTRIBUTION action as shown in the following example where we have a uniform distribution for argument 1 and and an exponential distribution for argument 2

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_EXPONENTIAL.tmp
td_uni: TD_UNIFORM

td_exp: TD_EXPONENTIAL  MINIMUM=-10.0  LAMBDA=0.5

td_pd: TD_PRODUCT_DISTRIBUTION DISTRIBUTIONS=td_uni,td_exp
```

Glossary of keywords and components

Compulsory keywords

MINIMUM	The minimum of the exponential distribution.
LAMBDA	The lambda parameter of the exponential distribution given as positive number.

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.5 TD_EXPONENTIALLY_MODIFIED_GAUSSIAN

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by a sum of exponentially modified Gaussian distributions (static).
Employ a target distribution that is given by a sum where each term is a product of one-dimensional

exponentially modified Gaussian distributions,

$$p(\mathbf{s}) = \sum_i w_i \prod_k^d \frac{\lambda_{k,i}}{2} \exp\left[\frac{\lambda_{k,i}}{2}(2\mu_{k,i} + \lambda_{k,i}\sigma_{k,i}^2 - 2s_k)\right] \operatorname{erfc}\left[\frac{\mu_{k,i} + \lambda_{k,i}\sigma_{k,i}^2 - s_k)}{\sqrt{2}\sigma_{k,i}}\right]$$

where $(\mu_{1,i}, \mu_{2,i}, \ldots, \mu_{d,i})$ are the centers of the Gaussian component, $(\sigma_{1,i}, \sigma_{2,i}, \ldots, \sigma_{d,i})$ are the standard deviations of the Gaussian component, $(\lambda_{1,i}, \lambda_{2,i}, \ldots, \lambda_{d,i})$ are the rate parameters of the exponential component, and $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$ is the complementary error function. The weights w_i are normalized to 1, $\sum_i w_i = 1$. The centers $(\mu_{1,i}, \mu_{2,i}, \ldots, \mu_{d,i})$ are given using the numbered CENTER keywords, the standard deviations $(\sigma_{1,i}, \sigma_{2,i}, \ldots, \sigma_{d,i})$ using the the numbered SIGMA keywords, and the rate parameters $(\lambda_{1,i}, \lambda_{2,i}, \ldots, \lambda_{d,i})$ using the numbered LAMBDA keywords. The weights are given using the WEIGHTS keywords, if no weights are given are all terms weighted equally.

Examples

An exponentially modified Gaussian distribution in one-dimension

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_EXPONENTIALLY_MODIFIED_GAUSSIAN.tmp
td1: TD_EXPONENTIALLY_MODIFIED_GAUSSIAN CENTER1=-10.0 SIGMA1=1.0 LAMBDA1=0.25
```

A sum of two one-dimensional exponentially modified Gaussian distributions

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_EXPONENTIALLY_MODIFIED_GAUSSIAN.tmp
TD_EXPONENTIALLY_MODIFIED_GAUSSIAN ...
 CENTER1=-10.0 SIGMA1=1.0 LAMBDA1=0.5
 CENTER2=+10.0 SIGMA2=1.0 LAMBDA2=1.0
 WEIGHTS=2.0,1.0
 LABEL=td1
... TD_EXPONENTIALLY_MODIFIED_GAUSSIAN
```

A sum of two two-dimensional exponentially modified Gaussian distributions

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_EXPONENTIALLY_MODIFIED_GAUSSIAN.tmp
TD_EXPONENTIALLY_MODIFIED_GAUSSIAN ...
 CENTER1=-5.0,+5.0 SIGMA1=1.0,1.0 LAMBDA1=0.5,0.5
 CENTER2=+5.0,+5.0 SIGMA2=1.0,1.0 LAMBDA2=1.0,1.0
 WEIGHTS=1.0,1.0
 LABEL=td1
... TD_EXPONENTIALLY_MODIFIED_GAUSSIAN
```

Glossary of keywords and components

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
CENTER	The center of each exponentially modified Gaussian distributions.. You can use multiple instances of this keyword i.e. CENTER1, CENTER2, CENTER3...
SIGMA	The sigma parameters for each exponentially modified Gaussian distributions.. You can use multiple instances of this keyword i.e. SIGMA1, SIGMA2, SIGM↩ A3...
LAMBDA	The lambda parameters for each exponentially modified Gaussian distributions. You can use multiple instances of this keyword i.e. LAMBDA1, LAMBDA2, L↩ AMBDA3...

WEIGHTS	The weights of the distributions. By default all are weighted equally.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.6 TD_GAUSSIAN

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by a sum of Gaussian kernels (static).
Employ a target distribution that is given by a sum of multivariate Gaussian (or normal) distributions, defined as

$$p(\mathbf{s}) = \sum_i w_i \, N(\mathbf{s}; \mu_i, \mathbf{\Sigma}_i)$$

where $\mu_i = (\mu_{1,i}, \mu_{2,i}, \ldots, \mu_{d,i})$ and $\mathbf{\Sigma}_i$ are the center and the covariance matrix for the i-th Gaussian. The weights w_i are normalized to 1, $\sum_i w_i = 1$.
By default the Gaussian distributions are considered as separable into independent one-dimensional Gaussian distributions. In other words, the covariance matrix is taken as diagonal $\mathbf{\Sigma}_i = (\sigma^2_{1,i}, \sigma^2_{2,i}, \ldots, \sigma^2_{d,i})$. The Gaussian distribution is then written as

$$N(\mathbf{s}; \mu_i, \sigma_i) = \prod_k^d \frac{1}{\sqrt{2\pi\sigma^2_{d,i}}} \exp\left(-\frac{(s_d - \mu_{d,i})^2}{2\sigma^2_{d,i}}\right)$$

where $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, \ldots, \sigma_{d,i})$ is the standard deviation. In this case you need to specify the centers μ_i using the numbered CENTER keywords and the standard deviations σ_i using the numbered SIGMA keywords.
For two arguments it is possible to employ `bivariate Gaussian kernels` with correlation between arguments, defined as

$$N(\mathbf{s}; \mu_i, \sigma_i, \rho_i) = \frac{1}{2\pi\sigma_{1,i}\sigma_{2,i}\sqrt{1-\rho_i^2}} \exp\left(-\frac{1}{2(1-\rho_i^2)}\left[\frac{(s_1-\mu_{1,i})^2}{\sigma^2_{1,i}} + \frac{(s_2-\mu_{2,i})^2}{\sigma^2_{2,i}} - \frac{2\rho_i(s_1-\mu_{1,i})(s_2-\mu_{2,i})}{\sigma_{1,i}\sigma_{2,i}}\right]\right)$$

where ρ_i is the correlation between s_1 and s_2 that goes from -1 to 1. In this case the covariance matrix is given as

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{1,i} & \rho_i\sigma_{1,i}\sigma_{2,i} \\ \rho_i\sigma_{1,i}\sigma_{2,i} & \sigma^2_{2,i} \end{bmatrix}$$

The correlation ρ is given using the numbered CORRELATION keywords. A value of $\rho = 0$ means that the arguments are considered as un-correlated, which is the default behavior.
The Gaussian distributions are always defined with the conventional normalization factor such that they are normalized to 1 over an unbounded region. However, in calculation within VES we normally consider bounded region on which the target distribution is defined. Thus, if the center of a Gaussian is close to the boundary of the region it can happen that the tails go outside the region. In that case it might be needed to use the NORMALIZE keyword to make sure that the target distribution is properly normalized to 1 over the bounded region. The code will issue a warning if that is needed.
For periodic CVs it is generally better to use Von Mises distributions instead of Gaussian kernels as these distributions properly account for the periodicity of the CVs.

Examples

One single Gaussian kernel in one-dimension.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GAUSSIAN.tmp
td: TD_GAUSSIAN CENTER1=-1.5 SIGMA1=0.8
```

Sum of three Gaussian kernels in two-dimensions with equal weights as no weights are given.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GAUSSIAN.tmp
TD_GAUSSIAN ...
 CENTER1=-1.5,+1.5 SIGMA1=0.8,0.3
 CENTER2=+1.5,-1.5 SIGMA2=0.3,0.8
 CENTER3=+1.5,+1.5 SIGMA3=0.4,0.4
 LABEL=td
... TD_GAUSSIAN
```

Sum of three Gaussian kernels in two-dimensions which are weighted unequally. Note that weights are automatically normalized to 1 so that WEIGHTS=1.0,2.0,1.0 is equal to specifying WEIGHTS=0.25,0.50,0.25.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GAUSSIAN.tmp
TD_GAUSSIAN ...
 CENTER1=-1.5,+1.5 SIGMA1=0.8,0.3
 CENTER2=+1.5,-1.5 SIGMA2=0.3,0.8
 CENTER3=+1.5,+1.5 SIGMA3=0.4,0.4
 WEIGHTS=1.0,2.0,1.0
 LABEL=td
... TD_GAUSSIAN
```

Sum of two bivariate Gaussian kernels where there is correlation of $\rho_2 = 0.75$ between the two arguments for the second Gaussian.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GAUSSIAN.tmp
TD_GAUSSIAN ...
 CENTER1=-1.5,+1.5 SIGMA1=0.8,0.3
 CENTER2=+1.5,-1.5 SIGMA2=0.3,0.8 CORRELATION2=0.75
 LABEL=td
... TD_GAUSSIAN
```

Glossary of keywords and components

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
CENTER	The centers of the Gaussian distributions.. You can use multiple instances of this keyword i.e. CENTER1, CENTER2, CENTER3...
SIGMA	The standard deviations of the Gaussian distributions.. You can use multiple instances of this keyword i.e. SIGMA1, SIGMA2, SIGMA3...
CORRELATION	The correlation for two-dimensional bivariate Gaussian distributions. Only works for two arguments. The value should be between -1 and 1. If no value is given the Gaussian kernels is considered as un-correlated (i.e. value of 0.0).. You can use multiple instances of this keyword i.e. CORRELATION1, CORR↩ ELATION2, CORRELATION3...
WEIGHTS	The weights of the Gaussian distributions. Have to be as many as the number of centers given with the numbered CENTER keywords. If no weights are given the distributions are weighted equally. The weights are automatically normalized to 1.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.7 TD_GENERALIZED_EXTREME_VALUE

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Generalized extreme value distribution (static).
Employ a target distribution given by a generalized extreme value distribution that is defined as

$$p(s) = \frac{1}{\sigma}\, t(s)^{\xi+1}\, e^{-t(s)},$$

where

$$t(s) = \{ \quad (1+\xi\left(\frac{s-\mu}{\sigma}\right)^{-1/\xi} \text{ if } \xi \neq 0 \exp\left(-\frac{s-\mu}{\sigma}\right) \text{ if } \xi = 0$$

, and μ is the location parameter which approximately determines the location of the maximum of the distribution, $\sigma > 0$ is the scale parameter that determines the broadness of the distribution, and ξ is the shape parameter that determines the tail behavior of the distribution. For $\xi = 0$, $\xi > 0$, and $\xi < 0$ the Gumbel, Frechet, and Weibull families of distributions are obtained, respectively.
The location parameter μ is given using the LOCATION keyword, the scale parameter σ using the SCALE keyword, and the shape parameter ξ using the SHAPE keyword.
This target distribution action is only defined for one dimension, for multiple dimensions it should be used in combination with TD_PRODUCT_DISTRIBUTION action.

Examples

Generalized extreme value distribution with $\mu = 0.0$, $\sigma = 2.0$, and $\xi = 0.0$ (Gumbel distribution)

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GENERALIZED_EXTREME_VALUE.tmp
td: TD_GENERALIZED_EXTREME_VALUE  LOCATION=0.0  SCALE=2.0 SHAPE=0.0
```

Generalized extreme value distribution with $\mu = -5.0$, $\sigma = 1.0$, and $\xi = 0.5$ (Frechet distribution)

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GENERALIZED_EXTREME_VALUE.tmp
td: TD_GENERALIZED_EXTREME_VALUE  LOCATION=-5.0  SCALE=1.0 SHAPE=0.5
```

Generalized extreme value distribution with $\mu = 5.0$, $\sigma = 2.0$, and $\xi = -0.5$ (Weibull distribution)

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GENERALIZED_EXTREME_VALUE.tmp
td: TD_GENERALIZED_EXTREME_VALUE  LOCATION=5.0  SCALE=1.0 SHAPE=-0.5
```

The generalized extreme value distribution is only defined for one dimension so for multiple dimensions we have to use it in combination with the TD_PRODUCT_DISTRIBUTION action as shown in the following example where we have a Generalized extreme value distribution for argument 1 and uniform distribution for argument 2

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GENERALIZED_EXTREME_VALUE.tmp
td_gev: TD_GENERALIZED_EXTREME_VALUE  LOCATION=-5.0  SCALE=1.0 SHAPE=0.5

td_uni: TD_UNIFORM

td_pd: TD_PRODUCT_DISTRIBUTION DISTRIBUTIONS=td_gev,td_uni
```

Glossary of keywords and components

Compulsory keywords

LOCATION	The mu parameter of the generalized extreme value distribution.
SCALE	The sigma parameter for the generalized extreme value distribution given as a positive number.
SHAPE	The xi parameter for the generalized extreme value distribution.

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.8 TD_GENERALIZED_NORMAL

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by a sum of generalized normal distributions (static).
Employ a target distribution that is given by a sum where each term is a product of one-dimensional generalized normal distributions (version 1, also know as an exponential power distribution), defined as

$$p(\mathbf{s}) = \sum_i w_i \prod_k^d \frac{\beta_{k,i}}{2\,\alpha_{k,i}\,\Gamma(1/\beta_{k,i})} \exp\left(-\left|\frac{s_k - \mu_{k,i}}{\alpha_{k,i}}\right|^{\beta_{k,i}}\right)$$

where $(\mu_{1,i}, \mu_{2,i}, \ldots, \mu_{d,i})$ are the centers of the distributions, $(\alpha_{1,i}, \alpha_{2,i}, \ldots, \alpha_{d,i})$ are the scale parameters of the distributions, $(\beta_{1,i}, \beta_{2,i}, \ldots, \beta_{d,i})$ are the shape parameters of the distributions, and $\Gamma(x)$ is the gamma function. The weights w_i are normalized to 1, $\sum_i w_i = 1$.
Employing $\beta = 2$ results in a Gaussian (normal) distributions with mean μ and variance $\alpha^2/2$, $\beta = 1$ gives the Laplace distribution, and the limit $\beta \to \infty$ results in a uniform distribution on the interval $[\mu - \alpha, \mu + \alpha]$.
The centers $(\mu_{1,i}, \mu_{2,i}, \ldots, \mu_{d,i})$ are given using the numbered CENTER keywords, the scale parameters $(\alpha_{1,i}, \alpha_{2,i}, \ldots, \alpha_{d,i})$ using the numbered SCALE keywords, and the shape parameters $(\beta_{1,i}, \beta_{2,i}, \ldots, \beta_{d,i})$ using the numbered SHAPE keywords. The weights are given using the WEIGHTS keywords, if no weights are given are all terms weighted equally.

Examples

A generalized normal distribution in one-dimensional

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GENERALIZED_NORMAL.tmp
td1: TD_GENERALIZED_NORMAL CENTER1=+20.0  ALPHA1=5.0  BETA1=4.0
```

A sum of two one-dimensional generalized normal distributions

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GENERALIZED_NORMAL.tmp
TD_GENERALIZED_NORMAL ...
 CENTER1=+20.0  ALPHA1=5.0  BETA1=4.0
 CENTER2=-20.0  ALPHA2=5.0  BETA2=3.0
 LABEL=td1
... TD_GENERALIZED_NORMAL
```

A sum of two two-dimensional generalized normal distributions

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GENERALIZED_NORMAL.tmp
TD_GENERALIZED_NORMAL ...
 CENTER1=-20.0,-20.0 ALPHA1=5.0,3.0 BETA1=2.0,4.0
 CENTER2=-20.0,+20.0 ALPHA2=3.0,5.0 BETA2=4.0,2.0
 WEIGHTS=2.0,1.0
 LABEL=td1
... TD_GENERALIZED_NORMAL
```

Glossary of keywords and components

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
CENTER	The center of each generalized normal distribution.. You can use multiple instances of this keyword i.e. CENTER1, CENTER2, CENTER3...
ALPHA	The alpha parameters for each generalized normal distribution.. You can use multiple instances of this keyword i.e. ALPHA1, ALPHA2, ALPHA3...
BETA	The beta parameters for each generalized normal distribution.. You can use multiple instances of this keyword i.e. BETA1, BETA2, BETA3...
WEIGHTS	The weights of the generalized normal distribution. By default all are weighted equally.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.9 TD_GRID

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution from an external grid file (static).
Using this keyword you can use a target distribution that is read from an external grid file that is in the proper PLUMED file format. You do not to give any information about the external grid file as all relevant information should be automatically detected. It is assumed that the distribution read in from the grid is a proper probability distribution, i.e. always non-negative and can be normalized.
By default the target distribution from the external grid is always normalized inside the code. You can disable this normalization by using DO_NOT_NORMALIZE keyword. However, be warned that this will generally lead to the wrong behavior if the distribution from the external grid is not properly normalized to 1.
If the distribution from the external grid file has for some reason negative values can you use the SHIFT keyword to shift the distribution by a given value. Another option is to use the SHIFT_TO_ZERO keyword to shift the minimum of the distribution to zero.
Note that the number of grid bins used in the external grid file do not have to be the same as used in the bias or action where the target distribution is employed as the code will employ a linear (or bilinear for two dimensions) interpolation to calculate values. Currently only one or two dimensional grids are supported.
It can happen that the intervals on which the target distribution is defined is larger than the intervals covered by the external grid file. In this case the default option is to consider the target distribution as continuous such that values outside the boundary of the external grid file are the same as at the boundary. This can be changed by using the ZERO_OUTSIDE keyword which will make values outside to be taken as zero.

Examples

Generally you only need to provide the the filename of the external grid file.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_GRID.tmp
td: TD_GRID FILE=input-grid.data
```

The input grid is then specified using the usual format employed by PLUMED an example of which is shown below:

```
#! FIELDS d1 external.bias der_d1
#! SET min_d1 1.14
#! SET max_d1 1.32
#! SET nbins_d1 6
#! SET periodic_d1 false
   1.1400   0.0031   0.1101
   1.1700   0.0086   0.2842
   1.2000   0.0222   0.6648
   1.2300   0.0521   1.4068
   1.2600   0.1120   2.6873
   1.2900   0.2199   4.6183
   1.3200   0.3948   7.1055
```

Glossary of keywords and components

Compulsory keywords

FILE	The name of the external grid file to be used as a target distribution.

Options

ZERO_OUTSIDE	(default=off) By default the target distribution is continuous such that values outside the boundary of the external grid file are the same as at the boundary. This can be changed by using this flag which will make values outside to be taken as zero.
DO_NOT_NORMALIZE	(default=off) By default the target distribution from the external grid is always normalized inside the code. You can use this flag to disable this normalization. However, be warned that this will generally lead to the wrong behavior if the distribution from the external grid is not properly normalized to 1.
SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
SHIFT	Shift the grid read in by some constant value. Due to normalization the final shift in the target distribution will generally not be the same as the value given here
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.10 TD_LINEAR_COMBINATION

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by linear combination of distributions (static or dynamic).

Employ a target distribution that is a linear combination of the other distributions, defined as

$$p(\mathbf{s}) = \sum_i w_i \, p_i(\mathbf{s})$$

where the weights w_i are normalized to 1, $\sum_i w_i = 1$.
The labels of the distributions $p_i(\mathbf{s})$ to be used in the linear combination are given in the DISTRIBUTIONS keyword. The weights w_i can be given using the WEIGHTS keyword. The distributions are weighted equally if no weights are given.
It is assumed that all the distributions $p_i(\mathbf{s})$ are normalized. If that is not the case for some reason should you normalize each distribution separately by using the NORMALIZE keyword when defining them in the input file (i.e. before the TD_LINEAR_COMBINATION action). Note that normalizing the overall linear combination will generally lead to different results than normalizing each distribution separately.
The linear combination will be a dynamic target distribution if one or more of the distributions used is a dynamic distribution, otherwise it will be a static distribution.

Examples

Here we employ a linear combination of a uniform and a Gaussian distribution. No weights are given so the two distributions will be weighted equally.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_LINEAR_COMBINATION.tmp
td_uni: TD_UNIFORM

td_gauss: TD_GAUSSIAN CENTER1=-2.0 SIGMA1=0.5

td_comb: TD_LINEAR_COMBINATION DISTRIBUTIONS=td_uni,td_gauss
```

Here we employ a linear combination of a uniform and two Gaussian distribution. The weights are automatically normalized to 1 such that giving WEIGHTS=1.0,1.0,2.0 as we do here is equal to giving WEIGHTS=0.25,0.25,0.50.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_LINEAR_COMBINATION.tmp
td_uni: TD_UNIFORM

td_gauss1: TD_GAUSSIAN CENTER1=-2.0,-2.0 SIGMA1=0.5,0.3

td_gauss2: TD_GAUSSIAN CENTER1=+2.0,+2.0 SIGMA1=0.3,0.5

TD_LINEAR_COMBINATION ...
 DISTRIBUTIONS=td_uni,td_gauss1,td_gauss2
 WEIGHTS=1.0,1.0,2.0
 LABEL=td_comb
... TD_LINEAR_COMBINATION
```

In the above example the two Gaussian kernels are given using two separate DISTRIBUTION keywords. As the TD_GAUSSIAN target distribution allows multiple centers is it also possible to use just one DISTRIBUTION keyword for the two Gaussian kernels. This is shown in the following example which will give the exact same result as the one above as the weights have been appropriately adjusted

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_LINEAR_COMBINATION.tmp
td_uni: TD_UNIFORM

TD_GAUSSIAN ...
 CENTER1=-2.0,-2.0  SIGMA1=0.5,0.3
 CENTER2=+2.0,+2.0  SIGMA2=0.3,0.5
 WEIGHTS=1.0,2.0
 LABEL=td_gauss
... TD_GAUSSIAN

TD_LINEAR_COMBINATION ...
 DISTRIBUTIONS=td_uni,td_gauss
 WEIGHTS=0.25,0.75
 LABEL=td_comb
... TD_LINEAR_COMBINATION
```

Glossary of keywords and components

Compulsory keywords

DISTRIBUTIONS	The labels of the target distribution actions to be used in the linear combination.

Options

NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
WEIGHTS	The weights of target distributions. Have to be as many as the number of target distribution labels given in DISTRIBUTIONS. If no weights are given the distributions are weighted equally. The weights are automatically normalized to 1.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.11 TD_MULTICANONICAL

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Multicanonical target distribution (dynamic).
Use the target distribution to sample the multicanonical ensemble [119] [38]. In this way, in a single molecular dynamics simulation one can obtain information about the system in a range of temperatures. This range is determined through the keywords MIN_TEMP and MAX_TEMP.
The collective variables (CVs) used to construct the bias potential must be:

1. the energy or,
2. the energy and an order parameter.

Other choices of CVs or a different order of the above mentioned CVs are nonsensical. The second CV, the order parameter, must be used when one aims at studying a first order phase transition in the chosen temperature interval [27].
The algorithm will explore the free energy at each temperature up to a predefined free energy threshold ϵ specified through the keyword THRESHOLD (in kT units). If only the energy is biased, i.e. no phase transition is considered, then THRESHOLD can be set to around 5. If also an order parameter is used then the THRESHOLD should be greater than the barrier for the transformation in kT. For small systems undergoing a freezing transition THRESH↩
OLD is typically between 20 and 50.
When only the potential energy is used as CV the method is equivalent to the Wang-Landau algorithm [36]. The advantage with respect to Wang-Landau is that instead of sampling the potential energy indiscriminately, an interval is chosen on the fly based on the minimum and maximum targeted temperatures.
The algorithm works as follows. The target distribution for the potential energy is chosen to be:

$$p(E) = \begin{cases} \frac{1}{E_2 - E_1} & \text{if} \quad E_1 < E < E_2 \\ 0 & \text{otherwise} \end{cases}$$

where the energy limits E_1 and E_2 are yet to be determined. Clearly the interval $E_1 \breve{} E_2$ chosen is related to the interval of temperatures $T_1 - T_2$. To link these two intervals we make use of the following relation:

$$\beta' F_{\beta'}(E) = \beta F_\beta(E) + (\beta' - \beta)E + C,$$

where $F_\beta(E)$ is determined during the optimization and we shall choose C such that $F_{\beta'}(E_m) = 0$ with E_m the position of the free energy minimum. Using this relation we employ an iterative procedure to find the energy interval.

At iteration k we have the estimates E_1^k and E_2^k for E_1 and E_2, and the target distribution is:

$$p^k(E) = \frac{1}{E_2^k - E_1^k} \quad \text{for} \quad E_1^k < E < E_2^k.$$

E_1^k and E_2^k are obtained from the leftmost solution of $\beta_2 F_{\beta_2}^{k-1}(E_1^k) = \epsilon$ and the rightmost solution of $\beta_1 F_{\beta_1}^{k-1}(E_2^k) = \epsilon$. The procedure is repeated until convergence. This iterative approach is similar to that in TD_WELLTEMPERED.
The version of this algorithm in which the energy and an order parameter are biased is similar to the one described in TD_MULTITHERMAL_MULTIBARIC.
The output of these simulations can be reweighted in order to obtain information at all temperatures in the targeted temperature interval. The reweighting can be performed using the action REWEIGHT_TEMP_PRESS.

Examples

The following input can be used to run a simulation in the multicanonical ensemble. The temperature interval to be explored is 400-600 K. The energy is used as collective variable. Legendre polynomials are used to construct the bias potential. The averaged stochastic gradient descent algorithm is chosen to optimize the VES functional. The target distribution is updated every 100 optimization steps (200 ps here) using the last estimation of the free energy.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_MULTICANONICAL.tmp
# Use energy and volume as CVs
energy: ENERGY

# Basis functions
bf1: BF_LEGENDRE ORDER=20 MINIMUM=-25000 MAXIMUM=-23500

# Target distributions
TD_MULTICANONICAL ...
 LABEL=td_multi
 MIN_TEMP=400
 MAX_TEMP=600
... TD_MULTICANONICAL

# Expansion
VES_LINEAR_EXPANSION ...
 ARG=energy
 BASIS_FUNCTIONS=bf1
 TEMP=500.0
 GRID_BINS=1000
 TARGET_DISTRIBUTION=td_multi
 LABEL=b1
... VES_LINEAR_EXPANSION

# Optimization algorithm
OPT_AVERAGED_SGD ...
  BIAS=b1
  STRIDE=500
  LABEL=o1
  STEPSIZE=1.0
  FES_OUTPUT=500
  BIAS_OUTPUT=500
  TARGETDIST_OUTPUT=500
  COEFFS_OUTPUT=10
  TARGETDIST_STRIDE=100
... OPT_AVERAGED_SGD
```

The multicanonical target distribution can also be used to explore a temperature interval in which a first order phase transitions is observed.

Glossary of keywords and components

Compulsory keywords

THRESHOLD	(default=5) Maximum exploration free energy in kT.
EPSILON	(default=10) The zeros of the target distribution are changed to e^-EPSILON.
MIN_TEMP	Minimum temperature.
MAX_TEMP	Maximum temperature.

Options

STEPS_TEMP	Number of temperature steps. Only for the 2D version, i.e. energy and order parameter.
SIGMA	The standard deviation parameters of the Gaussian kernels used for smoothing the target distribution. One value must be specified for each argument, i.e. one value per CV. A value of 0.0 means that no smoothing is performed, this is the default behavior.

8.8.3.12 TD_MULTITHERMAL_MULTIBARIC

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Multithermal-multibaric target distribution (dynamic).
Use the target distribution to sample the multithermal-multibaric ensemble [38] [120]. In this way, in a single molecular dynamics simulation one can obtain information about the simulated system in a range of temperatures and pressures. This range is determined through the keywords MIN_TEMP, MAX_TEMP, MIN_PRESSURE, and MA↩
X_PRESSURE. One should also specified the target pressure of the barostat with the keyword PRESSURE.
The collective variables (CVs) used to construct the bias potential must be:

1. the potential energy and the volume or,
2. the potential energy, the volume, and an order parameter.

Other choices of CVs or a different order of the above mentioned CVs are nonsensical. The third CV, the order parameter, must be used when the region of the phase diagram under study is crossed by a first order phase transition [27] .
The algorithm will explore the free energy at each temperature and pressure up to a predefined free energy threshold ϵ specified through the keyword THRESHOLD (in kT units). If only the energy and the volume are being biased, i.e. no phase transition is considered, then THRESHOLD can be set to around 5. If also an order parameter is used then the THRESHOLD should be greater than the barrier for the transformation in kT. For small systems undergoing a freezing transition THRESHOLD is typically between 20 and 50.
It is also important to specify the number of intermediate temperatures and pressures to consider. This is done through the keywords STEPS_TEMP and STEPS_PRESSURE. If the number of intermediate temperature and pressures is too small, then holes might appear in the target distribution. If it is too large, the performance will deteriorate with no additional advantage.
We now describe the algorithm more rigorously. The target distribution is given by

$$p(E,\mathcal{V},s) = \{\, 1 \,/\Omega_{E,\mathcal{V},s} if there is at least one \beta', P' such that \beta' F_{\beta',P'}(E,\mathcal{V},s) < \epsilon with \beta_1 > \beta' > \beta_2 and P_1 < P' < P_2 0 otherwis$$

with $F_{\beta',P'}(E,\mathcal{V},s)$ the free energy as a function of energy E and volume $\mathcal{V}$ (and optionally the order parameter s) at temperature β' and pressure P', $\Omega_{E,\mathcal{V},s}$ is a normalization constant, and ϵ is the THRESHOLD. In practice the condition $\beta' F_{\beta',P'}(E,\mathcal{V},s) < \epsilon$ is checked in equally spaced points in each dimension β' and P'. The number of points is determined with the keywords STEPS_TEMP and STEPS_PRESSURE. In practice the target distribution is never set to zero but rather to a small value controlled by the keyword EPSILON. The small value is e^-EPSILON.
Much like in the Wang-Landau algorithm [36] or in the multicanonical ensemble [119] , a flat histogram is targeted. The idea behind this choice of target distribution is that all regions of potential energy and volume (and optionally order parameter) that are relevant at all temperatures $\beta_1 < \beta' < \beta_2$ and pressure $P_1 < P' < P_2$ are included in the distribution.

The free energy at temperature β' and pressure P' is calculated from the free energy at β and P using:

$$\beta' F_{\beta',P'}(E,\mathcal{V},s) = \beta F_{\beta,P}(E,\mathcal{V},s) + (\beta' - \beta)E + (\beta' P' - \beta P)\mathcal{V} + C$$

with C such that $F_{\beta',P'}(E_m,\mathcal{V}_m,s_m) = 0$ with $E_m,\mathcal{V}_m,s_m$ the position of the free energy minimum. $\beta F_{\beta,P}(E,\mathcal{V},s)$ is not know from the start and is instead found during the simulation. Therefore $p(E,\mathcal{V},s)$ is determined iteratively as done in the well tempered target distribution [121].
The output of these simulations can be reweighted in order to obtain information at all temperatures and pressures in the targeted region of Temperature-Pressure plane. The reweighting can be performed using the action REWEIGHT_TEMP_PRESS.
The multicanonical ensemble (fixed volume) can be targeted using TD_MULTICANONICAL.

Examples

The following input can be used to run a simulation in the multithermal-multibaric ensemble. The region of the temperature-pressure plane that will be explored is 260-350 K and 1 bar- 300 MPa. The energy and the volume are used as collective variables. Legendre polynomials are used to construct the two dimensional bias potential. The averaged stochastic gradient descent algorithm is chosen to optimize the VES functional. The target distribution is updated every 100 optimization steps (200 ps here) using the last estimation of the free energy.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_MULTITHERMAL_MULTIBARIC.tmp
# Use energy and volume as CVs
energy: ENERGY
vol: VOLUME

# Basis functions
bf1: BF_LEGENDRE ORDER=10 MINIMUM=-14750 MAXIMUM=-12250
bf2: BF_LEGENDRE ORDER=10 MINIMUM=6.5 MAXIMUM=8.25

# Target distribution - 1 bar = 0.06022140857 and 300 MPa = 180.66422571
TD_MULTITHERMAL_MULTIBARIC ...
 MIN_TEMP=260
 MAX_TEMP=350
 MAX_PRESSURE=180.66422571
 MIN_PRESSURE=0.06022140857
 PRESSURE=0.06022140857
 LABEL=td_multi
... TD_MULTITHERMAL_MULTIBARIC

# Bias expansion
VES_LINEAR_EXPANSION ...
 ARG=energy,vol
 BASIS_FUNCTIONS=bf1,bf2
 TEMP=300.0
 GRID_BINS=200,200
 TARGET_DISTRIBUTION=td_multi
 LABEL=b1
... VES_LINEAR_EXPANSION

# Optimization algorithm
OPT_AVERAGED_SGD ...
  BIAS=b1
  STRIDE=500
  LABEL=o1
  STEPSIZE=1.0
  FES_OUTPUT=500
  BIAS_OUTPUT=500
  TARGETDIST_OUTPUT=500
  COEFFS_OUTPUT=100
  TARGETDIST_STRIDE=100
... OPT_AVERAGED_SGD
```

The multithermal-multibaric target distribution can also be used to explore regions of the phase diagram crossed by first order phase transitions. Consider a system of 250 atoms that crystallizes in the FCC crystal structure. The region of the temperature-pressure plane that will be explored is 350-450 K and 1bar-1GPa. We assume that inside this region we can find the liquid-FCC coexistence line that we would like to obtain. In this case in addition to the energy and volume, an order parameter must also be biased. The energy, volume, and an order parameter are

used as collective variables to construct the bias potential. We choose as order parameter the FCCUBIC. Legendre polynomials are used to construct the three dimensional bias potential. The averaged stochastic gradient descent algorithm is chosen to optimize the VES functional. The target distribution is updated every 100 optimization steps (200 ps here) using the last estimation of the free energy.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_MULTITHERMAL_MULTIBARIC.tmp
# Use energy, volume and FCCUBIC as CVs
energy: ENERGY
vol: VOLUME
fcc: FCCUBIC SPECIES=1-256 SWITCH={CUBIC D_0=0.4 D_MAX=0.5} MORE_THAN={RATIONAL R_0=0.45 NN=12 MM=24}

# Basis functions
bf1: BF_LEGENDRE ORDER=8 MINIMUM=-26500 MAXIMUM=-23500
bf2: BF_LEGENDRE ORDER=8 MINIMUM=8.0 MAXIMUM=11.5
bf3: BF_LEGENDRE ORDER=8 MINIMUM=0.0 MAXIMUM=256.0

# Target distribution
TD_MULTITHERMAL_MULTIBARIC ...
 LABEL=td_multitp
 MIN_TEMP=350.0
 MAX_TEMP=450.0
 MIN_PRESSURE=0.06022140857
 MAX_PRESSURE=602.2140857
 PRESSURE=301.10704285
 SIGMA=250.0,0.1,10.0
 THRESHOLD=15
 STEPS_TEMP=20
 STEPS_PRESSURE=20
... TD_MULTITHERMAL_MULTIBARIC

# Expansion
VES_LINEAR_EXPANSION ...
 ARG=energy,vol,fcc.morethan
 BASIS_FUNCTIONS=bf1,bf2,bf3
 TEMP=400.0
 GRID_BINS=40,40,40
 TARGET_DISTRIBUTION=td_multitp
 LABEL=b1
... VES_LINEAR_EXPANSION

# Optimization algorithm
OPT_AVERAGED_SGD ...
  BIAS=b1
  STRIDE=500
  LABEL=o1
  STEPSIZE=1.0
  FES_OUTPUT=500
  BIAS_OUTPUT=500
  TARGETDIST_OUTPUT=500
  COEFFS_OUTPUT=100
  TARGETDIST_STRIDE=500
... OPT_AVERAGED_SGD
```

Glossary of keywords and components

Compulsory keywords

THRESHOLD	(default=5) Maximum exploration free energy in kT.
EPSILON	(default=10) The zeros of the target distribution are changed to e^-EPSILON.
MIN_TEMP	Minimum energy.
MAX_TEMP	Maximum energy.
MIN_PRESSURE	Minimum pressure.
MAX_PRESSURE	Maximum pressure.

PRESSURE	Target pressure of the barostat used in the MD engine.
STEPS_TEMP	(default=20) Number of temperature steps.
STEPS_PRESSURE	(default=20) Number of pressure steps.

Options

SIGMA	The standard deviation parameters of the Gaussian kernels used for smoothing the target distribution. One value must be specified for each argument, i.e. one value per CV. A value of 0.0 means that no smoothing is performed, this is the default behavior.

8.8.3.13 TD_PRODUCT_COMBINATION

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by product combination of distributions (static or dynamic).
Employ a target distribution that is a product combination of the other distributions, defined as

$$p(\mathbf{s}) = \frac{\prod_i p_i(\mathbf{s})}{\int d\mathbf{s} \prod_i p_i(\mathbf{s})}$$

where the distributions $p_i(\mathbf{s})$ are in full dimensional space of the arguments used.
Note the difference between this target distribution and the one defined in TD_PRODUCT_DISTRIBUTION. Here we have a non-separable distribution given as a product of distribution $p_i(\mathbf{s})$ which are in full dimensional space of the arguments used.
The labels of the distributions $p_i(\mathbf{s})$ to be used in the product combination are given in the DISTRIBUTIONS keyword.
The target distribution resulting from the product combination will be automatically normalized. Therefore, the product combination needs to be a proper distribution that is non-negative and that can be normalized. The code will perform checks to make sure that this is indeed the case.
The product combination will be a dynamic target distribution if one or more of the distributions used is a dynamic distribution. Otherwise it will be a static distribution.

Examples

In the following example the overall interval on which the target distribution is defined is from 0.23 to 0.8. We employ a product combination of a well-tempered distribution and a uniform distribution that decays to zero at 0.6. This results in a target distribution that is well-tempered from 0.23 to 0.6 and then decays to zero. In other words, we cut off the tail of the well-tempered distribution at 0.6

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_PRODUCT_COMBINATION.tmp
td_welltemp: TD_WELLTEMPERED BIASFACTOR=5
td_uniform: TD_UNIFORM MINIMA=0.23 MAXIMA=0.6 SIGMA_MAXIMA=0.05
td_combination: TD_PRODUCT_COMBINATION DISTRIBUTIONS=td_uniform,td_welltemp
```

In the following example the overall interval on which the target distribution is defined is from -4 to 4. We employ a product of a Gaussian distribution with two centers and distribution that is uniform on the interval -3 to 3 and then smoothly decays to zero outside that interval. The overall effect will then be to cut off the tails of the Gaussian distribution

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_PRODUCT_COMBINATION.tmp
TD_GAUSSIAN ...
 CENTER1=-2.9 SIGMA1=1.0
 CENTER2=+2.9 SIGMA2=0.4
 LABEL=td_gauss
... TD_GAUSSIAN
```

```
TD_UNIFORM ...
 MINIMA=-3.0 SIGMA_MINIMA=0.20
 MAXIMA=+3.0 SIGMA_MAXIMA=0.15
 LABEL=td_uni
... TD_UNIFORM

td_pc: TD_PRODUCT_COMBINATION DISTRIBUTIONS=td_gauss,td_uni
```

Glossary of keywords and components

Compulsory keywords

DISTRIBUTIONS	The labels of the target distribution actions to be used in the product combination.

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.14 TD_PRODUCT_DISTRIBUTION

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by a separable product of one-dimensional distributions (static or dynamic).
Employ a target distribution that is a separable product of one-dimensional distributions, defined as

$$p(\mathbf{s}) = \prod_{k}^{d} p_k(s_k)$$

where d is the number of arguments used and $p_k(s_k)$ is the one-dimensional distribution corresponding to the k-th argument.
Note the difference between this target distribution and the one defined in TD_PRODUCT_COMBINATION. Here we have a separable distribution given as a product of one-dimensional distribution $p_k(s_k)$.
The labels of the one-dimensional distributions $p_k(s_k)$ to be used in the product distribution are given in the DIS↩TRIBUTIONS keyword. Note that the order of the labels is very important.
It is assumed that all the distributions to be used in the product distribution are normalized. If that is not the case you need to normalize the distributions by using the NORMALIZE keyword. Here it does not matter if you normalize each distribution separately or the overall product, it will give the same results.
The product distribution will be a dynamic target distribution if one or more of the distributions used is a dynamic distribution. Otherwise it will be a static distribution.

Examples

In the following example we employ a uniform distribution for argument 1 and a Gaussian distribution for argument 2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_PRODUCT_DISTRIBUTION.tmp
target_uniform: TD_UNIFORM

target_Gaussian: TD_GAUSSIAN CENTER1=-2.0 SIGMA1=0.5

td_pd: TD_PRODUCT_DISTRIBUTION DISTRIBUTIONS=target_uniform,target_Gaussian
```

Note that order of the labels is important, using DISTRIBUTIONS=target_Gaussian,target_uniform would mean that we would employ a Gaussian distribution for argument 1 and a uniform distribution for argument 2, which would lead to completely different results.

Glossary of keywords and components

Compulsory keywords

DISTRIBUTIONS	Labels of the one-dimensional target distribution actions for each argument to be used in the product distribution. Note that order of the labels is important.

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
NORMALIZE	(default=off) Renormalized the target distribution over the intervals on which it is defined to make sure that it is properly normalized to 1. In most cases this should not be needed as the target distributions should be normalized. The code will issue a warning (but still run) if this is needed for some reason.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.15 TD_UNIFORM

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Uniform target distribution (static).
Using this keyword you can define a uniform target distribution which is a product of one-dimensional distributions $p_k(s_k)$ that are uniform over a given interval $[a_k, b_k]$

$$p_k(s_k) = \begin{cases} \frac{1}{(b_k - a_k)} & \text{if } a_k \leq s_k \leq b_k \\ 0 & \text{otherwise} \end{cases}$$

The overall distribution is then given as

$$p(\mathbf{s}) = \prod_k^d p_k(s_k) = \begin{cases} \prod_k^d \frac{1}{(b_k - a_k)} & \text{if } a_k \leq s_k \leq b_k \text{ for all } k \\ 0 & \text{otherwise} \end{cases}$$

The distribution is thus uniform inside a rectangular for two arguments and a cube for a three arguments.
The limits of the intervals a_k and b_k are given with the MINIMA and MAXIMA keywords, respectively. If one or both of these keywords are missing the code should automatically detect the limits.
It is also possible to use one-dimensional distributions that go smoothly to zero at the boundaries. This is done by employing a function with Gaussian switching functions at the boundaries a_k and b_k

$$f_k(s_k) = \{ \exp\left(-\frac{(s_k - a_k)^2}{2\sigma_{a,k}^2}\right) \text{if } s_k < a_k 1 \text{if } a_k \leq s_k \leq b_k \exp\left(-\frac{(s_k - b_k)^2}{2\sigma_{b,k}^2}\right) \text{if } s_k > b_k$$

where the standard deviation parameters $\sigma_{a,k}$ and $\sigma_{b,k}$ determine how quickly the switching functions goes to zero. The overall distribution is then normalized

$$p(\mathbf{s}) = \prod_k^d p_k(s_k) = \prod_k^d \frac{f(s_k)}{\int ds_k \, f(s_k)}$$

To use this option you need to provide the standard deviation parameters $\sigma_{a,k}$ and $\sigma_{b,k}$ by using the SIGMA_MI↩ NIMA and SIGMA_MAXIMA keywords, respectively. Giving a value of 0.0 means that the boundary is sharp, which is the default behavior.

Examples

If one or both of the MINIMA or MAXIMA keywords are missing the code should automatically detect the limits not given. Therefore, if we consider a target distribution that is defined over an interval from 0.0 to 10.0 for the first argument and from 0.2 to 1.0 for the second argument are the following example

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
td: TD_UNIFORM
```

is equivalent to this one

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
TD_UNIFORM ...
 MINIMA=0.0,0.2
 MAXIMA=10.0,1.0
 LABEL=td
 ... TD_UNIFORM
```

and this one

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
td: TD_UNIFORM  MAXIMA=10.0,1.0
```

and also this one

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
td: TD_UNIFORM MINIMA=0.0,0,2
```

We can also define a target distribution that goes smoothly to zero at the boundaries of the uniform distribution. In the following we consider an interval of 0 to 10 for the target distribution. The following input would result in a target distribution that would be uniform from 2 to 7 and then smoothly go to zero from 2 to 0 and from 7 to 10.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
TD_UNIFORM ...
 MINIMA=2.0
 MAXIMA=+7.0
 SIGMA_MINIMA=0.5
 SIGMA_MAXIMA=1.0
 LABEL=td
... TD_UNIFORM
```

It is also possible to employ a smooth switching function for just one of the boundaries as shown here where the target distribution would be uniform from 0 to 7 and then smoothly go to zero from 7 to 10.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
TD_UNIFORM ...
 MAXIMA=+7.0
 SIGMA_MAXIMA=1.0
 LABEL=td
... TD_UNIFORM
```

Furthermore, it is possible to employ a sharp boundary by using

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
TD_UNIFORM ...
 MAXIMA=+7.0
 SIGMA_MAXIMA=0.0
 LABEL=td
... TD_UNIFORM
```

or

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_UNIFORM.tmp
td: TD_UNIFORM MAXIMA=+7.0
```

Glossary of keywords and components

Options

MINIMA	The minimum of the intervals where the target distribution is taken as uniform. You should give one value for each argument.
MAXIMA	The maximum of the intervals where the target distribution is taken as uniform. You should give one value for each argument.
SIGMA_MINIMA	The standard deviation parameters of the Gaussian switching functions for the minima of the intervals. You should give one value for each argument. Value of 0.0 means that switch is done without a smooth switching function, this is the default behavior.
SIGMA_MAXIMA	The standard deviation parameters of the Gaussian switching functions for the maximum of the intervals. You should give one value for each argument. Value of 0.0 means that switch is done without a smooth switching function, this is the default behavior.

8.8.3.16 TD_VONMISES

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target distribution given by a sum of Von Mises distributions (static).
Employ a target distribution that is given by a sum where each term is a product of one-dimensional Von Mises distributions,

$$p(\mathbf{s}) = \sum_i w_i \prod_k^d \frac{\exp\left(\kappa_{k,i}\,\cos(s_k - \mu_{k,i})\right)}{2\pi I_0(\kappa_{k,i})}$$

where $(\mu_{1,i}, \mu_{2,i}, \ldots, \mu_{d,i})$ are the centers of the distributions, $(\kappa_{1,i}, \kappa_{2,i}, \ldots, \kappa_{d,i})$ are parameters that determine the extend of each distribution, and $I_0(x)$ is the modified Bessel function of order 0. The weights w_i are normalized to 1, $\sum_i w_i = 1$.
The Von Mises distribution is defined for periodic variables with a periodicity of 2π and is analogous to the Gaussian distribution. The parameter $\sqrt{1/\kappa}$ is comparable to the standard deviation σ for the Gaussian distribution.

To use this target distribution you need to give the centers $(\mu_{1,i}, \mu_{2,i}, \ldots, \mu_{d,i})$ by using the numbered CENTER keywords and the "standard deviations" $(\sqrt{1/\kappa_{1,i}}, \sqrt{1/\kappa_{2,i}}, \ldots, \sqrt{1/\kappa_{d,i}})$ using the numbered SIGMA keywords.

Examples

Sum of two Von Mises distribution in one dimension that have equal weights as no weights are given.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_VONMISES.tmp
TD_VONMISES ...
 CENTER1=+2.0 SIGMA1=0.6
 CENTER2=-2.0 SIGMA2=0.7
 LABEL=td
... TD_VONMISES
```

Sum of two Von Mises distribution in two dimensions that have different weights. Note that the weights are automatically normalized to 1 such that specifying WEIGHTS=1.0,2.0 is equal to specifying WEIGHTS=0.33333,0.66667.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_VONMISES.tmp
TD_VONMISES ...
 CENTER1=+2.0,+2.0 SIGMA1=0.6,0.7
 CENTER2=-2.0,+2.0 SIGMA2=0.7,0.6
 WEIGHTS=1.0,2.0
 LABEL=td
... TD_VONMISES
```

Glossary of keywords and components

Options

SHIFT_TO_ZERO	(default=off) Shift the minimum value of the target distribution to zero. This can for example be used to avoid negative values in the target distribution. If this option is active the distribution will be automatically normalized.
CENTER	The centers of the Von Mises distributions.. You can use multiple instances of this keyword i.e. CENTER1, CENTER2, CENTER3...
SIGMA	The standard deviations of the Von Mises distributions.. You can use multiple instances of this keyword i.e. SIGMA1, SIGMA2, SIGMA3...
WEIGHTS	The weights of the Von Mises distributions. Have to be as many as the number of centers given with the numbered CENTER keywords. If no weights are given the distributions are weighted equally. The weights are automatically normalized to 1.
WELLTEMPERED_FACTOR	Broaden the target distribution such that it is taken as [p(s)]^(1/gamma) where gamma is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

8.8.3.17 TD_WELLTEMPERED

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Well-tempered target distribution (dynamic).
Use as a target distribution the well-tempered distribution [49] given by

$$p(\mathbf{s}) = \frac{e^{-(\beta/\gamma)F(\mathbf{s})}}{\int d\mathbf{s}\, e^{-(\beta/\gamma)F(\mathbf{s})}} = \frac{[P_0(\mathbf{s})]^{1/\gamma}}{\int d\mathbf{s}\, [P_0(\mathbf{s})]^{1/\gamma}}$$

where γ is a so-called bias factor and $P_0(\mathbf{s})$ is the unbiased canonical distribution of the CVs. This target distribution thus corresponds to a biased ensemble where, as compared to the unbiased one, the probability peaks have been broaden and the fluctuations of the CVs are enhanced. The value of the bias factor γ determines by how much the fluctuations are enhanced.
The well-tempered distribution can be view as sampling on an effective free energy surface $\tilde{F}(\mathbf{s}) = (1/\gamma)F(\mathbf{s})$ which has largely the same metastable states as the original $F(\mathbf{s})$ but with barriers that have been reduced by a factor of γ. Generally one should use a value of γ that results in effective barriers on the order of few $k_{\mathrm{B}}T$ such that thermal fluctuations can easily induce transitions between different metastable states.
At convergence the relationship between the bias potential and the free energy surface is given by

$$F(\mathbf{s}) = -\left(\frac{1}{1-\gamma^{-1}}\right) V(\mathbf{s})$$

This target distribution depends directly on the free energy surface $F(\mathbf{s})$ which is quantity that we do not know a-priori and want to obtain. Therefore, this target distribution is iteratively updated [121] according to

$$p^{(m+1)}(\mathbf{s}) = \frac{e^{-(\beta/\gamma)F^{(m+1)}(\mathbf{s})}}{\int d\mathbf{s}\, e^{-(\beta/\gamma)F^{(m+1)}(\mathbf{s})}}$$

where $F^{(m+1)}(\mathbf{s})$ is the current best estimate of the free energy surface obtained according to

$$F^{(m+1)}(\mathbf{s}) = -V^{(m+1)}(\mathbf{s}) - \frac{1}{\beta}\log p^{(m)}(\mathbf{s}) = -V^{(m+1)}(\mathbf{s}) + \frac{1}{\gamma}F^{(m)}(\mathbf{s})$$

The frequency of performing this update needs to be set in the optimizer used in the calculation. Normally it is sufficient to do it every 100-1000 bias update iterations.

Examples

Employ a well-tempered target distribution with a bias factor of 10

```
BEGIN_PLUMED_FILE working DATADIR=example-check/TD_WELLTEMPERED.tmp
td_welltemp: TD_WELLTEMPERED BIASFACTOR=10
```

Glossary of keywords and components

Compulsory keywords

BIASFACTOR	The bias factor used for the well-tempered distribution.

8.8.4 Optimizers

The following list contains the optimizers available in the VES code.

OPT_ADAM	Adaptive moment estimation (ADAM) optimizer.
OPT_AVERAGED_SGD	Averaged stochastic gradient decent with fixed step size.
OPT_DUMMY	Dummy optimizer for debugging.
OPT_ROBBINS_MONRO_SGD	Robbins-Monro stochastic gradient decent.

8.8.4.1 OPT_ADAM

	This is part of the ves module

It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Adaptive moment estimation (ADAM) optimizer.

Attention

This optimizer is still experimental and not fully documented. The syntax might change. Restarting does not work. We recommend to use the averaged stochastic gradient decent optimizer (OPT_AVERAGED_SGD) for now.

Examples

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Keyword	Description
gradrms	**MONITOR_INSTANTANEOUS_GRADIENT**	the root mean square value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradrms-#.
gradmax	**MONITOR_INSTANTANEOUS_GRADIENT**	the largest absolute value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradmax-#.

Compulsory keywords

BIAS	the label of the VES bias to be optimized
STRIDE	the frequency of updating the coefficients given in the number of MD steps.
COEFFS_FILE	(default=coeffs.data) the name of output file for the coefficients
COEFFS_OUTPUT	(default=100) how often the coefficients should be written to file. This parameter is given as the number of iterations.
STEPSIZE	the step size used for the optimization

Options

MONITOR_INSTANTANEOUS_GRADIENT	(default=off) if quantities related to the instantaneous gradient should be outputted.
MULTIPLE_WALKERS	(default=off) if optimization is to be performed using multiple walkers connected via MPI

AMSGRAD	(default=off) Use the AMSGrad variant
COEFFS_FMT	specify format for coefficient file(s) (useful for decrease the number of digits in regtests)
COEFFS_SET_ID_PREFIX	suffix to add to the filename given in FILE to identify the bias, should only be given if a single filename is given in FILE when optimizing multiple biases.
INITIAL_COEFFS	the name(s) of file(s) with the initial coefficients
TARGETDIST_AVERAGES_FILE	the name of output file for the target distribution averages. By default it is targetdist-averages.data.
TARGETDIST_AVERAGES_OUTPUT	how often the target distribution averages should be written out to file. Note that the value is given in terms of coefficient iterations. If no value is given are the averages only written at the beginning of the optimization
BIAS_OUTPUT	how often the bias(es) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_PROJ_OUTPUT	how often the projections of the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time
MASK_FILE	read in a mask file which allows one to employ different step sizes for different coefficients and/or deactivate the optimization of certain coefficients (by putting values of 0.0). One can write out the resulting mask by using the OUTPUT_MASK_FILE keyword.
OUTPUT_MASK_FILE	Name of the file to write out the mask resulting from using the MASK_FILE keyword. Can also be used to generate a template mask file.
TARGETDIST_STRIDE	stride for updating a target distribution that is iteratively updated during the optimization. Note that the value is given in terms of coefficient iterations.
TARGETDIST_OUTPUT	how often the dynamic target distribution(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
TARGETDIST_PROJ_OUTPUT	how often the projections of the dynamic target distribution(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
BETA_1	Parameter for the first moment estimate. Defaults to 0.9
BETA_2	Parameter for the second moment estimate. Defaults to 0.999
EPSILON	Small parameter to avoid division by zero. Defaults to 1e-8
ADAMW_WEIGHT_DECAY	Weight decay parameter for the AdamW variant. Defaults to 0

8.8.4.2 OPT_AVERAGED_SGD

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Averaged stochastic gradient decent with fixed step size.

Algorithm

This optimizer updates the coefficients according to the averaged stochastic gradient decent algorithm described in ref [122]. This algorithm considers two sets of coefficients, the so-called instantaneous coefficients that are updated according to the recursion formula given by

$$\alpha^{(n+1)} = \alpha^{(n)} - \mu \left[\nabla\Omega(\bar{\alpha}^{(n)}) + \mathbf{H}(\bar{\alpha}^{(n)})[\alpha^{(n)} - \bar{\alpha}^{(n)}] \right],$$

where μ is a fixed step size and the gradient $\nabla\Omega(\bar{\alpha}^{(n)})$ and the Hessian $\mathbf{H}(\bar{\alpha}^{(n)})$ depend on the averaged coefficients defined as

$$\bar{\alpha}^{(n)} = \frac{1}{n+1} \sum_{k=0}^{n} \alpha^{(k)}.$$

This means that the bias acting on the system depends on the averaged coefficients $\bar{\alpha}^{(n)}$ which leads to a smooth convergence of the bias and the estimated free energy surface. Furthermore, this allows for a rather short sampling time for each iteration, for classical MD simulations typical sampling times are on the order of few ps (around 1000-4000 MD steps).
Currently it is only supported to employ the diagonal part of the Hessian which is generally sufficient. Support for employing the full Hessian will be added later on.
The VES bias that is to be optimized should be specified using the BIAS keyword. The fixed step size μ is given using the STEPSIZE keyword. The frequency of updating the coefficients is given using the STRIDE keyword where the value is given in the number of MD steps. For example, if the MD time step is 0.02 ps and STRIDE=2000 will the coefficients be updated every 4 ps. The coefficients will be outputted to the file given by the COEFFS_FILE keyword. How often the coefficients are written to this file is controlled by the COEFFS_OUTPUT keyword.
If the VES bias employs a dynamic target distribution that needs to be iteratively updated (e.g. TD_WELLTEMPERED) [121], you will need to specify the stride for updating the target distribution by using the TARGETDIST_STRIDE keyword where the stride is given in terms coefficient iterations. For example if the MD time step is 0.02 ps and STRIDE=1000, such that the coefficients are updated every 2 ps, will TARGETDIST_STRIDE=500 mean that the target distribution will be updated every 1000 ps.
The output of the free energy surfaces and biases is controlled by the FES_OUTPUT and the BIAS_OUTPUT keywords. It is also possible to output one-dimensional projections of the free energy surfaces by using the FES↩ _PROJ_OUTPUT keyword but for that to work you will need to select for which argument to do the projections by using the numbered PROJ_ARG keyword in the VES bias that is optimized. You can also output dynamic target distributions by using the TARGETDIST_OUTPUT and TARGETDIST_PROJ_OUTPUT keywords.
It is possible to start the optimization from some initial set of coefficients that have been previously obtained by using the INITIAL_COEFFS keyword.
When restarting simulations it should be sufficient to put the RESTART action in the beginning of the input files (or some MD codes the PLUMED should automatically detect if it is a restart run) and keep the same input as before The restarting of the optimization should be automatic as the optimizer will then read in the coefficients from the file given in COEFFS_FILE. For dynamic target distribution the code will also read in the final target distribution from the previous run (which is always outputted even if the TARGETDIST_OUTPUT keyword is not used).
This optimizer supports the usage of multiple walkers where different copies of the system share the same bias potential (i.e. coefficients) and cooperatively sample the averages needed for the gradient and Hessian. This can significantly help with convergence in difficult cases. It is of course best to start the different copies from different positions in CV space. To activate this option you just need to add the MULTIPLE_WALKERS flag. Note that this is only supported if the MD code support running multiple replicas connected via MPI.
The optimizer supports the usage of a so-called mask file that can be used to employ different step sizes for different coefficients and/or deactivate the optimization of certain coefficients (by putting values of 0.0). The mask file is read in by using the MASK_FILE keyword and should be in the same format as the coefficient file. It is possible to generate a template mask file by using the OUTPUT_MASK_FILE keyword.

Examples

In the following input we employ an averaged stochastic gradient decent with a fixed step size of 1.0 and update the coefficient every 1000 MD steps (e.g. every 2 ps if the MD time step is 0.02 ps). The coefficient are outputted to the coefficients.data every 50 iterations while the FES and bias is outputted to files every 500 iterations (e.g. every 1000 ps).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPT_AVERAGED_SGD.tmp
phi:   TORSION ATOMS=5,7,9,15

bf1: BF_FOURIER ORDER=5 MINIMUM=-pi MAXIMUM=pi

VES_LINEAR_EXPANSION ...
 ARG=phi
 BASIS_FUNCTIONS=bf1
 LABEL=ves1
 TEMP=300.0
 GRID_BINS=100
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
  BIAS=ves1
  STRIDE=1000
  LABEL=o1
  STEPSIZE=1.0
  COEFFS_FILE=coefficients.data
  COEFFS_OUTPUT=50
  FES_OUTPUT=500
  BIAS_OUTPUT=500
... OPT_AVERAGED_SGD
```

In the following example we employ a well-tempered target distribution that is updated every 500 iterations (e.g. every 1000 ps). The target distribution is also output to a file every 2000 iterations (the TARGETDIST_OUTPUT keyword). Here we also employ MULTIPLE_WALKERS flag to enable the usage of multiple walkers.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPT_AVERAGED_SGD.tmp
#SETTINGS NREPLICAS=2
phi:   TORSION ATOMS=5,7,9,15
psi:   TORSION ATOMS=7,9,15,17

bf1: BF_FOURIER ORDER=5 MINIMUM=-pi MAXIMUM=pi
bf2: BF_FOURIER ORDER=4 MINIMUM=-pi MAXIMUM=pi

td1: TD_WELLTEMPERED BIASFACTOR=10

VES_LINEAR_EXPANSION ...
 ARG=phi,psi
 BASIS_FUNCTIONS=bf1,bf2
 LABEL=ves1
 TEMP=300.0
 GRID_BINS=100,100
 TARGET_DISTRIBUTION=td1
 PROJ_ARG1=phi
 PROJ_ARG2=psi
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
  BIAS=ves1
  STRIDE=1000
  LABEL=o1
  STEPSIZE=1.0
  MULTIPLE_WALKERS
  COEFFS_FILE=coefficients.data
  COEFFS_OUTPUT=50
  FES_OUTPUT=500
  FES_PROJ_OUTPUT=500
  BIAS_OUTPUT=500
  TARGETDIST_STRIDE=500
  TARGETDIST_OUTPUT=2000
... OPT_AVERAGED_SGD
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Keyword	Description
gradrms	**MONITOR_INSTANTANEOUS_GRADIENT**	the root mean square value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradrms-#.
gradmax	**MONITOR_INSTANTANEOUS_GRADIENT**	the largest absolute value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradmax-#.
avergradrms	**MONITOR_AVERAGE_GRADIENT**	the root mean square value of the averaged coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradrms-#.
avergradmax	**MONITOR_AVERAGE_GRADIENT**	the largest absolute value of the averaged coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradmax-#.

Compulsory keywords

BIAS	the label of the VES bias to be optimized
STRIDE	the frequency of updating the coefficients given in the number of MD steps.
COEFFS_FILE	(default=coeffs.data) the name of output file for the coefficients
COEFFS_OUTPUT	(default=100) how often the coefficients should be written to file. This parameter is given as the number of iterations.
STEPSIZE	the step size used for the optimization

Options

MONITOR_INSTANTANEOUS_GRADIENT	(default=off) if quantities related to the instantaneous gradient should be outputted.
MULTIPLE_WALKERS	(default=off) if optimization is to be performed using multiple walkers connected via MPI
START_OPTIMIZATION_AFRESH	(default=off) if the iterations should be started afresh when a restart has been triggered by the RESTART keyword or the MD code.
MONITOR_AVERAGE_GRADIENT	(default=off) if the averaged gradient should be monitored and quantities related to it should be outputted.
COEFFS_FMT	specify format for coefficient file(s) (useful for decrease the number of digits in regtests)

COEFFS_SET_ID_PREFIX	suffix to add to the filename given in FILE to identify the bias, should only be given if a single filename is given in FILE when optimizing multiple biases.
INITIAL_COEFFS	the name(s) of file(s) with the initial coefficients
TARGETDIST_AVERAGES_FILE	the name of output file for the target distribution averages. By default it is targetdist-averages.data.
TARGETDIST_AVERAGES_OUTPUT	how often the target distribution averages should be written out to file. Note that the value is given in terms of coefficient iterations. If no value is given are the averages only written at the beginning of the optimization
BIAS_OUTPUT	how often the bias(es) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_PROJ_OUTPUT	how often the projections of the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time
MASK_FILE	read in a mask file which allows one to employ different step sizes for different coefficients and/or deactivate the optimization of certain coefficients (by putting values of 0.0). One can write out the resulting mask by using the OUTPUT_MASK_FILE keyword.
OUTPUT_MASK_FILE	Name of the file to write out the mask resulting from using the MASK_FILE keyword. Can also be used to generate a template mask file.
MONITOR_AVERAGES_GRADIENT_EXP_DECAY	use an exponentially decaying averaging with a given time constant when monitoring the averaged gradient
TARGETDIST_STRIDE	stride for updating a target distribution that is iteratively updated during the optimization. Note that the value is given in terms of coefficient iterations.
TARGETDIST_OUTPUT	how often the dynamic target distribution(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
TARGETDIST_PROJ_OUTPUT	how often the projections of the dynamic target distribution(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
EXP_DECAYING_AVER	calculate the averaged coefficients using exponentially decaying averaging using the decaying constant given here in the number of iterations

8.8.4.3 OPT_DUMMY

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Dummy optimizer for debugging.
This is dummy optimizer that can be used for debugging. It will not update the coefficients but can be used to monitor the gradient and Hessian for a given VES bias.

Examples

In the following input we use the OPT_DUMMY to monitor the gradient and Hessian for a given VES bias every 1 iteration.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPT_DUMMY.tmp
phi:   TORSION ATOMS=5,7,9,15

bf1: BF_FOURIER ORDER=5 MINIMUM=-pi MAXIMUM=pi

VES_LINEAR_EXPANSION ...
 ARG=phi
 BASIS_FUNCTIONS=bf1
 LABEL=ves1
 TEMP=300.0
 GRID_BINS=100
... VES_LINEAR_EXPANSION

OPT_DUMMY ...
  BIAS=ves1
  STRIDE=1000
  LABEL=o1
  MONITOR_HESSIAN
  GRADIENT_FILE=gradient.data
  GRADIENT_OUTPUT=1
  GRADIENT_FMT=%12.6f
  HESSIAN_FILE=hessian.data
  HESSIAN_OUTPUT=1
  HESSIAN_FMT=%12.6f
... OPT_DUMMY
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Keyword	Description
gradrms	**MONITOR_INSTANTANEOUS_GRADIENT**	the root mean square value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradrms-#.
gradmax	**MONITOR_INSTANTANEOUS_GRADIENT**	the largest absolute value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradmax-#.
avergradrms	**MONITOR_AVERAGE_GRADIENT**	the root mean square value of the averaged coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradrms-#.
avergradmax	**MONITOR_AVERAGE_GRADIENT**	the largest absolute value of the averaged coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradmax-#.

Compulsory keywords

BIAS	the label of the VES bias to be optimized
STRIDE	the frequency of updating the coefficients given in the number of MD steps.
COEFFS_FILE	(default=coeffs.data) the name of output file for the coefficients
COEFFS_OUTPUT	(default=100) how often the coefficients should be written to file. This parameter is given as the number of iterations.

Options

MONITOR_INSTANTANEOUS_GRADIENT	(default=off) if quantities related to the instantaneous gradient should be outputted.
MULTIPLE_WALKERS	(default=off) if optimization is to be performed using multiple walkers connected via MPI
MONITOR_AVERAGE_GRADIENT	(default=off) if the averaged gradient should be monitored and quantities related to it should be outputted.
MONITOR_HESSIAN	(default=off) also monitor the Hessian
COEFFS_FMT	specify format for coefficient file(s) (useful for decrease the number of digits in regtests)
COEFFS_SET_ID_PREFIX	suffix to add to the filename given in FILE to identify the bias, should only be given if a single filename is given in FILE when optimizing multiple biases.
INITIAL_COEFFS	the name(s) of file(s) with the initial coefficients
TARGETDIST_AVERAGES_FILE	the name of output file for the target distribution averages. By default it is targetdist-averages.data.
TARGETDIST_AVERAGES_OUTPUT	how often the target distribution averages should be written out to file. Note that the value is given in terms of coefficient iterations. If no value is given are the averages only written at the beginning of the optimization
BIAS_OUTPUT	how often the bias(es) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_PROJ_OUTPUT	how often the projections of the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time
MONITOR_AVERAGES_GRADIENT_EXP_DECAY	use an exponentially decaying averaging with a given time constant when monitoring the averaged gradient

8.8.4.4 OPT_ROBBINS_MONRO_SGD

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Robbins-Monro stochastic gradient decent.

Attention

This optimizer is only included for reference. We recommend to use the averaged stochastic gradient decent optimizer (OPT_AVERAGED_SGD).

Examples

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
stepsize	the current value of step size used to update the coefficients. For multiple biases this component is labeled using the number of the bias as stepsize-#.

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
gradrms	**MONITOR_INSTANTANEOUS_GRADIENT**	the root mean square value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradrms-#.
gradmax	**MONITOR_INSTANTANEOUS_GRADIENT**	the largest absolute value of the coefficient gradient. For multiple biases this component is labeled using the number of the bias as gradmax-#.

Compulsory keywords

BIAS	the label of the VES bias to be optimized
STRIDE	the frequency of updating the coefficients given in the number of MD steps.
COEFFS_FILE	(default=coeffs.data) the name of output file for the coefficients
COEFFS_OUTPUT	(default=100) how often the coefficients should be written to file. This parameter is given as the number of iterations.
INITIAL_STEPSIZE	the initial step size used for the optimization

Options

MONITOR_INSTANTANEOUS_GRADIENT	(default=off) if quantities related to the instantaneous gradient should be outputted.

MULTIPLE_WALKERS	(default=off) if optimization is to be performed using multiple walkers connected via MPI
START_OPTIMIZATION_AFRESH	(default=off) if the iterations should be started afresh when a restart has been triggered by the RESTART keyword or the MD code.
COEFFS_FMT	specify format for coefficient file(s) (useful for decrease the number of digits in regtests)
COEFFS_SET_ID_PREFIX	suffix to add to the filename given in FILE to identify the bias, should only be given if a single filename is given in FILE when optimizing multiple biases.
INITIAL_COEFFS	the name(s) of file(s) with the initial coefficients
TARGETDIST_AVERAGES_FILE	the name of output file for the target distribution averages. By default it is targetdist-averages.data.
TARGETDIST_AVERAGES_OUTPUT	how often the target distribution averages should be written out to file. Note that the value is given in terms of coefficient iterations. If no value is given are the averages only written at the beginning of the optimization
BIAS_OUTPUT	how often the bias(es) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_PROJ_OUTPUT	how often the projections of the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time
MASK_FILE	read in a mask file which allows one to employ different step sizes for different coefficients and/or deactivate the optimization of certain coefficients (by putting values of 0.0). One can write out the resulting mask by using the OUTPUT_MASK_FILE keyword.
OUTPUT_MASK_FILE	Name of the file to write out the mask resulting from using the MASK_FILE keyword. Can also be used to generate a template mask file.
TARGETDIST_STRIDE	stride for updating a target distribution that is iteratively updated during the optimization. Note that the value is given in terms of coefficient iterations.
TARGETDIST_OUTPUT	how often the dynamic target distribution(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
TARGETDIST_PROJ_OUTPUT	how often the projections of the dynamic target distribution(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
DECAY_CONSTANT	the decay constant used for the step size.

8.8.5 Utilities

The following list contains various utilities available in the VES code.

VES_OUTPUT_BASISFUNCTIONS	Output basis functions to file.
VES_OUTPUT_FES	Tool to output biases and free energy surfaces for VES biases from previously obtained coefficients.
VES_OUTPUT_TARGET_DISTRIBUTION	Output target distribution to file.

8.8.5.1 VES_OUTPUT_BASISFUNCTIONS

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Output basis functions to file.
This action can be used to write out to a grid file the values and derivatives of given basis functions. This is normally used for debugging when programming new types of basis functions. For example, it is possible to calculate the derivatives numerically and compare to the analytically calculated derivatives.
This action is normally used through the driver.

Examples

In the following input we define a Legendre polynomials basis functions of order 14 over the interval -4.0 to 4.0 and output their values and derivatives to files called bfL.values.data and bfL.derivs.data.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/VES_OUTPUT_BASISFUNCTIONS.tmp
BF_LEGENDRE ...
 ORDER=14
 MINIMUM=-4.0
 MAXIMUM=4.0
 LABEL=bfL
... BF_LEGENDRE

VES_OUTPUT_BASISFUNCTIONS ...
 BASIS_FUNCTIONS=bfL
 GRID_BINS=200
 FORMAT_VALUES_DERIVS=%13.6f
... VES_OUTPUT_BASISFUNCTIONS
```

This input should be run through the driver by using a command similar to the following one where the trajectory/configuration file configuration.gro is needed to trick the code to exit correctly.

```
plumed driver --plumed plumed.dat --igro configuration.gro
```

Glossary of keywords and components

Compulsory keywords

BASIS_FUNCTIONS	the label of the basis functions that you want to use

Options

IGNORE_PERIODICITY	(default=off) if the periodicity of the basis functions should be ignored.
NUMERICAL_DERIVATIVES	(default=off) if the derivatives of the basis functions should be calculated numerically.
GRID_BINS	the number of bins used for the grid for writing the basis function values and derivatives. The default value is 1000.
GRID_MIN	the minimum of the grid for writing the basis function values and derivatives. By default it is the minimum of the interval on which the basis functions are defined.

GRID_MAX	the maximum of the grid for writing the basis function values and derivatives. By default it is the maximum of the interval on which the basis functions are defined.
FILE_VALUES	filename of the file on which the basis function values are written. By default it is BF_LABEL.values.data.
FILE_DERIVS	filename of the file on which the basis function derivatives are written. By default it is BF_LABEL.derivs.data.
FORMAT_VALUES_DERIVS	the numerical format of the basis function values and derivatives written to file. By default it is %15.8f. . You can also use FORMAT_V↩ ALUES and FORMAT_DERIVS to give the numerical formats separately.
FORMAT_VALUES	the numerical format of the basis function derivatives written to file. By default it is %15.8f.
FORMAT_DERIVS	the numerical format of the basis function values written to file. By default it is %15.8f.
FILE_TARGETDIST_AVERAGES	filename of the file on which the averages over the target distributions are written. By default it is BF_LABEL.targetdist-averages.data.
FORMAT_TARGETDIST_AVERAGES	the numerical format of the target distribution averages written to file. By default it is %15.8f.
FILE_TARGETDIST	filename of the files on which the target distributions are written. By default it is BF_LABEL.targetdist-#.data.
TARGET_DISTRIBUTION	the target distribution to be used.. You can use multiple instances of this keyword i.e. TARGET_DISTRIBUTION1, TARGET_DISTRIBU↩ TION2, TARGET_DISTRIBUTION3...

8.8.5.2 VES_OUTPUT_FES

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Tool to output biases and free energy surfaces for VES biases from previously obtained coefficients.
This action can be used to output to file biases and free energy surfaces for VES biases from previously obtained coefficients. It should be used through the driver and can only be used in post processing. The VES bias needs to be defined in the exact same way as during the simulation. At the current moment this action does not support dynamic target distributions (e.g. well-tempered).

Examples

In the following input we define a VES bias and then read in the coefficient file coeffs.input.data and output the FES and bias every 500 iterations.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/VES_OUTPUT_FES.tmp
phi:   TORSION ATOMS=5,7,9,15
psi:   TORSION ATOMS=7,9,15,17

bf1: BF_FOURIER ORDER=5 MINIMUM=-pi MAXIMUM=pi
bf2: BF_FOURIER ORDER=5 MINIMUM=-pi MAXIMUM=pi

VES_LINEAR_EXPANSION ...
 ARG=phi,psi
 BASIS_FUNCTIONS=bf1,bf2
 LABEL=ves1
 GRID_BINS=100,100
 PROJ_ARG1=phi
 PROJ_ARG2=psi
... VES_LINEAR_EXPANSION
```

```
VES_OUTPUT_FES ...
  BIAS=ves1
  FES_OUTPUT=500
  FES_PROJ_OUTPUT=500
  BIAS_OUTPUT=500
  COEFFS_INPUT=coeffs.input.data
... VES_OUTPUT_FES
```

The header of coeffs.input.data should look like the following:

```
#! FIELDS idx_phi idx_psi ves1.coeffs ves1.aux_coeffs index
#! SET time 100.000000
#! SET iteration  10
#! SET type LinearBasisSet
#! SET ndimensions  2
#! SET ncoeffs_total  121
#! SET shape_phi  11
#! SET shape_psi  11
```

This input should be run through the driver by using a command similar to the following one where the trajectory/configuration file configuration.gro is needed to correctly define the CVs

```
plumed driver --plumed plumed.dat --igro configuration.gro
```

Glossary of keywords and components

Compulsory keywords

BIAS	the label of the VES bias for to output the free energy surfaces and the bias files
COEFFS_INPUT	the name of input coefficient file

Options

BIAS_OUTPUT	how often the bias(es) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.
FES_PROJ_OUTPUT	how often the projections of the FES(s) should be written out to file. Note that the value is given in terms of coefficient iterations.

8.8.5.3 VES_OUTPUT_TARGET_DISTRIBUTION

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Output target distribution to file.
This action can be used to output target distributions to a grid file, for example to see how they look like before using them in a VES bias. This action only support static target distributions.
This action is normally used through the driver.

Examples

In the following input we define a target distribution that is uniform for argument 1 and a Gaussian for argument 2 and then output it to a file called targetdist-1.data.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/VES_OUTPUT_TARGET_DISTRIBUTION.tmp
t1_1: TD_UNIFORM  MINIMA=-4.0  MAXIMA=+4.0
t1_2: TD_GAUSSIAN  CENTER1=-2.0  SIGMA1=0.5
t1: TD_PRODUCT_DISTRIBUTION  DISTRIBUTIONS=t1_1,t1_2

VES_OUTPUT_TARGET_DISTRIBUTION ...
 GRID_MIN=-4.0,-4.0
 GRID_MAX=+4.0,+4.0
 GRID_BINS=100,100
 TARGET_DISTRIBUTION=t1
 TARGETDIST_FILE=targetdist-1.data
 LOG_TARGETDIST_FILE=targetdist-1.log.data
 FMT_GRIDS=%11.6f
... VES_OUTPUT_TARGET_DISTRIBUTION
```

This input should be run through the driver by using a command similar to the following one where the trajectory/configuration file configuration.gro is needed to trick the code to exit correctly.

```
plumed driver --plumed plumed.dat --igro configuration.gro
```

Glossary of keywords and components

Compulsory keywords

GRID_MIN	the lower bounds for the grid
GRID_MAX	the upper bounds for the grid
GRID_BINS	the number of bins used for the grid.
TARGETDIST_FILE	filename of the file for writing the target distribution
TARGET_DISTRIBUTION	the target distribution to be used.

Options

DO_1D_PROJECTIONS	(default=off) Also output the one-dimensional marginal distributions for multi-dimensional target distribution.
GRID_PERIODICITY	specify if the individual arguments should be made periodic (YES) or not (NO). By default all arguments are taken as not periodic.
LOG_TARGETDIST_FILE	filename of the file for writing the log of the target distribution
FMT_GRIDS	the numerical format of the target distribution grids written to file. By default it is %14.9f

8.8.6 Command Line Tools

The following list contains the command line tools available in the VES code.

ves_md_linearexpansion	Simple MD code for dynamics on a potential energy surface given by a linear basis set expansion.

8.8.6.1 ves_md_linearexpansion

	This is part of the ves module
	It is only available if you configure PLUMED with ./configure –enable-modules=ves . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Simple MD code for dynamics on a potential energy surface given by a linear basis set expansion.
This is simple MD code that allows running dynamics of a single particle on a potential energy surface given by some linear basis set expansion in one to three dimensions.
It is possible to run more than one replica of the system in parallel.

Examples

In the following example we perform dynamics on the Wolfe-Quapp potential that is defined as

$$U(x,y) = x^4 + y^4 - 2x^2 - 4y^2 + xy + 0.3x + 0.1y$$

To define the potential we employ polynomial power basis functions (BF_POWERS). The input file is given as

```
nstep             10000
tstep             0.005
temperature       1.0
friction          10.0
random_seed       4525
plumed_input      plumed.dat
dimension         2
replicas          1
basis_functions_1 BF_POWERS ORDER=4 MINIMUM=-3.0 MAXIMUM=+3.0
basis_functions_2 BF_POWERS ORDER=4 MINIMUM=-3.0 MAXIMUM=+3.0
input_coeffs       pot_coeffs_input.data
initial_position   -1.174,+1.477
output_potential        potential.data
output_potential_grid   150
output_histogram        histogram.data

# Wolfe-Quapp potential given by the equation
# U(x,y) = x**4 + y**4 - 2.0*x**2 - 4.0*y**2 + x*y + 0.3*x + 0.1*y
# Minima around (-1.174,1.477); (-0.831,-1.366); (1.124,-1.486)
# Maxima around (0.100,0.050)
# Saddle points around (-1.013,-0.036); (0.093,0.174); (-0.208,-1.407)
```

This input is then run by using the following command.

```
plumed ves_md_linearexpansion input
```

The corresponding pot_coeffs_input.data file is

```
#! FIELDS idx_dim1 idx_dim2 pot.coeffs index description
#! SET type LinearBasisSet
#! SET ndimensions  2
#! SET ncoeffs_total  25
#! SET shape_dim1  5
#! SET shape_dim2  5
       0       0         0.0000000000000000e+00       0  1*1
       1       0         0.3000000000000000e+00       1  s^1*1
       2       0        -2.0000000000000000e+00       2  s^2*1
       4       0         1.0000000000000000e+00       4  s^4*1
       0       1         0.1000000000000000e+00       5  1*s^1
       1       1        +1.0000000000000000e+00       6  s^1*s^1
       0       2        -4.0000000000000000e+00      10  1*s^2
       0       4         1.0000000000000000e+00      20  1*s^4
#!--------------------
```

One then uses the (x,y) position of the particle as CVs by using the POSITION action as shown in the following PLUMED input

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ves_md_linearexpansion.tmp
p: POSITION ATOM=1
ene: ENERGY
PRINT ARG=p.x,p.y,ene FILE=colvar.data FMT=%8.4f
```

Glossary of keywords and components

Compulsory keywords

nstep	(default=10) The number of steps of dynamics you want to run.
tstep	(default=0.005) The integration timestep.
temperature	(default=1.0) The temperature to perform the simulation at. For multiple replica you can give a separate value for each replica.
friction	(default=10.) The friction of the Langevin thermostat. For multiple replica you can give a separate value for each replica.
random_seed	(default=5293818) Value of random number seed.
plumed_input	(default=plumed.dat) The name of the plumed input file(s). For multiple replica you can give a separate value for each replica.
dimension	(default=1) Number of dimensions, supports 1 to 3.
initial_position	Initial position of the particle. For multiple replica you can give a separate value for each replica.
replicas	(default=1) Number of replicas.
basis_functions_1	Basis functions for dimension 1.
input_coeffs	(default=potential-coeffs.in.data) Filename of the input coefficient file for the potential. For multiple replica you can give a separate value for each replica.
output_coeffs	(default=potential-coeffs.out.data) Filename of the output coefficient file for the potential.
output_coeffs_fmt	(default=%30.16e) Format of the output coefficient file for the potential. Useful for regtests.
output_potential_grid	(default=100) The number of grid points used for the potential and histogram output files.
output_potential	(default=potential.data) Filename of the potential output file.
output_histogram	(default=histogram.data) Filename of the histogram output file.

Options

--help/-h	(default=off) print this help
basis_functions_2	Basis functions for dimension 2 if needed.
basis_functions_3	Basis functions for dimension 3 if needed.
coeffs_prefactor	prefactor for multiplying the coefficients with. For multiple replica you can give a separate value for each replica.
template_coeffs_file	only generate a template coefficient file with the filename given and exit.

8.8.7 Tutorials

The following tutorials are available for the VES code.
MARVEL-VES School February 2017

MARVEL-VES tutorial (Lugano Feb 2017): Metadynamics	Brief introduction to metadynamics.
MARVEL-VES tutorial (Lugano Feb 2017): VES 1	Introduction to VES, using different target distributions and basis sets.

MARVEL-VES tutorial (Lugano Feb 2017): VES 2	VES, well-tempered target distribution and 2 dimensional biases.
MARVEL-VES tutorial (Lugano Feb 2017): Kinetics	How to obtain kinetics from biased molecular simulations using VES.

8.8.7.1 MARVEL-VES School February 2017

Tutorials from the MARVEL School on Variationally Enhanced Sampling that was held in Lugano, February 14-17, 2017.

Suggested readings

Metadynamics:
Enhancing Important Fluctuations: Rare Events and Metadynamics from a Conceptual Viewpoint, Annual Reviews in Physical Chemistry 2016
Variationally Enhanced Sampling:
Variational Approach to Enhanced Sampling and Free Energy Calculations, Physical Review Letters 2014
Variationally Optimized Free-Energy Flooding for Rate Calculation, Physical Review Letters 2015

Tuesday February 14

Tutorial 1: Introduction to PLUMED and analyzing molecular simulations

Wednesday February 15

Tutorial 2: Biasing with metadynamics
Tutorial 3: Biasing with variationally enhanced sampling

Thursday February 16

Tutorial 4: Further on variationally enhanced sampling
Tutorial 5: Advanced collective variables

- Path CVs
- Multicolvar
- Dimensionality reduction

Friday February 17

Tutorial 6: Obtaining kinetics from molecular simulations

8.8.7.2 MARVEL-VES tutorial (Lugano Feb 2017): Metadynamics

8.8.7.2.1 Learning Outcomes Once this tutorial is completed students will learn to:

- Perform metadynamics simulations using PLUMED 2 and LAMMPS
- Construct a bias potential on 1 and 2 collective variables (CVs)
- Assess the convergence of the free energy surface
- Distinguish between good and bad CVs
- Reweight with more than one bias potential

8.8.7.2.2 Resources The `tarball` for this project contains the following folders:

- Example1 : Contains the input file for the unbiased simulation.
- Example2 : Contains the input files for one of the biased simulations. The rest of the biased simulations inputs should be created by modifying this one.

8.8.7.2.3 Instructions

8.8.7.2.3.1 The system We consider the association/dissociation of NaCl in aqueous solution. The dissociation barrier is expected to be around 2.5 $k_{\mathrm{B}}T$. One interesting aspect of the ion dissociation problem is that collective solvent motions play an important role in the transition. This problem has been considered in the original metadynamics paper [47] and also in reference [123] . We will use the potential developed in ref. [124] for NaCl and TIP3P water with parameters corrected to be used with long-range Coulomb solvers [125]. The system contains 1 Na, 1 Cl, and 106 water molecules (total 320 atoms).

8.8.7.2.3.2 Perform an unbiased simulation and control the distance Na-Cl We first perform a standard MD simulation and control the distance Na-Cl. All the files needed for this example are contained in the folder Example1 . The distance Na-Cl can be calculated in Plumed 2 using:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ves-lugano2017-01-metad.txt
d1:  DISTANCE ATOMS=319,320
```

The coordination number of Na with respect to O in water will also be calculated for later use. This variable will represent the collective motion of the solvent.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ves-lugano2017-01-metad.txt
COORDINATION ...
 GROUPA=319
 GROUPB=1-318:3
 SWITCH={RATIONAL R_0=0.315 D_MAX=0.5 NN=12 MM=24}
 NLIST
 NL_CUTOFF=0.55
 NL_STRIDE=10
 LABEL=coord
... COORDINATION
```

To run LAMMPS you can use the run.sh script:

```
#!/bin/bash

#############################################################################
# Definition of variables
#############################################################################
EXE=lmp_mpi
totalCores=2
#############################################################################

mpirun -np ${totalCores} ${EXE} < start.lmp > out.lmp
```

This command runs LAMMPS using 2 MPI threads. The use of partitions will be discussed when using multiple walkers
Once the simulation is launched, the so called COLVAR file is written. In this case it contains the following:

```
#! FIELDS time d1 coord
 0.200000 0.568067 5.506808
 0.400000 0.500148 4.994588
 0.600000 0.449778 4.931140
 0.800000 0.528272 5.105816
 1.000000 0.474371 5.089863
 1.200000 0.430620 5.091551
 1.400000 0.470374 4.993886
 1.600000 0.458768 4.940097
 1.800000 0.471886 4.952868
 2.000000 0.489058 4.897593
.
.
.
```

If you plot the time (column 1) vs the distance (column 2), for instance in gnuplot:

```
pl  "./COLVAR" u 1:2 w lp,
```

you will see that the ion pair is stuck in the dissociated state during the 1 ns simulation. It is unable to cross the $\sim 5k_{\rm B}T$ barrier located at a distance of approximately 0.4 nm. You can also observe this behavior in the trajectory using VMD:

```
vmd out.dcd -psf nacl.psf
```

The trajectory has been saved in unwrapped format in order to avoid bonds stretching from one side to the box to the other due to periodic boundary conditions. In VMD we can wrap the atoms without breaking the bonds and show the box using the commands:

```
pbc wrap -compound res -all
pbc box
```

You can play with different visualization styles and options that VMD has. Therefore, if we want the system to go back and forth between the associated and dissociated state, we will need enhanced sampling.

8.8.7.2.3.3 Construct a bias potential on the distance Na-Cl We now construct a bias potential $V(\mathbf{s})$ on the distance Na-Cl using well-tempered metadynamics. The files for this example are contained in the directory Example2. As argument for the construction of the potential we will use the distance Na-Cl (label d1). We choose a gaussian height of 1 kJ/mol which is slightly less than 0.5 $k_{\rm B}T$. The gaussian width is 0.02 nm, in the same order of the features in the FES. A rule of thumb for choosing the gaussian width is to use the standard deviation of the unbiased fluctuations of the CV. The bias factor is set to 5 since the largest barrier in the FES is expected to be roughly 5 $k_{\rm B}T$. Once the metadynamics simulation is converged, the bias will be (up to an arbitrary constant):

$$V(\mathbf{s}) = -\left(1 - \frac{1}{\gamma}\right) F(\mathbf{s})$$

and therefore the system will evolve under an effective free energy:

$$\tilde{F}(\mathbf{s}) = F(\mathbf{s}) + V(\mathbf{s}) = \frac{F(\mathbf{s})}{\gamma},$$

that is to say, the largest barrier will be of around 1 $k_{\rm B}T$. The input is:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ves-lugano2017-01-metad.txt
d1:  DISTANCE ATOMS=319,320
METAD ...
 LABEL=metad
 ARG=d1
 SIGMA=0.02
 HEIGHT=1.
 BIASFACTOR=5
 TEMP=300.0
 PACE=500
 GRID_MIN=0.2
 GRID_MAX=1.0
 GRID_BIN=300
 CALC_RCT
... METAD
```

Here the CALC_RCT keyword turns on the calculation of the time dependent constant $c(t)$ that we will use below when reweighting the simulations.
We will also limit the exploration of the CV space by introducing an upper wall bias $V_{wall}(\mathbf{s})$:

$$V_{wall}(s) = \kappa(s - s_0)^2 \text{ if } s > s_0 \text{ and } 0 \text{ otherwise.}$$

The wall will focus the sampling in the most interesting region of the free energy surface. The effect of this bias potential will have to be corrected later in order to calculate ensemble averages. The syntax in Plumed 2 is:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ves-lugano2017-01-metad.txt
d1:  DISTANCE ATOMS=319,320
UPPER_WALLS ...
 ARG=d1
 AT=0.6
 KAPPA=2000.0
 EXP=2
 EPS=1
 OFFSET=0.
 LABEL=uwall
... UPPER_WALLS
```

You can run the simulation with the run.sh script as done in the previous example.
It is possible to try different bias factors to check the effect that it has on the trajectory and the effective FES.
In principle the cost of a metadynamics simulation should increase as the simulation progresses due to the need of adding an increasing number of Gaussian kernels to calculate the bias. However, since a grid is used to build up the bias this effect is not observed. You can check what happens if you do not use the GRID_$*$ keywords. Remember that the bins in the grid should be small enough to describe the changes in the bias created by the Gaussian kernels. Normally a bin of size $\sigma/5$ (with σ the gaussian width) is small enough.

8.8.7.2.3.4 Assess convergence One way to identify if a well tempered metadynamics simulation has converged is observing the estimated free energy surface at different times. The FES is estimated by using the relation (again up to an arbitrary constant):

$$F(\mathbf{s}) = -\left(\frac{\gamma}{\gamma-1}\right) V(\mathbf{s},t)$$

and the bias potential is calculated as a sum of Gaussian kernels. This can be done with Plumed 2 using for instance:

```
plumed sum_hills --hills ../HILLS --min 0.1 --max 0.8 --bin 300 --stride 100
```

Most of the flags are self explanatory. The flag –stride 100 will result in the FES been written every 100 Gaussian kernels, i.e. 100 ps. Inside the folder FES_calculation you will find a script run.sh that executes the sum_hills command and a gnuplot script plot.gpi that can be used typing:.

```
gnuplot plot.gpi
```

After roughly 3 ns the free energy surface does not change significantly with time except for an immaterial constant $c(t)$ that grows in time. This is in line with well-tempered metadynamics asymptotic behavior:

$$V(\mathbf{s},t) = -\left(1-\frac{1}{\gamma}\right) F(\mathbf{s}) + c(t).$$

The behavior of $c(t)$ will be studied with greater detail later.
It should be stressed that we are actually not calculating the free energy $F(\mathbf{s})$ but rather $F(\mathbf{s}) + V_{wall}(\mathbf{s})$. For this reason for distances higher than 0.6 nm the free energy increases sharply.
An alternative way to observe the evolution of the bias is plotting the final free energy plus the instantaneous bias. The scripts for this example can be found in the folder Bias_calculation. In this case we observe only the first 200 ps of the simulation. The (negative) bias can be calculated using:

```
plumed sum_hills --hills ../HILLS --min 0.1 --max 0.8 --bin 300 --stride 10 --negbias
```

This plot illustrates clearly how the bias is constructed to progressively "fill" the FES.
It is also possible to track convergence by controlling the evolution of some quantity connected to the free energy surface. In this case we will calculate the dissociation barrier, e.g. the height of the barrier that separates the associated state from the dissociated one. The scripts for this example are found in the folder Barrier_calculation. Using the python script calculate_barrier.py we can compute the barrier, for instance:

```
import numpy as np
# Total number of fes files in folder
total_files=101
# Min and max initial guesses
min_min=50
min_max=90
max_min=90
max_max=130
for i in range(total_files):
        file_name="fes_" + str(i) + ".dat"
        matrix=np.genfromtxt(file_name)
        minimum=np.amin(matrix[min_min:min_max,1])
        maximum=np.amax(matrix[max_min:max_max,1])
        print(str(i) + " " + str(minimum) + " " + str(maximum) + " " + str(maximum-minimum))
```

The script can be executed using:

```
python barrier_calculation.py > barrier.txt
```

and the results can be plotted using the gnuplot script plot.gpi. After roughly 4 ns the barrier stabilizes around 3 and 3.5 $k_{\mathrm{B}}T$.
It is important to stress that it is only possible to calculate the free energy difference between two points if the system has gone back and forth between these points several times. This applies both for the calculation of a barrier and the difference in free energy between two basins. It is also important to understand that none of the free energy methods described in this series of tutorials will be able to calculate free energies of regions that have not been sampled, i.e. visited.
Reweighting the simulation on the same CV that was used for biasing can also be used as a test of convergence. We will show that in the next section.

8.8.7.2.3.5 Reweight the simulation We first reweight the simulation on the distance Na-Cl, the same CV used for biasing. This reweighting is useful to check convergence and to have an estimate of the free energy that does not rely on using kernels. For instance if some features of the FES could not be captured by the kernels, the reweighting procedure will show them. This will become clearer in the VES tutorial. The scripts to perform this calculation are found in the folder ReweightDistance. In metadynamics quasi-stationary limit the weight assigned to a given configuration is [3] :

$$w(\mathbf{R}) \propto e^{\beta(V(\mathbf{s},t)-c(t))}.$$

By plotting time (column 1) versus $c(t)$ (column 6) using the COLVAR file, the importance of taking $c(t)$ into account becomes clear. $c(t)$ keeps growing even after long times, reflecting the approximately rigid shift of the bias with time. Normally the first part of the trajectory is not used for reweighting since during this period the simulation has not reached the quasi-stationary limit. In this case we can discard the first 2 or 3 ns of simulation. To disregard the first 3 ns of simulation we can use sed to delete the first 15000 lines from the COLVAR file:

```
sed '2,15000d' ../COLVAR > COLVAR
```

We then use Plumed to calculate two histograms, one taking into account the wall bias and the other one neglecting it. The weights for the reweighting involving only the metadynamics bias have already been discussed while the weights considering both biases are:

$$w(\mathbf{R}) \propto e^{\beta(V(\mathbf{s},t)-c(t)+V_{wall}(\mathbf{s}))}.$$

The header of the COLVAR file should be:

```
#! FIELDS time d1 coord metad.bias metad.rbias metad.rct metad.work uwall.bias
```

The input script for Plumed is:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ves-lugano2017-01-metad.txt
# Read COLVAR file
distance:       READ FILE=COLVAR  IGNORE_TIME VALUES=d1
metad:       READ FILE=COLVAR IGNORE_TIME VALUES=metad.rbias
uwall:       READ FILE=COLVAR IGNORE_TIME VALUES=uwall.bias

# Define weights
weights1: REWEIGHT_METAD TEMP=300
weights2: REWEIGHT_BIAS TEMP=300 ARG=metad.rbias,uwall.bias

# Calculate histograms
HISTOGRAM ...
  ARG=distance
  GRID_MIN=0.2
  GRID_MAX=0.8
  GRID_BIN=100
  BANDWIDTH=0.002
  LOGWEIGHTS=weights1
  LABEL=hh1
... HISTOGRAM

HISTOGRAM ...
  ARG=distance
  GRID_MIN=0.2
  GRID_MAX=0.8
  GRID_BIN=100
  BANDWIDTH=0.002
  LOGWEIGHTS=weights2
  LABEL=hh2
... HISTOGRAM

# Print histograms to file
DUMPGRID GRID=hh1 FILE=histo FMT=%24.16e
DUMPGRID GRID=hh2 FILE=histo_wall FMT=%24.16e
```

This example can be run with:

```
plumed --no-mpi driver --plumed plumed.dat --noatoms > plumed.out
```

and will generate the files histo and histo_wall. The histograms represent the probability $p(\mathbf{s})$ of observing a given value of the CV $\mathbf{s}$. From the histograms the FES can be calculated using:

$$\beta F(\mathbf{s}) = -\log p(\mathbf{s})$$

and therefore we plot the FES in gnuplot using for instance:

```
pl  "./histo" u 1:(-log($2)) w lp
```

The next plot compares the estimations of the FES from sum_hills, reweighting with metadynamics bias, and reweighting using both the metadynamics bias and the upper wall bias You will find a gnuplot script plot.gpi to make this plot inside the ReweightDistance folder.
We can obtain important information of the system by reweighting on 2 CVs: The distance Na-Cl and the coordination of Na with O. This reweighting is similar to the one already done and the files that you will need are located in the ReweightBoth folder. Additionally the COLVAR file with the omitted first steps is required. The plot of the FES as a function of these 2 CVs provides important information of the association/dissociation mechanism. In the dissociated state, Na can have a coordination of 5 or 6, though it is more likely to find a coordination number of 6. However, in order to associate Na must have a coordination with O of 5. In the associated state Na can have a coordination of 3, 4 or 5. The transition state is characterized by a coordination number of $\sim$5.

8.8.7.2.3.6 Construct a bias potential on the coordination Na-O As an exercise, you can write the input files for a simulation in which a bias potential is constructed on the coordination Na-0, i.e. the solvent degree of freedom. You can use the same gaussian height as before and $\sigma = 0.1$.
You will find that the exploration of the CV space is not efficient. The reason is that there is a slow degree of freedom that it is not being biased: the distance Na-Cl. Furthermore you can see in the 2 CV reweighting that the coordination Na-O shows significant overlap between the associated and dissociated states.
Bear in mind that this is a rather trivial example since the existing barriers are relatively low. Real problems in materials science usually involve large barriers and are not as forgiving as this example; a bad CV may lead to huge hysteresis and problems in convergence.

8.8.7.2.3.7 Construct a bias potential on both CVs We will now construct a bias potential on both CVs. We have already calculated the FES as a function of both CVs through reweighting. In this example the FES will be calculated using the metadynamics bias potential. You can use the input files from Example2.tar and changed the plumed.dat file. To construct a 2 dimensional bias with metadynamics use the following input:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-01-metad.txt
__FILL__
METAD ...
 LABEL=metad
 ARG=d1,coord
 SIGMA=0.02,0.1
 HEIGHT=1.
 BIASFACTOR=5
 TEMP=300.0
 PACE=500
 GRID_MIN=0.15,2.
 GRID_MAX=0.9,9.
 GRID_BIN=400,400
 CALC_RCT
... METAD
```

Once that the simulation is completed you can run plumed sum_hills to calculate the FES:

```
plumed sum_hills --hills HILLS --mintozero
```

and plot the results using the following lines in gnuplot:

```
set pm3d map
set zr [0:15]
spl "fes.dat" u 1:2:3
```

8.8.7.2.4 Final remarks Some valuable tools for metadynamics simulations will be discussed in the VES tutorial. These include:

- Restarting a simulation.
- Using Plumed driver to calculate a CV that was not calculated during the simulation. A reweighting can then be performed on this CV.
- Constructing biased histograms, i.e. histograms without weights to calculate the effective FES $\tilde{F}(\mathbf{s}) = F(\mathbf{s}) + V(\mathbf{s})$.
- Use multiple walkers to improve the exploration of CV space.

8.8.7.3 MARVEL-VES tutorial (Lugano Feb 2017): VES 1

8.8.7.3.1 Learning Outcomes Once this tutorial is completed students will learn to:

- Use different target distributions and choose the most appropriate for their problem.
- Use different basis sets and order of the expansions. Select the appropriate order for their problem.
- Use the optimization algorithm and choose the parameters.
- Construct biases in 1 dimension.
- Assess the convergence of the simulation.
- Obtain biased and unbiased histograms.

8.8.7.3.2 Resources The `tarball` for this project contains the following folders:

- Example1 : Contains the input file for the first example.
- Example2 : Contains the input file for the second example.

8.8.7.3.3 Summary of theory Variationally enhanced sampling [76] is based on the the following functional of the bias potential:

$$\Omega[V] = \frac{1}{\beta}\log\frac{\int d\mathbf{s}\, e^{-\beta[F(\mathbf{s})+V(\mathbf{s})]}}{\int d\mathbf{s}\, e^{-\beta F(\mathbf{s})}} + \int d\mathbf{s}\, p(\mathbf{s})V(\mathbf{s}),$$

where $\mathbf{s}$ are the CVs to be biased, $p(\mathbf{s})$ is a predefined probability distribution that we will refer to as the target distribution, and $F(\mathbf{s})$ is the free energy surface. This functional can be shown to be convex and to have a minimum at:

$$V(\mathbf{s}) = -F(\mathbf{s}) - \frac{1}{\beta}\log p(\mathbf{s}).$$

The last equation states that once that the functional $\Omega[V]$ is minimized, the bias and the target distribution allow calculating the free energy. The target distribution $p(\mathbf{s})$ can be chosen at will and it is the distribution of the CVs once that $\Omega[V]$ has been minimized.
The variational principle is put to practice by expanding $V(\mathbf{s})$ in some basis set:

$$V(\mathbf{s}) = \sum_i \alpha_i\, f_i(\mathbf{s}),$$

where $f_i(\mathbf{s})$ are the basis functions and the α are the coefficients in the expansion. We then need to find the α that minimize $\Omega[V]$. In principle one could use any optimization algorithm. In practice the algorithm that has become the default choice for VES is the so-called averaged stochastic gradient descent algorithm [122]. In this algorithm the α are evolved iteratively according to:

$$\alpha^{(n+1)} = \alpha^{(n)} - \mu\left[\nabla\Omega(\bar{\alpha}^{(n)}) + H(\bar{\alpha}^{(n)})[\alpha^{(n)} - \bar{\alpha}^{(n)}]\right]$$

where μ is the step size, $\bar{\alpha}^{(n)}$ is the running average of $\alpha^{(n)}$ at iteration n, and $\nabla\Omega(\bar{\alpha}^{(n)})$ and $H(\bar{\alpha}^{(n)})$ are the gradient and Hessian of $\Omega[V]$ evaluated at the running average at iteration n, respectively. The behavior of the coefficients will become clear in the examples below.

8.8.7.3.4 Instructions

8.8.7.3.4.1 The system We will consider the same system employed for the metadynamics tutorial.

8.8.7.3.4.2 Example 1: First VES simulation For the first VES simulation we will revisit the problem of the ion pair dissociation but replacing the metadynamics bias with a VES bias. The bias potential will be constructed on the distance Na-Cl as done before. We will still use the upper wall used in the metadynamics tutorial to make the actual example as similar as possible to the previous one. We will then see that VES has a more natural way to deal with barriers. All files needed for this example can be found in the Example1 folder.
Every VES simulation has three key ingredients:

- Basis set
- Target distribution
- Optimization algorithm

For the basis set we will choose Legendre polynomials defined in the interval [0.23,0.7] nm. Legendre polynomials are a good choice for non-periodic CVs. A rule of thumb for choosing the order of the expansion is that an expansion of order N can capture features of the FES of approximately L/N where L is the length of the interval. In this case, an order of 10 is able to capture features of the order of around 0.05 nm. We will see afterwards that the order of the expansion is not critical as long as we obtain good sampling at convergence. If this is the case, it is possible to obtain finer features of the FES through reweighting. The syntax for this basis set in Plumed is:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-02-ves1.txt
__FILL__
BF_LEGENDRE ...
 ORDER=10
 MINIMUM=0.23
 MAXIMUM=0.8
 LABEL=bf1
... BF_LEGENDRE
```

We will use a uniform target distribution:

$$p(\mathbf{s}) = 1/C$$

with C a normalization constant. Once that $\Omega[V]$ is minimized, the bias potential satisfies (up to an arbitrary constant):

$$V(\mathbf{s}) = -F(\mathbf{s})$$

This is the same relation that holds for non-tempered metadynamics.
The syntax for the bias potential in Plumed is:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-02-ves1.txt
__FILL__
td1: TD_UNIFORM

VES_LINEAR_EXPANSION ...
 ARG=d1
 BASIS_FUNCTIONS=bf1
 GRID_BINS=300
 TARGET_DISTRIBUTION=td1
 LABEL=b1
... VES_LINEAR_EXPANSION
```

Finally we have to choose the optimization algorithm. The standard is the averaged stochastic gradient descent. One has to define two parameters: the stride and the step size. The stride is the number of steps in which samples are collected to calculate the gradient and hessian of $\Omega[V]$ and the step size is the step by which the coefficients are evolved at every optimization steps. Both of this parameters are connected. Increasing the stride will have an effect similar to reducing the step size. It has become traditional to choose a stride of around 500-2000 steps. It must be noted that we are not looking for an accurate estimation of the gradient, since for this we would need to sample all the CV space. The step size in the optimization has a strong connection with the height of typical barriers in the system. The larger the barriers, the larger the step size needed such that the bias can grow fast enough to overcome them. For this example we have chosen a stride of 500 steps and a step size of 0.5 kJ/mol. The syntax in Plumed is:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-02-ves1.txt
__FILL__
OPT_AVERAGED_SGD ...
  BIAS=b1
  STRIDE=500
  LABEL=o1
  STEPSIZE=0.5
  FES_OUTPUT=100
  BIAS_OUTPUT=500
  COEFFS_OUTPUT=10
... OPT_AVERAGED_SGD
```

Now that we have set the scene, we can run our simulation using the run.sh script in the Example1 folder. The simulation will produce several files:

- COLVAR: Just as in the metadynamics example.
- coeffs.data : Values of the coefficients α and $\bar{\alpha}$.
- bias.<bias-name>.iter-<iteration-number> : Bias potential as a function of s at iteration <iteration-number>.
- fes.<bias-name>.iter-<iteration-number> : FES at iteration <iteration-number>.
- targetdistribution.<bias-name>.data : Target distribution.

You can first observe how the system moves in the CV space in a fashion similar to metadynamics. Then we can see the evolution of α and $\bar{\alpha}$. The first lines of the file coeffs.data are:

```
#! FIELDS idx_d1 b1.coeffs b1.aux_coeffs index
#! SET time 0.000000
#! SET iteration  0
#! SET type LinearBasisSet
#! SET ndimensions  1
#! SET ncoeffs_total  11
#! SET shape_d1  11
       0         0.0000000000000000e+00         0.0000000000000000e+00       0
       1         0.0000000000000000e+00         0.0000000000000000e+00       1
       2         0.0000000000000000e+00         0.0000000000000000e+00       2
       3         0.0000000000000000e+00         0.0000000000000000e+00       3
       4         0.0000000000000000e+00         0.0000000000000000e+00       4
       5         0.0000000000000000e+00         0.0000000000000000e+00       5
       6         0.0000000000000000e+00         0.0000000000000000e+00       6
       7         0.0000000000000000e+00         0.0000000000000000e+00       7
       8         0.0000000000000000e+00         0.0000000000000000e+00       8
       9         0.0000000000000000e+00         0.0000000000000000e+00       9
      10         0.0000000000000000e+00         0.0000000000000000e+00      10
#!-------------------



#! FIELDS idx_d1 b1.coeffs b1.aux_coeffs index
#! SET time 10.000000
#! SET iteration  10
#! SET type LinearBasisSet
#! SET ndimensions  1
#! SET ncoeffs_total  11
#! SET shape_d1  11
       0         0.0000000000000000e+00         0.0000000000000000e+00       0
       1         5.1165453234702052e-01         1.1482045941475065e+00       1
       2        -1.0356798763597277e+00        -1.7365051185667855e+00       2
       3        -5.1830527698835660e-01        -1.1651638070736938e+00       3
       4         4.1754103138162207e-01         4.8203393927719917e-01       4
       5         3.2087945211009694e-01         6.6606116920677805e-01       5
       6        -1.5499943980403830e-01        -4.7946750842365812e-03       6
       7        -1.1433825688016251e-01        -1.5099503286093419e-01       7
       8         9.8787914656136719e-02         1.3156529595420300e-02       8
       9         4.4467081175713474e-03        -8.7160339645570323e-02       9
      10        -1.1504176822089783e-01        -1.5789737594248379e-01      10
#!-------------------
```

The first column are the coefficient indices, the second are the $\bar{\alpha}$, and the third are the α. Each block in the file corresponds to a different iteration, in this case iteration 0 and 10. We can plot the evolution of the coefficients using the gnuplot script plotCoeffs.gpi . The output should be similar to the next figure.
The α change fast and oscillate around some mean value. The $\bar{\alpha}$ evolve smoothly until they stabilize around some equilibrium value. It is important to remember that the bias is a function of $\bar{\alpha}$ and since these evolve smoothly, so will the bias. Once that the $\bar{\alpha}$ have stabilized, the simulation can be considered converged.
It is also interesting to observe how the estimation of the FES evolves in time. For this we will plot the FES using the files fes.b1.iter-<iteration-number>. There is a gnuplot script plotFes.gpi that you can use for this purpose. At

variance with metadynamics, in this case there is no growing offset in the bias and therefore we will have to shift the FES ourselves to distinguish several FES at different times in the same plot.
We can also calculate the height of the barrier as we did in the metadynamics tutorial. The files for carrying out this task can be found in the Barrier_calculation folder. Remember that the accuracy of this calculation is limited by the fact that we have chosen a small order in the basis set expansion. We will discuss this aspect in greater detail in the next example.

8.8.7.3.4.3 Example 2: Target distributions and basis sets In this example we will consider other choices of target distributions and we will understand the influence of the order of the basis set expansion. The files needed for this example are contained in the directory Example2. Instead of introducing a barrier as done in the example above, in this case we will use a uniform target distribution in the interval [0.23:0.6] nm and decaying to zero in the interval [0.6:0.8] nm. The expression is:

$$p(s) = \begin{cases} \frac{1}{C} & \text{if } s < s_0 \\ \frac{1}{C} e^{-\frac{(s-s_0)^2}{2\sigma^2}} & \text{if } s > s_0 \end{cases}$$

where $s_0 = 0.6$ nm and $\sigma = 0.05$. To define this $p(s)$ in Plumed the input is:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-02-ves1.txt
__FILL__
td1: TD_UNIFORM MINIMA=0.23 MAXIMA=0.6 SIGMA_MAXIMA=0.05

VES_LINEAR_EXPANSION ...
 ARG=d1
 BASIS_FUNCTIONS=bf1
 LABEL=b1
 TEMP=300.0
 GRID_BINS=300
 TARGET_DISTRIBUTION=td1
... VES_LINEAR_EXPANSION
```

We will choose a basis set of order 20 to be able to capture the features of the FES with detail. If you are doing this example in a group, each member of the group can choose a different order in the expansion, for instance 5, 10, 20, and 40. The syntax in Plumed is:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-02-ves1.txt
__FILL__
BF_LEGENDRE ...
 ORDER=20
 MINIMUM=0.23
 MAXIMUM=0.8
 LABEL=bf1
... BF_LEGENDRE
```

Once that you start running the simulation, a file named targetdist.b1.data will be created. This file contains the chosen target distribution. We can plot it to confirm that it is what we are looking for. There is a gnuplot script plotTargetDistrib.gpi that creates the following plot.
As the simulation runs, it is useful to control the evolution of the coefficients using the gnuplot script plotCoeffs.gpi.
The FES is calculated from the expression:

$$F(\mathbf{s}) = -V(\mathbf{s}) - \frac{1}{\beta} \log p(\mathbf{s}) = \{ -V(s) \text{ if } s < s_0 \; -V(s) + \frac{(s-s_0)^2}{2\beta\sigma^2} \text{ if } s > s_0$$

In other words the bias potential is forced to create the upper barrier that we were explicitly introducing in the first example. When the FES is calculated the effect of the barrier is "subtracted" through $p(s)$ and therefore the FES that we calculate does not include the barrier. This can be seen by plotting the fes.b1.iter-<iteration-number> files with gnuplot, for instance:

```
pl "./fes.b1.iter-10000.data" u 1:2 w l
```

This plot should be similar to the next figure.
Only the interval [0.23:0.7] is plotted since there is little sampling in the region [0.7:0.8] due to the small value of $p(s)$ in this region. As discussed before, if there is no sampling, it is not possible to obtain free energies.
When the simulation ends, it is interesting to check if in fact the sampled biased distribution is equal to the chosen target distribution. The scripts to calculate the sampled biased distribution are located in the directory Biased↩ Distribution. As usual, we will disregard the initial part of the simulation since in this period the bias is changing a lot. As done before, we get rid of the first 2 ns of simulation using sed:

```
sed '2,10000d' ../COLVAR > COLVAR
```

Once that the coefficients in the expansion have stabilized it is possible to calculate the biased distribution of CVs by constructing a histogram with equal weights for all points. This distribution should be equal to the target distribution $p(s)$. The histogram can be calculated in plumed using the following input in the plumed.dat file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-02-ves1.txt
__FILL__
distance: READ FILE=COLVAR  IGNORE_TIME VALUES=d1

HISTOGRAM ...
  ARG=distance
  GRID_MIN=0.2
  GRID_MAX=0.8
  GRID_BIN=100
  BANDWIDTH=0.004
  LABEL=hh1
... HISTOGRAM

DUMPGRID GRID=hh1 FILE=histo FMT=%24.16e
```

and running (or using the run.sh script):

```
plumed --no-mpi driver --plumed plumed.dat --noatoms > plumed.out
```

The next plot shows that in fact the sampled distribution agrees with the target distribution.
Once that the coefficients are stabilized it is possible to reweight using the standard umbrella sampling formula [126] . In this case the weight assigned to each configuration is:

$$w(\mathbf{R}) \propto e^{\beta V(\mathbf{s})}.$$

The files needed for this reweighting are contained in the folder ReweightDistance. The procedure to the the reweighting and plot the results is similar to the ones in the cases above and therefore it is not described in detail. The reweighted FES is plotted in the next figure and compared to the FES calculated from the formula $V(\mathbf{s}) = -F(\mathbf{s}) - \frac{1}{\beta}\log p(\mathbf{s})$.
The two curves do not differ much since the order of the expansion was relatively large. What happens if you chose a lower or higher order in the expansion?

8.8.7.3.4.4 Restarting a simulation In this section we will restart the simulation that we have performed in our second example. In directory Example2, cd to the folder Restart. To restart the simulation we will need:

- LAMMPS restart file, since it stores the last configuration
- COLVAR file, since new lines will be appended
- coeff.data file, containing the iteration number and the values of the coefficients.

Therefore we execute in the command line the following commands:

```
cp ../restart .
cp ../COLVAR .
cp ../coeffs.data .
```

In order to restart, the RESTART keyword must be added at the beginning of the input file for PLUMED named plumed.restart.dat:

```
RESTART

d1:  DISTANCE ATOMS=319,320
.
.
.
```

Then the simulation can be restarted using the script runRestart.sh . Check that the output of the new simulation is appended to the COLVAR file, that the starting time of the new simulation is the ending time of the old simulation, that CV values are coherent, and that coefficients evolve continuously.

8.8.7.3.4.5 Gaussian target distribution As an exercise, you can use a target distribution consisting in a gaussian centered at the dissociation barrier. The syntax in Plumed is:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-02-ves1.txt
__FILL__
td1: TD_GAUSSIAN CENTER1=0.325 SIGMA1=0.03

VES_LINEAR_EXPANSION ...
 ARG=d1
 BASIS_FUNCTIONS=bf1
 LABEL=b1
 TEMP=300.0
 GRID_BINS=300
 TARGET_DISTRIBUTION=td1
... VES_LINEAR_EXPANSION
```

Gaussian target distributions are useful to focus the sampling on a particular region of CV space. This has been used in protein folding problems to focus the sampling on the small but relevant folded state [127].

8.8.7.3.4.6 Optimization algorithm We suggest an exercise to gain experience in choosing the parameters of the optimization algorithm. The averaged stochastic gradient descent algorithm has two parameters: the stride and the step size. Normally a stride of around 500-2000 steps is used. However it is not always easy to choose the step size. Luckily, the algorithm is quite robust and will work for different step sizes.
Run different simulation using step sizes $\mu = 0.001$ and $\mu = 10$ and try to rationalize the behavior. Normally, when the step size is too large, the system gets stuck in CV space and coefficients oscillate wildly. When the step size is too small, the algorithm runs out of "steam" too fast and the simulation converges slowly. These two extreme cases should be avoided.

8.8.7.3.5 Final remarks The purpose of this first tutorial was to introduce the student to VES. At this point one can see that VES is a powerful and versatile enhanced sampling method. We suggest to explore different possibilities of basis sets and target distributions. It is also interesting to experiment with different optimization algorithms and parameters of these.
The next tutorial will deal with the use of the well-tempered target distribution and the construction of biases on 2 CVs.

8.8.7.4 MARVEL-VES tutorial (Lugano Feb 2017): VES 2

8.8.7.4.1 Learning Outcomes Once this tutorial is completed students will learn to:

- Use the well-tempered target distribution and understand its usefulness
- Construct biases in 1 and 2 dimensions.

8.8.7.4.2 Resources The `tarball` for this project contains the following folders:

- Example1 : Contains the input file for the the first example.
- Example2 : Contains the input file for the the second example.

8.8.7.4.3 Summary of theory One of the most useful target distribution is the well-tempered one. The well-tempered target distribution is [121] :

$$p(\mathbf{s}) = \frac{e^{-(\beta/\gamma)F(\mathbf{s})}}{\int d\mathbf{s}\, e^{-(\beta/\gamma)F(\mathbf{s})}}$$

where γ is the so-called bias factor. It is possible to show that:

$$p(\mathbf{s}) = \frac{[P(\mathbf{s})]^{1/\gamma}}{\int d\mathbf{s}\, [P(\mathbf{s})]^{1/\gamma}}$$

where $P(\mathbf{s})$ is the unbiased distribution of CVs. Therefore the target distribution is the unbiased distribution with enhanced fluctuations and lowered barriers. This is the same distribution as sampled in well-tempered metadynamics. The advantages of this distribution are that the features of the FES (metastable states) are preserved and that the system is not forced to sample regions of high free energy as it would if we had chosen the uniform target

distribution. This is specially important when biasing 2 CVs and there are large regions of very high free energy and therefore they represent un-physical configurations.
There is a caveat though, $p(\mathbf{s})$ depends on $F(\mathbf{s})$ that is the function that we are trying to calculate. One way to approach this problem is to calculate $p(\mathbf{s})$ self-consistently [121], for instance at iteration k:

$$p^{(k+1)}(\mathbf{s}) = \frac{e^{-(\beta/\gamma)F^{(k+1)}(\mathbf{s})}}{\int d\mathbf{s}\, e^{-(\beta/\gamma)F^{(k+1)}(\mathbf{s})}}$$

where:

$$F^{(k+1)}(\mathbf{s}) = -V^{(k)}(\mathbf{s}) - \frac{1}{\beta}\log p^{(k)}(\mathbf{s})$$

Normally $p^{(0)}(\mathbf{s})$ is taken to be uniform. Therefore the target distribution evolves in time until it becomes stationary when the simulation has converged. It has been shown that in some cases the convergence is faster using the well-tempered target distribution than using the uniform $p(\mathbf{s})$ [121].

8.8.7.4.4 Instructions

8.8.7.4.4.1 The system

We will consider the same system employed in previous tutorials.

8.8.7.4.4.2 Example 1: Enhancing fluctuations

We consider Example 2 of the VES 1 tutorial. In that case we used a uniform target distribution that at some value decayed to zero. In this case we will use a product of two distributions:

$$p(s) = \frac{p_{\mathrm{WT}}(s)\, p_{\mathrm{barrier}}(s)}{\int ds\, p_{\mathrm{WT}}(s)\, p_{\mathrm{barrier}}(s)}$$

where $p_{\mathrm{WT}}(s)$ is the well-tempered target distribution and:

$$p_{\mathrm{barrier}}(s) = \begin{cases} \frac{1}{C} & \text{if } s < s_0 \\ \frac{1}{C} e^{-\frac{(s-s_0)^2}{2\sigma^2}} & \text{if } s > s_0 \end{cases}$$

with C a normalization factor. The files needed for this exercise are in the directory Example1. This target distribution can be specified in plumed using:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-03-ves2.txt
__FILL__
td_uniform: TD_UNIFORM MINIMA=0.23 MAXIMA=0.6 SIGMA_MAXIMA=0.05
td_welltemp: TD_WELLTEMPERED BIASFACTOR=5
td_combination: TD_PRODUCT_COMBINATION DISTRIBUTIONS=td_uniform,td_welltemp

VES_LINEAR_EXPANSION ...
 ARG=d1
 BASIS_FUNCTIONS=bf1
 LABEL=b1
 TEMP=300.0
 GRID_BINS=300
 TARGET_DISTRIBUTION=td_combination
... VES_LINEAR_EXPANSION
```

As usual, we run the example using the run.sh script. As the simulation progresses we can track the evolution of the target distribution. At variance with previous simulations where $p(s)$ was stationary, in this case it evolves in time. The $p(s)$ is dumped every 500 steps in a file named targetdist.b1.iter-<iteration-number>.data. You can plot these files manually or using the script plotTargetDistrib.gpi. The result should be similar to the following plot where the distribution at different times has been shifted to see more clearly the difference.
To shed some light on the nature of the well-tempered target distribution, we will compare the unbiased and biased distribution of CVs. The unbiased distribution of CVs $P(s)$ is calculated by constructing a histogram of the CVs with weights given by:

$$w(\mathbf{R}) \propto e^{\beta V(\mathbf{s})}.$$

The biased distribution of CVs $p'(s)$ is calculated also by constructing a histogram of the CVs but in this case each point is assigned equal weights. The prime is added in $p'(s)$ to distinguish the biased distribution from the target distribution $p(s)$. If the simulation has converged then $p'(s) = p(s)$. The files needed for this calculation are located in the Reweight directory. Since this simulation converges fast as compared to previous ones, we only disregard the first 1 ns of simulation:

```
sed '2,5000d' ../COLVAR > COLVAR
```

To calculate the biased and unbiased distribution of CVs we use the following PLUMED input:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-03-ves2.txt
__FILL__
distance:       READ FILE=COLVAR  IGNORE_TIME VALUES=d1
ves:       READ FILE=COLVAR IGNORE_TIME VALUES=b1.bias

weights: REWEIGHT_BIAS TEMP=300 ARG=ves.bias

HISTOGRAM ...
  ARG=distance
  GRID_MIN=0.23
  GRID_MAX=0.8
  GRID_BIN=301
  BANDWIDTH=0.006
  LABEL=hh1
... HISTOGRAM

HISTOGRAM ...
  ARG=distance
  GRID_MIN=0.23
  GRID_MAX=0.8
  GRID_BIN=301
  BANDWIDTH=0.006
  LOGWEIGHTS=weights
  LABEL=hh2
... HISTOGRAM

DUMPGRID GRID=hh1 FILE=histo_biased FMT=%24.16e
DUMPGRID GRID=hh2 FILE=histo_unbiased FMT=%24.16e
```

If you do not understand what this input does, you might want to read once again the previous tutorials. The histograms histo_biased and histo_unbiased correspond to $p'(s)$ and $P(s)$, respectively. We are interested in comparing the unbiased distribution of CVs $P(s)$ and the well-tempered distribution $p_{\mathrm{WT}}(s)$. We know from the equations above that:

$$p_{\mathrm{WT}}(s) \propto [P(\mathbf{s})]^{1/\gamma},$$

and also,

$$p_{\mathrm{WT}}(s) \propto p(s)/p_{\mathrm{barrier}}(s) \propto p'(s)/p_{\mathrm{barrier}}(s).$$

Therefore we have two ways to calculate the well-tempered target distribution. We can compare $P(s)$ and the well-tempered target distribution calculated in two ways with the following gnuplot lines:

```
biasFactor=5.
invBiasFactor=1./biasFactor
pl "./histo_unbiased" u 1:(-log($2)) w l title "P(s)"
repl "./histo_unbiased" u 1:(-log($2**invBiasFactor)) w l title "[P(s)]^(1/gamma)"
repl "< paste ./histo_biased ../targetdist.b1.iter-0.data" u 1:(-log($2/$5)) w l title "Sampled p(s)"
```

There is also a gnuplot script plot.gpi that should do everything for you. The output should be similar to the next plot where we plot the negative logarithm of the distributions.
Notice that as expected both equations to calculate $p_{\mathrm{WT}}(s)$ agree. Also the association barrier of $5\ k_{\mathrm{B}}T$ becomes of $1\ k_{\mathrm{B}}T$ when the well-tempered target distribution is sampled.
The take home message of this tutorial is that when the well-tempered target distribution is employed, the biased distribution of CVs preserves all the features of the unbiased distribution, but barriers are lowered. Equivalently one may say that fluctuations are enhanced.

8.8.7.4.4.3 Example 2: Constructing a 2 dimensional bias In this example we will construct a 2 dimensional bias on the distance Na-Cl and the coordination number of Na with respect to O. The files to run this example are included in the Example2 folder. Two dimensional biases in VES can be written:

$$V(s_1, s_2; \alpha) = \sum_{i_1, i_2} \alpha_{i_1, i_2}\, f_{i_1}(s_1)\, f_{i_2}(s_2),$$

where $f_{i_1}(s_1)$ and $f_{i_2}(s_2)$ are the basis functions. We will choose to expand the bias potential in Legendre polynomials up to order 20 in both dimensions.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-03-ves2.txt
__FILL__
# CV1
BF_LEGENDRE ...
 ORDER=20
 MINIMUM=0.23
 MAXIMUM=0.8
 LABEL=bf1
... BF_LEGENDRE

# CV2
BF_LEGENDRE ...
 ORDER=20
 MINIMUM=2.5
 MAXIMUM=7.5
 LABEL=bf2
... BF_LEGENDRE
```

We have chosen the interval [0.23,0.8] nm for the distance and [2.5,7.5] for the coordination number. The total number of non-zero coefficients will be 400. In the coefficients file the indices i_1 and i_2 are given by the first two columns. We use the well-tempered target distribution together with a barrier at 0.6 nm on the distance Na-Cl.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-03-ves2.txt
__FILL__
td_uniform: TD_UNIFORM MINIMA=0.2,2.5 MAXIMA=0.6,7.5 SIGMA_MAXIMA=0.05,0.0
td_welltemp: TD_WELLTEMPERED BIASFACTOR=5
td_combination: TD_PRODUCT_COMBINATION DISTRIBUTIONS=td_uniform,td_welltemp

VES_LINEAR_EXPANSION ...
 ARG=d1,coord
 BASIS_FUNCTIONS=bf1,bf2
 LABEL=b1
 TEMP=300.0
 GRID_BINS=300,300
 TARGET_DISTRIBUTION=td_combination
... VES_LINEAR_EXPANSION
```

We can now run the simulation and control its progress. Since there are 400 coefficients we choose the largest (in absolute value) to control the convergence of the simulation. In this case we have chosen coefficients with indices (i_1, i_2) as (1,0), (0,1), (1,1), (2,1), and (0,2). You can plot the evolution of the coefficients using the gnuplot script plotCoeffs.gpi. This plot should look similar to the following one. The bias therefore converges smoothly as in one dimensional problems.
The estimated FES can also be plotted to control the progress of the simulation. For instance in gnuplot,

```
set xr [0.23:0.7]
set yr [3:7]
set zr [0:6]
set cbr [0:6]
set pm3d map
temp=2.494
spl "./fes.b1.iter-1000.data" u 1:2:($3/temp) w pm3d notitle
```

There is a gnuplot script plotFes.gpi that generates the following plot for the FES after 10 ns of simulation:
This FES agrees with that calculated through reweighting in the metadynamics tutorial.
As an exercise, you can repeat this simulation using the uniform target distribution instead of the well-tempered $p(s)$. Compare the convergence time of both simulations. Discuss the reasons why the algorithm converges faster to the well-tempered target distribution than to the uniform one. Does it make sense to sample all the CV space uniformly?

8.8.7.4.5 Final remarks At this point the student has acquired experience with several characteristics of the VES method. There is one tool that has proven to be instrumental for many problems and that has not yet been discussed in this series of tutorials: the use of multiple walkers. This tool will be the subject of another tutorial.

8.8.7.5 MARVEL-VES tutorial (Lugano Feb 2017): Kinetics

8.8.7.5.1 Aims The aim of this tutorial is to introduce the use of VES for obtaining kinetic information of activated processes. We will learn how to set up a biased simulation using a fixed flooding potential acting on a relevant slow degree of freedom. We can then scale the accelerated MD time in a post-processing procedure.

8.8.7.5.2 Learning Outcomes Once this tutorial is completed students will:

- Optimize a bias using VES with an energy cutoff to selectively fill low regions of the free energy surface.
- Use the optimized bias to observe several rare event transitions
- Post-process the accelerated trajectory to obtain an unbiased estimate of the transition rate
- Compute the empirical cumulative distribution of first passage times and compare to a theoretical model.

8.8.7.5.3 Resources The `tarball` for this project contains the following files:

- CH.airebo : Force field parameters for LAMMPS
- input : LAMMPS input script
- data.start : Starting configuration in LAMMPS format
- plumed.dat : Example PLUMED file
- time-reweighting.py : Python script for post-processing trajectory
- TRAJECTORIES-1700K : A directory containing many trajectories for post processing
- get-all-fpt.py : A python script to extract transition times from the trajectories
- cdf.analysis.py : A python script to compute the cumulative probability distribution and perform KS test

8.8.7.5.4 Requirements

- python with numpy, scipy, and statsmodels
- LAMMPS compiled with MANYBODY package
- VMD for visualization

8.8.7.5.5 Instructions

8.8.7.5.5.1 Exercise 1A. Preliminary investigation using VES As an example of an activated process in materials science, we will work with the Stone-Wales transformation in a carbon nanotube. The `tarball` for this project contains the inputs necessary to run a simulation in LAMMPS for a 480 atom carbon nanotube. We use the AIREBO (Adaptive Intermolecular Reactive Empirical Bond Order) force field parameters which can approximately describe C-C bond breakage and formation at reasonable computational cost. For CVs we can use the coordination number in PLUMED to measure the number of covalent bonds among different groups of atoms. The transformation involves breaking two C-C bonds and forming two alternative C-C bonds.A definition of these CVs as well as the relevant C-C bonds are depicted in Figure stone-wales. We prepare a PLUMED input file as follows.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ves-lugano2017-04-kinetics.txt
# set distance units to angstrom, time to ps, and energy to eV
UNITS LENGTH=A TIME=ps ENERGY=eV

# define two variables
COORDINATION GROUPA=229,219 GROUPB=238,207 R_0=1.8 NN=8 MM=16 PAIR LABEL=CV1
COORDINATION GROUPA=229,219 GROUPB=207,238 R_0=1.8 NN=8 MM=16 PAIR LABEL=CV2

# the difference between variables
COMBINE ARG=CV1,CV2 COEFFICIENTS=1,-1 POWERS=1,1 LABEL=d1 PERIODIC=NO
```

In the above, the first line sets the energy units. The second and third line define the two CVs for the C-C covalent bonds. (We have chosen atoms 238,207,229 and 219; however this choice is arbitrary and other atoms could equally well have been chosen.) Lastly, we define a simple approximate reaction coordinate given by the difference between CV1 and CV2 that we can use to monitor the transition.
Next we will use VES to drive the transformation at 1700 K. We bias the formation of bonds DB and AC shown in Figure stone-wales using CV2 which changes from 0 to 2 during the transformation. (Although a more rigorous treatment would bias both CVs, for this tutorial we will simplify things and work in only one dimension). We choose a Chebyshev polynomial basis set up to order 36

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-04-kinetics.txt
__FILL__
# The basis set to use
bf1: BF_CHEBYSHEV ORDER=36 MINIMUM=0.0 MAXIMUM=2.0
```

and we tell PLUMED to use VES acting on CV2 with a free energy cutoff

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-04-kinetics.txt
__FILL__
td_uniform: TD_UNIFORM

VES_LINEAR_EXPANSION ...
  ARG=CV2
  BASIS_FUNCTIONS=bf1
  LABEL=variational
  TEMP=1700
  BIAS_CUTOFF=15.0
  BIAS_CUTOFF_FERMI_LAMBDA=10.0
  TARGET_DISTRIBUTION=td_uniform
... VES_LINEAR_EXPANSION
```

Here we are biasing CV2 with the basis set defined above. The final two lines impose the cutoff at 15 eV. The cutoff is of the form

$$\frac{1}{1+e^{\lambda[F(s)-F_c]}}$$

where λ (inverse energy units) controls how sharply the function goes to zero. Above we have set $\lambda = 10.0$ to ensure the cutoff goes sharply enough to zero.
The following input updates the VES bias every 200 steps, writing out the bias every 10 iterations. To enforce the energy cutoff we also need to update the target distribution which we do every 40 iterations with the TARGETDI↩ ST_STRIDE flag.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-04-kinetics.txt
__FILL__
OPT_AVERAGED_SGD ...
  BIAS=variational
  STRIDE=200
  LABEL=var-S
  STEPSIZE=0.1
  COEFFS_FILE=coeffs.dat
  BIAS_OUTPUT=10
  TARGETDIST_STRIDE=40
  TARGETDIST_OUTPUT=40
  COEFFS_OUTPUT=1
... OPT_AVERAGED_SGD
```

Finally, we will stop the simulation when the transition occurs using the COMMITTOR in PLUMED

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-04-kinetics.txt
__FILL__
COMMITTOR ARG=d1 BASIN_LL1=-2.0 BASIN_UL1=-1.0 STRIDE=600
```

Run the simulation using LAMMPS

```
lmp_mpi < input
```

and plot the last few bias output files (bias.variational.iter-n.data) using gnuplot. What is the difference between the maximum and minimum values of the bias obtained during the simulation? Is the cutoff value sufficient to cross the barrier? Is the cutoff value too large?
Figure Figure1A shows an example bias potential after 90 iterations. Note that a cutoff of 15 eV is too large as the system transitions before the bias reaches the prescribed cutoff. From Figure Figure1A we see a barrier height of approximately 7.3 eV.
The output also produces an movie.xyz file which can be viewed in vmd. For better visualization, first change atom id from 1 to C by typing

```
sed "s/^1 /C /" movie.xyz > newmovie.xyz
```

Then load the newmovie.xyz into vmd and choose Graphics --> Representations and set Drawing Method to DynamicBonds. Create a second representation by clicking Create Rep and set the Drawing Method to VDW. Change the sphere scale to 0.3 and play the movie. You should observe the Stone-Wales transformation right before the end of the trajectory.

8.8.7.5.5.2 Exercise 1B. Set up and run a VES bias imposing a cutoff In this exercise we will run a VES simulation to fill the FES only up to a certain cutoff. This will be the first step in order to obtain kinetic information from biased simulations. In the previous section, we observed that a cutoff of 15 eV is too strong for our purpose. Change the cutoff energy from 15.0 to 6.0 eV by setting

```
BIAS_CUTOFF=6.0
```

and rerun the simulation from Exercise 1A. [Note: In practice one can use multiple walkers during the optimization by adding the flag MULTIPLE_WALKERS]
Plot some of the bias files (bias.variational.iter-n.data) that are printed during the simulation using gnuplot. At the end of the simulation, you should be able to reproduce something like Figure Figure1B. Is the bias converging? If so, how many iterations does it require to converge?
Figure Figure1B shows the bias potential after 70,80, and 90 iteration steps. Note that the bias has reached the cutoff and goes to zero at around 1 Angstrom.

8.8.7.5.5.3 Exercise 2. Using a fixed bias as a flooding potential to obtain rates In this exercise we will use the bias obtained above as a static umbrella potential. We will set up and run a new trajectory to measure the first passage time of escape from the well.
We can extract the coefficients that we need from the final iteration in Exercise-1B above.

```
tail -n 47 coeffs.dat > fixed-coeffs.dat
```

Now create a new directory from which you will run a new simulation and copy the necessary input files (including the fixed-coeffs.dat) into this directory. Modify the PLUMED file so that the optimized coefficients are read by the VES_LINEAR_EXPANSION

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-04-kinetics.txt
__FILL__
td_uniform: TD_UNIFORM

VES_LINEAR_EXPANSION ...
  ARG=CV2
  BASIS_FUNCTIONS=bf1
  LABEL=variational
  TEMP=1700
  BIAS_CUTOFF=6.0
  BIAS_CUTOFF_FERMI_LAMBDA=10.0
  TARGET_DISTRIBUTION=td_uniform
  COEFFS=fixed-coeffs.dat
... VES_LINEAR_EXPANSION
```

The final line specifies the coefficients to be read from a file.
Make sure to remove the lines for the stochastic optimization (OPT_AVERAGED_SGD) as we no longer wish to update the bias.
We will also perform metadynamics with an infrequent deposition stride to ensure that the trajectory does not get stuck in any regions where the bias potential is not fully converged The following implements metadynamics on both CV1 and CV2 with a deposition stride of 4000 steps and a hill height of 0.15 eV.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-04-kinetics.txt
__FILL__
METAD ...
   ARG=CV1,CV2
   SIGMA=0.2,0.2
   HEIGHT=0.15
   PACE=4000
   LABEL=metad
... METAD
```

Again we will use the COMMITTOR to stop the trajectory after the transition.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/ves-lugano2017-04-kinetics.txt
__FILL__
COMMITTOR ARG=d1 BASIN_LL1=-2.0 BASIN_UL1=-1.0
```

Now run a trajectory with the fixed bias. What is the time (biased) to escape? Plot the trajectory of the approximate reaction coordinate (column 4 in the COLVAR) in gnuplot. An example is shown in Figure Figure2.
Also look at the metadynamics bias in the column labeled metad.bias. How does the magnitude compare to the bias from VES in column variational.bias?

The crossing time for a single event doesn't tell us much because we don't have any statistics on the transition events. To obtain the mean first passage time, we have to repeat the calculation many times. To generate statistically independent samples, we have to change the seed for the random velocities that are generated in the LAMMPS input file. In a new directory, copy the necessary files and edit the following line in the input file

```
velocity      all create 1700.  495920
```

Choose a different 6 digit random number and repeat Exercise 2. How does the escape time compare to what you obtained before? Repeat the procedure several times with different velocity seeds to get a distribution of first passage times. Make sure you launch each simulation from a separate directory and keep all COLVAR files as you will need them in the next section where we will analyze the transition times.

8.8.7.5.5.4 Exercise 3. Post processing to obtain unbiased estimate for the transition rate In the previous section you generated several trajectories with different first passage times. However, these times need to be re-weighted to correct for the bias potential. We can scale the time according to the hyperdynamics formula

$$t^* = \Delta t_{MD} \sum_i^n e^{\beta V(s)}$$

Note that we need to add the total bias at each step, coming from both the VES bias and metadynamics. The python script time-reweighting.py will read the COLVAR from Exercise 2 and print the final reweighted time (in seconds). Open the script to make sure you understand how it works.
Run the script using

```
python time-reweighting.py
```

Note that the output first passage time is converted to seconds. What is the acceleration factor of our biased simulation? (i.e. the ratio of biased to unbiased transition times) The script also produces a time-reweighted trajectory COLVAR-RW for a specified CV (here we choose the approximate reaction coordinate, d1). Plot the reweighted COLVAR-RW in gnuplot and compare the original vs. time-reweighted trajectories. In particular, what effect does rescaling have on the time step?
Rerun the script for each of the trajectories you have run with the fixed bias and compute the mean first passage time from your data. The Stone-Wales transformation at 1700 K is estimated in fullerene to be $\sim$10 days. How does your average time compare to this value?
The distribution of first passage times for an activated processes typically follows an exponential distribution. Instead of directly making a histogram of the first passage times, we can look at the cumulative distribution function which maps a value x to the fraction of values less than or equal to x. Since we are computing the cumulative distribution from a data set, this is called the empirical cumulative distribution (ECDF). On the other hand, the theoretical cumulative distribution (CDF) of an exponentially distributed random process is

$$P(t) = 1 - e^{-t/\tau}$$

where τ is the mean first passage time.
In order to calculate the ECDF we need many trajectories. Several COLVAR files are included in the TRAJECT↩ ORIES-1700K directory. The script get-all-fpt.py is a modified version of time-reweighting.py and will calculate the first passage time (fpt) from all the simulation data and will output the times to a file fpt.dat. Run the script from the TRAJECTORIES-1700K directory

```
python get-all-fpt.py
```

You should obtain the output file fpt.dat which has a list of all the times. You can append your own values you obtained from Exercise 2 to the end of the list to increase the number of data points.
The script cdf-analysis.py will compute the ECDF and fit the distribution to the theoretical CDF. The script uses the statsmodels Python module. Run the script with

```
python cdf-analyysis.py
```

from the same directory where the fpt.dat file is located. The script prints both the mean first passage time of the data as well as the fit parameter τ. How do these two values compare?
Figure Figure3 shows an example fit of the ECDF to the theoretical CDF for a Poisson process. To test the reliability of the fit, we can generate some data set according the theoretical CDF and perform a two-sample Kolmogorov-↩ Smirnov (KS) test. The KS test provides the probability that the two sets of data are drawn from the same underlying distribution expressed in the so-called p-value. The null hypothesis is typically rejected for p-value < 0.05. The script cdf-analysis.py also performs the KS test and prints the p-value. What is the p-value we obtain from your dataset of transition times?

8.9 MAZE

maze is a module for PLUMED2, which implements enhanced sampling methods for ligand unbinding from protein tunnels. The maze module is developed and maintained by Jakub Rydzewski at the Institute of Physics, Nicolaus Copernicus University, Torun, Poland. See this link for additional information.
The maze module is an optional module for PLUMED2 that needs to be enabled when configuring the compilation of PLUMED2. You can either pass a flag '--enable-modules=maze' or a '--enable-modules=all' when running the configure script.
See the following sections for further information:

- Loss
- Optimizers
- Biases

8.9.1 Loss

The following list contains the loss functions available in the maze module.

MAZE_LOSS	

8.9.1.1 MAZE_LOSS

	This is part of the maze module
	It is only available if you configure PLUMED with ./configure –enable-modules=maze . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Define a coarse-grained loss function describing interactions in a ligand-protein complex, which is minimized during the simulation to obtain ligand unbinding pathways.
The loss function is the following:

$$\mathcal{L} = \sum_{i=1}^{N_p} r_i^{-\alpha} e^{-\beta r_i^{-\gamma}},$$

where N_p is the number of ligand-protein atom pairs, r is a re-scaled distance between the ith pair, and α, β, γ are the positive parameters defined in that order by the PARAMS keyword.

Examples

The loss function can be defined in the following way:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MAZE_LOSS.tmp
l: MAZE_LOSS PARAMS=1,1,1
```

Glossary of keywords and components

Compulsory keywords

PARAMS	Parameters for the loss function.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances

8.9.2 Optimizers

The following list contains the optimizers available in the maze module.

MAZE_MEMETIC_SAMPLING	
MAZE_RANDOM_ACCELERATION_MD	
MAZE_RANDOM_WALK	
MAZE_SIMULATED_ANNEALING	
MAZE_STEERED_MD	

8.9.2.1 MAZE_MEMETIC_SAMPLING

	This is part of the maze module
	It is only available if you configure PLUMED with ./configure –enable-modules=maze . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculates the biasing direction along which the ligand unbinds by minimizing the MAZE_LOSS function. The optimal biasing direction is determined by performing a population-based memetics search with local heuristics.

Examples

Every optimizer implemented in the maze module needs a loss function as an argument, and it should be passed using the MAZE_LOSS keyword.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MAZE_MEMETIC_SAMPLING.tmp
MAZE_MEMETIC_SAMPLING ...
  LABEL=ma

  LOSS=l

  N_ITER=10
  OPTIMIZER_STRIDE=200

  CAPACITY=20

  LOCAL_SEARCH_ON
  N_LOCAL_ITER=10
  LOCAL_SEARCH_RATE=0.1
  LOCAL_SEARCH_TYPE=stochastic_hill_climbing

  MUTATION_RATE=0.1
  MATING_RATE=0.7
  CAUCHY_ALPHA=0
  CAUCHY_BETA=0.1

  LIGAND=2635-2646
  PROTEIN=1-2634
... MAZE_MEMETIC_SAMPLING
```

As shown above, each optimizer should be provided with the LIGAND and the PROTEIN keywords.
The neighbor list can be turned on by providing the NLIST keyword.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
x	Optimal biasing direction; x component.
y	Optimal biasing direction; y component.
z	Optimal biasing direction; z component.
loss	Loss function value defined by the provided pairing function.
sr	Sampling radius. Reduces sampling to the local proximity of the ligand position.

The atoms involved can be specified using

LIGAND	Indices of ligand atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
PROTEIN	Indices of protein atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

N_ITER	Number of optimization steps. Required only for optimizers, do not pass this keyword to the fake optimizers (results in crash) , e.g., random walk, steered MD, or random acceleration MD.
OPTIMIZER_STRIDE	Optimizer stride. Sets up a callback function that launches the optimization process every OPTIMIZER_STRIDE.
CAPACITY	Sampling set size.
MUTATION_RATE	Probability of mutation.
MATING_RATE	Probability of mating.
CAUCHY_ALPHA	Mean of Cauchy distribution for sampling.
CAUCHY_BETA	Spread of Cauchy distribution for sampling.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the simulation in serial – used only for debugging purposes, should not be used otherwise.
PAIR	(default=off) Pair only the 1st element of the 1st group with the 1st element in the second, etc.

NLIST	(default=off) Use a neighbor list of ligand-protein atom pairs to speed up the calculating of the distances.
LOCAL_SEARCH_ON	(default=off) Turn local search on.
NL_CUTOFF	Neighbor list cut-off for the distances of ligand-protein atom pairs.
NL_STRIDE	Update stride for the ligand-protein atom pairs in the neighbor list.
LOSS	Loss function describing ligand-protein interactions required by every optimizer.
N_LOCAL_ITER	Number of local search iterations.
LOCAL_SEARCH_RATE	Rate of mutation in local search.
LOCAL_SEARCH_TYPE	Type of local search.

8.9.2.2 MAZE_RANDOM_ACCELERATION_MD

	This is part of the maze module
	It is only available if you configure PLUMED with ./configure –enable-modules=maze . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Performs random acceleration MD within the protein matrix.

Examples

Every optimizer implemented in the maze module needs a loss function as an argument, and it should be passed using the MAZE_LOSS keyword.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/MAZE_RANDOM_ACCELERATION_MD.tmp
MAZE_RANDOM_ACCELERATION_MD ...
  LABEL=rw

  OPTIMIZER_STRIDE=_
  LOSS=l
  RMIN=_

  LIGAND=2635-2646
  PROTEIN=1-2634
... MAZE_RANDOM_ACCELERATION_MD
```

As shown above, each optimizer should be provided with the LIGAND and the PROTEIN keywords.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
x	Optimal biasing direction; x component.
y	Optimal biasing direction; y component.
z	Optimal biasing direction; z component.
loss	Loss function value defined by the provided pairing function.
sr	Sampling radius. Reduces sampling to the local proximity of the ligand position.
dist	Distance traveled in one sampling interval.
tdist	Total distance traveled by biased atoms.

The atoms involved can be specified using

LIGAND	Indices of ligand atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
PROTEIN	Indices of protein atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

OPTIMIZER_STRIDE	Optimizer stride. Sets up a callback function that launches the optimization process every OPTIMIZER_STRIDE.
R_MIN	Minimal distance traveled before sampling a new direction of biasing.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the simulation in serial – used only for debugging purposes, should not be used otherwise.
PAIR	(default=off) Pair only the 1st element of the 1st group with the 1st element in the second, etc.
NLIST	(default=off) Use a neighbor list of ligand-protein atom pairs to speed up the calculating of the distances.
NL_CUTOFF	Neighbor list cut-off for the distances of ligand-protein atom pairs.
NL_STRIDE	Update stride for the ligand-protein atom pairs in the neighbor list.
LOSS	Loss function describing ligand-protein interactions required by every optimizer.

8.9.2.3 MAZE_RANDOM_WALK

	This is part of the maze module
	It is only available if you configure PLUMED with ./configure –enable-modules=maze . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Fake optimizer that can be used for debugging.
This is dummy optimizer that can be used for debugging and monitoring purposes. It returns a random direction of biasing, changed every OPTIMIZER_STRIDE.
Performs a random walk within the protein matrix.

Examples

Every optimizer implemented in the maze module needs a loss function as an argument, and it should be passed using the MAZE_LOSS keyword.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MAZE_RANDOM_WALK.tmp
MAZE_RANDOM_WALK ...
```

```
  LABEL=rw

  LOSS=l
  OPTIMIZER_STRIDE=200

  LIGAND=2635-2646
  PROTEIN=1-2634
... MAZE_RANDOM_WALK
```

As shown above, each optimizer should be provided with the LIGAND and the PROTEIN keywords.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
x	Optimal biasing direction; x component.
y	Optimal biasing direction; y component.
z	Optimal biasing direction; z component.
loss	Loss function value defined by the provided pairing function.
sr	Sampling radius. Reduces sampling to the local proximity of the ligand position.

The atoms involved can be specified using

LIGAND	Indices of ligand atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
PROTEIN	Indices of protein atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

OPTIMIZER_STRIDE	Optimizer stride. Sets up a callback function that launches the optimization process every OPTIMIZER_STRIDE.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the simulation in serial – used only for debugging purposes, should not be used otherwise.
PAIR	(default=off) Pair only the 1st element of the 1st group with the 1st element in the second, etc.

NLIST	(default=off) Use a neighbor list of ligand-protein atom pairs to speed up the calculating of the distances.
NL_CUTOFF	Neighbor list cut-off for the distances of ligand-protein atom pairs.
NL_STRIDE	Update stride for the ligand-protein atom pairs in the neighbor list.
LOSS	Loss function describing ligand-protein interactions required by every optimizer.

8.9.2.4 MAZE_SIMULATED_ANNEALING

	This is part of the maze module
	It is only available if you configure PLUMED with ./configure –enable-modules=maze . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculates the biasing direction along which the ligand unbinds by minimizing the MAZE_LOSS function. The optimal biasing direction is determined by performing simulated annealing.

Examples

Every optimizer implemented in the maze module needs a loss function as an argument, and it should be passed using the MAZE_LOSS keyword.
In the following example simulated annealing is launched for 1000 iterations as the optimizer for the loss function every 200 ps. The geometric cooling scheme is used.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/MAZE_SIMULATED_ANNEALING.tmp
UNITS LENGTH=A TIME=ps ENERGY=kcal/mol

MAZE_SIMULATED_ANNEALING ...
  LABEL=sa

  LOSS=l

  N_ITER=1000
  OPTIMIZER_STRIDE=200

  PROBABILITY_DECREASER=300
  COOLING=0.95
  COOLING_TYPE=geometric

  LIGAND=2635-2646
  PROTEIN=1-2634
... MAZE_SIMULATED_ANNEALING
```

As shown above, each optimizer should be provided with the LIGAND and the PROTEIN keywords.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
x	Optimal biasing direction; x component.
y	Optimal biasing direction; y component.
z	Optimal biasing direction; z component.
loss	Loss function value defined by the provided pairing function.
sr	Sampling radius. Reduces sampling to the local proximity of the ligand position.

The atoms involved can be specified using

LIGAND	Indices of ligand atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
PROTEIN	Indices of protein atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

N_ITER	Number of optimization steps. Required only for optimizers, do not pass this keyword to the fake optimizers (results in crash) , e.g., random walk, steered MD, or random acceleration MD.
OPTIMIZER_STRIDE	Optimizer stride. Sets up a callback function that launches the optimization process every OPTIMIZER_STRIDE.
PROBABILITY_DECREASER	Temperature-like parameter that is decreased during optimization to modify the Metropolis-Hastings acceptance probability.
COOLING	Reduction factor for PROBABILITY_DECREASER, should be in (0, 1].
COOLING_SCHEME	Cooling scheme: geometric.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the simulation in serial – used only for debugging purposes, should not be used otherwise.
PAIR	(default=off) Pair only the 1st element of the 1st group with the 1st element in the second, etc.
NLIST	(default=off) Use a neighbor list of ligand-protein atom pairs to speed up the calculating of the distances.
NL_CUTOFF	Neighbor list cut-off for the distances of ligand-protein atom pairs.
NL_STRIDE	Update stride for the ligand-protein atom pairs in the neighbor list.
LOSS	Loss function describing ligand-protein interactions required by every optimizer.

8.9.2.5 MAZE_STEERED_MD

	This is part of the maze module
	It is only available if you configure PLUMED with ./configure –enable-modules=maze . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Performs a linear unbinding along a predefined biasing direction that needs to be provided using the PULLING keyword.

Examples

Every optimizer implemented in the maze module needs a loss function as an argument, and it should be passed using the MAZE_LOSS keyword.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/MAZE_STEERED_MD.tmp
MAZE_STEERED_MD ...
  LABEL=smd

  LOSS=l
  PULLING=0.3,0.3,0.3
  OPTIMIZER_STRIDE=_

  LIGAND=2635-2646
  PROTEIN=1-2634
... MAZE_STEERED_MD
```

As shown above, each optimizer should be provided with the LIGAND and the PROTEIN keywords.

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
x	Optimal biasing direction; x component.
y	Optimal biasing direction; y component.
z	Optimal biasing direction; z component.
loss	Loss function value defined by the provided pairing function.
sr	Sampling radius. Reduces sampling to the local proximity of the ligand position.
tdist	Total distance traveled by biased atoms.

The atoms involved can be specified using

LIGAND	Indices of ligand atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
PROTEIN	Indices of protein atoms.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

OPTIMIZER_STRIDE	Optimizer stride. Sets up a callback function that launches the optimization process every OPTIMIZER_STRIDE.
PULLING	Constant biasing direction.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the simulation in serial – used only for debugging purposes, should not be used otherwise.
PAIR	(default=off) Pair only the 1st element of the 1st group with the 1st element in the second, etc.
NLIST	(default=off) Use a neighbor list of ligand-protein atom pairs to speed up the calculating of the distances.
NL_CUTOFF	Neighbor list cut-off for the distances of ligand-protein atom pairs.
NL_STRIDE	Update stride for the ligand-protein atom pairs in the neighbor list.
LOSS	Loss function describing ligand-protein interactions required by every optimizer.

8.9.3 Biases

The following list contains the biases available in the maze module.

MAZE_OPTIMIZER_BIAS	

8.9.3.1 MAZE_OPTIMIZER_BIAS

	This is part of the maze module
	It is only available if you configure PLUMED with ./configure –enable-modules=maze . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Biases the ligand along the direction calculated by the chosen MAZE_OPTIMIZER.
OptimizerBias is a class deriving from Bias, and it is used to adaptively bias a ligand-protein system toward an optimal solution found by the chosen optimizer.
Remember to define the loss function (MAZE_LOSS) and the optimizer (MAZE_OPTIMIZER) prior to the adaptive bias for the optimizer.
The adaptive bias potential is the following:

$$V(\mathbf{x}_t) = \alpha \left(wt - (\mathbf{x} - \mathbf{x}^*_{t-\tau}) \cdot \frac{\mathbf{x}^*_t - \mathbf{x}_t}{\|\mathbf{x}^*_t - \mathbf{x}_t\|} \right)^2,$$

where $\mathbf{x}^*_t$ is the optimal solution at time t, w is the biasing rate, τ is the interval at which the loss function is minimized, and α is a scaled force constant.

Examples

In the following example the bias potential biases a ligand atom (which have to be given as an argument) with the biasing rate equal to 0.02 A/ps, and the biasing constant equal to 3.6 kcal/(mol A). It also takes an optimizer (see MAZE_OPTIMIZER).

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/MAZE_OPTIMIZER_BIAS.tmp
UNITS LENGTH=A TIME=ps ENERGY=kcal/mol

p: POSITION ATOM=2635 NOPBC

MAZE_OPTIMIZER_BIAS ...
  LABEL=bias
```

```
  ARG=p.x,p.y,p.z

  BIASING_RATE=0.02
  ALPHA=3.6

  OPTIMIZER=opt
... MAZE_OPTIMIZER_BIAS
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
force2	Square of the biasing force.
x	Optimal biasing direction: x component.
y	Optimal biasing direction: y component.
z	Optimal biasing direction: z component.
tdist	Total distance traveled by biased atoms.

Compulsory keywords

BIASING_RATE	Biasing rate.
ALPHA	Rescaled force constant.
OPTIMIZER	Optimization technique to minimize the collective variable for ligand unbinding: RANDOM_↩ WALK, STEERED_MD, RANDOM_ACCELERATION_MD, SIMULATED_ANNEALING, M↩ EMETIC_SAMPLING

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...

8.10 OPES (On-the-fly Probability Enhanced Sampling)

8.10.1 Overview

The OPES module contains the implementation of the on-the-fly probability enhanced sampling mehtod (OPES) [78] [128] [79].
The OPES method aims at sampling a given target distribution over the configuration space, $p^{tg}(\mathbf{x})$, different from the equilibrium Boltzmann distribution, $P(\mathbf{x}) \propto e^{-\beta U(\mathbf{x})}$. To do so, it incrementally builds a bias potential $V(\mathbf{x})$, by estimating on-the-fly the needed probability distributions:

$$V(\mathbf{x}) = -\frac{1}{\beta}\log\frac{p^{tg}(\mathbf{x})}{P(\mathbf{x})}\,.$$

The bias quickly becomes quasi-static and the desired properties, such as the free energy, can be calculated with a simple reweighting REWEIGHT_BIAS.
Depending on the kind of target distribution one wishes to sample, different OPES biases can be used.

8.10.2 Installation

This module is not installed by default. Add '--enable-modules=opes' to your './configure' command when building PLUMED to enable these features. See also List of modules.

8.10.3 Usage

The OPES module contains three bias actions, OPES_METAD and OPES_METAD_EXPLORE that sample metadynamics-like target distributions (e.g. the well-tempered one), and OPES_EXPANDED that samples expanded ensembles target distributions (replica-exchange-like). It also contains various expansion collective variables (ECVs) to define such expanded targets.

8.10.4 Contents

8.10.5 Biases

The following list contains the biases available in the OPES module.

OPES_EXPANDED	On-the-fly probability enhanced sampling with expanded ensembles for the target distribution.
OPES_METAD	On-the-fly probability enhanced sampling with metadynamics-like target distribution.
OPES_METAD_EXPLORE	On-the-fly probability enhanced sampling with well-tempered target distribution in exploreation mode.

8.10.5.1 OPES_EXPANDED

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

On-the-fly probability enhanced sampling with expanded ensembles for the target distribution.

This method is similar to the OPES method (OPES) with expanded ensembles target distribution (replica-exchange-like) [79].
An expanded ensemble is obtained by summing a set of ensembles at slightly different termodynamic conditions, or with slightly different Hamiltonians. Such ensembles can be sampled via methods like replica exchange, or this OPES_EXPANDED bias action. A typical example is a multicanonical simulation, in which a whole range of temperatures is sampled instead of a single one.
In oreder to define an expanded target ensemble we use expansion collective variables (ECVs), $\Delta u_\lambda(\mathbf{x})$. The bias at step n is

$$V_n(\mathbf{x}) = -\frac{1}{\beta}\log\left(\frac{1}{N_{\{\lambda\}}}\sum_\lambda e^{-\Delta u_\lambda(\mathbf{x})+\beta\Delta F_n(\lambda)}\right).$$

See Ref.[79] for more details on the method.
Notice that the estimates in the DELTAFS file are expressed in energy units, and should be multiplied by β to be dimensionless as in Ref.[79]. The DELTAFS file also contains an estimate of $c(t) = \frac{1}{\beta}\log\langle e^{\beta V}\rangle$. Similarly to OPES_METAD, it is printed only for reference, since $c(t)$ reaches a fixed value as the bias converges, and should NOT be used for reweighting. Its value is also needed for restarting a simulation.
You can store the instantaneous $\Delta F_n(\lambda)$ estimates also in a more readable format using STATE_WFILE and STATE_WSTRIDE. Restart can be done either from a DELTAFS file or from a STATE_RFILE, it is equivalent.
Contrary to OPES_METAD, OPES_EXPANDED does not use kernel density estimation.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_EXPANDED.tmp
# simulate multiple temperatures, as in parallel tempering
ene: ENERGY
ecv: ECV_MULTITHERMAL ARG=ene TEMP_MAX=1000
opes: OPES_EXPANDED ARG=ecv.* PACE=500
PRINT FILE=COLVAR STRIDE=500 ARG=ene,opes.bias
```

You can easily combine multiple ECVs. The OPES_EXPANDED bias will create a multidimensional target grid to sample all the combinations.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_EXPANDED.tmp
# simulate multiple temperatures while biasing a CV
ene: ENERGY
dst: DISTANCE ATOMS=1,2

ecv1: ECV_MULTITHERMAL ARG=ene TEMP_SET_ALL=200,300,500,1000
ecv2: ECV_UMBRELLAS_LINE ARG=dst CV_MIN=1.2 CV_MAX=4.3 SIGMA=0.5
opes: OPES_EXPANDED ARG=ecv1.*,ecv2.* PACE=500 OBSERVATION_STEPS=1

PRINT FILE=COLVAR STRIDE=500 ARG=ene,dst,opes.bias
```

If an ECV is based on more than one CV you must provide all the output component, in the proper order. You can use Regular Expressions for that, like in the following example.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_EXPANDED.tmp
# simulate multiple temperatures and pressures while biasing a two-CVs linear path
ene: ENERGY
vol: VOLUME
ecv_mtp: ECV_MULTITHERMAL_MULTIBARIC ...
  ARG=ene,vol
  TEMP=300
  TEMP_MIN=200
  TEMP_MAX=800
  PRESSURE=0.06022140857*1000 #1 kbar
  PRESSURE_MIN=0
  PRESSURE_MAX=0.06022140857*2000 #2 kbar
...

cv1: DISTANCE ATOMS=1,2
cv2: DISTANCE ATOMS=3,4
ecv_umb: ECV_UMBRELLAS_LINE ARG=cv1,cv2 TEMP=300 CV_MIN=0.1,0.1 CV_MAX=1.5,1.5 SIGMA=0.2 BARRIER=70

opes: OPES_EXPANDED ARG=(ecv_.*) PACE=500 WALKERS_MPI PRINT_STRIDE=1000

PRINT FILE=COLVAR STRIDE=500 ARG=ene,vol,cv1,cv2,opes.bias
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
work	**CALC_WORK**	total accumulated work done by the bias

Compulsory keywords

ARG	the label of the ECVs that define the expansion. You can use an $*$ to make sure all the output components of the ECVs are used, as in the examples above
PACE	how often the bias is updated
OBSERVATION_STEPS	(default=100) number of unbiased initial PACE steps to collect statistics for initialization
FILE	(default=DELTAFS) a file with the estimate of the relative Delta F for each component of the target and of the global c(t)
PRINT_STRIDE	(default=100) stride for printing to DELTAFS file, in units of PACE

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
STORE_STATES	(default=off) append to STATE_WFILE instead of ovewriting it each time
CALC_WORK	(default=off) calculate the total accumulated work done by the bias since last restart
WALKERS_MPI	(default=off) switch on MPI version of multiple walkers
SERIAL	(default=off) perform calculations in serial
FMT	specify format for DELTAFS file
STATE_RFILE	read from this file the Delta F estimates and all the info needed to RESTART the simulation
STATE_WFILE	write to this file the Delta F estimates and all the info needed to RESTART the simulation
STATE_WSTRIDE	number of MD steps between writing the STATE_WFILE. Default is only on CPT events (but not all MD codes set them)
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

8.10.5.2 OPES_METAD

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

On-the-fly probability enhanced sampling with metadynamics-like target distribution.
This On-the-fly probability enhanced sampling (OPES) method with metadynamics-like target distribution is described in [78].
This OPES_METAD action samples target distributions defined via their marginal $p^{tg}(\mathbf{s})$ over some collective variables (CVs), $\mathbf{s} = \mathbf{s}(\mathbf{x})$. By default OPES_METAD targets the well-tempered distribution, $p^{WT}(\mathbf{s}) \propto [P(\mathbf{s})]^{1/\gamma}$, where γ is known as BIASFACTOR. Similarly to METAD, OPES_METAD optimizes the bias on-the-fly, with a given PACE. It does so by reweighting via kernel density estimation the unbiased distribution in the CV space, $P(\mathbf{s})$. A compression algorithm is used to prevent the number of kernels from growing linearly with the simulation time. The bias at step n is

$$V_n(\mathbf{s}) = (1-1/\gamma)\frac{1}{\beta}\log\left(\frac{P_n(\mathbf{s})}{Z_n}+\epsilon\right).$$

See Ref.[78] for a complete description of the method.
As an intuitive picture, rather than gradually filling the metastable basins, OPES_METAD quickly tries to get a coarse idea of the full free energy surface (FES), and then slowly refines its details. It has a fast initial exploration phase, and then becomes extremely conservative and does not significantly change the shape of the deposited bias any more, reaching a regime of quasi-static bias. For this reason, it is possible to use standard umbrella sampling reweighting (see REWEIGHT_BIAS) to analyse the trajectory. At this link you can find some python scripts that work in a similar way to sum_hills, but the preferred way to obtain a FES with OPES is via reweighting (see OPES_METAD Tutorial: Running and post-processing). The estimated $c(t)$ is printed for reference only, since it should converge to a fixed value as the bias converges. This $c(t)$ should NOT be used for reweighting. Similarly, the Z_n factor is printed only for reference, and it should converge when no new region of the CV-space is explored.
Notice that OPES_METAD is more sensitive to degenerate CVs than METAD. If the employed CVs map different metastable basins onto the same CV-space region, then OPES_METAD will remain stuck rather than completely reshaping the bias. This can be useful to diagnose problems with your collective variable. If it is not possible to improve the set of CVs and remove this degeneracy, then you might instead consider to use OPES_METAD_EXPLORE or METAD. In this way you will be able to obtain an estimate of the FES, but be aware that you most likely will not reach convergence and thus this estimate will be subjected to systematic errors (see e.g. Fig.3 in [129]). On the contrary, if your CVs are not degenerate but only suboptimal, you should converge faster by using OPES_METAD instead of METAD [78].
The parameter BARRIER should be set to be at least equal to the highest free energy barrier you wish to overcome. If it is much lower than that, you will not cross the barrier, if it is much higher, convergence might take a little longer. If the system has a basin that is clearly more stable than the others, it is better to start the simulation from there.
By default, the kernels SIGMA is adaptive, estimated from the fluctuations over ADAPTIVE_SIGMA_STRIDE simulation steps (similar to METAD ADAPTIVE=DIFF, but contrary to that, no artifacts are introduced and the bias will converge to the correct one). However, notice that depending on the system this might not be the optimal choice for SIGMA.
You can target a uniform flat distribution by explicitly setting BIASFACTOR=inf. However, this should be useful only in very specific cases.
It is possible to take into account also of other bias potentials besides the one of OPES_METAD during the internal reweighting for $P(\mathbf{s})$ estimation. To do so, one has to add those biases with the EXTRA_BIAS keyword, as in the example below. This allows one to define a custom target distribution by adding another bias potential equal to the desired target free energy and setting BIASFACTOR=inf (see example below). Another possible usage of EXT↩RA_BIAS is to make sure that OPES_METAD does not push against another fixed bias added to restrain the CVs range (e.g. UPPER_WALLS).
Through the EXCLUDED_REGION keywork, it is possible to specify a region of CV space where no kernels will be deposited. This can be useful for example for making sure the bias does not modify the transition region, thus allowing for rate calculation. See below for an example of how to use this keyword.
Restart can be done from a KERNELS file, but it might be not perfect (due to limited precision when printing kernels to file, or if adaptive SIGMA is used). For an exact restart you must use STATE_RFILE to read a checkpoint with all the needed info. To save such checkpoints, define a STATE_WFILE and choose how often to print them with ST↩ATE_WSTRIDE. By default this file is overwritten, but you can instead append to it using the flag STORE_STATES.

Multiple walkers are supported only with MPI communication, via the keyword WALKERS_MPI.

Examples

Several examples can be found on the PLUMED-NEST website, by searching for the OPES keyword. The OPES_METAD Tutorial: Running and post-processing can also be useful to get started with the method.
The following is a minimal working example:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_METAD.tmp
cv: DISTANCE ATOMS=1,2
opes: OPES_METAD ARG=cv PACE=200 BARRIER=40
PRINT STRIDE=200 FILE=COLVAR ARG=*
```

Another more articulated one:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_METAD.tmp
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
opes: OPES_METAD ...
  FILE=Kernels.data
  TEMP=300
  ARG=phi,psi
  PACE=500
  BARRIER=50
  SIGMA=0.15,0.15
  SIGMA_MIN=0.01,0.01
  STATE_RFILE=Restart.data
  STATE_WFILE=State.data
  STATE_WSTRIDE=500*100
  STORE_STATES
  WALKERS_MPI
  NLIST
...
PRINT FMT=%g STRIDE=500 FILE=Colvar.data ARG=phi,psi,opes.*
```

Next is an example of how to define a custom target distribution different from the well-tempered one. Here we chose to focus more on the transition state, that is around $\phi = 0$. Our target distribution is a Gaussian centered there, thus the target free energy we want to sample is a parabola, $F^{tg}(\mathbf{s}) = -\frac{1}{\beta}\log[p^{tg}(\mathbf{s})]$.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_METAD.tmp
phi: TORSION ATOMS=5,7,9,15
FtgValue: CUSTOM ARG=phi PERIODIC=NO FUNC=(x/0.4)^2
Ftg: BIASVALUE ARG=FtgValue
opes: OPES_METAD ...
  ARG=phi
  PACE=500
  BARRIER=50
  SIGMA=0.2
  BIASFACTOR=inf
  EXTRA_BIAS=Ftg.bias
...
PRINT FMT=%g STRIDE=500 FILE=COLVAR ARG=phi,Ftg.bias,opes.bias
```

Notice that in order to reweight for the unbiased $P(\mathbf{s})$ during postprocessing, the total bias `Ftg.bias+opes.bias` must be used.
Finally, an example of how to use the EXCLUDED_REGION keyword. It expects a characteristic function that is different from zero in the region to be excluded. You can use CUSTOM and a combination of the step function to define it. With the following input no kernel is deposited in the transition state region of alanine dipeptide, defined by the interval $\phi \in [-0.6, 0.7]$:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_METAD.tmp
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
xx: CUSTOM PERIODIC=NO ARG=phi FUNC=step(x+0.6)-step(x-0.7)
opes: OPES_METAD ...
  ARG=phi,psi
  PACE=500
  BARRIER=30
```

```
  EXCLUDED_REGION=xx
  NLIST
...
PRINT FMT=%g STRIDE=500 FILE=COLVAR ARG=phi,psi,xx,opes.*
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
rct	estimate of c(t). $\frac{1}{\beta}\log\langle e^{\beta V}\rangle$, should become flat as the simulation converges. Do NOT use for reweighting
zed	estimate of Z_n. should become flat once no new CV-space region is explored
neff	effective sample size
nker	total number of compressed kernels used to represent the bias

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
work	**CALC_WORK**	total accumulated work done by the bias
nlker	**NLIST**	number of kernels in the neighbor list
nlsteps	**NLIST**	number of steps from last neighbor list update

Compulsory keywords

TEMP	(default=-1) temperature. If not set, it is taken from MD engine, but not all MD codes provide it
PACE	the frequency for kernel deposition
SIGMA	(default=ADAPTIVE) the initial widths of the kernels. If not set, an adaptive sigma will be used with the given ADAPTIVE_SIGMA_STRIDE
BARRIER	the free energy barrier to be overcome. It is used to set BIASFACTOR, E↩ PSILON, and KERNEL_CUTOFF to reasonable values
COMPRESSION_THRESHOLD	(default=1) merge kernels if closer than this threshold, in units of sigma
FILE	(default=KERNELS) a file in which the list of all deposited kernels is stored

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NLIST	(default=off) use neighbor list for kernels summation, faster but experimental
NLIST_PACE_RESET	(default=off) force the reset of the neighbor list at each PACE. Can be useful with WALKERS_MPI

FIXED_SIGMA	(default=off) do not decrease sigma as the simulation proceeds. Can be added in a RESTART, to keep in check the number of compressed kernels
RECURSIVE_MERGE_OFF	(default=off) do not recursively attempt kernel merging when a new one is added
NO_ZED	(default=off) do not normalize over the explored CV space, Z_n=1
STORE_STATES	(default=off) append to STATE_WFILE instead of ovewriting it each time
CALC_WORK	(default=off) calculate the total accumulated work done by the bias since last restart
WALKERS_MPI	(default=off) switch on MPI version of multiple walkers
SERIAL	(default=off) perform calculations in serial
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
ADAPTIVE_SIGMA_STRIDE	number of steps for measuring adaptive sigma. Default is 10xPACE
SIGMA_MIN	never reduce SIGMA below this value
BIASFACTOR	the gamma bias factor used for the well-tempered target distribution. Set to 'inf' for uniform flat target
EPSILON	the value of the regularization constant for the probability
KERNEL_CUTOFF	truncate kernels at this distance, in units of sigma
NLIST_PARAMETERS	(default=3.0,0.5) the two cutoff parameters for the kernels neighbor list
FMT	specify format for KERNELS file
STATE_RFILE	read from this file the compressed kernels and all the info needed to RESTART the simulation
STATE_WFILE	write to this file the compressed kernels and all the info needed to RESTART the simulation
STATE_WSTRIDE	number of MD steps between writing the STATE_WFILE. Default is only on CPT events (but not all MD codes set them)
EXCLUDED_REGION	kernels are not deposited when the action provided here has a nonzero value, see example above
EXTRA_BIAS	consider also these other bias potentials for the internal reweighting. This can be used e.g. for sampling a custom target distribution (see example above)
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

8.10.5.3 OPES_METAD_EXPLORE

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

On-the-fly probability enhanced sampling with well-tempered target distribution in exploreation mode.
On-the-fly probability enhanced sampling with well-tempered target distribution (OPES) with well-tempered target distribution, exploration mode [128] .
This OPES_METAD_EXPLORE action samples the well-tempered target distribution, that is defined via its marginal $p^{WT}(\mathbf{s}) \propto [P(\mathbf{s})]^{1/\gamma}$ over some collective variables (CVs), $\mathbf{s} = \mathbf{s}(\mathbf{x})$. While OPES_METAD does so by estimating the unbiased distribution $P(\mathbf{s})$, OPES_METAD_EXPLORE instead estimates on-the-fly the target $p^{WT}(\mathbf{s})$ and uses it to define the bias. The bias at step n is

$$V_n(\mathbf{s}) = (\gamma - 1)\frac{1}{\beta}\log\left(\frac{p_n^{WT}(\mathbf{s})}{Z_n} + \epsilon\right) .$$

See Ref.[128] for a complete description of the method.
Intuitively, while OPES_METAD aims at quickly converging the reweighted free energy, OPES_METAD_EXPLORE aims at quickly sampling the target well-tempered distribution. Given enough simulation time, both will converge to the same bias potential but they do so in a qualitatively different way. Compared to OPES_METAD, OPES_METAD_EXPLORE is more similar to METAD, because it allows the bias to vary significantly, thus enhancing exploration. This goes at the expenses of a typically slower convergence of the reweight estimate. OPES_METAD_EXPLORE can be useful e.g.∼for simulating a new system with an unknown BARRIER, or for quickly testing the effectiveness of a new CV that might be degenerate.
Similarly to OPES_METAD, also OPES_METAD_EXPLORE uses a kernel density estimation with the same on-the-fly compression algorithm. The only difference is that the kernels are not weighted, since it estimates the sampled distribution and not the reweighted unbiased one.
All the options of OPES_METAD are also available in OPES_METAD_EXPLORE, except for those that modify the target distribution, since only a well-tempered target is allowed in this case.

Examples

The following is a minimal working example:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/OPES_METAD_EXPLORE.tmp
cv: DISTANCE ATOMS=1,2
opes: OPES_METAD_EXPLORE ARG=cv PACE=500 BARRIER=40
PRINT STRIDE=100 FILE=COLVAR ARG=cv,opes.*
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential
rct	estimate of c(t). $\frac{1}{\beta}\log\langle e^{\beta V}\rangle$, should become flat as the simulation converges. Do NOT use for reweighting
zed	estimate of Z_n. should become flat once no new CV-space region is explored
neff	effective sample size
nker	total number of compressed kernels used to represent the bias

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
work	**CALC_WORK**	total accumulated work done by the bias

nlker	**NLIST**	number of kernels in the neighbor list
nlsteps	**NLIST**	number of steps from last neighbor list update

Compulsory keywords

TEMP	(default=-1) temperature. If not set, it is taken from MD engine, but not all MD codes provide it
PACE	the frequency for kernel deposition
SIGMA	(default=ADAPTIVE) the initial widths of the kernels, divided by the square root of gamma. If not set, an adaptive sigma will be used with the given ADAPTIVE_SIGMA_STRIDE
BARRIER	the free energy barrier to be overcome. It is used to set BIASFACTOR, E↩ PSILON, and KERNEL_CUTOFF to reasonable values
COMPRESSION_THRESHOLD	(default=1) merge kernels if closer than this threshold, in units of sigma
FILE	(default=KERNELS) a file in which the list of all deposited kernels is stored

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NLIST	(default=off) use neighbor list for kernels summation, faster but experimental
NLIST_PACE_RESET	(default=off) force the reset of the neighbor list at each PACE. Can be useful with WALKERS_MPI
FIXED_SIGMA	(default=off) do not decrease sigma as the simulation proceeds. Can be added in a RESTART, to keep in check the number of compressed kernels
RECURSIVE_MERGE_OFF	(default=off) do not recursively attempt kernel merging when a new one is added
NO_ZED	(default=off) do not normalize over the explored CV space, Z_n=1
STORE_STATES	(default=off) append to STATE_WFILE instead of ovewriting it each time
CALC_WORK	(default=off) calculate the total accumulated work done by the bias since last restart
WALKERS_MPI	(default=off) switch on MPI version of multiple walkers
SERIAL	(default=off) perform calculations in serial
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
ADAPTIVE_SIGMA_STRIDE	number of steps for measuring adaptive sigma. Default is 10xPACE
SIGMA_MIN	never reduce SIGMA below this value

BIASFACTOR	the gamma bias factor used for the well-tempered target distribution. Cannot be 'inf'
EPSILON	the value of the regularization constant for the probability
KERNEL_CUTOFF	truncate kernels at this distance, in units of sigma
NLIST_PARAMETERS	(default=3.0,0.5) the two cutoff parameters for the kernels neighbor list
FMT	specify format for KERNELS file
STATE_RFILE	read from this file the compressed kernels and all the info needed to RESTART the simulation
STATE_WFILE	write to this file the compressed kernels and all the info needed to RESTART the simulation
STATE_WSTRIDE	number of MD steps between writing the STATE_WFILE. Default is only on CPT events (but not all MD codes set them)
EXCLUDED_REGION	kernels are not deposited when the action provided here has a nonzero value, see example above
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

8.10.6 Expansion Collective Variables

Expansion collective variables (ECVs) are used to define the expanded ensemble to be sampled by OPES_EXPANDED. See Ref.[79] for details. ECVs take as argument some underlying colvar and have as output components the same colvars.
The following list contains the expansion CVs available in the OPES module.

ECV_CUSTOM	Use some given CVs as a set of expansion collective variables (ECVs).
ECV_LINEAR	Linear expansion, according to a parameter lambda.
ECV_MULTITHERMAL	Expand a simulation to sample multiple temperatures simultaneously.
ECV_MULTITHERMAL_MULTIBARIC	Expand a simulation to sample multiple temperatures and pressures.
ECV_UMBRELLAS_FILE	Target a multiumbrella ensemble, by combining systems each with a parabolic bias potential at a different location.
ECV_UMBRELLAS_LINE	Target a multiumbrella ensemble, by combining systems each with a parabolic bias potential at a different location.

8.10.6.1 ECV_CUSTOM

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Use some given CVs as a set of expansion collective variables (ECVs).
This can be useful e.g. for quickly testing new ECVs, but from a performance point of view it is probably better to implement a new ECV class.
By default the ARGs are expeted to be energies, ΔU_i, and are then multiplied by the inverse temperature β

$$\Delta u_i = \beta \Delta U_i \, .$$

Use the DIMENSIONLESS flag to avoid this multiplication.
The flag ADD_P0 adds also the unbiased distribution to the target. It is possible to specify a BARRIER as in ECV_UMBRELLAS_LINE, to avoid a too high initial bias.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_CUSTOM.tmp
ene: ENERGY
t1: CUSTOM PERIODIC=NO ARG=ene FUNC=(300/500-1)*x
t2: CUSTOM PERIODIC=NO ARG=ene FUNC=(300/1000-1)*x
ecv: ECV_CUSTOM ARG=t1,t2 TEMP=300 ADD_P0
opes: OPES_EXPANDED ARG=ecv.* PACE=500
```

It is equivalent to the following:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_CUSTOM.tmp
ene: ENERGY
ecv: ECV_MULTITHERMAL ARG=ene TEMP=300 TEMP_SET_ALL=300,500,1000
opes: OPES_EXPANDED ARG=ecv.* PACE=500
```

Glossary of keywords and components

Compulsory keywords

TEMP	(default=-1) temperature. If not specified tries to get it from MD engine
ARG	the labels of the single ECVs. ΔU_i, in energy units

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ADD_P0	(default=off) add the unbiased Boltzmann distribution to the target distribution, to make sure to sample it
DIMENSIONLESS	(default=off) consider ARG as dimensionless rather than an energy, thus do not multiply it by β
BARRIER	a guess of the free energy barrier to be overcome (better to stay higher than lower)

8.10.6.2 ECV_LINEAR

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Linear expansion, according to a parameter lambda.
This can be used e.g. for thermodynamic integration, or for multibaric simulations, in which case lambda=pressure. It can also be used for multithermal simulations, but for simplicity it is more convenient to use ECV_MULTITHERMAL.
The difference in Hamiltonian ΔU is expected as ARG.

$$\Delta u_\lambda = \beta\lambda\Delta U \, .$$

Use the DIMENSIONLESS flag to avoid multiplying for the inverse temperature β.
By defauly the needed steps in lambda are automatically guessed from few initial unbiased MD steps, as descibed in [79]. Otherwise one can set this number with LAMBDA_STEPS. In both cases the steps will be uniformly distriuted. Finally, one can use instead the keyword LAMBDA_SET_ALL and explicitly provide each lambda value.

Examples

Typical multibaric simulation:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_LINEAR.tmp
vol: VOLUME
ecv: ECV_LINEAR ...
  ARG=vol
  TEMP=300
  LAMBDA=0.06022140857*2000 #2 kbar
  LAMBDA_MIN=0.06022140857  #1 bar
  LAMBDA_MAX=0.06022140857*4000 #4 kbar
...
opes: OPES_EXPANDED ARG=ecv.vol PACE=500
```

Typical thermodynamic integration:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_LINEAR.tmp
DeltaU: EXTRACV NAME=energy_difference
ecv: ECV_LINEAR ARG=DeltaU TEMP=300
opes: OPES_EXPANDED ARG=ecv.* PACE=100
```

Notice that by defauly LAMBDA=0, LAMBDA_MIN=0 and LAMBDA_MAX=1, which is the typical case for thermodynamic integration.

Glossary of keywords and components

Compulsory keywords

TEMP	(default=-1) temperature. If not specified tries to get it from MD engine
ARG	the label of the Hamiltonian difference. ΔU
LAMBDA	(default=0) the lambda at which the underlying simulation runs

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
DIMENSIONLESS	(default=off) ARG is considered dimensionless rather than an energy, thus is not multiplied by β
GEOM_SPACING	(default=off) use geometrical spacing in lambda instead of linear spacing
LAMBDA_MIN	(default=0) the minimum of the lambda range
LAMBDA_MAX	(default=1) the maximum of the lambda range
LAMBDA_STEPS	uniformly place the lambda values, for a total of LAMBDA_STEPS
LAMBDA_SET_ALL	manually set all the lamdbas

8.10.6.3 ECV_MULTITHERMAL

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Expand a simulation to sample multiple temperatures simultaneously.
The internal energy U of of the system should be used as ARG.

$$\Delta u_{\beta'} = (\beta' - \beta)U \, ,$$

where β' are the temperatures to be sampled and β is the temperature at which the simulation is conducted. In case of fixed volume, the internal energy is simply the potential energy given by the ENERGY colvar $U = E$, and you will run a multicanonical simulation. If instead the simulation is at fixed pressure p, the contribution of the volume must be added $U = E + pV$ (see example below).
By defauly the needed steps in temperatures are automatically guessed from few initial unbiased MD steps, as descibed in [79]. Otherwise you can manually set this number with TEMP_STEPS. In both cases the steps will be geometrically spaced in temperature. Use instead the keyword NO_GEOM_SPACING for a linear spacing in the inverse temperature (beta), that typically increases the focus on lower temperatures. Finally, you can use instead the keyword TEMP_SET_ALL and explicitly provide each temperature.
You can reweight the resulting simulation at any temperature in the chosen range, using e.g. REWEIGHT_TEMP_PRESS.
A similar target distribution can be sampled using TD_MULTICANONICAL.

Examples

Fixed volume, multicanonical simulation:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_MULTITHERMAL.tmp
ene: ENERGY
ecv: ECV_MULTITHERMAL ARG=ene TEMP=300 TEMP_MIN=300 TEMP_MAX=800
opes: OPES_EXPANDED ARG=ecv.ene PACE=500
```

which, if your MD code passes the temperature to PLUMED, is equivalent to:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_MULTITHERMAL.tmp
ene: ENERGY
ecv: ECV_MULTITHERMAL ARG=ene TEMP_MAX=800
opes: OPES_EXPANDED ARG=ecv.ene PACE=500
```

If instead the pressure is fixed and the volume changes, you shuld calculate the internal energy first, $U = E + pV$

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_MULTITHERMAL.tmp
ene: ENERGY
vol: VOLUME
intEne: CUSTOM PERIODIC=NO ARG=ene,vol FUNC=x+0.06022140857*y
ecv: ECV_MULTITHERMAL ARG=intEne TEMP_MAX=800
opes: OPES_EXPANDED ARG=ecv.intEne PACE=500
```

Notice that $p = 0.06022140857$ corresponds to 1 bar when using the default PLUMED units.

Glossary of keywords and components

Compulsory keywords

TEMP	(default=-1) temperature. If not specified tries to get it from MD engine
ARG	the label of the internal energy of the system. If volume is fixed it is calculated by the ENERGY colvar

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically

NO_GEOM_SPACING	(default=off) do not use geometrical spacing in temperature, but instead linear spacing in inverse temperature
TEMP_MIN	the minimum of the temperature range
TEMP_MAX	the maximum of the temperature range
TEMP_STEPS	the number of steps in temperature
TEMP_SET_ALL	manually set all the temperatures

8.10.6.4 ECV_MULTITHERMAL_MULTIBARIC

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Expand a simulation to sample multiple temperatures and pressures.
The potential ENERGY, E, and the VOLUME, V, of the system should be used as ARG.

$$\Delta u_{\beta',p'} = (\beta' - \beta)E + (\beta' p' - \beta p)V \, ,$$

where β', p' are the temperatures and pressures to be sampled, while β, p is the temperature and pressure at which the simulation is conducted.
If instead you wish to sample multiple temperatures and a single pressure, you should use ECV_MULTITHERMAL with as ARG the internal energy $U = E + pV$.
The TEMP_STEPS and PRESSURE_STEPS are automatically guessed from the initial unbiased steps (see OBSERVATION_STEPS in OPES_EXPANDED), unless explicitly set. The algorithm for this guess is described in [79] should provide a rough estimate useful for most applications. The pressures are uniformely spaced, while the temperatures steps are geometrically spaced. Use instead the keyword NO_GEOM_SPACING for a linear spacing in inverse temperature (beta). For more detailed control you can use instead TEMP_SET_ALL and/or PRESSURE_SET_ALL to explicitly set all of them. The temperatures and pressures are then combined in a 2D grid.
You can use CUT_CORNER to avoid a high-temperature/low-pressure region. This can be useful e.g. to increase the temperature for greater ergodicity, while avoiding water to vaporize, as in Ref.[79].
You can reweight the resulting simulation at any temperature and pressure in chosen target, using e.g. REWEIGHT_TEMP_PRESS. A similar target distribution can be sampled using TD_MULTITHERMAL_MULTIBARIC.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_MULTITHERMAL_MULTIBARIC.tmp
ene: ENERGY
vol: VOLUME
ecv: ECV_MULTITHERMAL_MULTIBARIC ...
  ARG=ene,vol
  TEMP=500
  TEMP_MIN=270
  TEMP_MAX=800
  PRESSURE=0.06022140857*2000 #2 kbar
  PRESSURE_MIN=0.06022140857  #1 bar
  PRESSURE_MAX=0.06022140857*4000 #4 kbar
  CUT_CORNER=500,0.06022140857,800,0.06022140857*1000
...
opes: OPES_EXPANDED ARG=ecv.* FILE=DeltaF.data PACE=500 WALKERS_MPI
```

Notice that $p = 0.06022140857$ corresponds to 1 bar only when using the default PLUMED units. If you modify them via the UNITS command, then the pressure has to be rescaled accordingly.

Glossary of keywords and components

Compulsory keywords

TEMP	(default=-1) temperature. If not specified tries to get it from MD engine
ARG	the labels of the potential energy and of the volume of the system. You can calculate them with ENERGY and VOLUME respectively
PRESSURE	pressure. Use the proper units

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NO_GEOM_SPACING	(default=off) do not use geometrical spacing in temperature, but instead linear spacing in inverse temperature
TEMP_MIN	the minimum of the temperature range
TEMP_MAX	the maximum of the temperature range
TEMP_STEPS	the number of steps in temperature
TEMP_SET_ALL	manually set all the temperatures
PRESSURE_MIN	the minimum of the pressure range
PRESSURE_MAX	the maximum of the pressure range
PRESSURE_STEPS	the number of steps in pressure
PRESSURE_SET_ALL	manually set all the pressures
SET_ALL_TEMP_PRESSURE	manually set all the target temperature_pressure pairs. An underscore separates temperature and pressure, while different points are comma-separated, e.g.: temp1_pres1,temp1_pres2,...
CUT_CORNER	avoid region of high temperature and low pressure. Exclude all points below a line in the temperature-pressure plane, defined by two points↩ : $T_{low}, P_{low}, T_{high}, P_{high}$

8.10.6.5 ECV_UMBRELLAS_FILE

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target a multiumbrella ensemble, by combining systems each with a parabolic bias potential at a different location. Any set of collective variables s can be used as ARG. The positions s_i and dimension σ_i of the umbrellas are read from file.

$$\Delta u_{\mathbf{s}_i}(\mathbf{s}) = \sum_j^{dim} \frac{([s]_j - [s_i]_j)^2}{2[\sigma_i]_j^2} \, .$$

Notice that σ_i is diagonal, thus only one SIGMA per CV has to be specified for each umbrella. You can choose the umbrellas manually, or place them on a grid, or along a path, similar to PATH. They must cover all the CV space that one wishes to sample.
The first column of the umbrellas file is always ignored and must be called "time". You can also use as input file a STATE file from an earlier OPES_METAD run (or an OPES_MEAD_EXPLORE run, if you combine it with other ECVs).
Similarly to ECV_UMBRELLAS_LINE, you should set the flag ADD_P0 if you think your umbrellas might not properly cover all the CV region relevant for the unbiased distribution. You can also use BARRIER to set the maximum barrier height to be explored, and avoid huge biases at the beginning of your simulation. See also Appendix B of Ref.[79] for more details on these last two options.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_UMBRELLAS_FILE.tmp
cv1: DISTANCE ATOMS=1,2
cv2: DISTANCE ATOMS=3,4
cv3: DISTANCE ATOMS=4,1
ecv: ECV_UMBRELLAS_FILE ARG=cv1,cv2,cv3 FILE=Umbrellas.data ADD_P0 BARRIER=70
opes: OPES_EXPANDED ARG=ecv.* PACE=500
PRINT FILE=COLVAR STRIDE=500 ARG=cv1,cv2,cv3,opes.bias
```

The umbrellas file might look like this:

```
#! FIELDS time cv1 cv2 cv3 sigma_cv1 sigma_cv2 sigma_cv3
1  1.17958  2.93697  1.06109  0.19707  0.28275  0.32427
2  2.04023  2.69714  1.84770  0.22307  0.25933  0.31783
3  1.99693  1.10299  1.13351  0.19517  0.26260  0.37427
4  1.15954  1.37447  2.25975  0.20096  0.27168  0.33353
5  1.10126  2.45936  2.40260  0.19747  0.24215  0.35523
```

Glossary of keywords and components

Compulsory keywords

TEMP	(default=-1) temperature. If not specified tries to get it from MD engine
FILE	the name of the file containing the umbrellas

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ADD_P0	(default=off) add the unbiased Boltzmann distribution to the target distribution, to make sure to sample it
LOWER_HALF_ONLY	(default=off) use only the lower half of each umbrella potentials
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
BARRIER	a guess of the free energy barrier to be overcome (better to stay higher than lower)

8.10.6.6 ECV_UMBRELLAS_LINE

	This is part of the opes module
	It is only available if you configure PLUMED with ./configure –enable-modules=opes . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Target a multiumbrella ensemble, by combining systems each with a parabolic bias potential at a different location. Any set of collective variables s can be used as ARG.

$$\Delta u_{\mathbf{s}_i}(\mathbf{s}) = \sum_j^{dim} \frac{([s]_j - [s_i]_j)^2}{2\sigma^2} .$$

The Gaussian umbrellas are placed along a line, from CV_MIN to CV_MAX. The umbrellas are placed at a distance SIGMA∗SPACING from each other, if you need more flexibility use ECV_UMBRELLAS_FILE. The unbiased fluctuations in the basin usually are a reasonable guess for the value of SIGMA. Typically, a SPACING greater than 1 can lead to faster convergence, by reducing the total number of umbrellas. The umbrellas can be multidimensional, but the CVs dimensions should be rescaled since a single SIGMA must be used.
The keyword BARRIER can be helpful to avoid breaking your system due to a too strong initial bias. If you think the placed umbrellas will not cover the whole unbiased probability distribution you should add it explicitly to the target, with the flag ADD_P0, for more robust convergence. See also Appendix B of Ref.[79] for more details on these last two options.
The flag LOWER_HALF_ONLY modifies the ECVs so that they are set to zero when $\mathrm{s} > \mathrm{s}_i$, as in LOWER_WALLS. This can be useful e.g. when the CV used is the ENERGY and one wants to sample a broad range of high energy values, similar to ECV_MULTITHERMAL but with a flat energy distribution as target. By pushing only from below one can avoid too extreme forces that could crash the simulation.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_UMBRELLAS_LINE.tmp
cv: DISTANCE ATOMS=1,2
ecv: ECV_UMBRELLAS_LINE ARG=cv CV_MIN=1.2 CV_MAX=4.3 SIGMA=0.5 SPACING=1.5
opes: OPES_EXPANDED ARG=ecv.* PACE=500
```

It is also possible to combine different ECV_UMBRELLAS_LINE to build a grid of CV values that will be sampled. For example the following code will sample a whole 2D region of cv1 and cv2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ECV_UMBRELLAS_LINE.tmp
cv1: DISTANCE ATOMS=1,2
ecv2: ECV_UMBRELLAS_LINE ARG=cv1 CV_MIN=1.2 CV_MAX=4.3 SIGMA=0.5

cv2: DISTANCE ATOMS=3,4
ecv1: ECV_UMBRELLAS_LINE ARG=cv2 CV_MIN=13.8 CV_MAX=21.4 SIGMA=4.3

opes: OPES_EXPANDED ARG=ecv1.*,ecv2.* PACE=500
```

Glossary of keywords and components

Compulsory keywords

TEMP	(default=-1) temperature. If not specified tries to get it from MD engine
CV_MIN	the minimum of the CV range to be explored
CV_MAX	the maximum of the CV range to be explored
SIGMA	sigma of the umbrella Gaussians
SPACING	(default=1) the distance between umbrellas, in units of SIGMA

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ADD_P0	(default=off) add the unbiased Boltzmann distribution to the target distribution, to make sure to sample it
LOWER_HALF_ONLY	(default=off) use only the lower half of each umbrella potentials
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
BARRIER	a guess of the free energy barrier to be overcome (better to stay higher than lower)

8.10.7 Tutorials

The following list contains the tutorials available for the OPES module. Other examples of how to use OPES can be found in the `PLUMED-NEST`.

PLUMED Masterclass 22.3 OPES method	Link to the PLUMED Masterclass about the OPES method.
OPES_METAD Tutorial: Running and post-processing	Simple example of how to run and post-process an OPES_METAD simulation.

8.10.7.1 PLUMED Masterclass 22.3 OPES method

8.10.7.1.1 OPES Masterclass The PLUMED Masterclass 22.3 provides a good overview of the OPES method, it can be found here: PLUMED Masterclass 22.3: OPES method.

8.10.7.2 OPES_METAD Tutorial: Running and post-processing

8.10.7.2.1 Aim The aim of this tutorial is to briefly present an example of an OPES_METAD simulation, together with some useful post-processing tools for obtaining an estimate of the free energy surface. This tutorial does not contain theory explanations. The OPES method is presented in detail in Ref. [78] and in its supporting information.

8.10.7.2.2 Resources The `TARBALL` for this project contains the following:

- input files for running a simple double-well model with pesmd
- plumed files for OPES_METAD and METAD simulations
- `FES_from_Reweighting.py`: script for estimating free energy surface via reweighting
- `FES_from_State.py`: analogous to sum_hills, but for OPES simulations
- `State_from_Kernels.py`: script for obtaining an OPES state file from a kernels file

8.10.7.2.3 Simulating and reweighting We will perform Langevin dynamics on the following two-dimensional model potential:
To run such simulations, we use the pesmd tool and define the potential directly inside the plumed input file.
We choose as collective variable (CV) the x coordinate only, in order to mimic a realistic situation in which some of the slow modes of the system are unknown and not accelerated by the biasing. The following is a typical plumed input for performing an OPES_METAD simulation:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/opes-metad.txt
UNITS NATURAL
p: DISTANCE ATOMS=1,2 COMPONENTS
opes: OPES_METAD ARG=p.x TEMP=1 PACE=500 BARRIER=10
PRINT FILE=COLVAR STRIDE=500 ARG=p.x,p.y,opes.bias
```

By default an OPES_METAD simulation generates a KERNELS file that, similarly to the HILLS file from METAD, contains all the Gaussians deposited during the run, and can be used for restarting a simulation. We also print a COLVAR file containing the system coordinates and the instantaneous value of the bias. Here is the time evolution of the x coordinate.
From this simulation we can estimate via reweighting the free energy surface (FES) along x. We use a weighted kernel density estimation with bandwidth `sigma`, and use block average for uncertainty estimation. You should run the following script from the same folder where the COLVAR file is stored.

```
../FES_from_Reweighting.py --kt 1 --sigma 0.03 --blocks 5
```

As value for `sigma` we used the one from the last line of the KERNELS file. Notice that a similar FES estimate could have been obtained using REWEIGHT_BIAS instead of the python script. For this simple system we can compare the result with the true FES (black dashed line):

Note

> If the COLVAR file is not available one can use instead the KERNELS file, choosing the `logweight` column as bias

We now run a well-tempered metadynamics simulation on the same system, to get an idea of the differences between the two methods. This is the plumed input used:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/opes-metad.txt
UNITS NATURAL
p: DISTANCE ATOMS=1,2 COMPONENTS
metad: METAD ...
  ARG=p.x
  TEMP=1
  PACE=500
  HEIGHT=1
  SIGMA=0.185815
  BIASFACTOR=10
  GRID_MIN=-3.5
  GRID_MAX=3.5
  GRID_BIN=200
  CALC_RCT
...
PRINT FILE=COLVAR STRIDE=500 ARG=p.x,p.y,metad.rbias,metad.rct
```

And here is the resulting x trajectory.
One can see that with metadynamics more transitions are present, especially in the initial part of the simulation. This is because the bias is changing quickly and can efficiently push the system through high free-energy pathways. Unfortunately, these out-of-equilibrium transitions cannot be efficiently reweighted and might lead to a systematic error.
To avoid this systematic error, one must remove the part of the metadynamics simulation where the bias is not quasi-static. You can do so using the option `--skiprows` in the `FES_from_Reweighting.py` script. Removing an initial transient might be useful also for OPES simulations, in particular when the chosen `BARRIER` is much higher than the real one.

Note

> Several different reweighting schemes have been proposed for metadynamics. The one we report here is among the most commonly used, but it is not guaranteed to be the most efficient.

8.10.7.2.4 Free energy estimate from the bias A quick way to get a FES estimate along the biased CVs in OPES, is to directly look at the instantaneous bias $V_t(x)$, since at convergence $F(x) = -(1-1/\gamma)^{-1}V(x)$. For metadynamics simulations one should use the `rbias` instead, which is the bias shifted by the $c(t)$ factor. In Fig. 6 we plot the COLVAR file using on the x-axis the CV x and on the y-axis this instantaneous FES estimate, $F_t(x)$. Points are colored according to the simulation time.
We can see that the running estimate from metadynamics has much broader oscillations. This can be seen also in Fig. 3 of Ref. [78], where only the free energy difference between the basins is plotted. It is also clearly visible that the OPES simulation does not explore free energy regions much higher than the given `BARRIER` parameter.
Another more common way of obtaining this direct FES estimate from the metadynamics bias is via the sum_hills tool, e.g. with the following command:

```
plumed sum_hills --hills HILLS --mintozero --stride 1000
```

An analogous procedure can be done also with OPES, starting from the STATE file printed during the simulation. This file contains the compressed kernels that are used internally to define the bias, together with all the information needed for a smooth restart. The syntax to print it is similar to the one used to print the bias grid in METAD :

```
BEGIN_PLUMED_FILE working DATADIR=example-check/opes-metad.txt
UNITS NATURAL
p: DISTANCE ATOMS=1,2 COMPONENTS
opes: OPES_METAD ...
  ARG=p.x
  TEMP=1
  PACE=500
  BARRIER=10
  STATE_WFILE=STATE
  STATE_WSTRIDE=500*1000
  STORE_STATES
...
```

If you did not dump the STATE file while running your simulation, you can recover it with the driver (see READ). The script `State_from_Kernels.py` contains the minimal input to do so, and only requires the KERNELS file.
If the STATE file is available, one can run the provided script with the following command:

```
../FES_from_State.py --kt 1 --all_stored
```

For OPES_METAD the resulting FES estimate will be very similar to the one obtained from reweighting.
In this tutorial a single simulation is presented for OPES and metadynamics, but you can check Fig. S2 and S3 in the supporting information of Ref. [78] to see the results obtained by averaging 100 independent realizations.

8.10.7.2.5 Some other considerations The OPES_METAD has a few quantities one can print to monitor the simulation:

- `rct`, contrary to the one of METAD, converges to a constant and can be useful to quickly check the status of the simulation, especially when running with many CVs. It should not be used for reweighting.
- `zed` should change only when a new region of CV-space is sampled.
- `neff` is an estimate of the effective sample size and should always grow.
- `nker` is the number of compressed kernels. If it is equal to the number of deposited kernels, it means that the simulation never comes back to where it has already been (too many CVs? too small `SIGMA`? too small `COMPRESSION_THRESHOLD`?).

Finally, here are some optional keywords that might be useful:

- `NLIST` activates a neighbor list over the kernels. It can considerably speed up the simulation, especially when using two or more CVs.
- `SIGMA_MIN` can be useful to reduce the total number of compressed kernels (`nker`) and thus speed up the simulation. It can help especially with long simulations, and can also be added after a restart (see also `FIXED_SIGMA`).

8.11 PIV collective variable

8.11.1 Overview

To be completed

8.11.2 Installation

This module is not installed by default. Add '--enable-modules=piv' to your './configure' command when building PLUMED to enable these features.

8.11.3 Usage

Currently, all features of the PIV module are included in a single PIV collective variable: PIV

8.11.4 Contents

8.11.5 CVs Documentation

The following list contains descriptions of biases developed for the PLUMED-PIV module. They can be used in combination with other actions outside of the PIV module.

PIV	Calculates the PIV-distance.

8.11.5.1 PIV

	This is part of the piv module
	It is only available if you configure PLUMED with ./configure –enable-modules=piv . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculates the PIV-distance.
PIV distance is the squared Cartesian distance between the PIV [80] [81] associated to the configuration of the system during the dynamics and a reference configuration provided as input (PDB file format). PIV can be used together with FUNCPATHMSD to define a path in the PIV space.

Examples

The following example calculates PIV-distances from three reference configurations in Ref1.pdb, Ref2.pdb and Ref3.pdb and prints the results in a file named colvar. Three atoms (PIVATOMS=3) with names (pdb file) A B and C are used to construct the PIV and all PIV blocks (AA, BB, CC, AB, AC, BC) are considered. SFACT↩OR is a scaling factor that multiplies the contribution to the PIV-distance given by the single PIV block. NLIST sets the use of neighbor lists for calculating atom-atom distances. The SWITCH keyword specifies the parameters of the switching function that transforms atom-atom distances. SORT=1 means that the PIV block elements are sorted (SORT=0 no sorting.) Values for SORT, SFACTOR and the neighbor list parameters have to be specified for each block. The order is the following: AA,BB,CC,AB,AC,BC. If ONLYDIRECT (ONLYCROSS) is used the order is AA,BB,CC (AB,AC,BC). The sorting operation within each PIV block is performed using the counting sort algorithm, PRECISION specifies the size of the counting array.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/PIV.tmp
PIV ...
LABEL=Pivd1
PRECISION=1000
NLIST
REF_FILE=Ref1.pdb
PIVATOMS=3
ATOMTYPES=A,B,C
```

```
SFACTOR=0.3,0.5,1.0,0.2,0.2,0.2
SORT=1,1,1,1,1,1
SWITCH1={RATIONAL R_0=0.6 MM=12 NN=4}
SWITCH2={RATIONAL R_0=0.4 MM=10 NN=5}
SWITCH3={RATIONAL R_0=0.4 MM=10 NN=5}
SWITCH4={RATIONAL R_0=0.5 MM=12 NN=6}
SWITCH5={RATIONAL R_0=0.5 MM=12 NN=6}
SWITCH6={RATIONAL R_0=0.5 MM=12 NN=6}
NL_CUTOFF=0.8,0.6,0.6,0.7,0.7,0.7
NL_STRIDE=10,10,10,10,10,10
NL_SKIN=0.1,0.1,0.1,0.1,0.1,0.1
... PIV
PIV ...
LABEL=Pivd2
PRECISION=1000
NLIST
REF_FILE=Ref2.pdb
PIVATOMS=3
ATOMTYPES=A,B,C
SFACTOR=0.3,0.5,1.0,0.2,0.2,0.2
SORT=1,1,1,1,1,1
SWITCH1={RATIONAL R_0=0.6 MM=12 NN=4}
SWITCH2={RATIONAL R_0=0.4 MM=10 NN=5}
SWITCH3={RATIONAL R_0=0.4 MM=10 NN=5}
SWITCH4={RATIONAL R_0=0.5 MM=12 NN=6}
SWITCH5={RATIONAL R_0=0.5 MM=12 NN=6}
SWITCH6={RATIONAL R_0=0.5 MM=12 NN=6}
NL_CUTOFF=0.8,0.6,0.6,0.7,0.7,0.7
NL_STRIDE=10,10,10,10,10,10
NL_SKIN=0.1,0.1,0.1,0.1,0.1,0.1
... PIV
PIV ...
LABEL=Pivd3
PRECISION=1000
NLIST
REF_FILE=Ref3.pdb
PIVATOMS=3
ATOMTYPES=A,B,C
SFACTOR=0.3,0.5,1.0,0.2,0.2,0.2
SORT=1,1,1,1,1,1
SWITCH1={RATIONAL R_0=0.6 MM=12 NN=4}
SWITCH2={RATIONAL R_0=0.4 MM=10 NN=5}
SWITCH3={RATIONAL R_0=0.4 MM=10 NN=5}
SWITCH4={RATIONAL R_0=0.5 MM=12 NN=6}
SWITCH5={RATIONAL R_0=0.5 MM=12 NN=6}
SWITCH6={RATIONAL R_0=0.5 MM=12 NN=6}
NL_CUTOFF=0.8,0.6,0.6,0.7,0.7,0.7
NL_STRIDE=10,10,10,10,10,10
NL_SKIN=0.1,0.1,0.1,0.1,0.1,0.1
... PIV

PRINT ARG=Pivd1,Pivd2,Pivd3 FILE=colvar
```

WARNING: Both the "CRYST" and "ATOM" lines of the PDB files must conform precisely to the official pdb format, including the width of each alphanumerical field:

```
CRYST1   31.028   36.957   23.143  89.93  92.31  89.99 P 1           1
ATOM      1  OW1 wate    1      15.630  19.750   1.520  1.00  0.00
```

In each pdb frame, atoms must be numbered in the same order and with the same element symbol as in the input of the MD program.
The following example calculates the PIV-distances from two reference configurations Ref1.pdb and Ref2.pdb and uses PIV-distances to define a Path Collective Variable (FUNCPATHMSD) with only two references (Ref1.pdb and Ref2.pdb). With the VOLUME keyword one scales the atom-atom distances by the cubic root of the ratio between the specified value and the box volume of the initial step of the trajectory file.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/PIV.tmp
PIV ...
LABEL=c1
PRECISION=1000
VOLUME=12.15
NLIST
```

```
REF_FILE=Ref1.pdb
PIVATOMS=2
ATOMTYPES=A,B
ONLYDIRECT
SFACTOR=1.0,0.2
SORT=1,1
SWITCH1={RATIONAL R_0=0.6 MM=12 NN=4}
SWITCH2={RATIONAL R_0=0.5 MM=10 NN=5}
NL_CUTOFF=1.2,1.2
NL_STRIDE=10,10
NL_SKIN=0.1,0.1
... PIV
PIV ...
LABEL=c2
PRECISION=1000
VOLUME=12.15
NLIST
REF_FILE=Ref2.pdb
PIVATOMS=2
ATOMTYPES=A,B
ONLYDIRECT
SFACTOR=1.0,0.2
SORT=1,1
SWITCH1={RATIONAL R_0=0.6 MM=12 NN=4}
SWITCH2={RATIONAL R_0=0.4 MM=10 NN=5}
NL_CUTOFF=1.2,1.2
NL_STRIDE=10,10
NL_SKIN=0.1,0.1
... PIV

p1: FUNCPATHMSD ARG=c1,c2 LAMBDA=0.180338
METAD ARG=p1.s,p1.z SIGMA=0.01,0.2 HEIGHT=0.8 PACE=500   LABEL=res
PRINT ARG=c1,c2,p1.s,p1.z,res.bias STRIDE=500  FILE=colvar FMT=%15.6f
```

When using PIV please cite [81] .
(See also PRINT)

Glossary of keywords and components

Compulsory keywords

SWITCH	The switching functions parameter.You should specify a Switching function for all PIV blocks.↩ Details of the various switching functions you can use are provided on switchingfunction.. You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...
PRECISION	the precision for approximating reals with integers in sorting.
REF_FILE	PDB file name that contains the ith reference structure.
PIVATOMS	Number of atoms to use for PIV.
SORT	Whether to sort or not the PIV block.
ATOMTYPES	The atom types to use for PIV.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
TEST	(default=off) Print the actual and reference PIV and exit
COM	(default=off) Use centers of mass of groups of atoms instead of atoms as specified in the Pdb file

ONLYCROSS	(default=off) Use only cross-terms (A-B, A-C, B-C, ...) in PIV
ONLYDIRECT	(default=off) Use only direct-terms (A-A, B-B, C-C, ...) in PIV
DERIVATIVES	(default=off) Activate the calculation of the PIV for every class (needed for numerical derivatives).
NLIST	(default=off) Use a neighbor list for distance calculations.
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
TIMER	(default=off) Perform timing analysis on heavy loops.
SFACTOR	Scale the PIV-distance by such block-specific factor
VOLUME	Scale atom-atom distances by the cubic root of the cell volume. The input volume is used to scale the R_0 value of the switching function.
UPDATEPIV	Frequency (in steps) at which the PIV is updated.
NL_CUTOFF	Neighbor lists cutoff.
NL_STRIDE	Update neighbor lists every NL_STRIDE steps.
NL_SKIN	The maximum atom displacement tolerated for the neighbor lists update.

8.12 S2 contact model collective variable

8.12.1 Overview

S2 contact model CV used in [82], based on NH order parameter from [130] and methyl order parameter from [131].

8.12.2 Installation

This module is not installed by default. Add '--enable-modules=s2cm' to your './configure' command when building PLUMED to enable these features.

8.12.3 Usage

Currently, all features of the S2 contact model module are included in a single S2 contact model collective variable: S2CM

8.12.4 Contents

8.12.5 CVs Documentation

S2CM	S2 contact model CV.

8.12.5.1 S2CM

	This is part of the s2cm module
	It is only available if you configure PLUMED with ./configure –enable-modules=s2cm . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

S2 contact model CV.
This CV was used in [82], based on NH order parameter from [130] and methyl order parameter from [131]. Input parameters can be found in the relevant papers.

Examples

Glossary of keywords and components

The atoms involved can be specified using

METHYL_ATOM	the methyl carbon atom of the residue (i). For more information on how to specify lists of atoms see Groups and Virtual Atoms
NH_ATOMS	the hydrogen atom of the NH group of the residue (i) and carbonyl oxygen of the preceding residue (i-1). For more information on how to specify lists of atoms see Groups and Virtual Atoms
HEAVY_ATOMS	the heavy atoms to be included in the calculation. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

R_EFF	the effective distance, r_eff in the equation, given in nm.
PREFACTOR_A	the prefactor, a in the equation
EXPONENT_B	the exponent, b in the equation
OFFSET_C	the offset, c in the equation
N_I	n_i in the equation

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SERIAL	(default=off) Perform the calculation in serial - for debug purpose
NLIST	(default=off) Use a neighbour list to speed up the calculation
NL_CUTOFF	The cutoff for the neighbour list
NL_STRIDE	The frequency with which we are updating the atoms in the neighbour list
R_SHIFT	shift all distances by given amount

8.13 SASA collective variable

8.13.1 Overview

This SASA module contains methods for the calculation of the solvent accessible surface area (SASA) of proteins using either the fast algorithm by Hasel et al. [83] or the LCPO algorithm [84]. This module can be used to include the SASA as a collective variable in metadynamics simulations, and also for implicit solvent simulations as described in [85], [86] and [87].

8.13.2 Installation

This module is not installed by default. Add '--enable-modules=sasa' to your './configure' command when building PLUMED to enable these features.

8.13.3 Usage

Currently, all features of the SASA module are included in two SASA functions: SASA_HASEL SASA_LCPO

8.13.4 Contents

8.13.5 CVs Documentation

The following list contains descriptions of biases developed for the PLUMED-SASA module. They can be used in combination with other biases outside of the SASA module.

SASA_HASEL	Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.
SASA_LCPO	Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.

8.13.5.1 SASA_HASEL

	This is part of the sasa module
	It is only available if you configure PLUMED with ./configure –enable-modules=sasa . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.
The atoms for which the SASA is desired should be indicated with the keyword ATOMS, and a pdb file of the protein must be provided in input with the MOLINFO keyword. The algorithm described in [83] is used for the calculation. The radius of the solvent is assumed to be 0.14 nm, which is the radius of water molecules. Using the keyword NL_STRIDE it is also possible to specify the frequency with which the neighbor list for the calculation of SASA is updated (the default is every 10 steps).
Different properties can be calculated and selected using the TYPE keyword:
1) the total SASA (TOTAL);
2) the free energy of transfer for the protein according to the transfer model (TRANSFER). This keyword can be used, for instance, to compute the transfer of a protein to different temperatures, as detailed in [86], or to different pressures, as detailed in [87], or to different osmolyte solutions, as detailed in [85].
When the TRANSFER keyword is used, a file with the free energy of transfer values for the sidechains and backbone atoms should be provided (using the keyword DELTAGFILE). Such file should have the following format:

```
----------------Sample DeltaG.dat file---------------------
ALA     0.711019999999962
ARG     -2.24832799999996
ASN     -2.74838799999999
ASP     -2.5626376
CYS     3.89864000000006
GLN     -1.76192
GLU     -2.38664400000002
GLY     0
HIS     -3.58152799999999
ILE     2.42634399999986
LEU     1.77233599999988
LYS     -1.92576400000002
MET     -0.262827999999956
PHE     1.62028800000007
PRO     -2.15598800000001
SER     -1.60934800000004
THR     -0.591559999999987
TRP     1.22936000000027
TYR     0.775547999999958
VAL     2.12779200000011
BACKBONE        1.00066920000002
-----------------------------------------------------------
```

where the second column is the free energy of transfer for each sidechain/backbone, in kJ/mol.

A Python script for the computation of free energy of transfer values to describe the effect of osmolyte concentration, temperature and pressure (according to [85], [86] and [87]) is freely available at https://github.com/andrea-arsiccio/DeltaG-calculation. The script automatically outputs a DeltaG.dat file compatible with this SASA module.
If the DELTAGFILE is not provided, the program computes the free energy of transfer values as if they had to take into account the effect of temperature according to approaches 2 or 3 in the paper [86]. Please read and cite this paper if using the transfer model for computing the effect of temperature in implicit solvent simulations. For this purpose, the keyword APPROACH should be added, and set to either 2 or 3.
The SASA usually makes sense when atoms used for the calculation are all part of the same molecule. When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.2, by considering the ordered list of atoms and rebuilding the broken entities using a procedure that is equivalent to that done in WHOLEMOLECULES. Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.
In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.
The SASA may also be computed using the SASA_LCPO collective variable, which makes use of the LCPO algorithm [84]. SASA_LCPO is more accurate then SASA_HASEL, but the computation is slower.

Examples

The following input tells plumed to print the total SASA for atoms 10 to 20 in a protein chain.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SASA_HASEL.tmp
SASA_HASEL TYPE=TOTAL ATOMS=10-20 NL_STRIDE=10 LABEL=sasa
PRINT ARG=sasa STRIDE=1 FILE=colvar
```

The following input tells plumed to compute the transfer free energy for the protein chain containing atoms 10 to 20. Such transfer free energy is then used as a bias in the simulation (e.g., implicit solvent simulations). The free energy of transfer values are read from a file called DeltaG.dat.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SASA_HASEL.tmp
SASA_HASEL TYPE=TRANSFER ATOMS=10-20 NL_STRIDE=10 DELTAGFILE=DeltaG.dat LABEL=sasa

bias: BIASVALUE ARG=sasa

PRINT ARG=sasa,bias.* STRIDE=1 FILE=colvar
```

The following input tells plumed to compute the transfer free energy for the protein chain containing atoms 10 to 20. Such transfer free energy is then used as a bias in the simulation (e.g., implicit solvent simulations). The free energy of transfer values are computed according to [86], and take into account the effect of temperature using approach 2 as described in the paper.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SASA_HASEL.tmp
SASA_HASEL TYPE=TRANSFER ATOMS=10-20 NL_STRIDE=10 APPROACH=2 LABEL=sasa

bias: BIASVALUE ARG=sasa

PRINT ARG=sasa,bias.* STRIDE=1 FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the group of atoms that you are calculating the SASA for. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

TYPE	(default=TOTAL) The type of calculation you want to perform. Can be TOTAL or TRANSFER
NL_STRIDE	The frequency with which the neighbor list for the calculation of SASA is updated.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
DELTAGFILE	a file containing the free energy of transfer values for backbone and sidechains atoms. Necessary only if TYPE = TRANSFER. A Python script for the computation of free energy of transfer values to describe the effect of osmolyte concentration, temperature and pressure is freely available at https://github.com/andrea-arsiccio/DeltaG-calculation. The script automatically outputs a DeltaG.dat file compatible with this SASA module. If TYPE = TRANSFER and no DELTAGFILE is provided, the free energy values are those describing the effect of temperature, and are computed using the temperature value passed by the MD engine
APPROACH	either approach 2 or 3. Necessary only if TYPE = TRANSFER and no DELTAGFILE is provided. If TYPE = TRANSFER and no DELTAGFILE is provided, the free energy values are those describing the effect of temperature, and the program must know if approach 2 or 3 (as described in Arsiccio and Shea, Protein Cold Denaturation in Implicit Solvent Simulations: A Transfer Free Energy Approach, J. Phys. Chem. B, 2021) needs to be used to compute them

8.13.5.2 SASA_LCPO

	This is part of the sasa module
	It is only available if you configure PLUMED with ./configure –enable-modules=sasa . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.
The atoms for which the SASA is desired should be indicated with the keyword ATOMS, and a pdb file of the protein must be provided in input with the MOLINFO keyword. The LCPO algorithm is used for the calculation (please, read and cite [84]). The radius of the solvent is assumed to be 0.14 nm, which is the radius of water molecules. Using the keyword NL_STRIDE it is also possible to specify the frequency with which the neighbor list for the calculation of the SASA is updated (the default is every 10 steps).
Different properties can be calculated and selected using the TYPE keyword:
1) the total SASA (TOTAL);
2) the free energy of transfer for the protein according to the transfer model (TRANSFER). This keyword can be used, for instance, to compute the transfer of a protein to different temperatures, as detailed in [86], or to different pressures, as detailed in [87], or to different osmolyte solutions, as detailed in [85].
When the TRANSFER keyword is used, a file with the free energy of transfer values for the sidechains and backbone atoms should be provided (using the keyword DELTAGFILE). Such file should have the following format:

```
----------------Sample DeltaG.dat file---------------------
ALA     0.711019999999962
ARG     -2.24832799999996
ASN     -2.74838799999999
ASP     -2.5626376
CYS     3.89864000000006
```

```
GLN     -1.76192
GLU     -2.38664400000002
GLY     0
HIS     -3.58152799999999
ILE     2.42634399999986
LEU     1.77233599999988
LYS     -1.92576400000002
MET     -0.262827999999956
PHE     1.62028800000007
PRO     -2.15598800000001
SER     -1.60934800000004
THR     -0.591559999999987
TRP     1.22936000000027
TYR     0.775547999999958
VAL     2.12779200000011
BACKBONE        1.00066920000002
------------------------------------------------------------
```

where the second column is the free energy of transfer for each sidechain/backbone, in kJ/mol.
A Python script for the computation of free energy of transfer values to describe the effect of osmolyte concentration, temperature and pressure (according to [85], [86] and [87]) is freely available at https://github.com/andrea-arsiccio/DeltaG-calculation. The script automatically outputs a DeltaG.dat file compatible with this SASA module.
If the DELTAGFILE is not provided, the program computes the free energy of transfer values as if they had to take into account the effect of temperature according to approaches 2 or 3 in the paper [86]. Please read and cite this paper if using the transfer model for computing the effect of temperature in implicit solvent simulations. For this purpose, the keyword APPROACH should be added, and set to either 2 or 3.
The SASA usually makes sense when atoms used for the calculation are all part of the same molecule. When running with periodic boundary conditions, the atoms should be in the proper periodic image. This is done automatically since PLUMED 2.2, by considering the ordered list of atoms and rebuilding the broken entities using a procedure that is equivalent to that done in WHOLEMOLECULES. Notice that rebuilding is local to this action. This is different from WHOLEMOLECULES which actually modifies the coordinates stored in PLUMED.
In case you want to recover the old behavior you should use the NOPBC flag. In that case you need to take care that atoms are in the correct periodic image.
The SASA may also be computed using the SASA_HASEL collective variable, which makes use of the algorithm described in [83]. SASA_HASEL is less accurate then SASA_LCPO, but the computation is faster.

Examples

The following input tells plumed to print the total SASA for atoms 10 to 20 in a protein chain.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SASA_LCPO.tmp
SASA_LCPO TYPE=TOTAL ATOMS=10-20 NL_STRIDE=10 LABEL=sasa
PRINT ARG=sasa STRIDE=1 FILE=colvar
```

The following input tells plumed to compute the transfer free energy for the protein chain containing atoms 10 to 20. Such transfer free energy is then used as a bias in the simulation (e.g., implicit solvent simulations). The free energy of transfer values are read from a file called DeltaG.dat.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SASA_LCPO.tmp
SASA_LCPO TYPE=TRANSFER ATOMS=10-20 NL_STRIDE=10 DELTAGFILE=DeltaG.dat LABEL=sasa

bias: BIASVALUE ARG=sasa

PRINT ARG=sasa,bias.* STRIDE=1 FILE=colvar
```

The following input tells plumed to compute the transfer free energy for the protein chain containing atoms 10 to 20. Such transfer free energy is then used as a bias in the simulation (e.g., implicit solvent simulations). The free energy of transfer values are computed according to [86], and take into account the effect of temperature using approach 2 as described in the paper.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/SASA_LCPO.tmp
SASA_LCPO TYPE=TRANSFER ATOMS=10-20 NL_STRIDE=10 APPROACH=2 LABEL=sasa

bias: BIASVALUE ARG=sasa

PRINT ARG=sasa,bias.* STRIDE=1 FILE=colvar
```

Glossary of keywords and components

The atoms involved can be specified using

ATOMS	the group of atoms that you are calculating the SASA for. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

TYPE	(default=TOTAL) The type of calculation you want to perform. Can be TOTAL or TRANSFER
NL_STRIDE	The frequency with which the neighbor list is updated.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
DELTAGFILE	a file containing the free energy values for backbone and sidechains. Necessary only if TYPE = TRANSFER. A Python script for the computation of free energy of transfer values to describe the effect of osmolyte concentration, temperature and pressure is freely available at https://github.↩ com/andrea-arsiccio/DeltaG-calculation. The script automatically outputs a DeltaG.dat file compatible with this SASA module. If TYPE = TRANSFER and no DELTAGFILE is provided, the free energy values are those describing the effect of temperature, and are computed using the temperature value passed by the MD engine
APPROACH	either approach 2 or 3. Necessary only if TYPE = TRANSFER and no DELT↩ AGFILE is provided. If TYPE = TRANSFER and no DELTAGFILE is provided, the free energy values are those describing the effect of temperature, and the program must know if approach 2 or 3 (as described in Arsiccio and Shea, Protein Cold Denaturation in Implicit Solvent Simulations: A Transfer Free Energy Approach, J. Phys. Chem. B, 2021) needs to be used to compute them

8.14 Funnel-Metadynamics (FM)

8.14.1 Overview

FM is a combination of Metadynamics bias potential [47] with a funnel-shape restraint potential applied to the target structure of a binding interaction. The latter is composed of a cone restraint, which covers the ligand binding site, and a cylindrical one that heads towards the solvent [88]. When inside the funnel volume, the ligand does not feel any restraint potential, proceeding as regular Metadynamics. Upon reaching the boundaries of the funnel, a repulsive bias is applied forcing the ligand to remain in the allowed funnel space. The result is an acceleration in the sampling of the binding/unbinding process, leading to a swift convergence of the calculation and a well-defined binding free-energy surface.

8.14.2 Installation

This module is not installed by default. Add '--enable-modules=funnel' to your './configure' command when building PLUMED to enable these features.

8.14.3 Usage

This module is a direct evolution of the original FM [88] since it incorporates an alignment function that removes the necessity to block the target macromolecule in the simulation box.
The user can follow a comprehensive protocol [89], which will help in all stages of the simulation, including pre- and post-processing. An example of input file can be found on `FUNNEL-NEST's webpage`

8.14.4 Contents

The funnel module is composed of a collective variable that calculates the position of a ligand with respect to a line and a potential that creates a funnel-shape restraint centered on the line (FUNNEL_PS and FUNNEL, respectively).

- CV documentation
- Bias Documentation

8.14.5 CV documentation

The following list contains descriptions of the collective variables developed for the PLUMED-FUNNEL module. They should be used in combination with the funnel-shaped restraint potential and Metadynamics to enable Funnel-Metadynamics.

FUNNEL_PS	FUNNEL_PS implements the Funnel-Metadynamics (FM) technique in PLUMED 2.

8.14.5.1 FUNNEL_PS

	This is part of the funnel module
	It is only available if you configure PLUMED with ./configure –enable-modules=funnel . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

FUNNEL_PS implements the Funnel-Metadynamics (FM) technique in PLUMED 2.
Please read the FM [88] [89] papers to better understand the notions hereby reported.
This colvar evaluates the position of a ligand of interest with respect to a given line, built from the two points A and B, and should be used together with the FUNNEL bias. The constructed line represents the axis of the funnel-shape restraint potential, which should be placed so as to include the portion of a macromolecule (i.e., protein, DNA, etc.) that should be explored. Since it is important that the position of the line is updated based on the motion of the macromolecule during the simulation, this colvar incorporates an alignment method. The latter uses the TYPE=O↩
PTIMAL option to remove motions due to rotation and translation of the macromolecule with respect to a reference structure, which is provided by the user. In order to accomplish the task, an optimal alignment matrix is calculated using the Kearsley [21] algorithm. The reference structure should be given as a pdb file, containing only the atoms of the macromolecule or a selection of them (e.g., the protein CA atoms of secondary structures). In contrast to the methods reported in the Distances from reference configurations, the values reported in the occupancy and beta-factor columns of the pdb file are not important since they will be overwritten and replaced by the value 1.00 during the procedure. It is important to understand that all atoms in the file will be used for the alignment, even if they display 0.00 in the occupancy column.
The ligand can be represented by one single atom or the center of mass (COM) of a group of atoms that should be provided by the user.
By default FUNNEL_PS is computed taking into account periodic boundary conditions. Since PLUMED 2.↩
5, molecules are rebuilt using a procedure that is equivalent to that done in WHOLEMOLECULES. We note that this action is local to this colvar, thus it does not modify the coordinates stored in PLUMED. Moreover, FUNNEL_PS requires an ANCHOR atom to be specified in order to facilitate the reconstruction of periodic boundary conditions. This atom should be the closest macromolecule's atom to the ligand and it should reduce the risk of ligand "warp-

ing" in the simulation box. Nevertheless, we highly recommend to add to the PLUMED input file a custom line of WHOLEMOLECULES, in order to be sure of reconstructing the ligand together with the macromolecule (please look the examples). In this case, the user can use the NOPBC flag to turn off the internal periodic boundary condition reconstruction.
FUNNEL_PS is divided in two components (fps.lp and fps.ld) which evaluate the projection of the ligand along the funnel line and the distance from it, respectively. The values attributed to these two components are then used together with the potential file created by the FUNNEL bias to define if the ligand is within or not in the funnel-shape restraint potential. In the latter case, the potential will force the ligand to enter within the funnel boundaries.

Examples

The following input tells plumed to print the FUNNEL_PS components for the COM of a ligand. The inputs are a reference structure, which is the structure used for the alignment, the COM of the molecule we want to track, and 2 points in the Cartesian space (i.e., x, y, and z) to draw the line representing the funnel axis.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FUNNEL_PS.tmp
lig: COM ATOMS=2446,2447,2448,2449,2451
fps: FUNNEL_PS REFERENCE=protein.pdb LIGAND=lig POINTS=5.3478,-0.7278,2.4746,7.3785,6.7364,-9.3624
PRINT ARG=fps.lp,fps.ld
```

It is recommended to add a line to force the reconstruction of the periodic boundary conditions. In the following example, WHOLEMOLECULES was added to make sure that the ligand was reconstructed together with the protein. The list contains all the atoms reported in the start.pdb file followed by the ANCHOR atom and the ligand. All atoms should be contained in the same entity and the correct order.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FUNNEL_PS.tmp
WHOLEMOLECULES ENTITY0=54,75,212,228,239,258,311,328,348,372,383,402,421,463,487,503,519,657,674,690,714,897,9
974,985,1007,1018,1037,1247,1264,1283,1302,1324,1689,1708,1727,1738,1962,1984,1994,2312,2329,2349,2467,2490,25
2547,2554,2569,2575,2591,2607,2635,2657,2676,2693,2700,2719,2735,2746,2770,2777,2788,2795,2805,2815,2832,2854,
2911,2927,2948,2962,2472,3221,3224,3225,3228,3229,3231,3233,3235,3237
lig: COM ATOMS=3221,3224,3225,3228,3229,3231,3233,3235,3237
fps: FUNNEL_PS LIGAND=lig REFERENCE=start.pdb ANCHOR=2472 POINTS=4.724,5.369,4.069,4.597,5.721,4.343
PRINT ARG=fps.lp,fps.ld
```

Glossary of keywords and components

Description of components

By default the value of the calculated quantity can be referenced elsewhere in the input file by using the label of the action. Alternatively this Action can be used to calculate the following quantities by employing the keywords listed below. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
lp	the position along the funnel line
ld	the distance from the funnel line

The atoms involved can be specified using

LIGAND	This MUST be a single atom, normally the COM of the ligand. For more information on how to specify lists of atoms see Groups and Virtual Atoms
ANCHOR	Closest protein atom to the ligand, picked to avoid pbc problems during the simulation. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

REFERENCE	a file in pdb format containing the structure you would like to align.
POINTS	6 values defining x, y, and z of the 2 points used to construct the line. The order should be A_x,A_y,A_z,B_x,B_y,B_z.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
SQUARED-ROOT	(default=off) Used to initialize the creation of the alignment variable

8.14.6 Bias Documentation

The following list contains descriptions of biases developed for the PLUMED-FUNNEL module. They should be used in combination with the collective variable to calculate the position relative to the funnel-shape restraint potential and Metadynamics to enable Funnel-Metadynamics.

FUNNEL	Calculate a funnel-shape restraint potential that is defined on a grid that is read during the setup.

8.14.6.1 FUNNEL

	This is part of the funnel module
	It is only available if you configure PLUMED with ./configure –enable-modules=funnel . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate a funnel-shape restraint potential that is defined on a grid that is read during the setup.
If the input file is not already present, it will create one with the name specified in the FILE flag. The potential has a two-dimensional resolution since it has been devised to be used with the two components of FUNNEL_PS (i.e., fps.lp and fps.ld) and it is divided in two sections, a cone shape attached to a cylindrical one. The user can customize the shape of both the sections by modifying a number of flags. In particular the cone section of the funnel is calculated with the following formula:

$$MAX_Z = R_{cyl} + tg_{alpha} * (z_{cc} - MIN_S)$$

where MAX_Z is the radius of the cone base, R_{cyl} is the radius of the cylinder part, tg_{alpha} is the angle regulating how steep the cone is, z_{cc} is the switching point between cone and cylinder, and MIN_S is the lowest possible value assumed by fps.lp of FUNNEL_PS. As for the cylinder, it starts from the value of z_{cc} and stops at the value of MAX_S with a section of $pi * r_{cyl}^2$.
There is the option of transforming the cone region into a sphere with the use of the SPHERE flag. In this case, the new shape will have a radius of z_{cc}. It might be necessary tuning the SAFETY option to select how much the potential extends from the sphere.

Examples

The following is an input for a calculation with a funnel potential that is defined in the file BIAS and that acts on the collective variables defined by FUNNEL_PS.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FUNNEL.tmp
lig: COM ATOMS=3221,3224,3225,3228,3229,3231,3233,3235,3237
```

```
fps: FUNNEL_PS LIGAND=lig REFERENCE=start.pdb ANCHOR=2472 POINTS=4.724,5.369,4.069,4.597,5.721,4.343

FUNNEL ARG=fps.lp,fps.ld ZCC=1.8 ALPHA=0.55 RCYL=0.1 MINS=-0.5 MAXS=3.7 KAPPA=35100 NBINS=500 NBINZ=500 FILE=B
```

The BIAS will then look something like this:

```
#! FIELDS fps.lp fps.ld funnel.bias der_fps.lp der_fps.ld
#! SET min_fps.lp -0.500000
#! SET max_fps.lp 3.700000
#! SET nbins_fps.lp 500.000000
#! SET periodic_fps.lp false
#! SET min_fps.ld 0.000000
#! SET max_fps.ld 1.510142
#! SET nbins_fps.ld 500.000000
#! SET periodic_fps.ld false
    -0.500000      0.000000      0.000000      0.000000      0.000000
    -0.500000      0.003020      0.000000      0.000000      0.000000
    -0.500000      0.006041      0.000000      0.000000      0.000000
    -0.500000      0.009061      0.000000      0.000000      0.000000
    -0.500000      0.012081      0.000000      0.000000      0.000000
    -0.500000      0.015101      0.000000      0.000000      0.000000
```

The Funnel potential should always be used in combination with the collective variable FUNNEL_PS, since it is constructed to take as inputs fps.lp and fps.ld (the former linepos and linedist of Funnel-Metadynamics **[FM]**). In the first block of data the value of fps.lp (the value in the first column) is kept fixed and the value of the function is given at 500 equally spaced values for fps.ld between 0 and 1.51. In the second block of data fps.lp is fixed at $-0.5 + \frac{4.2}{500}$ and the value of the function is given at 500 equally spaced values for fps.ld between 0 and 1.51. In the third block of data the same is done but fps.lp is fixed at $-0.5 + \frac{8.4}{100}$ and so on until you get to the five hundredth block of data where fps.lp is fixed at 3.7.
It is possible to switch the shape of the cone region, transforming it in a sphere, with the flag SPHERE.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/FUNNEL.tmp
lig: COM ATOMS=545,546,547,548,549,550,551,552,553
fps: FUNNEL_PS LIGAND=lig REFERENCE=ref.pdb ANCHOR=52 POINTS=2.793,3.696,3.942,3.607,4.298,3.452

FUNNEL ARG=fps.lp,fps.ld ZCC=4.0 RCYL=0.1 MINS=0.2 MAXS=4.9 KAPPA=100000 NBINS=500 NBINZ=500 SPHERE SAFETY=1.0
```

Glossary of keywords and components

Description of components

By default this Action calculates the following quantities. These quantities can be referenced elsewhere in the input by using this Action's label followed by a dot and the name of the quantity required from the list below.

Quantity	Description
bias	the instantaneous value of the bias potential

Compulsory keywords

SCALE	(default=1.0) a factor that multiplies the external potential, useful to invert free energies
MAXS	(default=MAXS) maximum value assumed by fps.lp
ZCC	(default=ZCC) switching point between cylinder and cone
FILE	name of the Funnel potential file

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOSPLINE	(default=off) specifies that no spline interpolation is to be used when calculating the energy and forces due to the external potential
SPARSE	(default=off) specifies that the external potential uses a sparse grid
SPHERE	(default=off) The Funnel potential including the binding site can be spherical instead of a cone
WALKERS_MPI	(default=off) To be used when gromacs + multiple walkers are used
ARG	the input for this action is the scalar output from one or more other actions. The particular scalars that you will use are referenced using the label of the action. If the label appears on its own then it is assumed that the Action calculates a single scalar value. The value of this scalar is thus used as the input to this new action. If $*$ or $*.*$ appears the scalars calculated by all the proceeding actions in the input file are taken. Some actions have multi-component outputs and each component of the output has a specific label. For example a DISTANCE action labelled dist may have three components x, y and z. To take just the x component you should use dist.x, if you wish to take all three components then use dist.$*$.More information on the referencing of Actions can be found in the section of the manual on the PLUMED Getting Started. Scalar values can also be referenced using POSIX regular expressions as detailed in the section on Regular Expressions. To use this feature you you must compile PLUMED with the appropriate flag.. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
NBINS	number of bins along fps.lp
NBINZ	number of bins along fps.ld
MINS	minimum value assumed by fps.lp, if the ligand is able to go beyond this value the simulation will crash
KAPPA	constant to be used for the funnel-shape restraint potential
RCYL	radius of the cylindrical section
SAFETY	To be used in case the SPHERE flag is chosen, it regulates how much the potential extends (in nm)
SLOPE	Adjust the behavior of the potential outside the funnel, greater values than 1.0 will tend to push the ligand more towards the cylinder and vice versa
ALPHA	angle to change the width of the cone section

8.15 Membrane Fusion

8.15.1 Overview

Membrane fusion process, when two separate membranes merge, is crucial in life. The fusion of lipid bilayers follows a series of discrete steps with two relevant intermediates: hemifusion structures and fusion pores. The hemifusion structures mix lipids from the involved membranes without cargo exchange, while the fusion pore require an aqueous channel to connect the contents.
To study the hemifusion stage computationally, Hub and Awasthi developed a CV that initially nucleated hydrophilic pores in lipid bilayers [91] and later extended it to induce the hemifusion stalks [93]. Di Bartolo and Masone implemented that CV in PLUMED [90].
Then, to nucleate and expand the fusion pore, based on Hub's work in single lipid bilayers [92], Di Bartolo and Masone implemented two others CVs to nucleate and expand fusion pores.

8.15.2 Installation

This module is not installed by default. Add '--enable-modules=membranefusion' to your './configure' command when building PLUMED to enable these features.

8.15.3 Usage

This module contains three CVs to:

- Induce a hemifusion stalk: MEMFUSIONP
- Nucleate a fusion pore: FUSIONPORENUCLEATIONP
- Expand a fusion pore: FUSIONPOREEXPANSIONP

8.15.4 Contents

8.15.5 CVs Documentation

The following list contains descriptions of biases developed for the PLUMED-MEMBRANEFUSION module. They can be used in combination with other biases outside of the MEMBRANEFUSION module.

FUSIONPOREEXPANSIONP	A CV for inducing the expansion of a fusion pore from a nucleated fusion pore.
FUSIONPORENUCLEATIONP	A CV for inducing the nucleation of the fusion pore from a hemifusion stalk.
MEMFUSIONP	Calculate a CV that can induce the formation of the hemifusion stalk between two initially flat and planar bilayers.

8.15.5.1 FUSIONPOREEXPANSIONP

	This is part of the membranefusion module
	It is only available if you configure PLUMED with ./configure –enable-modules=membranefusion . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

A CV for inducing the expansion of a fusion pore from a nucleated fusion pore.
Calculate the collective variable designed by Hub [92] and implemented into PLUMED by Masone and collaborators. This CV is capable of inducing the expansion of the fusion pore from a nucleated fusion pore.

$$\xi_e = \frac{R(r) - R_0}{R_0}$$

Where ξ_e is the CV, R_0 is a normalization constant that makes zero the initial value of ξ_e, and $R(r)$ is the approximate radius of the fusion pore, which is defined by the number of waters and phosphateoxygens beads within a horizontal layer in the center of both membranes.

Examples

This example induces the expansion of a nucleated fusion pore ($\xi_e = 0.75$) from a just nucleated fusion pore ($\xi_e = 0.00$).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FUSIONPOREEXPANSIONP.tmp
lMem: GROUP ATOMS=1-10752,21505-22728,23953-24420 #All the lower membrane beads.
uMem: GROUP ATOMS=10753-21504,22729-23952,24421-24888 #All the upper membrane beads.
tails: GROUP ATOMS=8-23948:12,12-23952:12,23966-24884:18,23970-24888:18 #All the lipid tails beads (from the l
waters: GROUP ATOMS=24889-56589  #All the water beads.
po4: GROUP ATOMS=2-23942:12,23957-24875:18 #All the lipid phosphateoxygens beads.
```

```
fusionPoreExpansion: FUSIONPOREEXPANSIONP UMEMBRANE=uMem LMEMBRANE=lMem TAILS=tails WATERS=waters PHOSPHATEOXY

MOVINGRESTRAINT ...
    ARG=fusionPoreExpansion
    STEP0=0 AT0=0.0 KAPPA0=10000.0
    STEP1=500000 AT1=0.75 KAPPA1=10000.0
...

PRINT ARG=fusionPoreExpansion FILE=COLVAR STRIDE=1
```

Glossary of keywords and components

The atoms involved can be specified using

UMEMBRANE	all the beads of the upper membrane.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
LMEMBRANE	all the beads of the lower membrane.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
TAILS	all the tail beads of the system.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
WATERS	all the water beads of the system.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
PHOSPHATEOXYGENS	all the lipid phosphateoxygens beads of the system.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NSMEM	the number of slices of the membrane fusion cylinder.
D	horizontal layer thickness, it depends on the Z separation of the membranes.
R0	normalization constant that makes 0 the initial value of the CV.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
DSMEM	(default=0.1) thickness of the slices of the membrane fusion cylinder.
HMEM	(default=0.25) parameter of the step function (x,h) for the membrane fusion.
VO	(default=0.076879) beads' molecular volume.
H	(default=0.1) parameter of the step function (x,h) for the fusion pore expansion.
RMAX	(default=2.5) to avoid effects of membrane undulations in large membranes (more than 256 lipids).
XCYL	X coordinate of the fixed cylinder, if not present this will be calculated.
YCYL	X coordinate of the fixed cylinder, if not present this will be calculated.

8.15.5.2 FUSIONPORENUCLEATIONP

	This is part of the membranefusion module
	It is only available if you configure PLUMED with ./configure –enable-modules=membranefusion . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

A CV for inducing the nucleation of the fusion pore from a hemifusion stalk.
Calculate the collective variable designed by Hub and collaborators [91] and implemented into PLUMED by Masone and collaborators. This CV is capable of inducing the nucleation of the fusion pore from a hemifusion stalk.

$$\xi_n = \frac{1}{N_{sn}} \sum_{s=0}^{N_{sn}-1} \delta_{sn}(N_{sn}^{(p)})$$

Where ξ_n is the CV, N_{sn} is the number of slices of the cylinder that make up the CV, δ_{sn} is a continuos function in the interval [0 1] ($\delta_{sf} = 0$ for no beads in the slice s, and $\delta_{sf} = 1$ for 1 or more beads in the slice s) and $N_{sf}^{(p)}$ accounts for the number of water and phosphateoxygens beads within the slice s.

Examples

This example induces the nucleation of the fusion pore ($\xi_n = 1.0$) from a hemifusion stalk ($\xi_n = 0.2$).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FUSIONPORENUCLEATIONP.tmp

lMem: GROUP ATOMS=1-10752,21505-22728,23953-24420 #All the lower membrane beads.
uMem: GROUP ATOMS=10753-21504,22729-23952,24421-24888 #All the upper membrane beads.
tails: GROUP ATOMS=8-23948:12,12-23952:12,23966-24884:18,23970-24888:18 #All the lipid tails beads (from the l
waters: GROUP ATOMS=24889-56490 #All the water beads.
po4: GROUP ATOMS=2-23942:12,23957-24875:18 #All the lipid phosphateoxygens beads.

fusionPoreNucleation: FUSIONPORENUCLEATIONP UMEMBRANE=uMem LMEMBRANE=lMem TAILS=tails WATERS=waters PHOSPHATEO

MOVINGRESTRAINT ...
    ARG=fusionPoreNucleation
    STEP0=0 AT0=0.2 KAPPA0=10000.0
    STEP1=500000 AT1=1.0 KAPPA1=10000.0
...

PRINT ARG=fusionPoreNucleation FILE=COLVAR STRIDE=1
```

Glossary of keywords and components

The atoms involved can be specified using

UMEMBRANE	all the beads of the upper membrane.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
LMEMBRANE	all the beads of the lower membrane.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
TAILS	all the tail beads of the system.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
WATERS	all the water beads of the system.. For more information on how to specify lists of atoms see Groups and Virtual Atoms
PHOSPHATEOXYGENS	all the lipid phosphateoxygens beads of the system.. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NSMEM	the number of slices of the membrane fusion cylinder.
NS	the number of slices of the membrane-spanning cylinder in such a way that when the bilayers are flat and parallel the CV is equal to 0.2.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
DSMEM	(default=0.1) thickness of the slices of the membrane fusion cylinder.
HMEM	(default=0.25) parameter of the step function (x,h) for the membrane fusion.
DS	(default=0.25) thickness of the slices of the membrane-spanning cylinder.
HCH	(default=0.25) parameter of the step function (x,h) for the CV.
RCYL	(default=0.8) the radius of the membrane-spanning cylinder.
ZETA	(default=0.75) parameter of the switch function (x,).
ONEOVERS2C2CUTOFF	(default=500) cut off large values for the derivative of the atan2 function to avoid violate energy.
XCYL	X coordinate of the fixed cylinder, if not present this will be calculated.
YCYL	X coordinate of the fixed cylinder, if not present this will be calculated.

8.15.5.3 MEMFUSIONP

	This is part of the membranefusion module
	It is only available if you configure PLUMED with ./configure –enable-modules=membranefusion . Furthermore, this feature is still being developed so take care when using it and report any problems on the mailing list.

Calculate a CV that can induce the formation of the hemifusion stalk between two initially flat and planar bilayers. Calculate the collective variable designed by Hub and collaborators [91] and implemented into PLUMED by Masone and collaborators [90] . This CV is capable of inducing the formation of the hemifusion stalk between two initially flat and planar bilayers surrounded by water molecules.

$$\xi_f = \frac{1}{N_{sf}} \sum_{s=0}^{N_{sf}-1} \delta_{sf}(N_{sf}^{(p)})$$

Where ξ_f is the CV, N_{sf} is the number of slices of the cylinder that make up the CV, δ_{sf} is a continuos function in the interval [0 1] ($\delta_{sf} = 0$ for no beads in the slice s, and $\delta_{sf} = 1$ for 1 or more beads in the slice s) and $N_{sf}^{(p)}$ accounts for the number of tail beads within the slice s.

Examples

This example induces a hemifusion stalk ($\xi_f = 0.85$) from a pair of initially flat membranes ($\xi_f = 0.2$).

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MEMFUSIONP.tmp
lMem: GROUP ATOMS=1-12288 #All the lower membrane beads.
uMem: GROUP ATOMS=12289-24576 #All the upper membrane beads.
tails: GROUP ATOMS=8-24572:12,12-24576:12 #All the lipid tails beads (from the lower and upper membrane).
```

```
memFusion: MEMFUSIONP UMEMBRANE=uMem LMEMBRANE=lMem TAILS=tails NSMEM=70 DSMEM=0.1 HMEM=0.25 RCYLMEM=1.75 ZETA

MOVINGRESTRAINT ...
    ARG=memFusion
    STEP0=0 AT0=0.2 KAPPA0=10000.0
    STEP1=500000 AT1=0.85 KAPPA1=10000.0
...

PRINT ARG=memFusion FILE=COLVAR STRIDE=1
```

You can test this CV with another example in this GitHub folder.

Glossary of keywords and components

The atoms involved can be specified using

UMEMBRANE	all the beads of the upper membrane. For more information on how to specify lists of atoms see Groups and Virtual Atoms
LMEMBRANE	all the beads of the lower membrane. For more information on how to specify lists of atoms see Groups and Virtual Atoms
TAILS	all the tail beads of the system. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

NSMEM	the number of slices of the membrane fusion cylinder in such a way that when the bilayers are flat and parallel the CV is equal to 0.2.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
DSMEM	(default=0.1) thickness of the slices of the membrane fusion cylinder.
HMEM	(default=0.25) parameter of the step function (x,h) for the membrane fusion.
RCYLMEM	(default=1.75) the radius of the membrane fusion cylinder.
ZETAMEM	(default=0.5) occupation factor.
ONEOVERS2C2CUTOFF	(default=500) cut off large values for the derivative of the atan2 function.
XCYL	X coordinate of the fixed cylinder, if not present this will be calculated.
YCYL	Y coordinate of the fixed cylinder, if not present this will be calculated.

Chapter 9

Command Line Tools

PLUMED contains a number of simple command line tools. To use one of these tools you issue a command something like:

```
plumed <toolname> <list of input flags for that tool>
```

The following is a list of the various standalone tools that PLUMED contains.

completion	Dumps the body of a bash function to be used for auto completion.
config	inquire plumed about how it was configure
driver	driver is a tool that allows one to to use plumed to post-process an existing trajectory.
driver-float	Equivalent to driver, but using single precision reals.
gen_example	gen_example is a tool that you can use to construct an example for the manual that users can interact with to understand
gen_json	gen_json constructs a json file that includes a dictionary of actions, the keywords for those actions and the components and outputs this to standard output
gentemplate	gentemplate is a tool that you can use to construct template inputs for the various actions
info	This tool allows you to obtain information about your plumed version
kt	Print out the value of k_BT at a particular temperature
manual	manual is a tool that you can use to construct the manual page fora particular action
mklib	compile a .cpp file into a shared library
newcv	create a new collective variable from a template
partial_tempering	scale parameters in a gromacs topology to implement solute or partial tempering
patch	patch an MD engine
pathtools	pathtools can be used to construct paths from pdb data
pdbrenumber	Modify atom numbers in a PDB, possibly using hybrid-36 coding.
pesmd	Pesmd allows one to do (biased) Langevin dynamics on a two-dimensional potential energy surface.
selector	create lists of serial atom numbers
simplemd	simplemd allows one to do molecular dynamics on systems of Lennard-Jones atoms.
sum_hills	sum_hills is a tool that allows one to to use plumed to post-process an existing hills/colvar file
vim2html	convert plumed input file to colored html using vim syntax

In addition to the keywords above, by enabling optional modules you can access to the following keywords:

drr_tool	(from Extended-System Adaptive Biasing Force module) - Extract .grad and .count files from the binary output .drrstate - Merge windows
ves_md_linearexpansion	(from Variationally Enhanced Sampling (VES code) module) Simple MD code for dynamics on a potential energy surface given by a linear basis set expansion.

For all these tools and to use PLUMED as a plugin in an MD calculation you will need an input file.

9.1 completion

	This is part of the cltools module

Dumps the body of a bash function to be used for auto completion.
Users will typically not need this command. See more at Using bash autocompletion

Examples

```
plumed completion
```

Glossary of keywords and components

Options

--help/-h	(default=off) print this help

9.2 config

inquire plumed about how it was configure

Note

This command line tool is implemented as a shell script. Its help message is pasted below:

```
Options:
  -h, --help
                    print this help and exit
  -q, --quiet
                    don't write anything, just return true of false
  show
                    dump a full configuration file
  has [word1 [word2]..]
                    check if plumed has features words
  module [word1 [word2]..]
                    check if plumed has enables modules words
  makefile_conf
                    dumps the Makefile.conf file

Examples:

Check if plumed as dlopen enabled
> plumed config has dlopen
Check if plumed as dlopen AND zlib enabled
> plumed config has dlopen zlib
Check if plumed as module colvar active
> plumed config module colvar
```

Glossary of keywords and components

9.3 driver

	This is part of the cltools module

driver is a tool that allows one to to use plumed to post-process an existing trajectory.
The input to driver is specified using the command line arguments described below.
In addition, you can use the special READ command inside your plumed input to read in colvar files that were generated during your MD simulation. The values read in can then be treated like calculated colvars.

Warning

Notice that by default the driver has no knowledge about the masses and charges of your atoms! Thus, if you want to compute quantities depending charges (e.g. DHENERGY) or masses (e.g. COM) you should pass the proper information to the driver. You can do it either with the –pdb option or with the –mc option. The latter will read a file produced by DUMPMASSCHARGE .

Examples

The following command tells plumed to post process the trajectory contained in `trajectory.xyz` by performing the actions described in the input file `plumed.dat`. If an action that takes the stride keyword is given a stride equal to n then it will be performed only on every nth frames in the trajectory file.

```
plumed driver --plumed plumed.dat --ixyz trajectory.xyz
```

Notice that `xyz` files are expected to be in internal PLUMED units, that is by default nm. You can change this behavior by using the `--length-units` option:

```
plumed driver --plumed plumed.dat --ixyz trajectory.xyz --length-units A
```

The strings accepted by the `--length-units` options are the same ones accepted by the UNITS action. Other file formats typically have their default coordinates (e.g., `gro` files are always in nm) and it thus should not be necessary to use the `--length-units` option. Additionally, consider that the units used by the `driver` might be different by the units used in the PLUMED input file `plumed.dat`. For instance consider the command:

```
plumed driver --plumed plumed.dat --ixyz trajectory.xyz --length-units A
```

where `plumed.dat` is

```
BEGIN_PLUMED_FILE working DATADIR=example-check/driver.tmp
# no explicit UNITS action here
d: DISTANCE ATOMS=1,2
PRINT ARG=d FILE=colvar
```

In this case, the driver reads the `xyz` file assuming it to contain coordinates in Angstrom units. However, the resulting `colvar` file contains a distance expressed in nm.
The following command tells plumed to post process the trajectory contained in trajectory.xyz. by performing the actions described in the input file plumed.dat.

```
plumed driver --plumed plumed.dat --ixyz trajectory.xyz --trajectory-stride 100 --timestep 0.001
```

Here though `--trajectory-stride` is set equal to the frequency with which frames were output during the trajectory and the `--timestep` is equal to the simulation timestep. As such the `STRIDE` parameters in the `plumed.dat` files are referred to the original timestep and any files output resemble those that would have been generated had we run the calculation we are running with driver when the MD simulation was running.
PLUMED can read xyz files (in PLUMED units) and gro files (in nm). In addition, PLUMED includes by default support for a subset of the trajectory file formats supported by VMD, e.g. xtc and dcd:

```
plumed driver --plumed plumed.dat --pdb diala.pdb --mf_xtc traj.xtc --trajectory-stride 100 --timestep 0.001
```

where `--mf_` prefixes the extension of one of the accepted molfile plugin format. If PLUMED has been installed with full molfile support, other formats will be available. Just type `plumed driver --help` to see which plugins are available.
Molfile plugin require periodic cell to be triangular (i.e. first vector oriented along x and second vector in xy plane). This is true for many MD codes. However, it could be false if you rotate the coordinates in your trajectory before reading them in the driver. Also notice that some formats (e.g. amber crd) do not specify atom number. In this case you can use the `--natoms` option:

```
plumed driver --plumed plumed.dat --imf_crd trajectory.crd --natoms 128
```

Check the available molfile plugins and limitations at this link.
Additionally, you can use the xdrfile implementation of xtc and trr. To this aim, just download and install properly the xdrfile library (see this link). If the xdrfile library is installed properly the PLUMED configure script should be able to detect it and enable it. Notice that the xdrfile implementation of xtc and trr is more robust than the molfile one, since it provides support for generic cell shapes. In addition, it allows DUMPATOMS to write compressed xtc files.

Glossary of keywords and components

The input trajectory is specified using one of the following

--ixyz	the trajectory in xyz format
--igro	the trajectory in gro format
--idlp4	the trajectory in DL_POLY_4 format
--ixtc	the trajectory in xtc format (xdrfile implementation)
--itrr	the trajectory in trr format (xdrfile implementation)
--mf_dcd	molfile: the trajectory in dcd format
--mf_crd	molfile: the trajectory in crd format
--mf_crdbox	molfile: the trajectory in crdbox format
--mf_gro	molfile: the trajectory in gro format
--mf_g96	molfile: the trajectory in g96 format
--mf_trr	molfile: the trajectory in trr format
--mf_trj	molfile: the trajectory in trj format
--mf_xtc	molfile: the trajectory in xtc format
--mf_pdb	molfile: the trajectory in pdb format

The following must be present

--plumed	(default=plumed.dat) specify the name of the plumed input file
--timestep	(default=1.0) the timestep that was used in the calculation that produced this trajectory in picoseconds
--trajectory-stride	(default=1) the frequency with which frames were output to this trajectory during the simulation (0 means that the number of the step is read from the trajectory file, currently working only for xtc/trr files read with –ixtc/–trr)
--multi	(default=0) set number of replicas for multi environment (needs MPI)

The following options are available

--help/-h	(default=off) print this help
--help-debug	(default=off) print special options that can be used to create regtests
--noatoms	(default=off) don't read in a trajectory. Just use colvar files as specified in plumed.dat
--parse-only	(default=off) read the plumed input file and stop
--restart	(default=off) makes driver behave as if restarting
--dump-full-virial	(default=off) with –dump-forces, it dumps the 9 components of the virial
--shortcut-ofile	the name of the file to output info on the way shortcuts have been expanded. If there are no shortcuts in your input file nothing is output
--length-units	units for length, either as a string or a number
--mass-units	units for mass in pdb and mc file, either as a string or a number
--charge-units	units for charge in pdb and mc file, either as a string or a number

--kt	set k_BT, it will not be necessary to specify temperature in input file
--dump-forces	dump the forces on a file
--dump-forces-fmt	(default=%f) the format to use to dump the forces
--pdb	provides a pdb with masses and charges
--mc	provides a file with masses and charges as produced with DUMPMASSCHARGE
--box	comma-separated box dimensions (3 for orthorhombic, 9 for generic)
--natoms	provides number of atoms - only used if file format does not contain number of atoms
--initial-step	provides a number for the initial step, default is 0
--debug-forces	output a file containing the forces due to the bias evaluated using numerical derivatives and using the analytical derivatives implemented in plumed

9.3.1 READ

This is part of the generic module

Read quantities from a colvar file.
This Action can be used with driver to read in a colvar file that was generated during an MD simulation

Description of components

The READ command will read those fields that are labelled with the text string given to the VALUE keyword. It will also read in any fields that are labeled with the text string given to the VALUE keyword followed by a dot and a further string. If a single Value is read in this value can be referenced using the label of the Action. Alternatively, if multiple quantities are read in, they can be referenced elsewhere in the input by using the label for the Action followed by a dot and the character string that appeared after the dot in the title of the field.

Examples

This input reads in data from a file called input_colvar.data that was generated in a calculation that involved PLU↩ MED. The first command reads in the data from the column headed phi1 while the second reads in the data from the column headed phi2.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/READ.tmp
rphi1:       READ FILE=input_colvar.data  VALUES=phi1
rphi2:       READ FILE=input_colvar.data  VALUES=phi2
PRINT ARG=rphi1,rphi2 STRIDE=500  FILE=output_colvar.data
```

The file input_colvar.data is just a normal colvar file as shown below

```
#! FIELDS time phi psi metad.bias metad.rbias metad.rct
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
 0.000000  -1.2379   0.8942   0.0000   0.0000   0.0000
 1.000000  -1.4839   1.0482   0.0000   0.0000   0.0089
 2.000000  -1.3243   0.6055   0.0753   0.0664   0.0184
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which the file should be read.
EVERY	(default=1) only read every nth line of the colvar file. This should be used if the colvar was written more frequently than the trajectory.
VALUES	the values to read from the file
FILE	the name of the file from which to read these quantities

Options

IGNORE_TIME	(default=off) ignore the time in the colvar file. When this flag is not present read will be quite strict about the start time of the simulation and the stride between frames
IGNORE_FORCES	(default=off) use this flag if the forces added by any bias can be safely ignored. As an example forces can be safely ignored if you are doing post processing that does not involve outputting forces
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

9.4 driver-float

This is part of the cltools module

Equivalent to driver, but using single precision reals.
The purpose of this tool is just to test what PLUMED does when linked from a single precision code.

Examples

```
plumed driver-float --plumed plumed.dat --ixyz trajectory.xyz
```

See also examples in driver

Glossary of keywords and components

The input trajectory is specified using one of the following

--ixyz	the trajectory in xyz format
--igro	the trajectory in gro format
--idlp4	the trajectory in DL_POLY_4 format
--ixtc	the trajectory in xtc format (xdrfile implementation)
--itrr	the trajectory in trr format (xdrfile implementation)
--mf_dcd	molfile: the trajectory in dcd format
--mf_crd	molfile: the trajectory in crd format
--mf_crdbox	molfile: the trajectory in crdbox format
--mf_gro	molfile: the trajectory in gro format
--mf_g96	molfile: the trajectory in g96 format
--mf_trr	molfile: the trajectory in trr format

--mf_trj	molfile: the trajectory in trj format
--mf_xtc	molfile: the trajectory in xtc format
--mf_pdb	molfile: the trajectory in pdb format

The following must be present

--plumed	(default=plumed.dat) specify the name of the plumed input file
--timestep	(default=1.0) the timestep that was used in the calculation that produced this trajectory in picoseconds
--trajectory-stride	(default=1) the frequency with which frames were output to this trajectory during the simulation (0 means that the number of the step is read from the trajectory file, currently working only for xtc/trr files read with –ixtc/–trr)
--multi	(default=0) set number of replicas for multi environment (needs MPI)

The following options are available

--help/-h	(default=off) print this help
--help-debug	(default=off) print special options that can be used to create regtests
--noatoms	(default=off) don't read in a trajectory. Just use colvar files as specified in plumed.dat
--parse-only	(default=off) read the plumed input file and stop
--restart	(default=off) makes driver behave as if restarting
--dump-full-virial	(default=off) with –dump-forces, it dumps the 9 components of the virial
--shortcut-ofile	the name of the file to output info on the way shortcuts have been expanded. If there are no shortcuts in your input file nothing is output
--length-units	units for length, either as a string or a number
--mass-units	units for mass in pdb and mc file, either as a string or a number
--charge-units	units for charge in pdb and mc file, either as a string or a number
--kt	set k_BT, it will not be necessary to specify temperature in input file
--dump-forces	dump the forces on a file
--dump-forces-fmt	(default=%f) the format to use to dump the forces
--pdb	provides a pdb with masses and charges
--mc	provides a file with masses and charges as produced with DUMPMASSCHARGE
--box	comma-separated box dimensions (3 for orthorhombic, 9 for generic)
--natoms	provides number of atoms - only used if file format does not contain number of atoms
--initial-step	provides a number for the initial step, default is 0
--debug-forces	output a file containing the forces due to the bias evaluated using numerical derivatives and using the analytical derivatives implemented in plumed

9.5 gen_example

This is part of the cltools module

gen_example is a tool that you can use to construct an example for the manual that users can interact with to understand
The example constructed by this action is in html. In all probability you will never need to use this tool. However, it is used within the scripts that generate the html manual for PLUMED. If you need to use this tool outside those scripts the input is specified using the following command line arguments.

Examples

The following generates an example based on the contents of the plumed file plumed.dat

```
plumed gen_example --plumed plumed.dat --status working
```

Glossary of keywords and components

Compulsory keywords

--plumed	(default=plumed.dat) convert the input in this file to the html manual
--out	(default=example.html) the file on which to output the example in html
--name	(default=ppp) the name to use for this particular input
--status	(default=nobadge) whether or not the input file works
--multi	(default=0) set number of replicas for multi environment (needs MPI)

Options

--help/-h	(default=off) print this help

9.6 gen_json

This is part of the cltools module

gen_json constructs a json file that includes a dictionary of actions, the keywords for those actions and the components and outputs this to standard output

Examples

The following command generates the json file

```
plumed gen_json
```

Glossary of keywords and components

Compulsory keywords

--actions	a file containing one line descriptions of the various actions

Options

--help/-h	(default=off) print this help

9.7 gentemplate

This is part of the cltools module

gentemplate is a tool that you can use to construct template inputs for the various actions
The templates generated by this tool are primarily for use with Toni Giorgino's VMD GUI. It may be useful however to use this tool as a quick aid memoir.

Examples

The following generates template input for the action DISTANCE.

```
plumed gentemplate --action DISTANCE
```

Glossary of keywords and components

Options

--help/-h	(default=off) print this help
--list	(default=off) print a list of the available actions
--include-optional	(default=off) also print optional modifiers
--action	print the template for this particular action

9.8 info

This is part of the cltools module

This tool allows you to obtain information about your plumed version
You can specify the information you require using the following command line arguments

Examples

The following command returns the root directory for your plumed distribution.

```
plumed info --root
```

Glossary of keywords and components

Options

--help/-h	(default=off) print this help
--configuration	(default=off) prints the configuration file
--root	(default=off) print the location of the root directory for the plumed source
--user-doc	(default=off) print the location of user manual (html)
--developer-doc	(default=off) print the location of user manual (html)
--version	(default=off) print the version number
--long-version	(default=off) print the version number (long version)
--git-version	(default=off) print the version number (git version, if available)
--include-dir	(default=off) print the location of the include dir
--soext	(default=off) print the extension of shared libraries (so or dylib)

9.9 kt

This is part of the cltools module

Print out the value of k_BT at a particular temperature

Examples

The following command will tell you the value of k_BT when T is equal to 300 K in eV

```
plumed kt --temp 300 --units eV
```

Glossary of keywords and components

Compulsory keywords

--temp	print the manual for this particular action
--units	(default=kj/mol) the units of energy can be kj/mol, kcal/mol, j/mol, eV or the conversion factor from kj/mol

Options

--help/-h	(default=off) print this help

9.10 manual

	This is part of the cltools module

manual is a tool that you can use to construct the manual page for a particular action
The manual constructed by this action is in html. In all probability you will never need to use this tool. However, it is used within the scripts that generate the html manual for PLUMED. If you need to use this tool outside those scripts the input is specified using the following command line arguments.

Examples

The following generates the html manual for the action DISTANCE.

```
plumed manual --action DISTANCE
```

Glossary of keywords and components

Compulsory keywords

--action	print the manual for this particular action

Options

--help/-h	(default=off) print this help
--vim	(default=off) print the keywords in vim syntax
--spelling	(default=off) print a list of the keywords and component names for the spell checker

9.11 mklib

compile a .cpp file into a shared library

Note

This command line tool is implemented as a shell script. Its help message is pasted below:

```
ERROR
type 'plumed mklib file.cpp'
```

Glossary of keywords and components

9.12 newcv

create a new collective variable from a template

Note

This command line tool is implemented as a shell script. Its help message is pasted below:

```
ERROR
type 'plumed newcv directive classname'
E.g. 'plumed newcv TORSION Torsion'
```

Glossary of keywords and components

9.13 partial_tempering

scale parameters in a gromacs topology to implement solute or partial tempering

Note

This command line tool is implemented as a shell script. Its help message is pasted below:

```
Usage:

  plumed partial_tempering [--gromacs4] scale < processed.top

where scale is the Hamiltonian scaling factor and
processed.top is a post-processed topology file (i.e. produced with grompp -pp)
where each "hot" atom has a "_" appended to the atom type, e.g.:

     1 amber99_43_     1    RC5    O5'      1    -0.6223         16   ; qtot -0.6223

Notice that the section that should be edited is the [atoms] section for all the
molecules that you wish to affect (typically only for the solute, but you may also
want to change solvent parameters).

Also remember to first produce the processed.top file with grompp -pp. Editing a normal
topol.top file will not work, because it does not contain all the parameters.
The processed.top file should not have any "#include" statement.

# produce a processed topology
grompp -pp
# choose the "hot" atoms
vi processed.top
# generate the actual topology
plumed partial_tempering $scale < processed.top > topol$i.top

WARNING: It's not very robust and there might be force-field dependent issues!
A few tests are strongly suggested.

WARNING: This script requires gawk to be available on your system.

1. Compare partial_tempering with scale=1.0 to non-scaled force field. E.g.
grompp -o topol-unscaled.tpr
grompp -pp
vi processed.top # choose the "hot" atoms appending "_". You can choose whatever.
plumed partial_tempering 1.0 < processed.top > topol-scaled.top # scale with factor 1
grompp -p topol-scaled.top -o topol-scaled.tpr
# Then do a rerun on a trajectory
mdrun -s topol-unscaled.tpr -rerun rerun.trr
mdrun -s topol-scaled.tpr -rerun rerun.trr
# and compare the resuling energy files. they should be identical

2. Compare partial_tempering with scale=0.5 to non-scaled force field.
Repeat the same procedure but using "plumed partial_tempering 0.5".
Choose all the atoms in all the relevant [atoms] sections (e.g. solute, solvent and ions).
In the two resulting energy files you should see:
long range electrostatics, LJ, and dihedral energy is *half* in the scaled case
all other terms (bonds/bends) are identical.
```

Glossary of keywords and components

9.14 patch

patch an MD engine

Note

This command line tool is implemented as a shell script. Its help message is pasted below:

```
Actions (choose one):
  -h, --help
                    print this help and exit
  -p, --patch
                    patch
  -r, -R, --revert
                    revert
  -l, --list-engines
                    print a list of available MD engines
  -s, --save
                    save, this needs *.preplumed files (*)
  --save-originals
                    same as save, but save also original files (*)
  -n NEWENGINE, --new NEWENGINE
                    create a new patch named NEWENGINE (*)
  -i, --info
                    output information on the patching procedure for a particular code
Options:
  -e ENGINE, --engine ENGINE
                    set MD engine to ENGINE (default: choose interactively)
  -m MODE, --mode MODE (default: shared)
                    set link mode to MODE, which can be either static, shared or runtime
  --static
                    same as --mode static
  --shared
                    same as --mode shared
  --runtime
                    same as --mode runtime
  -d FILE, --diff FILE
                    set the path to diff file (default: ROOT/patches/ENGINE.diff) (*)
  -q, --quiet
                    do not write loggin information; useful with -i to print just
                    the patching information
  -I, --include
                    use include files rather than symbolic links
  -f, --force
                    force patching (*)

(*) These options are for developers or for testing only. Be sure to know what
    you are doing before you use them.
```

Glossary of keywords and components

9.15 pathtools

	This is part of the mapping module

pathtools can be used to construct paths from pdb data
The path CVs in PLUMED are curvilinear coordinates through a high dimensional vector space. Enhanced sampling calculations are often run using the progress along the paths and the distance from the path as CVs as this provides a convenient way of defining a reaction coordinate for a complicated process. This method is explained in the documentation for PATH.
The path itself is an ordered set of equally-spaced, high-dimensional frames the way in which these frames should be constructed will depend on the problem in hand. In other words, you will need to understand the reaction you wish to study in order to select a sensible set of frames to use in your path CV. This tool provides two methods that may be useful when it comes to constructing paths; namely:

- A tool that takes in an initial guess path in which the frames are not equally spaced. This tool adjusts the positions of the frames in order to make them equally spaced so that they can be used as the basis for a path CV.
- A tool that takes two frames as input and that allows you to return a linear path connecting these two frames. The output from this method may be useful as an initial guess path. It is arguable that a linear path rather defeats the purpose of the path CV method, however, as the whole purpose is to be able to define non-linear paths.

Notice that you can use these two methods and take advantage of all the ways of measuring Distances from reference configurations that are available within PLUMED. The way you do this with each of these tools described above is explained in the example below.

Examples

The example below shows how you can take a set of unequally spaced frames from a pdb file named in_path.↩ pdb and use pathtools to make them equally spaced so that they can be used as the basis for a path CV. The file containing this final path is named final_path.pdb.

```
plumed pathtools --path in_path.pdb --metric EUCLIDEAN --out final_path.pdb
```

The example below shows how can create an initial linear path connecting the two pdb frames in start.pdb and end.pdb. In this case the path output to path.pdb will consist of 6 frames: the initial and final frames that were contained in start.pdb and end.pdb as well as four equally spaced frames along the vector connecting start.pdb to end.pdb.

```
plumed pathtools --start start.pdb --end end.pdb --nframes 4 --metric OPTIMAL --out path.pdb
```

Often the idea with path collective variables is to create a path connecting some initial state A to some final state B. You would in this case have representative configurations from your A and B states defined in the input files to pathtools that we have called start.pdb and end.pdb in the example above. Furthermore, it may be useful to have a few frames before your start frame and after your end frame. You can use path tools to create these extended paths as shown below. In this case the final path would now consist of 8 frames. Four of these frames would lie on the vector connecting state A to state B, there would be one frame each at start.pdb and end.pdb as well as one frame just before start.pdb and one frame just after end.pdb. All these frames would be equally spaced.

```
plumed pathtools --start start.pdb --end end.pdb --nframes 4 --metric OPTIMAL --out path.pdb --nframes-before-
```

Notice also that when you re-parameterize paths you must choose two frames to fix. Generally you chose to fix the states that are representative of your states A and B. By default pathtools will fix the first and last frames. You can, however, change the states to fix by taking advantage of the fixed flag as shown below.

```
plumed pathtools --path inpath.pdb --metric EUCLIDEAN --out outpath.pdb --fixed 2,12
```

Glossary of keywords and components

The atoms involved can be specified using

--start	a pdb file that contains the structure for the initial frame of your path. For more information on how to specify lists of atoms see Groups and Virtual Atoms
--end	a pdb file that contains the structure for the final frame of your path. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Or alternatively by using

--path	a pdb file that contains an initial path in which the frames are not equally spaced

Compulsory keywords

--fixed	(default=0) the frames to fix when constructing the path using –path
--metric	the measure to use to calculate the distance between frames
--out	the name of the file on which to output your path
--arg-fmt	(default=f) the format to use for argument values in your frames
--tolerance	(default=1E-4) the tolerance to use for the algorithm that is used to re-parameterize the path
--nframes-before-start	(default=1) the number of frames to include in the path before the first frame
--nframes	(default=1) the number of frames between the start and end frames in your path
--nframes-after-end	(default=1) the number of frames to put after the last frame of your path

Options

--help/-h	(default=off) print this help

9.16 pdbrenumber

This is part of the cltools module

Modify atom numbers in a PDB, possibly using hybrid-36 coding.
When reading a PDB files, PLUMED honors the serial number of each atom. This command can be used to process a PDB file changing the atom serial numbers. Notice that the resulting list might have gaps. It is however fundamental that atom numbers correspond to those used within the MD code. Importantly, if the serial number of an atom is greater than 99999, it is written in hybrid-36 notation (see pdbreader). The main use of pdbrenumber is thus that of producing files where atoms are numbered using hybrid-36 convention.
The output PDB file is identical to the input PDB file, except for the atom number field. The rest of the line is written unchanged to the output file, even if it is incorrectly formatted. Residue numbers are not touched, and atom numbers in the input file are ignored.

Examples

By default, pdbreader just sets the numbers as progressive starting from 1. For instance the following command:

```
> plumed pdbrenumber --ipdb input.pdb --opdb output.pdb
```

will copy file `input.pdb` to `output.pdb` replacing all the serial atoms with increasing numbers starting from one. Atoms that have an index that is greater than 99999 will be written in the output PDB file in hybrid-36 code.
It is possible to set a different serial number for the first atom, letting the following ones grow by one at each line. Here for instance the first atom will be assigned serial 1000, the second serial 1001, etc:

```
> plumed pdbrenumber --ipdb input.pdb --opdb output.pdb --firstatomnumber 1000
```

If the first atom number is >99999, it should be given as a decimal number (not in hybrid-36 code). However, numbers >99999 in the output PDB file will be written in hybrid-36 code.
As an alternative, one can provide a list of atoms as one per line in an auxiliary file.

```
> plumed pdbrenumber --ipdb input.pdb --opdb output.pdb --atomnumbers list.txt
```

The `list.txt` file might be something like this

```
120000
120001
120002
1
2
3
```

Numbers >99999 in the list should be provided as decimal numbers (not in hybrid-36 code). However, numbers >99999 in the output PDB file will be written in hybrid-36 code. Notice that there should be at least enough lines in `list.txt` as many atoms in the PDB file. Additional lines in `list.txt` will just be ignored.

Glossary of keywords and components

Compulsory keywords

--ipdb	specify the name of the input PDB file
--opdb	specify the name of the output PDB file

Options

--help/-h	(default=off) print this help
--firstatomnumber	specify the desired serial number of the first atom of the output file
--atomnumbers	specify the desired serial numbers of the atoms of the output file using a separate list

9.17 pesmd

This is part of the cltools module

Pesmd allows one to do (biased) Langevin dynamics on a two-dimensional potential energy surface.
The energy landscape that you are moving about on is specified using a plumed input file. The directives that are available for this command line tool are as follows:

Examples

You run a Langevin simulation using pesmd with the following command:

```
plumed pesmd < input
```

The following is an example of an input file for a pesmd simulation. This file instructs pesmd to do 50 steps of Langevin dynamics on a 2D potential energy surface at a temperature of 0.722

```
temperature 0.722
tstep 0.005
friction 1
dimension 2
nstep 50
ipos 0.0 0.0
```

If you run the following a description of all the directives that can be used in the input file will be output.

```
plumed pesmd --help
```

The energy landscape to explore is given within the plumed input file. For example the following example input uses MATHEVAL to define a two dimensional potential.

```
d1: DISTANCE ATOMS=1,2 COMPONENTS
ff: MATHEVAL ARG=d1.x,d1,y PERIODIC=NO FUNC=()
bb: BIASVALUE ARG=ff
```

Atom 1 is placed at the origin. The x and y components on our surface are the positions of the particle on our two dimensional energy landscape. By calculating the vector connecting atom 1 (the origin) to atom 2 (the position of our particle) we are thus getting the position of the atom on the energy landscape. This is then inserted into the function that is calculated on the second line. The value of this function is then used as a bias.
We can also specify a potential on a grid and look at the dynamics on this function using pesmd. A plumed input for an example such as this one might look something like this:

```
d1: DISTANCE ATOMS=1,2 COMPONENTS
bb: EXTERNAL ARG=d1.x,d1,y FILE=fes.dat
```

In this way we can use pesmd to do a dynamics on a free energy surface calculated using metadynamics and sum_hills. On a final note once we have defined our potential we can use all the biasing functions within plumed in addition in order to do a biased dynamics on the potential energy landscape of interest.

Glossary of keywords and components

Compulsory keywords

nstep	The number of steps of dynamics you want to run
temperature	(default=NVE) the temperature at which you wish to run the simulation in LJ units
friction	(default=off) The friction (in LJ units) for the Langevin thermostat that is used to keep the temperature constant
tstep	(default=0.005) the integration timestep in LJ units
dimension	the dimension of your energy landscape
plumed	(default=plumed.dat) the name of the plumed input file containing the potential
ipos	(default=0.0) the initial position of the system
idum	(default=0) The random number seed

Options

periodic	(default=off) are your input coordinates periodic
min	minimum value the coordinates can take for a periodic domain
max	maximum value the coordinates can take for a periodic domain

9.18 selector

create lists of serial atom numbers

Note

This command line tool is implemented as a shell script. Its help message is pasted below:

```
Example:
echo "
mda:nucleic
mdt:resname RG
" | selector.sh --pdb ref.pdb
```

Glossary of keywords and components

9.19 simplemd

This is part of the cltools module

simplemd allows one to do molecular dynamics on systems of Lennard-Jones atoms.
The input to simplemd is specified in an input file. Configurations are input and output in xyz format. The input file should contain one directive per line. The directives available are as follows:

Examples

You run an MD simulation using simplemd with the following command:

```
plumed simplemd < in
```

The following is an example of an input file for a simplemd calculation. This file instructs simplemd to do 50 steps of MD at a temperature of 0.722

```
inputfile input.xyz
outputfile output.xyz
temperature 0.722
tstep 0.005
friction 1
forcecutoff 2.5
listcutoff  3.0
nstep 50
nconfig 10 trajectory.xyz
nstat   10 energies.dat
```

If you run the following a description of all the directives that can be used in the input file will be output.

```
plumed simplemd --help
```

Glossary of keywords and components

Compulsory keywords

nstep	The number of steps of dynamics you want to run
temperature	(default=NVE) the temperature at which you wish to run the simulation in LJ units
friction	(default=off) The friction (in LJ units) for the Langevin thermostat that is used to keep the temperature constant
tstep	(default=0.005) the integration timestep in LJ units
epsilon	(default=1.0) LJ parameter
sigma	(default=1.0) LJ parameter

inputfile	An xyz file containing the initial configuration of the system
forcecutoff	(default=2.5)
listcutoff	(default=3.0)
outputfile	An output xyz file containing the final configuration of the system
nconfig	(default=10) The frequency with which to write configurations to the trajectory file followed by the name of the trajectory file
nstat	(default=1) The frequency with which to write the statistics to the statistics file followed by the name of the statistics file
idum	(default=0) The random number seed
ndim	(default=3) The dimensionality of the system (some interesting LJ clusters are two dimensional)
wrapatoms	(default=false) If true, atomic coordinates are written wrapped in minimal cell

9.20 sum_hills

This is part of the cltools module

sum_hills is a tool that allows one to to use plumed to post-process an existing hills/colvar file

Examples

a typical case is about the integration of a hills file:

```
plumed sum_hills  --hills PATHTOMYHILLSFILE
```

The default name for the output file will be fes.dat Note that starting from this version plumed will automatically detect the number of the variables you have and their periodicity. Additionally, if you use flexible hills (multivariate Gaussian kernels), plumed will understand it from the HILLS file.
The sum_hills tool will also accept multiple files that will be integrated one after the other

```
plumed sum_hills  --hills PATHTOMYHILLSFILE1,PATHTOMYHILLSFILE2,PATHTOMYHILLSFILE3
```

if you want to integrate out some variable you do

```
plumed sum_hills  --hills PATHTOMYHILLSFILE   --idw t1 --kt 0.6
```

where with –idw you define the variables that you want all the others will be integrated out. –kt defines the temperature of the system in energy units. (be consistent with the units you have in your hills: plumed will not check this for you) If you need more variables then you may use a comma separated syntax

```
plumed sum_hills  --hills PATHTOMYHILLSFILE   --idw t1,t2 --kt 0.6
```

You can define the output grid only with the number of bins you want while min/max will be detected for you

```
plumed sum_hills --bin 99,99 --hills PATHTOMYHILLSFILE
```

or full grid specification

```
plumed sum_hills --bin 99,99 --min -pi,-pi --max pi,pi --hills PATHTOMYHILLSFILE
```

You can of course use numbers instead of -pi/pi.
You can use a –stride keyword to have a dump each bunch of hills you read

```
plumed sum_hills --stride 300 --hills PATHTOMYHILLSFILE
```

You can also have, in case of well tempered metadynamics, only the negative bias instead of the free energy through the keyword –negbias

```
plumed sum_hills --negbias --hills PATHTOMYHILLSFILE
```

Here the default name will be negativebias.dat
From time to time you might need to use HILLS or a COLVAR file as it was just a simple set of points from which you want to build a free energy by using -(1/beta)log(P) then you use –histo

```
plumed sum_hills --histo PATHTOMYCOLVARORHILLSFILE  --sigma 0.2,0.2 --kt 0.6
```

in this case you need a –kt to do the reweighting and then you need also some width (with the –sigma keyword) for the histogram calculation (actually will be done with Gaussian kernels, so it will be a continuous histogram) Here the default output will be histo.dat. Note that also here you can have multiple input files separated by a comma.
Additionally, if you want to do histogram and hills from the same file you can do as this

```
plumed sum_hills --hills --histo PATHTOMYCOLVARORHILLSFILE  --sigma 0.2,0.2 --kt 0.6
```

The two files can be eventually the same
Another interesting thing one can do is monitor the difference in blocks as a metadynamics goes on. When the bias deposited is constant over the whole domain one can consider to be at convergence. This can be done with the –nohistory keyword

```
plumed sum_hills --stride 300 --hills PATHTOMYHILLSFILE  --nohistory
```

and similarly one can do the same for an histogram file

```
plumed sum_hills --histo PATHTOMYCOLVARORHILLSFILE  --sigma 0.2,0.2 --kt 0.6 --nohistory
```

just to check the hypothetical free energy calculated in single blocks of time during a simulation and not in a cumulative way
Output format can be controlled via the –fmt field

```
plumed sum_hills --hills PATHTOMYHILLSFILE  --fmt %8.3f
```

where here we chose a float with length of 8 and 3 digits
The output can be named in a arbitrary way :

```
plumed sum_hills --hills PATHTOMYHILLSFILE  --outfile myfes.dat
```

will produce a file myfes.dat which contains the free energy.
If you use stride, this keyword is the suffix

```
plumed sum_hills --hills PATHTOMYHILLSFILE  --outfile myfes_ --stride 100
```

will produce myfes_0.dat, myfes_1.dat, myfes_2.dat etc.
The same is true for the output coming from histogram

```
plumed sum_hills --histo HILLS --kt 2.5 --sigma 0.01 --outhisto myhisto.dat
```

is producing a file myhisto.dat while, when using stride, this is the suffix

```
plumed sum_hills --histo HILLS --kt 2.5 --sigma 0.01 --outhisto myhisto_ --stride 100
```

that gives myhisto_0.dat, myhisto_1.dat, myhisto_3.dat etc..

Glossary of keywords and components

Options

--help/-h	(default=off) print this help
--help-debug	(default=off) print special options that can be used to create regtests
--negbias	(default=off) print the negative bias instead of the free energy (only needed with well tempered runs and flexible hills)
--nohistory	(default=off) to be used with –stride: it splits the bias/histogram in pieces without previous history

--mintozero	(default=off) it translate all the minimum value in bias/histogram to zero (useful to compare results)
--hills	specify the name of the hills file
--histo	specify the name of the file for histogram a colvar/hills file is good
--stride	specify the stride for integrating hills file (default 0=never)
--min	the lower bounds for the grid
--max	the upper bounds for the grid
--bin	the number of bins for the grid
--spacing	grid spacing, alternative to the number of bins
--idw	specify the variables to be used for the free-energy/histogram (default is all). With –hills the other variables will be integrated out, with –histo the other variables won't be considered
--outfile	specify the output file for sumhills
--outhisto	specify the output file for the histogram
--kt	specify temperature in energy units for integrating out variables
--sigma	a vector that specify the sigma for binning (only needed when doing histogram
--fmt	specify the output format

9.21 vim2html

convert plumed input file to colored html using vim syntax

Note

This command line tool is implemented as a shell script. Its help message is pasted below:

```
Usage:
  plumed vim2html [options] < input > output
  plumed vim2html [options] input > output
  plumed vim2html [options] input output
(one of the three)

Options can be:
--annotate-syntax Reports annotated syntax on output.
                  Also reports non-annotated regions on stderr
--pdf             To produce a pdf file with syntax highlighting.
--crop            Crop the pdf file to the only written part. Useful to insert the pdf in a LaTex file as imag
--fs              Specify the fontsize of the pdf output.
--colors          Specify the color palette. Allowed values are: default/ac
```

Glossary of keywords and components

Chapter 10

Miscellaneous

10.1 Comments

If you are an organized sort of person who likes to remember what the hell you were trying to do when you ran a particular simulation you might find it useful to put comments in your input file. In PLUMED you can do this as comments can be added using a # sign. On any given line everything after the # sign is ignored so erm... yes add lines of comments or trailing comments to your hearts content as shown below (using Shakespeare is optional):

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
# This is the distance between two atoms:
d1: DISTANCE ATOMS=1,2
UPPER_WALLS ARG=d1 AT=3.0 KAPPA=3.0 LABEL=Snout # In this same interlude it doth befall.
# That I, one Snout by name, present a wall.
```

(see DISTANCE and UPPER_WALLS)
An alternative to including comments in this way is to use the command ENDPLUMED. Everything in the PLUMED input after this keyword will be ignored.

10.1.1 ENDPLUMED

Terminate plumed input.
Can be used to effectively comment out the rest of the input file. It can be useful to quickly ignore part of a long input file. However, one should keep in mind that when opening the file it might be difficult to find where the commented out part begins. Regular comments (with #) are usually easier to read. Notice that VIM syntax should be able to detect this command and properly mark the rest of the file as a comment, although since vim doesn't parse the whole file it might fail in doing so for long input files.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/ENDPLUMED.tmp
d: DISTANCE ATOMS=1,10
PRINT ARG=d FILE=COLVAR STRIDE=10
ENDPLUMED
commands here are ignored
PRINT ARG=d FILE=COLVAR STRIDE=1
```

Glossary of keywords and components

10.2 Continuation lines

If your input lines get very long then editing them using vi and other such text editors becomes a massive pain in the arse.
We at PLUMED are aware of this fact and thus have provided a way of doing line continuations so as to make your life that much easier - aren't we kind? Well no not really, we have to use this code too. Anyway, you can do continuations by using the "..." syntax as this makes this:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
DISTANCES ATOMS1=1,300 ATOMS2=1,400 ATOMS3=1,500 LABEL=dist
```

(see DISTANCES)
equivalent to this:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
DISTANCES ...
  LABEL=dist
# we can also insert comments here
  ATOMS1=1,300
# multiple kewords per line are allowed
  ATOMS2=1,400 ATOMS3=1,500
#empty lines are also allowed

... DISTANCES
```

Notice that the closing `...` is followed by the word `DISTANCES`. This is optional, but might be useful to find more easily which is the matching start of the statement. The following is equally correct

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
dist: DISTANCES ...
# we can also insert comments here
  ATOMS1=1,300
# multiple kewords per line are allowed
  ATOMS2=1,400 ATOMS3=1,500
#empty lines are also allowed

...
```

Notice that PLUMED makes a check that the word following the closing `...` is actually identical to the first word in the line with the first `...`. If not, it will throw an error. Also notice that you might put more than one word in the first line. E.g.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
DISTANCES LABEL=dist ...
# we can also insert comments here
```

```
  ATOMS1=1,300
# multiple kewords per line are allowed
  ATOMS2=1,400 ATOMS3=1,500
#empty lines are also allowed
...
```

or, equivalently,

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
dist: DISTANCES ...
# we can also insert comments here
  ATOMS1=1,300
# multiple kewords per line are allowed
  ATOMS2=1,400 ATOMS3=1,500
#empty lines are also allowed
...
```

10.3 Using VIM syntax file

For the impatient use:

- Add the following to your .vimrc file:

  ```
  " Enable syntax
  :syntax on
  " This allows including the proper PLUMED syntax file:
  :let &runtimepath.=',.$PLUMED_VIMPATH
  " The former command requires PLUMED_VIMPATH to be set. Alternatively, use this:
  " let &runtimepath.=',/usr/local/lib/plumed/vim'
  " properly adjusted to the path where PLUMED is installed.
  " This makes autocompletion work in the expected way:
  :set completeopt=longest,menuone
  " This enables bindings of F2/F3/F4 to plumed specific commands:
  :let plumed_shortcuts=1
  ```

- When you open a PLUMED input file, you can enable syntax highlighting with:

  ```
  :set ft=plumed
  ```

  This will also enable autocompletion. Use <CTRL-X><CTRL-O> to autocomplete a word.

- If you want to fold multi-line statements, type

  ```
  :setlocal foldmethod=syntax
  ```

- While editing a plumed input file, you can use command :PHelp (or shortcut <F2>) to show in a split window a short help about the action defined in the line where the cursor is. Typing :PHelp again (or pushing <F2>) you will close that window. With <CTRL-W><CTRL-W> you go back and forth between the two windows.

- When you open a file starting with #! FIELDS, VIM will automatically understand it is a PLUMED output file (VIM filetype = plumedf) and will color fields and data columns with alternating colors. Typing :PPlus and :PMinus (or pushing <F3> and <F4>) you can move a highlighted column.

See below for more detailed instructions.

Configuration

When PLUMED is compiled, directories help and syntax will appear in builddir/vim. They contain a VIM plugin that can be used to highlight proper PLUMED instructions in a PLUMED input file and to quickly retrieve help. There is also a file builddir/vim/scripts.vim that helps VIM in recognizing PLUMED output files.

Warning

Notice that these file do not appear if you are cross compiling. In this case, you must copy the plugin files from another machine.

To make VIM aware of these files, you should copy them to your `$HOME/.vim` directory. Later you can enable plumed syntax with the command

```
:set ft=plumed
```

If you work in an environment where several PLUMED versions are installed (e.g. using env modules), we recommend the following procedure:

- Install PLUMED
- Add to your `.vimrc` file the following line:

```
:let &runtimepath.=',.$PLUMED_VIMPATH
```

The modulefile provided with PLUMED should set the PLUMED_VIMPATH environment variable to the proper path. Thus, when working with a given PLUMED module loaded, you should be able to enable to proper syntax by just typing

```
:set ft=plumed
```

in VIM. Notice that the variable `PLUMED_VIMPATH` is also set in the `sourceme.sh` script in the build directory. Thus, if you modify your `.vimrc` file as suggested, you will be able to use the correct syntax both when using an installed PLUMED and when running from a just compiled copy. Finally, in case you have both a pre-installed PLUMED **and** you have your development version the following command would give you the optimal flexibility:

```
:let &runtimepath.=',.$PLUMED_VIMPATH.',/opt/local/lib/plumed/vim/'
```

The environment variable `PLUMED_VIMPATH`, if set, will take the precedence. Otherwise, vim will resort to the hard coded path. In this case we assumed that there is a PLUMED installed in `/opt/local/` (e.g. using Mac↩ Ports), but you can override it sourcing a `sourceme.sh` file in the compilation directory or loading a PLUMED module with `module load plumed`.
If you are tired of typing `:set ft=plumed`, you can use a modeline. Add to your `.vimrc` file the following commands

```
:set modeline
:set modelines=5
```

Then, at the beginning of your PLUMED input file, put the following comment:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
# vim:ft=plumed
d: DISTANCE ATOMS=1,2
RESTRAINT ARG=d AT=0.0 KAPPA=1.0
```

Now, every time you open this file, you will see it highlighted.

Syntax highlighting

The syntax file contains a definition of all possible PLUMED actions and keywords. It is designed to allow for a quick validation of the PLUMED input file before running it. As such, all the meaningful words in the input should be highlighted:

- Valid action names (such as `METAD`) and labels (such as `m:` or `LABEL=m`) will be highlighted in the brightest way (`Type` in VIM). Those are the most important words.
- Keyword and flag names (such as `ATOMS=` or `COMPONENTS` when part of the action DISTANCE) will be highlighted with a different color (`Statement` in VIM).
- Values provided by users (such as the number of the atoms following `ATOMS=`) will be highlighted with a different color (`String` in VIM).
- Comments (see Comments) will be highlighted as comments (`Comment` in VIM).
- String `__FILL__` (extensively used in tutorials to indicate parts to be completed) is highlighted (`Todo` in VIM).

If you see something that is not highlighted and appears in black, this is likely going to result in an error at runtime. Think of this as a sort of preliminary spell-check. For this checks to be effective, we recommend to use a syntax file generated with exactly the same version of PLUMED that you are using. In case you find that parts of an input file that is valid are not highlighted, then please report it as a bug. On the contrary, you cannot expect the VIM syntax file to recognize all possible errors in a PLUMED input. Thus, a file for which the highlighting looks correct might still contain errors.

Multi-line folding

Notice that syntax highlighting also allow VIM to properly fold multi-line actions. Try to do the following:

- Open a PLUMED input file
- Enable PLUMED syntax

  ```
  :set ft=plumed
  ```

- Enable syntax-based folding

  ```
  :setlocal foldmethod=syntax
  ```

Now look at what happened to all the multi-line statements in PLUMED (i.e. those using Continuation lines). As you can see, they will be folded into single lines. Folded lines can be expanded with `zo` and folded with `zc`. Look at VIM documentation to learn more. In case you want to use this feature, we suggest you to put both label and action type on the first line of multi-line statements. E.g.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
d: DISTANCE ATOMS=1,2

m: METAD ...
  ARG=d
  HEIGHT=1.0
  SIGMA=0.5
  PACE=100
...
```

will be folded to

```
d: DISTANCE ATOMS=1,2

+--  6 lines: m: METAD ...-------------------------------------------------------
```

and

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
d: DISTANCE ATOMS=1,2

METAD LABEL=m ...
  ARG=d
  HEIGHT=1.0
  SIGMA=0.5
  PACE=100
...
```

will be folded to

```
d: DISTANCE ATOMS=1,2

+--  6 lines: METAD LABEL=m ...-------------------------------------------------
```

This will allow you to easily identify the folded lines by seeing the most important information, that is the action type (`METAD`) and its label (`m`). This feature is convenient if you want to browse files that contain a lot of actions defined on multiple lines.

Autocompletion

Another VIM feature that comes when you load PLUMED syntax is autocompletion of PLUMED actions and keywords. Open your favorite PLUMED input file and set it to PLUMED syntax highlighting with

```
:set ft=plumed
```

Now go into insert mode pressing `i` and type `DU` followed by `<CTRL+X><CTRL+O>`. Here `<CTRL+X>` stands for autocompletion and `<CTRL+O>` for omnifunc autocompletion. You will see a short menu listing the following actions

```
DUMPATOMS
DUMPDERIVATIVES
DUMPFORCES
DUMPMASSCHARGE
DUMPMULTICOLVAR
DUMPPROJECTIONS
```

That is, all the actions starting with `DU`. You can navigate it with up and down arrows so as to choose the best match. Notice that the default behavior of VIM is to use the first match by default. In the first example (`DU<CTRL+X><CTRL+O`), it would be `DUMPATOMS`. The following settings make it work as most of the people expect:

```
:set completeopt=longest,menuone
```

With these settings, in the first example (`DU<CTRL+X><CTRL+O`) VIM will only complete up to the longest common part (`DUMP`).
As you can imagine, if you use autocompletion after you have typed the word `DISTANCE` followed by a space you will see a menu listing `LABEL=`, `COMPONENTS`, etc. Basically, all the keywords that are possibly used within a `DISTANCE` line will be shown. This is very useful if you do not remember the exact name of the keywords associated with a given action.

Quick help

You can also retrieve quick explanation of the input options for a specific action. Try to do the following. Enable plumed syntax:

```
:set ft=plumed
```

Then add the following line

```
DISTANCE
```

Now, in normal mode, go with the cursor on the `DISTANCE` line and type

```
:PHelp
```

A new split window should appear containing some documentation about the DISTANCE collective variable. You can go back and forth between the two windows with `<CTRL+W><CTRL+W>`, as usually in vim. Notice that if you are in the help window and type `:PHelp` this window will be closed.
To make the navigation easier, you can add a shortcut in your .vimrc file. For example, adding:

```
: nmap <F2> : PHelp<CR>
```

you should be able to open and close the manual hitting the F2 key. This is done automatically in the PLUMED syntax file if you add `let plumed_shortcuts=1` to your vimrc file.

Displaying output files

Most of the PLUMED output files look like this

```
#! FIELDS A B C
1 2 3
```

This is useful since in the header you can see the name of the quantities that are printed in the data lines. However, when you have an output file with many columns it might be a bit error prone to count them. To simplify this, when PLUMED syntax for VIM is configured properly VIM should be able to:

- Detect that this file is a PLUMED output file with fields, automatically setting its type to `plumedf`. If not, just type `:set ft=plumedf`.
- Show this file with syntax highlighting to increase its readability.

Notice that the syntax file for the output files (`plumedf.vim`) is not the same one that is used for the PLUMED input file (`plumed.vim`).
To make output files more readable, vim will show `FIELDS` and `SET` words in a different color, and data columns with alternating colors (e.g. dark/light/dark/light). The colors in the columns are consistent with those shown in the FIELD line. In the example above, 1, 2, and 3 will be of the same color as A, B, and C respectively. This should make it much easier to find which columns correspond to a given quantity.
It is also possible to highlight a specific field of the file. Typing

```
:5PCol
```

you will highlight the fifth field. Notice that in the `FIELDS` line (the first line of the file) the seventh word of the line will be highlighted, which is the one containing the name of the field. This allows for easy matching of values shown in the file and tags provided in the `FIELDS` line. The highlighted column can be moved back and forth using `:PPlus` and `:PMinus`. Adding a count to the command will move the highlighted column more. E.g. `:2PPlus` will move the column to the right twice.
If you have a long output file, it might be convenient to split it with `:split` so that one of the two windows will only show the header. The other window can be used to navigate the file.
To make the navigation easier, you can add a shortcut in your .vimrc file. For example, adding:

```
: map <F3> :PMinus<CR>
: map <F4> :PPlus<CR>
```

you should be able to move the highlight column using F3 and F4 buttons. This is done automatically in the PLUMED syntax file if you add `let plumed_shortcuts=1` to your vimrc file.

10.4 Using bash autocompletion

When possible, PLUMED tries to install bash autocompletion so that you do not have to do anything. Just use the <TAB> key to complete plumed commands (e.g. `plumed dr<TAB>`) or even options (e.g. `plumed driver --i<TAB>`). In case this does not work, you might have to add the following lines to your .bashrc file:

```
_plumed() { eval "$(plumed --no-mpi completion 2>/dev/null)";}
complete -F _plumed -o default plumed
```

Effect

When typing on the the shell you should observe the following behavior.

```
> plumed <TAB>
```

will autocomplete with the names of the available PLUMED commands (e.g. `driver`, `help`, etc).

```
> plumed -<TAB>
```

will autocomplete with the available PLUMED options (e.g. `--no-mpi`, etc).
PLUMED also knows which are the options available for each command (e.g. `plumed driver --natoms`). So, the following

```
> plumed driver -<TAB>
```

(notice the −) will autocomplete to the options of `plumed driver`. On the contrary

```
> plumed driver --ixtc <TAB>
```

(notice the there is no − before <TAB>) will autocomplete to the files in the current directory.
Also notice that every time you use the <TAB> key to autocomplete the command `plumed` will be invoked. This should allow the correct commands and options to be reported depending on the exact `plumed` command in the current execution path. For instance, if you have multiple PLUMED versions installed with env modules, you should be able to see the commands available in the currently loaded version. Clearly, this feature will only be available if `plumed` can run on this machine (that is: will not work if you are cross compiling). This is not a problem since you are not expecting to run the `plumed` command in this specific case.

Technicalities

At configure time if the variable `BASH_COMPLETION_DIR` is defined it will be used to decide where PLUMED autocompletion should be installed. Otherwise, configure will look for the presence of the `bash-completion` package and, in case it is installed on the same prefix as PLUMED, also PLUMED autocompletion will be installed. Finally, if none of these two conditions are satisfied, autocompletion will not be enabled. You will have to change your bashrc file once adding the following lines:

```
_plumed() { eval "$(plumed --no-mpi completion 2>/dev/null)";}
complete -F _plumed -o default plumed
```

The command `plumed completion` just writes on its standard output the body of a bash function that is then used by bash to construct the autocompletion. The `--no-mpi` flag makes it more likely that the command can be executed correctly e.g. when you are on the login node of a cluster and PLUMED was compiled with MPI but the login node does not support MPI. In other cases, it is harmless. The `-o default` options will make sure that if `plumed --no-mpi completion` returns an error the default bash completion will be used. This is what will happen if you load an older PLUMED version for which the `completion` command is not available yet. In future PLUMED versions the `plumed completion` command might return more sophisticated functions. You should be able to benefit of these features without ever changing your bash configuration file again.

Multiple versions and suffixes

In case you have multiple versions of PLUMED installed in separate env modules there is nothing more to do. However, if you have have multiple versions of PLUMED installed with different suffixes you should consistently add more lines to your profile file. For instance, if you installed two executables named `plumed` and `plumed_mpi` your configuration file should look like:

```
_plumed() { eval "$(plumed --no-mpi completion 2>/dev/null)";}
complete -F _plumed -o default plumed
_plumed_mpi() { eval "$(plumed_mpi --no-mpi completion 2>/dev/null)";}
complete -F _plumed_mpi -o default plumed_mpi
```

10.5 Including other files

If, for some reason, you want to spread your PLUMED input over a number of files you can use INCLUDE as shown below:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/MiscelaneousPP.md
INCLUDE FILE=filename
```

So, for example, a single "plumed.dat" file:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
DISTANCE ATOMS=1,2 LABEL=dist
RESTRAINT ARG=dist AT=2.0 KAPPA=1.0
```

could be split up into two files as shown below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
DISTANCE ATOMS=1,2 LABEL=dist
INCLUDE FILE=toBeIncluded.inc
```

plus a "toBeIncluded.inc" file

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/MiscelaneousPP.md
#SETTINGS FILENAME=toBeIncluded.inc
# this is toBeIncluded.inc
RESTRAINT ARG=dist AT=2.0 KAPPA=1.0
```

However, when you do this it is important to recognize that INCLUDE is a real directive that is only resolved after all the Comments have been stripped and the Continuation lines have been unrolled. This means it is not possible to do things like:

```
BEGIN_PLUMED_FILE
# this is wrong:
DISTANCE INCLUDE FILE=options.dat
RESTRAINT ARG=dist AT=2.0 KAPPA=1.0
```

10.5.1 INCLUDE

This is part of the generic module

Includes an external input file, similar to #include in C preprocessor.
Useful to split very large plumed.dat files. Notice that in PLUMED 2.4 this action cannot be used before the initial setup part of the file (e.g. in the part with UNITS, MOLINFO, etc). As of PLUMED 2.5, INCLUDE can be used in any position of the file.

Examples

This input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
c1: COM ATOMS=1-100
c2: COM ATOMS=101-202
d: DISTANCE ATOMS=c1,c2
PRINT ARG=d
```

can be replaced with this input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
INCLUDE FILE=pippo.dat
d: DISTANCE ATOMS=c1,c2
PRINT ARG=d
```

where the content of file pippo.dat is

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
#SETTINGS FILENAME=pippo.dat
# this is pippo.dat
c1: COM ATOMS=1-100
c2: COM ATOMS=101-202
```

The files in this example are rather short, but imagine a case like this one:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
INCLUDE FILE=groups.dat
c: COORDINATION GROUPA=groupa GROUPB=groupb R_0=0.5
METAD ARG=c HEIGHT=0.2 PACE=100 SIGMA=0.2 BIASFACTOR=5
```

Here `groups.dat` could be a huge file containing group definitions. This groups.dat file might look something like the following example but with more atom indices in the groups.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
#SETTINGS FILENAME=groups.dat
# this is groups.dat
groupa: GROUP ...
  ATOMS={
    10
    50
    60
    70
    80
    120
  }
...
groupb: GROUP ...
  ATOMS={
    11
    51
    61
    71
    81
    121
  }
...
```

So, included files are the best place where one can store long definitions.
Another case where INCLUDE is very useful is when running multi-replica simulations. Here different replicas might have different input files, but perhaps a large part of the input is shared. This part can be put in a common included file. For instance you could have `common.dat`:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
#SETTINGS FILENAME=common.dat
# this is common.dat
t: TORSION ATOMS=1,2,3,4
```

Then `plumed.0.dat`:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
# this is plumed.0.dat
INCLUDE FILE=common.dat
RESTRAINT ARG=t AT=1.0 KAPPA=10
```

And `plumed.1.dat`:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
# this is plumed.1.dat
INCLUDE FILE=common.dat
RESTRAINT ARG=t AT=1.2 KAPPA=10
```

Warning

> Remember that when using multi replica simulations whenever plumed tried to open a file for reading it looks for a file with the replica suffix first. This is true also for files opened by INCLUDE!

As an example, the same result of the inputs above could have been obtained using the following `plumed.dat` file:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/INCLUDE.tmp
#SETTINGS NREPLICAS=2
t: TORSION ATOMS=1,2,3,4
INCLUDE FILE=other.inc
```

Then `other.0.inc`:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/INCLUDE.tmp
#SETTINGS FILENAME=other.0.inc
# this is other.0.inc
RESTRAINT ARG=t AT=1.0 KAPPA=10
```

And `other.1.inc`:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/INCLUDE.tmp
#SETTINGS FILENAME=other.1.inc
# this is other.1.inc
RESTRAINT ARG=t AT=1.2 KAPPA=10
```

Glossary of keywords and components

Compulsory keywords

FILE	file to be included

10.6 Loading shared libraries

You can introduce new functionality into PLUMED by placing it directly into the src directory and recompiling the PLUMED libraries. Alternatively, if you want to keep your code independent from the rest of PLUMED (perhaps so you can release it independently - we won't be offended), then you can create your own dynamic library. To use this in conjunction with PLUMED you can then load it at runtime by using the LOAD keyword as shown below:

```
BEGIN_PLUMED_FILE loads DATADIR=example-check/MiscelaneousPP.md
LOAD FILE=library.so
```

N.B. If your system uses a different suffix for dynamic libraries (e.g. macs use .dylib) then PLUMED will try to automatically adjust the suffix accordingly.

10.6.1 LOAD

	This is part of the setup module

Loads a library, possibly defining new actions.
It is available only on systems allowing for dynamic loading. It can also be fed with a cpp file, in which case the file is compiled first.

Examples

If you have a shared object named extensions.so and want to use the functions implemented within it within PLUMED you can load it with the following syntax

```
BEGIN_PLUMED_FILE loads DATADIR=example-check/LOAD.tmp
LOAD FILE=extensions.so
```

As a more practical example, imagine that you want to make a small change to one collective variable that is already implemented in PLUMED, say DISTANCE . Copy the file `src/colvar/Distance.cpp` into your work directory, rename it as `Distance2.cpp` and edit it as you wish. It might be better to also replace any occurrence of the string DISTANCE within the file with DISTANCE2, so that both old and new implementation will be available with different names. Then you can compile it into a shared object using

```
> plumed mklib Distance2.cpp
```

This will generate a file `Distance2.so` (or `Distance2.dylib` on a mac) that can be loaded. Now you can use your new implementation with the following input

```
BEGIN_PLUMED_FILE loads DATADIR=example-check/LOAD.tmp
# load the new library
LOAD FILE=Distance2.so
# compute standard distance
d: DISTANCE ATOMS=1,10
# compute modified distance
d2: DISTANCE2 ATOMS=1,10
# print them on a file
PRINT ARG=d,d2 FILE=compare-them
```

You can even skip the initial step and directly feed PLUMED with the `Distance2.cpp` file: it will be compiled on the fly.

```
BEGIN_PLUMED_FILE loads DATADIR=example-check/LOAD.tmp
# load the new definition
# this is a cpp file so it will be compiled
LOAD FILE=Distance2.cpp
# compute standard distance
d: DISTANCE ATOMS=1,10
# compute modified distance
d2: DISTANCE2 ATOMS=1,10
# print them on a file
PRINT ARG=d,d2 FILE=compare-them
```

This will allow to make quick tests while developing your own variables. Of course, after your implementation is ready you might want to add it to the PLUMED source tree and recompile the whole PLUMED.

Glossary of keywords and components

Compulsory keywords

FILE	file to be loaded

10.7 Embed a separate PLUMED instance

PLUMED

10.7.1 PLUMED

This is part of the generic module

Embed a separate PLUMED instance.
This command can be used to embed a separate PLUMED instance. Only required atoms will be passed to that instance, using an interface that is similar to the one used when calling PLUMED from the NAMD engine.
Notice that the two instances are running in the same UNIX process, so that they cannot be perfectly isolated. However, most of the features are expected to work correctly.
Notes:

- The LOAD action will not work correctly since registers will be shared among the two instances. In particular, the loaded actions will be visible to both guest and host irrespective of where they are loaded from. This can be fixed and will probably be fixed in a later version.
- `CHDIR` is not thread safe. However, in most implementations there will be a single process running PLUMED, with perhaps multiple OpenMP threads spawn in order to parallelize the calculation of individual variables. So, this is likely not a problem.
- MPI is working correctly. However, notice that the guest PLUMED will always run with a single process. Multiple replicas should be handled correctly.

As an advanced feature, one can use the option `KERNEL` to select the version of the guest PLUMED instance. In particular, an empty `KERNEL` (default) implies that the guest PLUMED instance is the same as the host one (no library is loaded). On the other hand, `KERNEL=/path/to/libplumedKernel.so` will allow specifying a library to be loaded for the guest instance. In addition to those mentioned above, this feature has limitations mostly related to clashes in the symbols defined in the different instances of the PLUMED library:

- On OSX, if you load a KERNEL with version >=2.5 there should be no problem thanks to the use of two-level namespaces.
- On OSX, if you load a KERNEL with version <=2.4 there should be clashes in symbol resolution. The only possible workarounds are:
  - If you are are using PLUMED with an MD code, it should be patched with `--runtime` and you should `export PLUMED_LOAD_NAMESPACE=LOCAL` before starting the MD engine.
  - If you are using PLUMED driver, you should launch the `plumed-runtime` executable (contained in the `prefix/lib/plumed/` directory), export `PLUMED_KERNEL` equal to the path of the host kernel library (as usual in runtime loading) and `export PLUMED_LOAD_NAMESPACE=LOCAL` before launching `plumed-runtime driver`.
- On Linux, any `KERNEL` should in principle work correctly. To achieve namespace separation we are loading the guest kernel with `RTLD_DEEPBIND`. However, this might create difficult to track problems in other linked libraries.
- On Unix systems where `RTLD_DEEPBIND` is not available kernels will not load correctly.
- In general, there might be unexpected crashes. Particularly difficult are situations where different kernels were compiled with different libraries.

A possible solution for the symbol clashes (not tested) could be to recompile the alternative PLUMED versions using separate C++ namespaces (e.g. `./configure CPPFLAGS=-DPLMD=PLMD_2_3`).

Todo
- Add support for multiple time stepping (`STRIDE` different from 1).
- Add the possibility to import CVs calculated in the host PLUMED instance into the guest PLUMED instance. Will be possible after #83 will be closed.
- Add the possibility to export CVs calculated in the guest PLUMED instance into the host PLUMED instance. Could be implemented using the `DataFetchingObject` class.

Examples

Here an example plumed file:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PLUMED.tmp
# plumed.dat
p: PLUMED FILE=plumed2.dat
PRINT ARG=p.bias FILE=COLVAR
```

`plumed2.dat` can be an arbitrary plumed input file, for instance

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PLUMED.tmp
#SETTINGS FILENAME=plumed2.dat
# plumed2.dat
d: DISTANCE ATOMS=1,10
RESTRAINT ARG=d KAPPA=10 AT=2
```

Now a more useful example. Imagine that you ran simulations using two different PLUMED input files. The files are long and complex and there are some clashes in the name of the variables (that is: same names are used in both files, same files are written, etc). In addition, files might have been written using different units (see UNITS`). If you want to run a single simulation with a bias potential that is the sum of the two bias potentials, you can:

- Place the two input files, as well as all the files required by plumed, in separate directories `directory1` and `directory2`.
- Run with the following input file in the parent directory:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/PLUMED.tmp
# plumed.dat
PLUMED FILE=plumed.dat CHDIR=directory1
PLUMED FILE=plumed.dat CHDIR=directory2
```

Glossary of keywords and components

Description of components

Quantity	Description
bias	the instantaneous value of the bias potential

Compulsory keywords

STRIDE	(default=1) stride different from 1 are not supported yet

Options

NOREPLICAS	(default=off) run multiple replicas as isolated ones, without letting them know that the host has multiple replicas
FILE	input file for the guest PLUMED instance
KERNEL	kernel to be used for the guest PLUMED instance (USE WITH CAUTION!)
LOG	log file for the guest PLUMED instance. By default the host log is used
CHDIR	run guest in a separate directory

10.8 Debugging the code

The DEBUG action provides some functionality for debugging the code that may be useful if you are doing very intensive development of the code of if you are running on a computer with a strange architecture.

10.8.1 DEBUG

	This is part of the generic module

Set some debug options.
Can be used while debugging or optimizing plumed.

Examples

```
BEGIN_PLUMED_FILE working DATADIR=example-check/DEBUG.tmp
# print detailed (action-by-action) timers at the end of simulation
DEBUG DETAILED_TIMERS
# dump every two steps which are the atoms required from the MD code
DEBUG logRequestedAtoms STRIDE=2
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) the frequency with which this action is to be performed

Options

logActivity	(default=off) write in the log which actions are inactive and which are inactive
logRequestedAtoms	(default=off) write in the log which atoms have been requested at a given time
NOVIRIAL	(default=off) switch off the virial contribution for the entirety of the simulation
DETAILED_TIMERS	(default=off) switch on detailed timers
FILE	the name of the file on which to output these quantities

10.9 Changing exchange patterns in replica exchange

Using the RANDOM_EXCHANGES keyword it is possible to make exchanges between randomly chosen replicas. This is useful e.g. for bias exchange metadynamics [132].

10.9.1 RANDOM_EXCHANGES

	This is part of the generic module

Set random pattern for exchanges.
In this way, exchanges will not be done between replicas with consecutive index, but will be done using a random pattern. Typically used in bias exchange [132].

Examples

Using the following three input files one can run a bias exchange metadynamics simulation using a different angle in each replica. Exchanges will be randomly tried between replicas 0-1, 0-2 and 1-2
Here is plumed.0.dat

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RANDOM_EXCHANGES.tmp
RANDOM_EXCHANGES
t: TORSION ATOMS=1,2,3,4
METAD ARG=t HEIGHT=0.1 PACE=100 SIGMA=0.3
```

Here is plumed.1.dat

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RANDOM_EXCHANGES.tmp
RANDOM_EXCHANGES
t: TORSION ATOMS=2,3,4,5
METAD ARG=t HEIGHT=0.1 PACE=100 SIGMA=0.3
```

Here is plumed.2.dat

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RANDOM_EXCHANGES.tmp
RANDOM_EXCHANGES
t: TORSION ATOMS=3,4,5,6
METAD ARG=t HEIGHT=0.1 PACE=100 SIGMA=0.3
```

Warning

Multi replica simulations are presently only working with gromacs.

The directive should appear in input files for every replicas. In case SEED is specified, it should be the same in all input files.

Glossary of keywords and components

Options

SEED	seed for random exchanges

10.10 List of modules

The functionality in PLUMED 2 is divided into a small number of modules. Some users may only wish to use a subset of the functionality available within the code while others may wish to use some of the more complicated features that are available. For this reason the plumed source code is divided into modules, which users can activate or deactivate to their hearts content.

You can activate a module at configure time using the keyword `--enable-modules`. For example:

```
./configure --enable-modules=modulename
```

will enable module called modulename. A module that is on by default can be disabled using the following syntax

```
./configure --enable-modules=-modulename
```

To enable or disable multiple modules one should provide them as a : separated list. Notice that `+modulename` and `modulename` both activate the module, whereas `-modulename` deactivates it. E.g.

```
./configure --enable-modules=+crystallization:-colvar
```

will disable the colvar module and enable the crystallization module. Also notice that `:` can be omitted when using + or −. Thus, the same can be obtained with

```
./configure --enable-modules=+crystallization-colvar
```

If you repeat the `--enable-modules` keyword only the last instance will be used. Thus `./configure --enable-modules=crystallization --enable-modules=-colvar` will *not* do what you expect! There are also some shortcuts available:

- `./configure --enable-modules=all` to enable all optional modules. This includes the maximum number of features in PLUMED, including modules that might not be properly functional.
- `./configure --enable-modules=none` or `./configure --disable-modules` to disable all optional modules. This produces a minimal PLUMED which can be used as a library but which has no command line tools and no collective variables or biasing methods.
- `./configure --enable-modules=reset` or `./configure --enable-modules` to enable the default modules.

The two kinds of syntax can be combined and, for example, `./configure --enable-modules=none:colvar` will result in a PLUMED with all the modules disabled with the exception of the colvar module.
Some modules are active by default in the version of PLUMED 2 that you download from the website while others are inactive. The following lists all of the modules that are available in plumed and tells you whether or not they are active by default.

Module name	Default behavior
adjmat	off
analysis	on
annfunc	off
bias	on
cltools	on
colvar	on
crystallization	off
dimred	off
drr	off
eds	off
fisst	off
function	on
funnel	off
generic	on
gridtools	on
isdb	on
logmfd	off
manyrestraints	off
mapping	on
maze	off
membranefusion	off
molfile	on
multicolvar	on
opes	off
pamm	off
piv	off
pytorch	off
reference	on
s2cm	off
sasa	off
secondarystructure	on
setup	on
vatom	on
ves	off
vesselbase	on

xdrfile	on

Until PLUMED 2.2, it was also possible to switch on or off modules by adding files in the plumed2/src directory. Since PLUMED 2.3 this is discouraged, since any choice made in this manner will be overwritten next time `./configure` is used.

10.11 Special replica syntax

(this part of the manual is based on Using special syntax for multiple replicas).
In many cases, we need to run multiple replicas with almost identical PLUMED files. These files might be prepared with cut-and-paste, which is very error prone, or could be set up with some smart bash or python script. Additionally, one can take advantage of the INCLUDE keyword so as to have a shared input file with common definitions and specific input files with replica-dependent keywords. However, as of PLUMED 2.4, we introduced a simpler manner to manipulate multiple replica inputs with tiny differences. Look at the following example:

```
BEGIN_PLUMED_FILE  working DATADIR=example-check/MiscelaneousPP.md
#SETTINGS NREPLICAS=3
# Compute a distance
d: DISTANCE ATOMS=1,2

# Apply a restraint.
RESTRAINT ARG=d AT=@replicas:1.0,1.1,1.2 KAPPA=1.0
# On replica 0, this means:
#   RESTRAINT ARG=d AT=1.0 KAPPA=1.0
# On replica 1, this means:
#   RESTRAINT ARG=d AT=1.1 KAPPA=1.0
# On replica 2, this means:
#   RESTRAINT ARG=d AT=1.2 KAPPA=1.0
```

If you prepare a single `plumed.dat` file like this one and feeds it to PLUMED while using 3 replicas, the 3 replicas will see the very same input except for the `AT` keyword, that sets the position of the restraint. Replica 0 will see a restraint centered at 1.0, replica 1 centered at 1.1, and replica 2 centered at 1.2.
The `@replicas:` keyword is not special for RESTRAINT or for the `AT` keyword. Any keyword in PLUMED can accept that syntax. For instance, the following single input file can be used to setup a bias exchange metadynamics [132] simulations:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
#SETTINGS NREPLICAS=2
# Compute distance between atoms 1 and 2
d: DISTANCE ATOMS=1,2

# Compute a torsional angle
t: TORSION ATOMS=30,31,32,33

# Metadynamics.
METAD ...
  ARG=@replicas:d,t
  HEIGHT=1.0
  PACE=100
  SIGMA=@replicas:0.1,0.3
  GRID_MIN=@replicas:0.0,-pi
  GRID_MAX=@replicas:2.0,pi
...
# On replica 0, this means:
#  METAD ARG=d HEIGHT=1.0 PACE=100 SIGMA=0.1 GRID_MIN=0.0 GRID_MAX=2.0
# On replica 1, this means:
#  METAD ARG=t HEIGHT=1.0 PACE=100 SIGMA=0.3 GRID_MIN=-pi GRID_MAX=pi
```

This would be a typical setup for a bias exchange simulation. Notice that even though variables `d` and `t` are both read in both replicas, `d` is only computed on replica 0 (and `t` is only computed on replica 1). This is because variables that are defined but not used are never actually calculated by PLUMED.
If the value that should be provided for each replica is a vector, you should use curly braces as delimiters. For instance, if the restraint acts on two variables, you can use the following input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
#SETTINGS NREPLICAS=3
# Compute distance between atoms 1 and 2
```

```
d: DISTANCE ATOMS=10,20

# Compute a torsional angle
t: TORSION ATOMS=30,31,32,33

# Apply a restraint:
RESTRAINT ...
  ARG=d,t
  AT=@replicas:{{1.0,2.0} {3.0,4.0} {5.0,6.0}}
  KAPPA=1.0,3.0
...
# On replica 0 this means:
#  RESTRAINT ARG=d AT=1.0,2.0 KAPPA=1.0,3.0
# On replica 1 this means:
#  RESTRAINT ARG=d AT=3.0,4.0 KAPPA=1.0,3.0
# On replica 2 this means:
#  RESTRAINT ARG=d AT=5.0,6.0 KAPPA=1.0,3.0
```

Notice the double curly braces. The outer ones are used by PLUMED to know there the argument of the `AT` keyword ends, whereas the inner ones are used to group the values corresponding to each replica. Also notice that the last example can be split in multiple lines exploiting the fact that within multi-line statements (enclosed by pairs of `...`) newlines are replaced with simple spaces:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
#SETTINGS NREPLICAS=3
d: DISTANCE ATOMS=10,20
t: TORSION ATOMS=30,31,32,33
RESTRAINT ...
  ARG=d,t
# indentation is not required (this is not python!)
# but makes the input easier to read
  AT=@replicas:{
    {1.0,2.0}
    {3.0,4.0}
    {5.0,6.0}
  }
  KAPPA=1.0,3.0
...
```

In short, whenever there are keywords that should vary across replicas, you should set them using the `@replicas:` keyword. As mentioned above, you can always use the old syntax with separate input file, and this is recommended when the number of keywords that are different is large.

10.12 Parsing constants

You might have noticed that from time to time constants are specified using strings rather than numbers. An example is the following

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
#SETTINGS MOLFILE=regtest/basic/rt65/AA.pdb
MOLINFO STRUCTURE=AA.pdb  MOLTYPE=rna
e1: TORSION ATOMS=@epsilon-1
t: METAD ARG=e1 SIGMA=0.15 PACE=10 HEIGHT=2 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=200
```

Notice that the boundaries for `GRID_MIN` and `GRID_MAX` are `-pi` and `pi`. Until PLUMED 2.3, we used a very dummy parses that could recognize only `pi` as a special string, plus strings such as `0.5pi` and `-pi`. However, as of version 2.4, we use the Lepton library in order to parse every constant that we read. This means that you can also employ more complicated expressions such as `1+2` or `exp(10)`:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/MiscelaneousPP.md
#SETTINGS MOLFILE=regtest/basic/rt65/AA.pdb
MOLINFO STRUCTURE=AA.pdb  MOLTYPE=rna
e1: TORSION ATOMS=@epsilon-1
RESTRAINT ARG=e1 AT=1+0.5
```

Notice that this applies to any quantity read by plumed as a real number, but does not apply yet to integer numbers (e.g.: the PACE argument of METAD).

10.13 Frequently used tools

histogrambead	INTERNAL	A function that can be used to calculate whether quantities are between fixed upper and lower bounds.
kernelfunctions	INTERNAL	Functions that are used to construct histograms
pdbreader	INTERNAL	PLUMED can use the PDB format in several places
switchingfunction	INTERNAL	Functions that measure whether values are less than a certain quantity.

Regular Expressions		POSIX regular expressions can be used to select multiple actions when using ARG (i.e. PRINT).
Files		Dealing with Input/Output

10.13.1 histogrambead

A function that can be used to calculate whether quantities are between fixed upper and lower bounds. A function that can be used to calculate whether quantities are between fixed upper and lower bounds.
If we have multiple instances of a variable we can estimate the probability density function for that variable using a process called kernel density estimation:

$$P(s) = \sum_i K\left(\frac{s - s_i}{w}\right)$$

In this equation K is a symmetric function that must integrate to one that is often called a kernel function and w is a smearing parameter. From a probability density function calculated using kernel density estimation we can calculate the number/fraction of values between an upper and lower bound using:

$$w(s) = \int_a^b \sum_i K\left(\frac{s - s_i}{w}\right)$$

All the input to calculate a quantity like $w(s)$ is generally provided through a single keyword that will have the following form:
KEYWORD={TYPE UPPER= a LOWER= b SMEAR= $\frac{w}{b-a}$}
This will calculate the number of values between a and b. To calculate the fraction of values you add the word NORM to the input specification. If the function keyword SMEAR is not present w is set equal to $0.5(b-a)$. Finally, type should specify one of the kernel types that is present in plumed. These are listed in the table below:

TYPE	FUNCTION
GAUSSIAN	$\frac{1}{\sqrt{2\pi}w}\exp\left(-\frac{(s-s_i)^2}{2w^2}\right)$
TRIANGULAR	$\frac{1}{2w}\left(1. - \left\lvert\frac{s-s_i}{w}\right\rvert\right) \quad \frac{s-s_i}{w} < 1$

Some keywords can also be used to calculate a discrete version of the histogram. That is to say the number of values between a and b, the number of values between b and c and so on. A keyword that specifies this sort of calculation would look something like
KEYWORD={TYPE UPPER= a LOWER= b NBINS= n SMEAR= $\frac{w}{n(b-a)}$}
This specification would calculate the following vector of quantities:

$$w_j(s) = \int_{a+\frac{j-1}{n}(b-a)}^{a+\frac{j}{n}(b-a)} \sum_i K\left(\frac{s - s_i}{w}\right)$$

Glossary of keywords and components

10.13.2 kernelfunctions

Functions that are used to construct histograms Functions that are used to construct histograms

Constructing histograms is something you learned to do relatively early in life. You perform an experiment a number of times, count the number of times each result comes up and then draw a bar graph that describes how often each of the results came up. This only works when there are a finite number of possible results. If the result a number between 0 and 1 the bar chart is less easy to draw as there are as many possible results as there are numbers between zero and one - an infinite number. To resolve this problem we replace probability, P with probability density, π, and write the probability of getting a number between a and b as:

$$P = \int_a^b \mathrm{d}x \pi(x)$$

To calculate probability densities from a set of results we use a process called kernel density estimation. Histograms are accumulated by adding up kernel functions, K, with finite spatial extent, that integrate to one. These functions are centered on each of the n-dimensional data points, $\mathbf{x}_i$. The overall effect of this is that each result we obtain in our experiments contributes to the probability density in a finite sized region of the space.
Expressing all this mathematically in kernel density estimation we write the probability density as:

$$\pi(\mathbf{x}) = \sum_i K\left[(\mathbf{x}-\mathbf{x}_i)^T \Sigma (\mathbf{x}-\mathbf{x}_i)\right]$$

where Σ is an $n \times n$ matrix called the bandwidth that controls the spatial extent of the kernel. Whenever we accumulate a histogram (e.g. in HISTOGRAM or in METAD) we use this technique.
There is thus some flexibility in the particular function we use for $K[\mathbf{r}]$ in the above. The following variants are available.

TYPE	FUNCTION
gaussian	$f(r) = \frac{1}{(2\pi)^n \sqrt{\lvert\Sigma^{-1}\rvert}} \exp\left(-0.5r^2\right)$
truncated-gaussian	$f(r) = \frac{1}{(2\pi)^n \sqrt{\lvert\Sigma^{-1}\rvert}\left(\frac{\mathrm{erf}(-6.25/sqrt2)-\mathrm{erf}(-6.25/sqrt2)}{2}\right)^n} \exp\left(-0.5r^2\right)$
triangular	$f(r) = \frac{3}{V}(1-\lvert r\rvert)H(1-\lvert r\rvert)$
uniform	$f(r) = \frac{1}{V}H(1-\lvert r\rvert)$

In the above $H(y)$ is a function that is equal to one when $y > 0$ and zero when $y \leq 0$. n is the dimensionality of the vector $\mathbf{x}$ and V is the volume of an ellipse in an n dimensional space which is given by:

$$\begin{aligned} V &= \lvert\Sigma^{-1}\rvert \frac{\pi^{\frac{n}{2}}}{\left(\frac{n}{2}\right)!} \qquad \text{for even} \quad n \\ V &= \lvert\Sigma^{-1}\rvert \frac{2^{\frac{n+1}{2}} \pi^{\frac{n-1}{2}}}{n!!} \end{aligned}$$

In METAD the normalization constants are ignored so that the value of the function at $r = 0$ is equal to one. In addition in METAD we must be able to differentiate the bias in order to get forces. This limits the kernels we can use in this method. Notice also that Gaussian kernels should have infinite support. When used with grids, however, they are assumed to only be non-zero over a finite range. The difference between the truncated-gaussian and regular gaussian is that the truncated gaussian is scaled so that its integral over the grid is equal to one when it is normalized. The integral of a regular gaussian when it is evaluated on a grid will be slightly less that one because of the truncation of a function that should have infinite support.

Glossary of keywords and components

10.13.3 pdbreader

PLUMED can use the PDB format in several places PLUMED can use the PDB format in several places

- To read molecular structure (MOLINFO).
- To read reference conformations (RMSD, but also many other methods in Distances from reference configurations, FIT_TO_TEMPLATE, etc).

The implemented PDB reader expects a file formatted correctly according to the PDB standard. In particular, the following columns are read from ATOM records

```
columns | content
1-6     | record name (ATOM or HETATM)
7-11    | serial number of the atom (starting from 1)
13-16   | atom name
18-20   | residue name
22      | chain id
23-26   | residue number
31-38   | x coordinate
39-46   | y coordinate
47-54   | z coordinate
55-60   | occupancy
61-66   | beta factor
```

PLUMED parser is slightly more permissive than the official PDB format in the fact that the format of real numbers is not fixed. In other words, any real number that can be parsed is OK and the dot can be placed anywhere. However, **columns are interpret strictly**. A sample PDB should look like the following

```
ATOM      2  CH3 ACE     1      12.932 -14.718  -6.016  1.00  1.00
ATOM      5  C   ACE     1      21.312  -9.928  -5.946  1.00  1.00
ATOM      9  CA  ALA     2      19.462 -11.088  -8.986  1.00  1.00
```

Notice that serial numbers need not to be consecutive. In the three-line example above, only the coordinates of three atoms are provided. This is perfectly legal and indicates PLUMED that information about these atoms only is available. This could be both for structural information in MOLINFO, where the other atoms would have no name assigned, and for reference structures used in RMSD, where only the provided atoms would be used to compute RMSD.

Occupancy and beta factors

PLUMED reads also occupancy and beta factors that however are given a very special meaning. In cases where the PDB structure is used as a reference for an alignment (that's the case for instance in RMSD and in FIT_TO_TEMPLATE), the occupancy column is used to provide the weight of each atom in the alignment. In cases where, perhaps after alignment, the displacement between running coordinates and the provided PDB is computed, the beta factors are used as weight for the displacement. Since setting the weights to zero is the same as **not** including an atom in the alignment or displacement calculation, the two following reference files would be equivalent when used in an RMSD calculation. First file:

```
ATOM      2  CH3 ACE     1      12.932 -14.718  -6.016  1.00  1.00
ATOM      5  C   ACE     1      21.312  -9.928  -5.946  1.00  1.00
ATOM      9  CA  ALA     2      19.462 -11.088  -8.986  0.00  0.00
```

Second file:

```
ATOM      2  CH3 ACE     1      12.932 -14.718  -6.016  1.00  1.00
ATOM      5  C   ACE     1      21.312  -9.928  -5.946  1.00  1.00
```

However notice that many extra atoms with zero weight might slow down the calculation, so removing lines is better than setting their weights to zero. In addition, weights for alignment need not to be equivalent to weights for displacement. Starting with PLUMED 2.7, if all the weights are set to zero they will be normalized to be equal to the inverse of the number of involved atoms. This means that it will be possible to use files with the weight columns set to zero obtaining a meaningful result. In previous PLUMED versions, setting all weights to zero was resulting in an error instead.

Systems with more than 100k atoms

Notice that it very likely does not make any sense to compute the RMSD or any other structural deviation **using** so many atoms. However, if the protein for which you want to compute RMSD has atoms with large serial numbers (e.g. because it is located **after** solvent in the sorted list of atoms) you might end up with troubles with the limitations of the PDB format. Indeed, since there are 5 columns available for atom serial number, this number cannot be larger than 99999. In addition, providing MOLINFO with names associated to atoms with a serial larger than 99999 would be impossible.
Since PLUMED 2.4 we allow hybrid 36 format to be used to specify atom numbers. This format is not particularly widespread, but has the nice feature that it provides a one-to-one mapping between numbers up to approximately 80 millions and strings with 5 characters, plus it is backward compatible for numbers smaller than 100000. This is not true for notations like the hex notation exported by VMD. Using the hybrid 36 format, the ATOM records for atom ranging from 99997 to 100002 would read like these:

```
ATOM  99997  Ar      X   1      45.349  38.631  15.116  1.00  1.00
ATOM  99998  Ar      X   1      46.189  38.631  15.956  1.00  1.00
ATOM  99999  Ar      X   1      46.189  39.471  15.116  1.00  1.00
ATOM  A0000  Ar      X   1      45.349  39.471  15.956  1.00  1.00
ATOM  A0000  Ar      X   1      45.349  38.631  16.796  1.00  1.00
ATOM  A0001  Ar      X   1      46.189  38.631  17.636  1.00  1.00
```

There are tools that can be found to translate from integers to strings and back using hybrid 36 format (a simple python script can be found here). In addition, as of PLUMED 2.5, we provide a command line tool that can be used to renumber atoms in a PDB file.

Glossary of keywords and components

10.13.4 switchingfunction

Functions that measure whether values are less than a certain quantity. Functions that measure whether values are less than a certain quantity.
Switching functions $s(r)$ take a minimum of one input parameter r_0. For $r \leq d_0$ $s(r) = 1.0$ while for $r > d_0$ the function decays smoothly to 0. The various switching functions available in PLUMED differ in terms of how this decay is performed.
Where there is an accepted convention in the literature (e.g. COORDINATION) on the form of the switching function we use the convention as the default. However, the flexibility to use different switching functions is always present generally through a single keyword. This keyword generally takes an input with the following form:

```
KEYWORD={TYPE <list of parameters>}
```

The following table contains a list of the various switching functions that are available in PLUMED 2 together with an example input.

TYPE	FUNCTION	EXAMPLE INPUT	DEFAULT PARAMETERS
RATIONAL	$s(r) = \frac{1-\left(\frac{r-d_0}{r_0}\right)^n}{1-\left(\frac{r-d_0}{r_0}\right)^m}$	{RATIONAL R_0= r_0 D_0= d_0 NN= n MM= m}	$d_0 = 0.0$, $n = 6$, $m = 2n$
EXP	$s(r) = \exp\left(-\frac{r-d_0}{r_0}\right)$	{EXP R_0= r_0 D_0= d_0}	$d_0 = 0.0$
GAUSSIAN	$s(r) = \exp\left(-\frac{(r-d_0)^2}{2r_0^2}\right)$	{GAUSSIAN R_0= r_0 D_0= d_0}	$d_0 = 0.0$
SMAP	$s(r) = \left[1 + (2^{a/b} - 1)\left(\frac{r-d_0}{r_0}\right)^a\right]^{-b/a}$	{SMAP R_0= r_0 D_0= d_0 A= a B= b}	$d_0 = 0.0$
Q	$s(r) = \frac{1}{1+\exp(\beta(r_{ij}-\lambda r_{ij}^0))}$	{Q REF= r_{ij}^0 BETA= β LAM↩ BDA= λ }	$\lambda = 1.8$, $\beta = 50nm^-1$ (all-atom) $\lambda = 1.5$, $\beta = 50nm^-1$ (coarse-grained)
CUBIC	$s(r) = (y-1)^2(1+2y)$ where $y = \frac{r-r_1}{r_0-r_1}$	{CUBIC D_0= r_1 D_MAX= r_0}	
TANH	$s(r) = 1 - \tanh\left(\frac{r-d_0}{r_0}\right)$	{TANH R_0= r_0 D_0= d_0}	
COSINUS	$s(r) = \begin{cases} 1 & \text{if} r \leq d_0 \\ 0.5\left(\cos(\frac{r-d_0}{r_0}\pi) + 1\right) & \text{if} d_0 < r \leq d_0 + r_0 \\ 0 & \text{if} r < d_0 + r_0 \end{cases}$	{COSINUS R_0= r_0 D_0= d_0}	
CUSTOM	$s(r) = FUNC$	{CUSTOM FUNC=1/(1+x^6) R_0= r_0 D_0= d_0}	

Notice that for backward compatibility we allow using MATHEVAL instead of CUSTOM. Also notice that if the a CUSTOM switching function only depends on even powers of x it can be made faster by using x2 as a variable. For instance

```
{CUSTOM FUNC=1/(1+x2^3) R_0=0.3}
```

is equivalent to

```
{CUSTOM FUNC=1/(1+x^6) R_0=0.3}
```

but runs faster. The reason is that there is an expensive square root calculation that can be optimized out.

Attention

> With the default implementation CUSTOM is slower than other functions (e.g., it is slower than an equivalent RATIONAL function by approximately a factor 2). Checkout page Making lepton library faster to see how to improve its performance.

For all the switching functions in the above table one can also specify a further (optional) parameter using the parameter keyword D_MAX to assert that for $r > d_{\max}$ the switching function can be assumed equal to zero. In this case the function is brought smoothly to zero by stretching and shifting it.

```
KEYWORD={RATIONAL R_0=1 D_MAX=3}
```

the resulting switching function will be $s(r) = \frac{s'(r)-s'(d_{max})}{s'(0)-s'(d_{max})}$ where $s'(r) = \frac{1-r^6}{1-r^{12}}$ Since PLUMED 2.2 this is the default. The old behavior (no stretching) can be obtained with the NOSTRETCH flag. The NOSTRETCH keyword is only provided for backward compatibility and might be removed in the future. Similarly, the STRETCH keyword is still allowed but has no effect.
Notice that switching functions defined with the simplified syntax are never stretched for backward compatibility. This might change in the future.

Glossary of keywords and components

10.13.5 Regular Expressions

When you use need to pass many arguments to a PLUMED action, being them components of a few collective variables or also multiple collective variables, you might find it convenient to use regular expressions.
Since version 2.1, plumed takes advantage of a configuration scripts that detects libraries installed on your system. If regex library is found, then you will be able to use regular expressions to refer to collective variables or function names.
Regular expressions are enclosed in round braces and must not contain spaces (the components names have no spaces indeed, so why use them?).
As an example the command:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RegexPP.md
d1: DISTANCE ATOMS=1,2 COMPONENTS
PRINT ARG=(d1\.[xy])   STRIDE=100 FILE=colvar FMT=%8.4f
```

will cause both the d1.x and d1.y components of the DISTANCE action to be printed.
Notice that selection does not happen in alphabetic order, nor in the order in which [xy] are listed, but rather in the order in which the two variables have been created by PLUMED. Also notice that the . character must be escaped as \. in order to interpret it as a literal .. An un-escaped dot is a wildcard which is matched by any character, So as an example

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RegexPP.md
d1: DISTANCE ATOMS=1,2 COMPONENTS
dxy: DISTANCE ATOMS=1,3

# this will match d1.x,d1.y,dxy
PRINT ARG=(d1.[xy])   STRIDE=100 FILE=colvar FMT=%8.4f

# while this will match d1.x,d1.y only
PRINT ARG=(d1\.[xy])   STRIDE=100 FILE=colvar FMT=%8.4f
```

You can concatenate more than one regular expression by using comma separated regular expressions. The resulting matches will be concatenated:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RegexPP.md
t1: TORSION ATOMS=5,7,9,15
t2: TORSION ATOMS=7,9,15,17
d1: DISTANCE ATOMS=7,17 COMPONENTS

# The first expression matches d1.x and d1.y
# The second expression matches t1 and t2
PRINT ARG=(d1\.[xy]),(t[0-9]) STRIDE=100 FILE=colvar FMT=%8.4f
# Thus this is the same as ARG=d1.x,d1.y,t1,t2
```

Be aware that if you have overlapping selections they will be duplicated. As an alternative you could use the "or" operator |:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RegexPP.md
t1: TORSION ATOMS=5,7,9,15
t2: TORSION ATOMS=7,9,15,17
d1: DISTANCE ATOMS=7,17 COMPONENTS

# Here is a single regular expression
PRINT ARG=(d1\.[xy]|t[0-9]) STRIDE=100 FILE=colvar FMT=%8.4f
# Thus this is the same as ARG=t1,t2,d1.x,d1.y
```

this selects the same set of arguments as the previous example.

Note

Be careful you do not confuse regular expressions, which are triggered by the parenthesis `()` and only available when PLUMED has been compiled with the regex library, with the capability of PLUMED to use $*$ as a wildcard in arguments:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/RegexPP.md
d1: DISTANCE ATOMS=1,2 COMPONENTS
# this is a regular expression that selects all components of d1
# i.e. d1.x d1.y and d1.z
PRINT ARG=(d1\..*)   STRIDE=100 FILE=colvar_reg FMT=%8.4f

# this is a wildcard that selects all the components of d1 as well
PRINT ARG=d1.*   STRIDE=100 FILE=colvar_wild FMT=%8.4f
```

Regular expressions are way more flexible than wildcards!

You can check the log to see whether or not your regular expression is picking the set of components you desire.
For more information on regular expressions visit http://www.regular-expressions.info/reference.↩html.

10.13.6 Files

We tried to design PLUMED in such a manner that input/output is done consistently irrespective of the file type. Most of the files written or read by PLUMED thus follow the very same conventions discussed below.

10.13.6.1 Restart

Whenever the RESTART option is used, all the files written by PLUMED are appended. This makes it easy to analyze results of simulations performed as a chain of several sub-runs. Notice that most of the PLUMED textual files have a header. The header is repeated at every restart. Additionally, several files have time in the first column. PLUMED just takes the value of the physical time from the MD engine. As such, you could have that time starts again from zero upon restart or not.
An exception from this behavior is given by files which are not growing as the simulation proceeds. For example, grids written with METAD with GRID_WFILE are overwritten by default during the simulation. As such, when restarting, there is no point in appending the file. Internally, PLUMED opens the file in append mode but then rewinds it every time a new grid is dumped.

10.13.6.2 Backup

Whenever the RESTART option is not used, PLUMED tries to write new files. If an old file is found in the way, P↩LUMED takes a backup named "bck.X.filename" where X is a progressive number. Notice that by default PLUMED only allows a maximum of 100 backup copies for a file. This behavior can be changed by setting the environment

variable PLUMED_MAXBACKUP to the desired number of copies. E.g. export PLUMED_MAXBACKUP=10 will fail after 10 copies. PLUMED_MAXBACKUP=-1 will never fail - be careful since your disk might fill up quickly with this setting.

10.13.6.3 Replica suffix

When running with multiple replicas (e.g., with GROMACS, -multi option) PLUMED adds the replica index as a suffix to all the files. The following command will thus print files named COLVAR.0, COLVAR.1, etc for the different replicas.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FilesPP.md
d: DISTANCE ATOMS=1,2
PRINT ARG=d FILE=COLVAR
```

When reading a file, PLUMED will try to add the suffix. If the file is not found, it will fall back to the name without suffix. The most important case is the reading of the plumed input file. If you provide a file for each replica (e.g. plumed.0.dat, plumed.1.dat, etc) you will be able to setup plumed differently on each replica. On the other hand, using a single plumed.dat will make all the replicas read the same file.

Warning

> This rule is true for almost all the files read by PLUMED. As of PLUMED version 2.4, the only exception is PDB files, where the replica suffix is not added.

Notice that when PLUMED adds the replica suffix, it recognizes the file extension and add the suffix *before* the extension. Before PLUMED 2.2, the only recognized suffix was ".gz". Since 2.2, any suffix with length less or equal to five letters is recognized.
This means that using in a multi-replica context an input such as

```
BEGIN_PLUMED_FILE working DATADIR=example-check/FilesPP.md
d: DISTANCE ATOMS=1,2
PRINT ARG=d FILE=COLVAR.gz
METAD ARG=d FILE=test.HILLS SIGMA=0.1 HEIGHT=0.1 PACE=100
```

PLUMED will write files named COLVAR.0.gz, COLVAR.1.gz, test.0.HILLS, test.1.HILLS, etc etc. This is useful since the preserved extension makes it easy to process the files later.

Chapter 11

Tutorials

The following pages describe how to perform a variety of tasks using PLUMED

Master ISDD tutorial 2024: Brief introduction to PLUMED	This tutorial explains the syntax of the PLUMED input file and how to use PLUMED to analyze CVs
Master ISDD tutorial 2024: Metadynamics simulations with PLUMED	This tutorial explains how to use PLUMED to run metadynamics simulations
PLUMED Masterclass 22.1: Funnel-Metadynamics	This Masterclass explains how to use PLUMED to run and analyze Funnel-Metadynamics simulations
PLUMED Masterclass 22.2: Analysis of Plumed output by Metadynminer	This Masterclass explains the use of R package Metadynminer to analyze metadynamics results.
PLUMED Masterclass 22.3: OPES method	This Masterclass shows how to use PLUMED to run and analyze OPES simulations
PLUMED Masterclass 22.5: Machine learning collective variables with PyTorch	This Masterclass shows how to train machine learning-based CVs in PyTorch and deploy them in PLUMED.
PLUMED Masterclass 22.6: EDS module + Coarse-Grained simulations	This Masterclass describes how to bias simulations to agree with experimental data using experiment directed simulation.
PLUMED Masterclass 22.8: Modelling Concentration-driven processes with PLUMED	This Masterclass discusses the analysis and simulation of processes driven by concentration such as nucleation, diffusion and growth with PLUMED.
PLUMED Masterclass 22.9: Using path collective variables to explore free energy landscapes	This Masterclass explains the syntax of the PLUMED input file and how to use PLUMED to analyze CVs
PLUMED Masterclass 22.10: Hamiltonian replica exchange with PLUMED and GROMACS	This Masterclass explains how to use Hamiltonian replica exchange simulations with GROMACS + PLUMED
PLUMED Masterclass 22.11: Variationally enhanced sampling with PLUMED	This Masterclass explains how to run variationally enhanced sampling simulations with PLUMED
PLUMED Masterclass 22.12: Liquid-solid chemical potential differences with the environment similarity CV	This Masterclass explains how to compute chemical potential differences between the liquid and a solid with the environment similarity CV
PLUMED Masterclass 22.13: SASA module and the application of PLUMED for implicit solvent simulations	This Masterclass explains how to use the SASA module of PLUMED to perform implicit solvent simulations
PLUMED Masterclass 22.15: FISST module and application of statistical learning in PLUMED	This Masterclass explains how to use the SASA module of PLUMED to perform implicit solvent simulations
PLUMED Masterclass 21.1: PLUMED syntax and analysis	This Masterclass explains the syntax of the PLUMED input file and how to use PLUMED to analyze CVs
PLUMED Masterclass 21.2: Statistical errors in MD	This Masterclass explains how to report averages, histograms and the associated error bars so that others can reproduce your results.

PLUMED Masterclass 21.3: Umbrella sampling	This Masterclass explains how to use PLUMED to run simple restrained simulations and account for the bias in the analysis
PLUMED Masterclass 21.4: Metadynamics	This Masterclass explains how to use PLUMED to run and analyze metadynamics simulations
PLUMED Masterclass 21.5: Simulations with multiple replicas	This Masterclass explains how to use PLUMED to run multiple-replica simulations
PLUMED Masterclass 21.6: Dimensionality reduction	This Masterclass explains how to select collective variables in a variety of contexts
PLUMED Masterclass 21.7: Optimizing PLUMED performances	This Masterclass explains how to use optimize PLUMED performances
Advanced Methods in MD 2023: Metadynamics simulations with PLUMED	This tutorial explains how to use PLUMED to run metadynamics simulations
Lugano tutorial: Brief guide to PLUMED syntax and analysis tutorial	This tutorial explains the syntax of the PLUMED input file and how to use PLUMED to analyze CVs
Lugano tutorial: Using restraints	This tutorial explains how to use PLUMED to run simple restrained simulations and account for the bias in the analysis
Lugano tutorial: Metadynamics simulations with PLUMED	This tutorial explains how to use PLUMED to run metadynamics simulations
Lugano tutorial: Calculating error bars	Calculating error bars
Lugano tutorial: Dimensionality reduction	How to perform dimensionality reduction using PLUMED
Lugano tutorial: Using PLUMED with LAMMPS	An exercise on running PLUMED with LAMMPS
Lugano tutorial: Binding of a ion and a dinucleotide	An exercise to compute binding free energies
Lugano tutorial: Computing proton transfer in water	An exercise to run a simple proton transfer calculation with CP2K
Lugano tutorial: Folding free energy for a protein described by a coil	This tutorial propose a more complex case to test your own skill with a more realistic example
Using Hamiltonian replica exchange with GROMACS	This tutorial explains how to use Hamiltonian replica exchange in GROMACS
Julich tutorial: Developing CVs in plumed	Implementing new collective variables in plumed
ESDW Workshop Lyon 2019: A very simple introduction to PLUMED	This tutorial gives a brief introduction to PLUMED by showing how it can be used to study a cluster of 7 Lennard Jones atoms.
MARVEL tutorial: Analyzing CVs	This tutorial explains how to use PLUMED to analyze CVs
MARVEL tutorial: Path CVs	This tutorial explains how to use various kinds of path collective variables
new-munster	This tutorial explains how to use PLUMED to run metadynamics simulations
Optimizing PLUMED performance	A short 1.5 hours tutorial that introduces performance optimization

In addition, the following websites contain resources that might be helpful

http://www.youtube.com/watch?v=iDvZmbWE5ps	A short video introduction to the use of multicolvars in PLUMED 2
http://www.youtube.com/watch?v=PxJP16qNCYs	A short video introduction to the syntax of the PLUMED 2 input file
http://en.wikipedia.org/wiki/Metadynamics	A wikipedia article on metadynamics

Some older tutorials are also available here (some of them cover topics not covered by the recent tutorial but syntax

may be outdated):

- Old Tutorials

11.1 Master ISDD tutorial 2024: Brief introduction to PLUMED

11.1.1 Aim

The aim of this tutorial is to introduce users to the PLUMED syntax. We will go through the preparation of input files to calculate and print simple collective variables on a pre-calculated trajectory. This tutorial has been prepared by Max Bonomi (adapting a lot of material from other tutorials and Masterclass) for the `Master In Silico Drug Design`, held at Université Paris Cité on April 8th, 2024.

11.1.2 Objectives

Once this tutorial is completed, users will be able to:

- Write a simple PLUMED input file and use it with the PLUMED driver to analyze a trajectory.
- Print collective variables such as distances (DISTANCE), torsional angles (TORSION), and gyration radius (GYRATION) using the PRINT action.
- Use MOLINFO shortcuts.

11.1.3 Software

In this and in the next tutorial, we will use two pieces of software: `PLUMED` version 2.9.0 and `GROMACS` version 2020.7. First, we need to install the software on your machine using `conda`.

11.1.3.1 Installation

You should have all the software installed in a conda environment. To check this, you can type:

```
# activate environment
conda activate ISDD-tutorial-2024
```

If the environment is not found, you can prepare it yourself. First, you need to create the conda environment with:

```
# create environment
conda create --name ISDD-tutorial-2024
```

and activate it with:

```
# activate environment
conda activate ISDD-tutorial-2024
```

Finally, we can proceed with the installation of the required software:

```
conda install --strict-channel-priority -c plumed/label/isdd-2024 -c conda-forge plumed
conda install --strict-channel-priority -c plumed/label/isdd-2024 -c conda-forge gromacs
conda install -c conda-forge numpy
```

The `--strict-channel-priority` might be necessary in case your `conda` install is configured to download packages from the `bioconda` channel. Indeed, `bioconda` contains a version of GROMACS that is **not** patched with PLUMED and would thus not work here.
Please keep in mind that every time you open a new shell, in order to use PLUMED and GROMACS you need to activate the `ISDD-tutorial-2024` environment using the following command:

```
conda activate ISDD-tutorial-2024
```

11.1.3.2 PLUMED overview

PLUMED is a library that can be incorporated into many MD codes by adding a relatively simple and well documented interface. Once it is incorporated you can use PLUMED to perform a variety of different analyses on-the-fly and to bias the sampling in MD simulations. Additionally, PLUMED can be used as a standalone code for analyzing trajectories. If you are using the code in this way you can run the PLUMED executable by issuing the command:

```
plumed <instructions>
```

Let's start by getting a feel for the range of calculations that PLUMED can do. Issue the following command now:

```
plumed --help
```

The output of this command is a list of tasks that PLUMED can perform. Among these, there are commands that allow you to patch an MD code, postprocess metadynamics simulations, and build the manual. In this tutorial we will mostly use PLUMED to analyze trajectories. In order to do so, we will learn how to use the driver tool. Let's look at the options of PLUMED driver by issuing the following command:

```
plumed driver --help
```

As you can see we can do a number of things with PLUMED driver. For all of these options, however, we are going to need to write a PLUMED input file. The syntax of the PLUMED input file is the same that we will use later to run enhanced sampling simulations during a MD simulation, so all the things that you will learn now will be useful later when you will run PLUMED coupled to GROMACS.

11.1.3.3 GROMACS overview

GROMACS is a molecular dynamics package mainly designed for simulations of proteins, lipids, and nucleic acids. More info can be found here.
You can check that GROMACS is properly working by typing:

```
gmx_mpi mdrun -h
```

If you inspect the long output generated by this command, you will notice a `-plumed` option, which indicates that GROMACS has properly been compiled with PLUMED. Great, now you are ready to go!

11.1.4 Your first PLUMED input file

The main goal of PLUMED is to compute collective variables (or CVs), which are complex descriptors of the system that can be used to describe the conformational change of a protein or a chemical reaction. This can be done either on-the-fly during a molecular dynamics simulations or *a posteriori* on a pre-calculated trajectory using PLUMED as a post-processing tool. In both cases one, you should create an input file with a specific PLUMED syntax. Have a look at the sample input file below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-master-ISDD-1.txt
# Compute distance between atoms 1 and 10.
# Atoms are ordered as in the trajectory files and their numbering starts from 1.
# The distance is called "d" for future reference.
d: DISTANCE ATOMS=1,10

# Compute the torsional angle between atoms 1, 10, 20, and 30.
# The angle is called "phi1" for future reference.
phi1: TORSION ATOMS=1,10,20,30

# The same CV defined above can be split into multiple lines
# The angle is called "phi2" for future reference.
TORSION ...
LABEL=phi2
ATOMS=1,10,20,30
...

# Print "d" on a file named "COLVAR1" every 10 steps.
PRINT ARG=d FILE=COLVAR1 STRIDE=10

# Print "phi1" and "phi2" on another file named "COLVAR2" every 100 steps.
PRINT ARG=phi1,phi2 FILE=COLVAR2 STRIDE=100
```

In the input file above, each line defines a so-called action. In this simple example, actions are used to compute a distance, a dihedral angle, or print some values on a file. Each action supports a number of keywords, whose value is specified. Action names are highlighted in green and, by clicking on them, you can go to the corresponding page in the manual that contains a detailed description of each keyword. Actions that support the keyword `STRIDE` are those that determine how frequently things are done. Notice that the default value for `STRIDE` is always 1. In the example above, omitting `STRIDE` keywords the corresponding `COLVAR` files would have been written for every frame of the analyzed trajectory. All the other actions in the example above, i.e. DISTANCE and TORSION, do not support the `STRIDE` keyword and are only calculated when requested. That is, `d` will be computed every 10 frames, and `phi1` and `phi2` every 100 frames.
Variables should be given a name (in the example above, `d`, `phi1`, and `phi2`), which is then used to refer to these variables in subsequent actions, such as the PRINT command. A lists of atoms should be provided as comma separated numbers, with no space.
You can find more information on the PLUMED syntax at Getting Started page of the manual. The complete documentation for all the supported collective variables can be found at the Collective Variables page.

11.1.4.1 The PLUMED internal units

By default the PLUMED inputs and outputs quantities in the following units:

- Length: nanometers
- Energy: kJ/mol
- Time: picoseconds
- Mass: amu
- Charge: e

If you want to change these units, you can do this using the UNITS keyword.

11.1.5 Resources

The `TARBALL` for this tutorial contains the following files:

- `GB1_native.pdb`: a PDB file with the native structure of the GB1 protein.
- `traj-whole.xtc`: a trajectory in `xtc` format. GB1 has already been made whole by fixing discontinuities due to periodic boundary conditions.
- `traj-broken.xtc`: the same trajectory as it was originally produced by GROMACS. Here GB1 is broken by periodic boundary conditions.

After dowloading the compressed archive to your local machine, you can unpack it using the following command:

```
tar xvzf master-ISDD-1.tar.gz
```

Once unpacked, all the files can be found in the `master-ISDD-1` directory. To keep things clean, it is recommended to run each exercise in a separate sub-directory that you can create inside `master-ISDD-1`.

Note

This tutorial has been tested with PLUMED version 2.8.2.

11.1.6 Exercises

11.1.6.1 Exercise 1: Computing and printing simple collective variables

In this exercise, we will learn how to compute and print collective variables on a pre-calculated MD trajectory. To analyze the trajectories provided here, we will:

- create a PLUMED input file with a text editor (typically called `plumed.dat`);
- run the PLUMED driver utility.

Notice that you can also visualize trajectories with VMD (always a good idea!). For example, the trajectory traj-whole.xtc can be visualized with the command:

```
vmd GB1_native.pdb traj-whole.xtc
```

Let's now prepare a PLUMED input file to calculate:

- the gyration radius of all CA protein atoms (GYRATION). Look in the GB1_native.pdb file to retrieve the list of indexes of the CA atoms;
- the total number of contacts (COORDINATION) between all protein CA atoms. For COORDINATION, set reference distance R_0 to 8.0 Angstrom (be careful with units!).

Below you can find a sample plumed.dat file that you can use as a template. Whenever you see an highlightedFILL string, be careful because this is a string that you must replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-master-ISDD-1.txt
# Compute gyration radius on CA atoms:
r: GYRATION ATOMS=__FILL__

# Compute number of contacts between CA atoms
co: COORDINATION GROUPA=__FILL__ R_0=__FILL__

# Print the two collective variables on COLVAR file every step
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

Once your plumed.dat file is complete, you can run the PLUMED driver as follows:

```
plumed driver --plumed plumed.dat --mf_xtc traj-broken.xtc
```

Scroll in your terminal to read the PLUMED log. As you can see, PLUMED gives a lot of feedback about the input that it is reading and the actions that it will execute. Please take your time to inspect the log file and check if PLUMED is actually doing what you intend to do.
The command above will create a COLVAR file like this one:

```
#! FIELDS time r co
 0.000000 2.458704 165.184127
 1.000000 2.341932 164.546604
 2.000000 2.404708 162.606975
 3.000000 2.454297 143.850122
 4.000000 2.569342 147.110408
 5.000000 2.304027 163.608703
```

Notice that the first line informs you about the content of each column.
In case you obtain different numbers, check your input, you might have made some mistake!
This file can then be visualized using gnuplot (for more info about this tool, look here):

```
gnuplot> p "COLVAR" u 1:2, "" u 1:3
```

Now, look at what happens if you run the exercise twice. The second time, PLUMED will *back up* the previously produced file so as not to overwrite it. You can also *concatenate* your files by using the action RESTART at the beginning of your input file.
Finally, let's try to use the same input file with traj-whole.xtc and examine the results. Did you find the same results as with the previous trajectory? Why?

11.1.6.2 Exercise 2: MOLINFO shortcuts

PLUMED provides some shortcuts to select atoms with specific properties. To use this feature, you should specify the MOLINFO action along with a reference PDB file. This command is used to provide information on the molecules that are present in your system.
Let's try to use this functionality to calculate the backbone dihedral angle ϕ (phi) of residue 2 of the GB1 protein. This CV is defined by the action TORSION and a set of 4 atoms. For residue i, the dihedral ϕ is defined by these atoms: C(i-1),N(i),CA(i),C(i) (see Fig. master-ISDD-1-dih-fig).
After consulting the manual and inspecting GB1_native.pdb, let's define the dihedral angle ϕ of residue 2 in two different ways:

1. specifying an explicit list of 4 atoms (`t1`).
2. using the MOLINFO shortcut (`t2`);

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-master-ISDD-1.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__

# Define the dihedral phi of residue 2 as an explicit list of 4 atoms
t1: TORSION ATOMS=__FILL__
# Define the same dihedral using MOLINFO shortcuts
t2: TORSION ATOMS=__FILL__

# Print the two collective variables on COLVAR file every 10 steps
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

After completing the PLUMED input file above, let's use it to analyze the trajectory `traj-whole.xtc` using the driver tool:

```
plumed driver --plumed plumed.dat --mf_xtc traj-whole.xtc
```

You can use `gnuplot` to visualize the trajectory of the two CVs calculated with the PLUMED input file above and written in the `COLVAR` file. **Are the two trajectories identical?**

11.1.7 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Analyze previously calculated trajectories using the driver utility.
- Define MOLINFO shortcuts.

11.2 Master ISDD tutorial 2024: Metadynamics simulations with PLUMED

11.2.1 Aim

The aim of this tutorial is to train users to perform and analyze metadynamics simulations with PLUMED. This tutorial has been prepared by Max Bonomi (adapting a lot of material from other tutorials) for the `Master In Silico Drug Design`, held at Université Paris Cité on April 8th, 2024.

11.2.2 Objectives

Once this tutorial is completed users will be able to:

- Write the PLUMED input file to perform metadynamics simulations.
- Calculate the free energy as a function of the metadynamics collective variables.
- Unbias metadynamics simulations.
- Estimate the error in the reconstructed free energies using block analysis.
- Assess the convergence of metadynamics simulations.

11.2.3 Resources

The `TARBALL` for this tutorial contains the following files:

- `diala.pdb`: a PDB file for alanine dipeptide in vacuo.
- `topol.tpr`: a GROMACS run (binary) file to perform MD simulations of alanine dipeptide.
- `do_block_fes.py`: a python script to perform error analysis of metadynamics simulations.

After dowloading the compressed archive to your local machine, you can unpack it using the following command:

```
tar xvzf master-ISDD-2.tar.gz
```

Once unpacked, all the files can be found in the `master-ISDD-2` directory. To keep things clean, it is recommended to run each exercise in a separate sub-directory that you can create inside `master-ISDD-2`.

Note

This tutorial has been tested with PLUMED version 2.9.0 and GROMACS version 2020.7.

11.2.4 Introduction

In the previous tutorial, we have seen that PLUMED can be used to compute collective variables (CVs) on a pre-calculated trajectory. However, PLUMED is most often use to add forces on the CVs during a MD simulation, for example, in order to accelerate sampling. To this aim, we have implemented a variety of possible biases acting on CVs. The complete documentation for all the biasing methods available in PLUMED can be found at the Bias page. In the following we will learn how to use PLUMED to perform and analyze a metadynamics simulation. Here you can find a brief recap of the metadynamics theory.
In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s},t\to\infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)),k\tau)}{k_B\Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s},t\to\infty) = -\frac{\Delta T}{T+\Delta T}F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T+\Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard MD, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T+\Delta T$) and the system temperature (T):

$$\gamma = \frac{T+\Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135] [136].
We will play with a toy system, alanine dipeptide simulated in vacuo using the AMBER99SB-ILDN force field (see Fig. master-ISDD-2-ala-fig). This rather simple molecule is useful to understand data analysis and free-energy methods. This system is a nice example because it presents two metastable states separated by a high free-energy barrier. It is conventional use to characterize the two states in terms of Ramachandran dihedral angles, which are denoted with ϕ (phi) and ψ (psi) in Fig. master-ISDD-2-transition-fig.

11.2.5 Exercises

11.2.5.1 Exercise 1: My first metadynamics simulation

In this exercise we will setup and perform a well-tempered metadynamics run using the backbone dihedral ϕ as collective variable. During the calculation, we will also monitor the behavior of the other backbone dihedral ψ.
Here you can find a sample `plumed.dat` file that you can use as a template. Whenever you see an highlightedF↩ ILL string, this is a string that you must replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-master-ISDD-2.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=diala.pdb

# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
# you might want to use MOLINFO shortcuts
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
# here also you might want to use MOLINFO shortcuts
psi: TORSION ATOMS=__FILL__

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 500 time steps, with initial height equal to 1.2 kJ/mol
  PACE=500 HEIGHT=1.2
# The bias factor should be wisely chosen
  BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
  SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
...

# Print both collective variables on COLVAR file every 10 steps
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

The syntax for the command METAD is simple. The directive is followed by a keyword `ARG` followed by the labels of the CVs on which the metadynamics bias potential will act. The keyword `PACE` determines the stride of Gaussian deposition in number of time steps, while the keyword `HEIGHT` specifies the height of the Gaussian. For each CVs, one has to specify the width of the Gaussian by using the keyword `SIGMA`. Gaussian will be written to the file indicated by the keyword `FILE`.
In this example, the bias potential will be stored on a grid, whose boundaries are specified by the keywords `GRI↩ D_MIN` and `GRID_MAX`. Notice that you can provide either the number of bins for every CV (`GRID_BIN`) or the desired grid spacing (`GRID_SPACING`). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither `GRID_BIN` nor `GRID_SPACING`) and if Gaussian width is fixed, PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.
Once your `plumed.dat` file is complete, you can run a 10-ns long metadynamics simulations with the following command:

```
gmx_mpi mdrun -s topol.tpr -nsteps 5000000 -plumed plumed.dat -ntomp 1
```

During the metadynamics simulation, PLUMED will create two files, named `COLVAR` and `HILLS`. The `COLVAR` file contains all the information specified by the PRINT command, in this case the value of the backbone dihedrals ϕ and ψ every 10 steps of simulation. We can use `gnuplot` to visualize the behavior of the metadynamics CV ϕ during the simulation:

```
gnuplot> p "COLVAR" u 1:2
```

By inspecting Figure master-ISDD-2-phi-fig, we can see that the system is initialized in one of the two metastable states of alanine dipeptide. After a while (t=0.1 ns), the system is pushed by the metadynamics bias potential to visit the other local minimum. As the simulation continues, the bias potential fills the underlying free-energy landscape, and the system is able to diffuse in the entire phase space.
The `HILLS` file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content:

```
#! FIELDS time phi sigma_phi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
```

The line starting with FIELDS tells us what is displayed in the various columns of the HILLS file: the simulation time, the instantaneous value of ϕ, the Gaussian width and height, and the bias factor. We can use the HILLS file to visualize the decrease of the Gaussian height during the simulation, according to the well-tempered recipe:
If we look carefully at the scale of the y-axis, we will notice that in the beginning the value of the Gaussian height is higher than the initial height specified in the input file, which should be 1.2 kJ/mol. In fact, this column reports the height of the Gaussian scaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy.

Warning

> The fact that the Gaussian height is decreasing to zero should not be used as a measure of convergence of your metadynamics simulation!

11.2.5.2 Exercise 2: Estimating the free energy as a function of the metadynamics CVs

One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills can be used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.

To calculate the free energy as a function of ϕ, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called fes.dat in which the free-energy surface as function of ϕ is calculated on a regular grid. One can modify the default name for the free-energy file, as well as the boundaries and bin size of the grid, by using the following sum_hills options:

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

The result should look like this:
To give a preliminary assessment of the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The sum_hills option --stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option --mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --stride 100 --mintozero
```

one free energy is calculated every 100 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. The resulting plot should look like the following:
These two qualitative observations:

1. the system is diffusing rapidly in the entire CV space (Figure master-ISDD-2-phi-fig)
2. the estimated free energy does not significantly change as a function of time (Figure master-ISDD-2-metad-phifest-fig)

suggest that the simulation **might** be converged.

Warning

> The two conditions listed above are necessary, but not sufficient to declare convergence. For a quantitative analysis of the convergence of metadynamics simulations, please have a look below at Exercise 4: Estimating the error in free energies using block-analysis.

11.2.5.3 Exercise 3: Reweighting (unbiasing) a metadynamics simulation

In the previous exercise we biased ϕ and computed the free energy as a function of the same variable directly from the metadynamics bias potential using the sum_hills utility. However, in many cases you might decide which variable should be analyzed *after* having performed a metadynamics simulation. For example, you might want to calculate the free energy as a function of CVs other than those biased during the metadynamics simulation, such as

the dihedral ψ. At variance with standard MD simulations, you cannot simply calculate histograms of other variables directly from your metadynamics trajectory, because the presence of the metadynamics bias potential has altered the statistical weight of each frame. To remove the effect of this bias and thus be able to calculate properties of the system in the unbiased ensemble, you must reweight (unbias) your simulation.
There are multiple ways to calculate the correct statistical weight of each frame in your metadynamics trajectory and thus to reweight your simulation. For example:

1. weights can be calculated by considering the time-dependence of the metadynamics bias potential [3];
2. weights can be calculated using the metadynamics bias potential obtained at the end of the simulation and assuming a constant bias during the entire course of the simulation [50].

In this exercise we will use the second method, which resembles the umbrella-sampling reweighting approach. In order to compute the weights we will use the driver tool.
First of all, you need to prepare a `plumed_reweight.dat` file that is identical to the one you used for running your metadynamics simulation except for a few modifications. First, you need to add the keyword `RESTART=YES` to the METAD command. This will make this action behave as if PLUMED was restarting, i.e. PLUMED will read from the `HILLS` file the Gaussians that have previously been accumulated. Second, you need to set the Gaussian `HEIGHT` to zero and the `PACE` to a large number. This will actually avoid adding new Gaussians (and even if they are added they will have zero height). Finally, you need to modify the PRINT statement so that you write every frame and that, in addition to `phi` and `psi`, you also write `metad.bias`. You might also want to change the name of the output file to `COLVAR_REWEIGHT`.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-master-ISDD-2.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=diala.pdb

__FILL__ # here goes the definitions of the phi and psi CVs

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 10000000 time steps (never!), with initial height equal to 0.0 kJ/mol
  PACE=10000000 HEIGHT=0.0 # <- this is the new stuff!
# The bias factor should be wisely chosen
  BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
  SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
# Say that METAD should be restarting (= reading an existing HILLS file)
  RESTART=YES # <- this is the new stuff!
...

# Print out the values of phi, psi and the metadynamics bias potential
# Make sure you print out the 3 variables in the specified order at every step
PRINT ARG=__FILL__ FILE=COLVAR_REWEIGHT STRIDE=__FILL__  # <- also change this one!
```

Then run the driver tool using this command:

```
plumed driver --mf_xtc traj_comp.xtc --plumed plumed_reweight.dat --kt 2.494339
```

Notice that you have to specify the value of k_BT in energy units. While running your simulation this information was communicated by the MD code.
As a result, PLUMED will produce a new `COLVAR_REWEIGHT` file with one additional column containing the metadynamics bias potential $V(s)$ calculated using all the Gaussians deposited along the entire trajectory. The beginning of the file should look like this:

```
#! FIELDS time phi psi metad.bias
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
 0.000000 -1.497988 0.273498 110.625670
 1.000000 -1.449714 0.576594 110.873141
 2.000000 -1.209587 0.831417 109.742353
 3.000000 -1.475975 1.279726 110.752327
```

You can easily obtain the weight w of each frame using the expression $w \propto \exp\left(\frac{V(s)}{k_B T}\right)$ (umbrella-sampling-like reweighting). At this point, you can read the COLVAR_REWEIGHT file using, for example, a python script and compute a weighted histogram. Alternatively, if you want PLUMED to do the weighted histograms for you, you can add the following lines at the end of the plumed_reweight.dat file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-master-ISDD-2.txt
# Use the metadynamics bias as argument
as: REWEIGHT_BIAS ARG=__FILL__

# Calculate histograms of phi and psi dihedrals every 50 steps
# using the weights obtained from the metadynamics bias potentials (umbrella-sampling-like reweighting)
# Look at the manual to understand the parameters of the HISTOGRAM action!
hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=50 BANDWIDTH=0.05 LOGWEIGHTS=as
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=50 BANDWIDTH=0.05 LOGWEIGHTS=as

# Convert histograms h(s) to free energies F(s) = -kBT * log(h(s))
ffphi: CONVERT_TO_FES GRID=hhphi
ffpsi: CONVERT_TO_FES GRID=hhpsi

# Print out the free energies F(s) to file once the entire trajectory is processed
DUMPGRID GRID=ffphi FILE=ffphi.dat
DUMPGRID GRID=ffpsi FILE=ffpsi.dat
```

and plot the result using gnuplot:

```
gnuplot> p "ffphi.dat" u 1:2 w lp
gnuplot> p "ffpsi.dat" u 1:2 w lp
```

You can now compare the free energies as a function of ϕ calculated:

1. directly from the metadynamics bias potential using sum_hills as done in Exercise 2: Estimating the free energy as a function of
2. using the reweighting procedure introduced in this exercise.

The results should be identical (see Fig. master-ISDD-2-fescomp-fig).
.

11.2.5.4 Exercise 4: Estimating the error in free energies using block-analysis

In the previous exercise, we calculated the *final* bias $V(s)$ on the entire metadynamics trajectory and we used this quantity to calculate the correct statistical weight of each frame that we need to reweight the biased simulation. In this exercise, we will see how this information can be used to calculate the error in the reconstructed free energies and assess whether our simulation is converged or not. Let's first calculate the un-biasing weights $w \propto \exp\left(\frac{V(s)}{k_B T}\right)$ from the COLVAR_REWEIGHT file obtained at the end of Exercise 3: Reweighting (unbiasing) a metadynamics simulation:

```
# Find maximum value of bias to avoid numerical errors when calculating the un-biasing weights
bmax=`awk 'BEGIN{max=0.}{if($1!="#!" && $4>max)max=$4}END{print max}' COLVAR_REWEIGHT`

# Print phi values and un-biasing (un-normalized) weights
awk '{if($1!="#!") print $2,exp(($4-bmax)/kbt)}' kbt=2.494339 bmax=$bmax COLVAR_REWEIGHT > phi.weight
```

If you inspect the phi.weight file, you will see that each line contains the value of the dihedral ϕ along with the corresponding (un-normalized) weight w for each frame of the metadynamics trajectory:

```
0.907347 0.0400579
0.814296 0.0169656
1.118951 0.0651276
1.040781 0.0714174
1.218571 0.0344903
1.090823 0.0700568
1.130800 0.0622998
```

At this point we can apply the block-analysis technique (for more info about the theory, have a look at Trieste tutorial: Averaging, histograms and block analysis) to calculate the average free energy across the blocks and the error as a function of block size. For your convenience, you can use the do_block_fes.py python script to read the phi.weight file and produce the desired output. We use a bash loop to test block sizes ranging from 1 to 1000:

```
# Arguments of do_block_fes.py
# - input file with CV value and weight for each frame of the trajectory: phi.weight
# - number of CVs: 1
# - CV range (min, max): (-3.141593, 3.141593)
# - # points in output free energy: 51
# - kBT (kJoule/mol): 2.494339
# - Block size: 1<=i<=1000 (every 10)
#
for i in `seq 1 10 1000`; do python3 do_block_fes.py phi.weight 1 -3.141593 3.141593 51 2.494339 $i; done
```

For each value of block size `N`, you will obtain a separate `fes.N.dat` file, containing the value of the ϕ variable on a grid, the average free energy across the blocks with its associated error (in kJ/mol) on each point of the grid:

```
-3.141593       23.184653     0.080659
-3.018393       17.264462     0.055181
-2.895194       13.360259     0.047751
-2.771994       10.772696     0.043548
-2.648794        9.403544     0.042022
```

Finally, we can calculate the average error along each free-energy profile as a function of the block size:

```
for i in `seq 1 10 1000`; do a=`awk '{tot+=$3}END{print tot/NR}' fes.$i.dat`; echo $i $a; done > err.blocks
```

and visualize it using `gnuplot`:

```
gnuplot> p "err.blocks" u 1:2 w lp
```

As expected, the error increases with the block size until it reaches a plateau in correspondence of a dimension of the block that exceeds the correlation between data points (Fig. master-ISDD-2-block-phi).
What can we learn from this analysis about the convergence of the metadynamics simulation?

11.2.6 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Setup and run a metadynamics calculation.
- Compute free energies from the metadynamics bias potential using the sum_hills utility.
- Reweight a metadynamics simulation.
- Calculate errors and assess convergence.

11.3 PLUMED Masterclass 22.1: Funnel-Metadynamics

Authors

Stefano Raniolo

Date

January 31, 2022

11.3.1 Aims

The purpose of this Masterclass is to understand and test Funnel-Metadynamics (FM) in the study of a binding event for a paradigmatic model with PLUMED.

11.3.2 Objectives

Once this Masterclass is completed, users will be able to:

- Understand the basics to perform FM simulations.
- Experience the use of Visual Molecular Dynamics (VMD) extensions to create the PLUMED input and some post-processing files.

- Compute the binding free energy.
- Assess the convergence of FM simulations.
- Obtain a basic understanding about possible pitfalls in the approach and how to circumvent them.
- Obtain energetic and structural data on the ligand binding mechanism useful to run ligand binding kinetic calculations.

11.3.3 Prerequisites

In order to easily follow this masterclass, users should be able to:

- Have a basic understanding of how a PLUMED input file is structured.
- Have experience with molecular dynamics engines, such as Gromacs (preferably), Amber, or NAMD.
- Have a basic knowledge about PLUMED outputs and post-processing tools (e.g., driver).
- Know how to operate in VMD to visualize molecules.

Most of the required information can be found in PLUMED Masterclass 21.1: PLUMED syntax and analysis and PLUMED Masterclass 21.4: Metadynamics.

11.3.4 Overview of the theory

Unveiling the binding mechanism of a ligand to its molecular target is an important information to perform a successful drug design campaign. PLUMED allows to accurately compute binding free energies, thus allowing us to thoroughly identify the lowest energy ligand binding modes and binding pathways. The FM is a variant of Metadynamics, which enhances the sampling of ligand binding/unbinding towards its molecular target through the use of a funnel-shaped restraint. The latter is constructed as an external potential that limits the sampling of the ligand to the binding pocket and a slim region in the solvent, accelerating convergence and avoiding wasting time in sampling useless unbound states. In the following, we will learn how to use PLUMED to perform a FM simulation following the FM Advanced Protocol (FMAP) [89]. Below it is reported a recap of the basic metadynamics theory.
In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t\to\infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t\to\infty) = -\frac{\Delta T}{T+\Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard MD, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T + \Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135] [136].

11.3.5 Setting up the software

The users can refer to the procedure introduced in PLUMED Masterclass 21.1: PLUMED syntax and analysis and PLUMED Masterclass 21.3: Umbrella sampling to install the required software. It is suggested that an optimized version of `GROMACS` is properly installed since the exercises will also require a simulation as input. Please remember to activate the module for the FM as reported `here`.
Please also install a working version of VMD that will be used together with two graphical user interfaces (GUIs) to create the PLUMED input file and analyse the results during the exercises. More information to download and install VMD can be found `here`.
The GUIs require the tcl tooltip library, which is not implemented in the software. You can download the library from its `github` repository. You should also download the GUIs themselves, which can be found in the folder funnel_gui in their `github` repository. No further installation is necessary, but in order to source the GUI the following command must always precede:

```
source /place/where/you/downloaded/tooltip.tcl
```

Therefore, an example on how to open the GUI to set up a FM simulation (e.g., funnel.tcl) would be to write the following lines in the VMD Tk console:

```
source /folder/where/you/downloaded/tooltip.tcl
source /folder/where/you/downloaded/funnel.tcl
```

During this masterclass, a request to source a GUI (funnel.tcl or ffs.tcl) will always translate to the above commands. You can find a detailed version of the procedure in the FMAP protocol paper [89].

11.3.6 Resources

The data needed to conduct the exercise shoulb be generated with VMD and a molecular dynamics engine. In case of problems in generating the requested data, an example can be found in `Zenodo`. Inside you will find:

- `FMAP_procedure.pdf`: a pdf resuming all the steps in FMAP;
- `README.md`: a file containing general information about installation and setup;
- `Supplementary_data_2`: a folder containing example data for the exercises.

In details, the latter includes:

- `2D-distance.dat`: an example of a 2D free-energy surface (FES);
- `BFES`: a folder containing 80 files representing the 1D FES with respect to time (1 FES per 10 ns);
- `COLVAR`: an example of output file form the FM simulation;
- `driver.dat`: an example of input file for driver;
- `fes_lr.dat`: an example of low resolution FES;
- `HILLS_800ns`: an example of output file for a 800 ns FM run;
- `plumed.dat`: an example of PLUMED input file for a FM simulation;

- `start.pdb`: an example of coordinates file for the FM simulation;
- `trj.trr`: a strided version of a trajectory obtained through FM simulation.

However, we strongly suggest to create new data in order to experience first-hand all protocol's steps.
All the compulsory files for the system of interest can be found in a dedicated repository:

```
git clone https://github.com/h2nch2co2h/masterclass-22-1
```

This repositoy contains 2 folders:

- `data`: GROMACS input files, topologies, and index file for the system that we are going to simulate;
- `slides`: the presentation of the FM theory presented during the first lesson.

We will simulate a paradigmatic system generally used to test and benchmark new binding free-energy approaches, the benzamidine-trypsin system (see Fig. masterclass-22-1-system.png). Although benzamidine is one of the simplest small molecules you could find in the ligand-target binding world, this system represents a good compromise between user-friendliness and biological importance. The binding happens in a very well-known pocket of trypsin, which has been thoroughly outlined by several X-ray structures. The amidine group in benzamidine is positively charged in physiological conditions and interacts with a negatively charged residue in the pocket (i.e., ASP171). In this masterclass we will try to analyse the binding/unbinding events between these two partners, with a particular focus in the estimate of the binding free energy. Additional information about convergence checks and the study of the saddle points in the lowest binding pathway will also be provided.

11.3.7 Exercises

11.3.7.1 Exercise 1: FM input generation

As first exercise, we will generate a PLUMED input file that could be used to run a FM simulation. For this purpouse, we will need a starting structure of the system under study that will be used to set up the funnel-shaped potential. As a general advice, this structure should be the output of the equilibration step (i.e., with the system brought at the correct simulation parameters and ready for the production run). For this exercise, the structure has been provided in the masterclass github (start.gro). Open VMD and source the first GUI (funnel.tcl) as explained in Setting up the software. Load the structure of the system and, when the simulation box will be visualised, call the interface by using the following command in the Tk console of VMD:

```
funnel_tk
```

This should open the interface and display a tentative orange funnel in VMD masterclass-22-1-funnel. The user is now able to customise the funnel with respect to the benzamidine-trypsin system.
An accurate description of all the steps to set up the funnel can be found in the Nature Protocols paper raniolo2020ligand. Here, we will briefly discuss the compulsory points for the simulation to start.
Hint for the user: in VMD all values are reported in Angstrom, whereas PLUMED works in nm. The conversion should be taken into account by the interface but always be careful about possible inconsistencies.
The first thing to correct is the orientation of the funnel, which can be done by selecting atoms in VMD or by changing the Cartesian coordinates of the points A and B that create the funnel. You can find these values in the boxes in the upper part of the interface, whereas the option to assign A or B to an atom can be activated through the switches directly below. For the sake of this exercise, the provided structure already has the ligand in the binding pocket, which will ease the setup process. However, this knowledge might be missing in other systems, thus requiring preceding docking calculations or an extensive scanning of the protein surface with FM.
Hint for the user: hovering over the options in the interface should display a small text explaining what each one does.
By visualizing the protein as new cartoon and the ligand (resname MOL) as licorice, it is possible to clearly see the binding pocket. Try to provide an acceptable orientation for the funnel, with the conical section faced towards the protein and the cylindrical one describing a possible unbindng route for benzamidine (see Fig. masterclass-22-1-funnel).
Hint for the user: choosing to assign points A and B by picking atoms is the fastest choice, just remember to press r after the selection to deactivate the pick mode.
At this point we only corrected the direction, so we should adapt all the other parameters to adjust the cone/cylinder ratios and depth. This is done through the following options:

- `Zcc`: switching point between cone and cylinder, value 0 for Zcc corresponds to point A;
- `Alpha`: angle defining the width of the cone;
- `RCyl`: radius of the cylinder.

Select a set of parameters that allow for benzamidine to explore the binding pocket (inside the conical section of the funnel) and unbind without colliding with the protein on the way out (inside the cylindrical part). A good choice in positions of points A and B and in parameters will allow for a swift convergence without possible artefacts in the simulation. Overall, the funnel-shaped restraint must cover completely the entrance of the binding pocket, including the residues at the edge since they can have a recruiting effect for the ligand. During the sampling the ligand will be simplified to its center of mass and will explore all the allowed volume. This should be considered for relatively big ligands, since covering large areas with the funnel might mean the loss of simulation time in exploring states that are not interesting in the binding process. For the current exercise, try not to have the funnel covering pure-solvent sections of the box and be as strict as possible to the binding site.
Once the funnel has been finalised, a few other parameters must be specified in order for the FM simulation to start. The funnel potential will be constructed as a grid file, so we need to provide dimensions to delimit it. This dimensions are defined by the following parameters:

- `Min fps.lp`: minimum value along the funnel axis (taking point A as the 0);
- `Max fps.lp`: maximum value along the funnel axis (taking point A as the 0).

At this point a bit of history. In the first version of FM, the dimensions of the grid were called linepos and linedist, which stands for the position on the funnel axis and distance from it, respectively. They have been transformed in fps.lp and fps.ld, with fps that stands for funnel positioning system. Limits for ld must not be specified since the minimum is 0 by definition and the maximum is computed based on the other parameters that have been provided (i.e., Alpha and min fps.lp).
Hint for the user: Min fps.lp can take either positive or negative values. For example if point A has been positioned deep in the bulk of the protein, positive values might be advisable since the ligand will never be able to pierce the protein. As for max fps.lp, the ligand should be able to sample portions of the solvent outside of the influence of the protein (e.g., the cutoff in the case of simulations). Since the solvent is sampled in the cylindrical part of the funnel, around 3-4 Angstroms of fps.lp should be assigned for the solvent sampling, outside of the protein electrostatic and vdw cutoffs.
With the edges of the grid decided, two walls are applied to enforce the sampling in the allowed region. This is necessary because if the center of mass of the ligand goes out of grid for any reason, the algorithm will output an error and the entire FM simulation will crash. For this reason, you can find the Low. wall and Up. wall parameters in the interface that should be set leaving 2-3 Angstroms from the grid edges. For example, with Min fps.lp and Max fps.lp of 0 and 40, a Low. wall of 2 and an Up. wall of 38 might be a good compromise.
After all the previous steps have been completed, make sure that the value at the right of ID is the same index as the structure that you used to set up the funnel, write in the box at the right of Ligand the atoms that should be recognised as part of the ligand in the VMD syntax and press enter. For example:

```
resname MOL
```

will consider all atoms of the residue called MOL (which in this case are all atoms of benzamidine).

```
resname MOL and noh
```

will instead select only the heavy atoms of residue MOL.
When ready, click on Apply and a PLUMED input file should be generated with all required options. You can print it to a file through the export feature. Please, check that all values have been correctly trasformed in the measurement unit accepted by PLUMED.
The newly created file contains all the compulsory lines to run a FM simulation. Few commands like WHOLE↩ MOLECULES and RESTART can also be commented and will be discussed in the next exercise. Others must be completed by the user either with the interface, a text editor, or by hand in the VMD canvas. Here, we will briefly describe the missing parameters for the funnel, leaving to the user the job to complete the rest (i.e., METAD, LOWER_WALLS, and UPPER_WALLS).

- `REFERENCE`: requires the name of the file that will be used to align the funnel to the protein during the simulation. Save the current system as a pdb and provide here the name of the pdb;

- ANCHOR: a protein atom that is close to the ligand along the supposed binding pathway. This is necessary in some cases to avoid periodic boundary conditions (PBC) problems and should be interactively chosen as previously done for point A and B;

- KAPPA: potential constant to generate the funnel-shaped restraint, requires a number in kJ/mol/nm^2 (for benzamidine around 35000 is fine).

Hint for the user: the drop-down menu contains all parameters that require an entry in order to work. Depending on the option, VMD selections or numbers should be supplied. Please remember to always press enter after selecting an option and providing the wanted value. If you changed anything, please remember to export again the input file. The user is now free to customise his/her input. Be aware that modifications done by hand on the canvas will be overwritten upon pressing Apply.
This concludes the first exercise.

11.3.7.2 Exercise 2: My first FM simulation

In this exercise we will finally run the FM simulation (applause noises). But before that, all necessary lines in the PLUMED input file created in Exercise 1: FM input generation must be completed. We remember that FM rely upon metadynamics to work, therefore the description of collective variables (CV) and all the necessary knownledge to set up a well-tempered Metadynamics are given for granted. The user can still refresh his/her memory with PLUMED Masterclass 21.4: Metadynamics. Suffice to say that in this case a distance between protein and ligand or fps.lp itself could be used as an argument for METAD.
The file runme.tpr has already been prepared for you to run and we will imply that the file from exercise 1 has been called plumed.dat. All you need to do is execute the following command:

```
gmx mdrun -s runme.tpr -deffnm firstFM -nsteps 125000000 -plumed plumed.dat
```

This represents 250 ns of FM simulation, which is unlikely to finish with a single-task launch. The objective of the exercise is not to reach convergence, but, depending from the computational resource at one disposal, an MPI job could achieve better results:

```
mpirun -np X gmx mdrun -s runme.tpr -deffnm firstFM -nsteps 125000000 -plumed plumed.dat
```

We also remember that gpu could be called with the -gpu_id flag and proper resource management can be controlle with the various -ntomp, -ntmpi, etc. For maximum performances, we would like to mention that FM supports WALKERS_MPI (the flag should be added both in METAD and FUNNEL in order for it to work).
Hint for the user: please remember that to use multiple walkers, the BIAS file, where the funnel potential is stored, and the HILLS file should be placed in the parent folder of the walkers by setting FILE=../BIAS and FILE=../HILLS as per last policy of GROMACS at this juncture. Please also be aware that several GB of files will be created, so avoid running such a calculation on a laptop.
Similarly to what has been seen for standard metadynamics, we can check the simulation through the following command:

```
plumed sum_hills --hills HILLS --stride 500 --mintozero
```

This should generate a free-energy surface (FES) file each 0.5 ns of simulation and comparing them with respect to time we can easily note if the profile does not change considerably. This is by no means a clear indication of convergence but it is one of the many checks that can be carried out to understand when to stop the simulation. Theoretically, depending also on the metadynamics parameters, 200 ns might not be enough to obtain full convergence, but could still be used for the post-processing phase.
At this point the user should just wait for the simulation to finish or to gather as much statistics as possible. During the simulation, the FM might crash giving an out of grid error. This is caused by the ligand jumping to a periodic image and thus abruptly leaving the grid representing the funnel potential. Two main scenario could happen:

- the error happens at the very first step of the simulation: please check that the pdb file provided in REFERENCE contains only protein atoms, ANCHOR has been correclty defined, or use WHOLEMOLECULES as a last resort.

- the error happens in the middle of the simulation: change ANCHOR atom or again employ WHOLEMOLECULES to solve the problem.

Hint for the user: an imperfect setup of FM can drastically decrease the performance of the simulation. However, FM can be as fast as a standard Metadynamics with the correct precautions. One of the biggest time sinks is the alignment repeated for each simulation step to ensure the correct funnel placement with respect to a rotating and translating protein. If a full pdb is provided in REFERENCE, computing the roto-translational matrix will become a heavy task. We suggest to limit the number of atoms used to only CA atoms of structured portions of the protein. The user is encouraged to try and run few thousand steps (-nsteps flag) with different pdb files to see the difference in performance.
Regardless of convergence and time simulated, congratulations! You run your first FM simulation.

11.3.7.3 Exercise 3: Post processing

With the production run terminated, we can focus on analysis to check for convergence and compute the binding free energy for the benzamidine- trypsin complex. Firstly, we will repeat the same analysis done in exercise 2 but with the VMD interface (see Fig. masterclass-22-1-ffs.png). Please open VMD and source the second GUI called ffs.tcl. To display the interface you can write in the Tk console of VMD the command:

```
ffs_tk
```

The user can immediately see that all options are on the right of the interface and most of the space is occupied by the central canvas that is used to plot graphs. As previously said, let's try generating FES files starting from the HILLS file created during the production run. For a sum_hills calculation, compulsory parameters are:

- the plumed binary (first option);
- the HILLS file (sixth option);
- the mintozero (seventh option). Position for the output files can be selected with the `Output` button and the last thing to do is to press `RUN sum_hills LR` or `RUN sum_hills HR`, depending on the required resolution of the FES file (low or high resolution, respectively). This would produce a single FES file that takes into account the entire simulation time.

Our objective though is to check the differences of the FES with respect to time. Therefore, we can provide a stride value to print the FES every n steps and click on `RUN Sum_hills stride`. The process might take some time to complete, so wait for VMD to finish all calculations.
Hint for the user: the log of each job (e.g., `RUN Sum_hills LR`, `RUN Sum_hills HR`, and `RUN Sum_↩ hills stride`) will be written in the Tk console of VMD. From there, the user can check that everything went smoothly.
By pressing `Plot FES`, one of the generated FES can be plotted in the VMD canvas. Depending on the graph resolution, the time requested to visualise it might change and up to 3D plots can be represented (the third dimension being transformed into a chromatic scale).
Each and every FES can provide a value for the binding free energy with respect to time and we can calculate the difference in binding free energy with the following formula:

$$\Delta G_b = -1/\beta ln(C_0 \pi r_{cyl}^2) \int_{bound} e^{-\beta(W(z)-W_{ref})} dz,$$

with C0 being the standard concentration of 1 M for all reacting molecules, β the inverse of the Boltzmann constant multiplied by the temperature in Kelvin, W(z) the value of the potential mean force (PMF) in the bound poses, and Wref the PMF in the unbound region.
If you used fps.lp as your CV of selection for metadynamics then you are set. Otherwise, the user is requested to reweight with respect to it and create again the FES files in function of fps.lp. Instructions on how to reweight on an auxiliary CV can be found in PLUMED Masterclass 21.4: Metadynamics. This step is necessary because the entropic correction contained in the above formula considers the variable of intergration as the position along the funnel.
Hint for the user: if you reweighted your simulation, make sure that the created files have the same name as the files generated by sum_hills (e.g., fes_1.dat). The interface automatically takes all the files with that format and will not recognise any other FES file.
Before, the user had to create his/her own script to compute the bindng free energy, now though the value is provided by clicking on `Calculate!`. The bound interval W(z) can be defined by using the `min_x` and `max_x` boxes, whereas the reference value Wref for the unbound must be written in the corresponding box. The user should try to compute a free energy from a FES of choice picking the intervals more appropriate to the results obtained from

the production run. The estimate will apper at the right of the button, already rescaled in kcal/mol and taking into consideration the entropic correction coming from the presence of the funnel. Repeating this analysis for different simualtion times will give the behaviour of the binding free energy with respect to time, which is an additional check to verify convergence of the simulation. In particular, a converged simulation should present a stable value for the free energy with the progress of time. This can be done by pressing the `Convergence` button, which will take all the fes files generated by sum_hills in the working directory (spedified by the `Output` option). Upon clicking, the user should see the new plot of the free energy in function of time. Moreover, the user can find the average-on-the-fly and the standard deviation of the computation in the VMD TK console.
Hint for the user: the calculation of the binding free energy with respect to time is initially done taking the whole dataset, however the first steps in metadynamics are generally discarded from statistical measurements. This can be done by specifying a value for the reject time in units of FES stride numbers. Therefore if you generated one FES per ns of simulation, the first 100 ns in a 800 ns production run can be erased by putting 100 on the box at the right of `Rej. time`. This will most likely reduce the error of the calculation and provide a more accurate estimate.
As a further instrument to check convergence, the interface incorporates a block bootstrap analysis. This approach takes the same interval considered for the calculation of the average-on-the-fly and standard deviation, but divides the remaining dataset in 10 blocks. Each of them will be subjected to a bootstrap analysis, obtaining in the end the mean and standard error for each block (visible in the Tk console) and a visual representation of how the distribution of values behaves in the block. For converged simulations, averages of the last blocks should be very similar and their distribution should resemble a normal distribution. Deviation from this behaviour might indicate missing convergence.
The user is encouraged to try different reject times with both the convergence tool and the block bootstrap analysis.
Hint for the user: this analysis might not be possible if you could not simulate a statistically-relevant amount of time. In fact, the approach requires a minimum of 500 points in total (50 per block) and will stop otherwise. This problem can be circumvented by reducing the stride in the generation of the FES, but this workaround is advised against since there would not be much difference among FES files not due to convergence but because the system would not have enough time to evolve. The user can use the 800 FES that have been prepared on Zenodo as a substitution.
Another analysis that can be done to check convergence in a binding free-energy calculation is to plot the sampling of the selected CVs with respect to time to see if the ligand was able to bind/unbind a number of times, thus gathering sufficient statistics on the binding process. This can be achieved by plotting with a software of choice the columns of time and collective variable (CV) used for the FM simulation (e.g., gnuplot, xmgrace, matplotlib, etc.).
Hint for the user: using multiple walkers, it is possible to experience a compartmentalization effect, each walker exploring only a subset of the whole CV space. To check if the potential has been correctly placed, the user can launch a single walker loading all the bias that has been added and should observe the binding/unbinding in a relatively low number of steps.
We will finish this exercise with two more interactive features of the interface. Up to now, we only spoke about graphs and distributions, but the user should always observe what happened in the simulation box. For this reason, please process your production run aligning the protein at the center of the box, displaying only benzamidine and trypsin (group `Complex` in the index file complex.ndx), and striding the simulation so as to have around 100 frames. This can be done through the `trjconv` function of GROMACS and the use of the flags `-pbc` and `-fit`.
Example:

```
gmx_mpi trjconv -f simulation.xtc -n complex.ndx -s runme.tpr -pbc nojump -o sim_nojump.xtc -dt 10000
gmx_mpi trjconv -f sim_nojump.xtc -n complex.ndx -s runme.tpr -fit rot+trans -o sim_aligned.xtc
```

The user can employ a combination of his/her own choice, it will just help in visualising the structures in VMD. The newly created trajectory can be loaded in VMD together with a coordinate file through the `New Molecule` and `Load Data into Molecule` options in `File`. Once this is done, we can track the state of each frame in the FES. For this, we will need one of the FES generated in the previous steps and a COLVAR file that has a number of lines equal to the frames in the loaded simulation. This job can be achieved by using the `driver` functionality of PLUMED either by hand or through the interface. Since the former is given for granted, we will proceed following the latter. The compulsory options to run a `driver` calculation on the interface are:

- the plumed binary (first option);
- the PLUMED input file (second option);
- the trajectory file (third option);
- the trajectory extension type (fourth option);

- a pdb file (containing masses that will be used to compute CVs, fifth option).

As always the output shall be saved in the folder specified by `Output` and when everything is ready you can just press `RUN Driver`. The log of the process will be displayed in the Tk console.
As for the PLUMED input file for the driver, we can use a revised version of the input used for the production run, extracting only the lines that are necessary to what we are interested (i.e., the value of a selected CV in the strided trajectory). For this exercise, we will use the same CV used in the FM simulation, therefore the PLUMED input file should resemble something like this:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-masterclass-22-01.txt
# Put here the CV that you used for the FM simulation (e.g., fps.lp)

lig: COM ATOMS=__FILL__

fps: FUNNEL_PS LIGAND=lig REFERENCE=__FILL__ ANCHOR=__FILL__ POINTS=__FILL__

# Print with the same stride as the trajectory that we processed

PRINT ARG=fps.lp FILE=COLVAR STRIDE=1
```

Hint for the user: for the feature we are going to see, the interface takes a file named COLVAR as input, this could conflict with the COLVAR generated during the production run. It is suggested to change the name of the FM COLVAR to avoid overwriting files.
Once the driver finishes the job, plot the last FES of you FM simulation with the `Plot FES` option and click `Trace` on the interface. A vertical line should have appeared in the canvas, signaling the position of the visualised frame in the FES. This function can be used to navigate through states and analyse the most important interactions in all the energetic minima explored by FM. Changing frames will automatically update the position on the FES.
However, broad minima could contain several different states and the user might be interested in clustering them to better define the structure families. The interface allows to extract poses from the displayed simulation by right clicking and dragging a selection interval in the canvas of VMD or by providing an extraction interval in the same boxes used for the convergence calculation. By setting the interval of interest in `Min_x` and `Max_x` and pressing `Extract`, a folder will be created in the working directory (defined by the `Output` option) with inside pdb structures of all the frames belonging to the interval of choice. At this point the user can employ this selection of structure for a cluster analysis with `gmx cluster` ot whatever tool of his/her own choice.
In addition to all the analyses that we listed, all the checks described in previous masterclasses for Metadynamics-based approaches apply to FM too (e.g., the height of the gaussians in the well-tempered regime), but this concludes the basic protocol to post-process a FM calculation. In the end the user will have an accurate estimate of the binding free energy, a thorough collection of binding modes for the ligand, and a qualitative information about the minimum energy path for the ligand to go from bound to unbound and vice versa.
This analysis could be completed with an estimation of the kinetic rates starting from all the states of minimum extracted from the interface and reconstructing a flux diagram with data coming from Infrequent Metadynamics and Committor analyses.
This ends the exercise.

11.3.7.4 Bonus exercise 4: Infrequent Metadynamics

This bonus exercise will give an idea on how to proceed if a more thorough study of the kinetic is in order. We will only see one application starting from the absolute minimum of our benzamidine-trypsin system, but potentially all minima extracted from the last point of Exercise 3: Post processing could be studied.
In order to correctly identify a transition state and estimate the barrier connecting two basins, we must not compromise it by adding bias. This is a delicate procedure since we still would like to speed up the sampling but up to the point when the system could jump from one state to the other on its own. In order to do that, we need to slighlty revise the input of a normal Metadynamics run, selecting a relatively low height for the gaussians and extending the rate of deposition. An example would be:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-01.txt
# Put here the CV that you used for the FM simulation (e.g., fps.lp)

dist: DISTANCE ATOMS=2472,3224

METAD ...
LABEL=metad ARG=dist PACE=4000 HEIGHT=0.6276 SIGMA=0.001
BIASFACTOR=10.0 FILE=HILLS TEMP=300.0
```

```
GRID_MIN=0.0 GRID_MAX=4.0
GRID_WFILE=grid_w.dat
GRID_WSTRIDE=50000000
... METAD

PRINT FILE=COLVAR STRIDE=4000 ARG=*
```

In this input, the distance between an atom of the protein and one of the ligand will be a reliable CV to determine the jump between different metastable states. This kind of simulations should be run in batches in order to gather enough statistics. Considering the low amount of bias introduced, this could seem as a big sink of computational power, but consider that the simulations will be interrupted as soon as we see the transition of interest.
However, let's see an example to better understand what to do. In masterclass-22-1-infr we can see the sampling of the distance in one of the replica that was run for the benzamidine-trypsin system starting from the absolute minimum.
width 60%
Although the simulation lasted around 65 ns, our point of interest lies at around 22 ns (orange line), when we have a clear change in the sampling of the distance. Here, the absolute minimum was left thanks to the metadynamics bias added. We can measure the added bias up to that point by looking at the HILLS file. This information is important because, albeit low, the infrequent Metadynamics provides an acceleration booster to the simulation that must be estimated in order to extract the unbiased time for the transition.
The acceleration factor can be calculated using the following formula:

$$\alpha =< e^{\beta V(s,t)} >,$$

where s is the collective variable of choice, β is the inverse of the Boltzmann constant multiplied the temperature in Kelvin, and V(s,t) is the bias felt at time t of transition [54]. Rescaling the transition time by this term will yield the real transition time that one would expect from an unbiased simulation. But how to obtain the characteristic transition time of the system starting from corrected transition times?
One practical way is to construct an empirical cumulative distribution function with the various corrected time values found in the independent replicas of infrequent Metadynamics. This is done by plotting on the x axis the time and on the y axis the percentage of simulations that evolved from our starting state to another. This scatter plot must then be fitted by a theoretical cumulative distribution function, which has the following equation:

$$TCDF = 1 - exp(-t/\tau),$$

where t is our transition time and τ is the characteristic transition time for the system of interest. The reliability of the fitting can be checked through the Kolmogorov-Smirnoff test.
The value thus obtained corresponds to the time requested on average for the system to move from our minimum to another well-defined state (state P) [54].
The same exercise can be done waiting for the ligand to leave the binding site, in which case we would be measuring the time requested for unbinding. The rate constant between the bound and unbound states can be approximated by dividing the number of transitions between the two by the total time the system spent in the bound state (calculated using the metadynamics simulation time and the acceleration factor as reported above).
This concludes the masterclass, thank you for your participation.

11.4 PLUMED Masterclass 22.2: Analysis of Plumed output by Metadynminer

Authors

Vojtech Spiwok

Date

February 14, 2022

11.4.1 Aims

The aim of this Masterclass is to introduce users to the R package Metadynminer for analysis of results from metadynamics in Plumed.

11.4.2 Objectives

Once this Masterclass is completed, users will be able to:

- Install R and its packages.
- Load metadynamics results (`HILLS` file) to Metadynminer.
- Calculate a free energy surface and plot it in a publication quality.
- Locate free energy minima and evaluate their relative free energies.
- Make an animation of the evolution of a free energy surface.
- Use data in Metadynminer objects to do advanced calculations (e.g., reweighting).

11.4.3 Setting up the software

11.4.3.1 Installation

We will use `R` installed from `CRAN`. Go to `CRAN` and follow the installation instructions for your operating system. R is Open Source. It is not necessary to use a graphical user interface but it is very useful, especially for making illustrations.
If you plan to use a graphical user interface, the best option is `Rstudio`. `Desktop version of Rstudio` is Open Source. Follow the installation instructions for your operating system.
To install metadynminer, open R (in commandline type `R` or open GUI by typing `rstudio` or using an icon) and type:

```
install.packages("metadynminer")
```

The output should contain: `* DONE (metadynminer).` If you have an output: `installation of package metadynminer had non-zero exit status` there was a problem with installation. You may try:

- On MS Windows install `Rtools`.
- If you have permission issues, you can try the commands `Sys.getenv("R_HOME")` and `Sys.getenv("R_LIBS_USER")` and check the permission of the printed directories. You may change installation directories by corresponding `Sys.setenv(...)` commands.

You may check whether Metadynminer was installed correctly by typing:

```
library(metadynminer)
```

This command should have no output. Installation is done only once (or can be repreated for version upgrades). Loading of the package by `library` command must be done in every R seance.

11.4.4 Resources

The data needed for this Masterclass can be found on `GitHub`. You can clone this repository using the following command:

```
git clone https://github.com/plumed/masterclass-22-02.git
```

The repository contains hills and colvar files, respectively, from 10 or 1 ns metadynamics simulations of alanine dipeptide in water calculated in Plumed. Other input data are available from `Zenodo` but are not required in this tutorial. The repo contains:

- `HILLS.cv12` and `COLVAR.cv12` - hills and collective variable + bias report from metadynamics with phi and psi dihedral angles
- `HILLS.cv1` and `COLVAR.cv1` - hills and collective variable + bias report from metadynamics with phi dihedral angle
- `HILLS.cv2` and `COLVAR.cv2` - hills and collective variable + bias report from metadynamics with psi dihedral angle

This data are independent of Plumed version (for version >=2.0).

11.4.5 Using R

R is a program and scrpting language predominantly developed for statistical data analysis. It is popular in statistics, economy, ecology, bioinformatics, and other fields. Let us quickly introduce R. Open R as described above and type:

```
1+1
2-1
3*3
5/2
```

This shows addition, subtraction, multiplication, and division. Numbers and other objects can be saved to variables. We advice users to avoid the names of variables that can be confused with existing functions in R. In this masterclass, we will use the prefix "my" (e.g., `myhills`) to avoid this. Variable names are case-sensitive. The operator used to save something to is an arrow `<-` (less than followed by minus sign). You may see R codes with a normal equal sign, but we will be puristic in this. Let us show the variables in R:

```
x <- 1+1
x
```

In the outputs above, there is an unexpected string `[1]`. This is the index of the first item on the line. This is useful when dealing with vectors. Vector can be created by implicitly typing its elements to the function `c()`, by `:` operator and other ways:

```
x <- c(1, 3, 2)
x
x <- 1:50
x
x <- 50:1
x
```

R can also work with 2D and multidimensional matrices. R can work with strings (written in quotes, e.g. "blue"), Booleans (`TRUE` and `FALSE`, or shortened as `T` and `F`).
R uses functions written as the name of function followed by its arguments in brackets:

```
exp(1)
cos(pi)
```

Functions can be applied element-wise:

```
x<-0:10
cos(x)
```

Functions with more variables use the specifications set by `=`, separated by `,`:

```
x<-0:100/10
y<-cos(x)
plot(x, y)
plot(x, y, xlab="x axis", ylab="y axis", ylim=c(-2,2))
```

Important difference from many other programming environments is that R can freely change number types, e.g. `3/2` is evaluated as 1.5, unlike Pyhton, which gives 1, because 3 and 2 are integers, and not floats.
Another important difference is that R uses indexes of vector items starting with 1 (not 0 line in C/C++, Python etc.). For example:

```
x <- c("Carlo", "Gareth", "Giovanni", "Max")
x[2]
```

will return Gareth. In contrast, Python returns Giovanni:

```
x = ["Carlo", "Gareth", "Giovanni", "Max"]
x[2]
```

Finally, R can use other types of objects, including objects defined by a user. Their instances can be accessed by a string operator. For example, the function `prcomp` performs a principal component analysis. Its output is an object with multiple instances, for example, with a rotation matrix:

```
pcamodel <- prcomp(outer(1:3,1:2))
pcamodel
pcamodel$rotation
```

Finally, to quit R you can use the function `q()`. The program asks the user whether he/she wants to save the data (i.e., variables created in the previous course of the run). We advise users to NOT save data, i.e. choose the option "n" or function `q(save="n")`.

11.4.5.1 Exercise 1: Calculation of the free energy surface from metadynamics hills

Open R (in Rstudio or command line) and load Metadynminer:

```
library(metadynminer)
```

Now load the file `HILLS.cv12` into Metadynminer:

```
myhills12 <- read.hills("HILLS.cv12", per=c(T, T))
```

At this point, it is necessary to set the periodicity of CVs. The free energy surface was calculated using torsion angles as CVs, which are periodic. It is necessary to set the periodicity to `TRUE` (or simplified as `T`) as a vector `per`. If you get an error message that the file was not found, it means that the home directory is not set properly. In Linux command line, the home is the directory where you start R by the command `R`. In Rstudio, you may navigate to the directory with the file using the File menu in the right bottom frame. Alternatively, you can use `getwd()` and `setwd("/path/to/your/dir")`.
Hills can be also loaded from a HILLS file posted online by replacing the file name by an URL.
To get help to any function in R, for example, `read.hills`, you can use either `help(read.hills)` or `?read.hills)`.
If you type the name of the hills file variable `myhills12` you get an information about the number of hills and CVs. The function `summary` (`summary(myhills12)`) returns the same information plus the ranges of CVs.
A hills file can be plotted by the `plot` function, i.e., `plot(myhills12)`. For a metadynamics with one CV it plots the evolution of the CV as a function of time. For a metadynamics with two CVs it plots a scatter plot with CV1 on the horizontal axis and CV2 on the vertical axis. The plot function applied on metadynamics hills file is an extension of the standard plot function in R. You may set parameters `xlab`, `ylab`, `main`, `sub`, `xlim`, `ylim`, `pch`, `col`, `bg`, `cex`, `lwd` and `asp` (see `?plot.hillsfile`). It is possible to stack plots on top of each other by using the command `plot(...)`, followed by a command `points(...)` or `lines(...)`.
For well-tempered metadynamics, it may be useful to check the evolution of heights of hills. This can be done by the command:

```
plotheights(myhills12)
```

Hills files are often obtained from multiple metadynamics runs. This may cause that time collumn does not correspond to the real time. The option `ignoretime=T` used either in `read.hills` or in `plotheights` replaces time by the hill number.
The main point is the calculation of the free energy surface from the hills file as an negatve image of the sum of Gaussian hills (scaled in well-tempered metadynamics, which is pre-scaled in the Plumed output).
This may be time consuming with hundreds of thousands of hills. In metadynminer you can use two functions for this purpose, `fes` and `fes2`. The former is fast and approximative, the latter is slow and more accurate. The function `fes` uses a pre-computed Gaussian hill which is simply being shifted on the canvas of the CV space for each hill and the bias potential is summed. The function `fes2` explicitly evaluates the Gaussian function. The function `fes` cannot be used in metadynamics with variable hills widths (the user would be warned).
In our tutorial we can do:

```
myfes12 <- fes(myhills12)
```

(or the same with `fes2` for analyses, final illustations etc). Typing the name of the free energy surface variable `myfes12` returns the number of CVs, number of bins (by default 256 for one CV, and 256x256 for two CVs), free energy maximum and minimum.
It is possible to calculate the value of minimum and maximum of the free energy surface by functions `min` and `max`, respectively. You may add, subtract, multiply, and divide the free energy surface by a constant. For example, you can set the free energy minimum to zero by subtracting the minimum:

```
myfes12 <- myfes12 - min(myfes12)
```

Similarly, to convert the free energy surface from kJ/mol to kcal/mol you may divide it by 4.18.
It is also possible to sum and subtract two free energy surfaces. This will be demonstrated later.
The important function of Metadynaminer is plotting of the free energy surface. You can use:

```
plot(myfes12)
```

Again, you may use options `xlim`, `ylim`, `zlim`, `main`, `sub`, `xlab`, `ylab`, `labcex` and `drawlabels` (see `?plot.fes`). You may control the type of the plot by the `plottype`. The option `plottype="image"` plots a heat map, `plottype="contour"` plots contours, and `plottype="both"` plots both.
It is possible to switch on the free energy scale by:

```
plot(myfes12, colscale=T)
```

You may change the title of the scale by `colscalelab="your label"`.
It is often necessary to control the levels of contours. For example, to plot the free energy surface with 20 contours by 10 kJ/mol type:

```
myfes12 <- myfes12 - min(myfes12)
plot(myfes12, zlim=c(0,200), nlevels=20)
```

You may also explicitly set levels as a vector by the parameter `levels`.
The default color palette for a 2D free energy surface in Metadynminer is `rainbow(135)[100:1]` (i.e., rainbow without violet color). If you prefer your own color palette, you may set it as `col` paramtere. For example, this command plots the free energy surface in gray colors:

```
plot(myfes12, col=gray.colors(50))
```

Metadynminer can identify minima on the free energy surface. This can be done by the function:

```
myminima12 <- fesminima(myfes12)
myminima12
```

The program should find approximately five free energy minima for alalnine dipeptide in water and print their idetifier (letters A, B, C, ... followed by AA, AB, AC, ...), location on the free energy surface (in bin ID and in CVs), and the free energy (for the final free energy surface). The function `summary` prints also populations (temperature and energy units can be controlled by `temp` and `eunit`, respectively).
Some metadynamics simulations may result in rough free energy surfaces with hundreds of free energy minima. It is not useful to identify all of them. It is better to find a reasonable number of minima scattered on the whole free energy surface. The function `fesminima` splits the canvas of the free energy surface into a certain number of sections (by default 8x8 for 2D and 8 for 1D free energy surfaces, it may be controled by `nbins`). A global minimum is found in each section and then it is checked whether it is a local minimum of the entire free energy surface. If not, it is discarded.
It may happen that the function `fesminima` cannot identify some minimum or it is necessary to add a reference point on a free energy surface, which is not a minimum. For this purpose, it is possible to make a minimum by `oneminimum` function and add it to the existing minima (by addition operation).
The output of `fesminima` can be plotted by `plot` function. The plot is similar to the free energy surface and it contains additional letters in the points of minima.

```
plot(myminima12)
```

Convergency of metadynamics can be assessed by the time profile of the differences between free energy minima. Stable difference between free energy minima indicates that the free energy surface is converged and thus accurate. It must be kept in mind that this is a necessary but not the only requirement of convergence. Please follow the PLUMED Masterclass 21.2: Statistical errors in MD for more details. The time profile of the differences between free energy minima can be caluclated and plotted as:

```
myprof12 <- feprof(myminima12)
plot(myprof12)
```

The program takes free energy minima (as identified by the function `fesminima`) and calculates the values of free energy in the same point in the CV space along the simulation. Typing the name of the time profile returns the same output as for minima. The function `summary` prints the same output as the summary of free energy minima together with minimal and maximal free energy relative to the global free energy minimum. The minimum A is always the global minimum and its relative free energy is 0 (also its minimum and maximum is zero). The column `tail` contains the relative free energy at the end of the simulation. It is possible to specify the time window in which the minimum and maximum are calculated, for example the second half of the simulation, by `imind` and `imaxd`. By default they are set to 1 and the number of hills, respectively, i.e. to whole simulation.
Finally, metadynminer provides a function for the nudged elastic band:

```
myneb<-neb(myminima12, min1="A", min2="D")
myneb
plot(myneb)
plot(myminima12)
linesonfes(myneb)
```

11.4.5.2 Exercise 2: Making high resolution plots and movies

In Rstudio it is possible to save plotted figures in different vector and bitmap formats and it is possible to customize their resolution. As an alternative, it is possible to save figures by the functions `png`, `bmp`, `jpeg`, `tiff`, `svg`, `pdf` or `eps`. These functions use the file name as an argument and it switches the plotting environment, so that the file is plotted into the file and not on the screen. This option must be switched off by the function `dev.off()`. These functions work in the basic R (i.e., without Rstudio) and can be used for example at high-performance computing clusters, grids, and clouds without GUI. Our suggestion to make a nice publication quality figure in the size of 8x8 cm (typical half page) is:

```
png("filename.png", height=8, width=8, units='cm', res=600, pointsize=6)
plot(myfes12)
dev.off()
```

The functions for image saving can be used with regular expressions. This is useful for making movies. A movie of free energy evolution can be simply made by:

```
tfes<-fes(myhills12, imax=50)
png("snap%04d.png")
plot(tfes, zlim=c(-200,0))
for(i in 1:199) {
  tfes<-tfes+fes(myhills12, imin=50*i+1, imax=50*(i+1))
  plot(tfes, zlim=c(-200,0))
}
dev.off()
```

Functions `imin` and `imax` make it possible to restrict the summation to certain hill range. The first line calculates the sum of the first 50 hills. The second call of the `fes` function calculates the sum of hills 51 to 100 and adds it to the total free energy surface. The resulting 200 files named `snap0001.png`, `snap0002.png`, etc can be concatenated to a movie by means of some movie editing software.

11.4.5.3 Exercise 3: Making a movie of flooding

The first task in this tutorial is to calculate the free energy surface of alanine dipeptide calculated in the space of two collective variables phi and psi. This can be done by typing:

```
library(metadynminer)
myhills12 <- read.hills("HILLS.cv12", per=c(T, T))
myfes12 <- fes(myhills12)
plot(myfes12)
```

The 2D free energy surface can be converted to 1D by conversion of the free energy to probability, integration of probabilities along one CV and conversion back to free energy. In metadynminer you can do this by the function `fes2d21d`:

```
myfes12.1d <- fes2d21d(myhills12, remdim=2)
```

The option `remdim` sets which CV should be removed (`remdim=2` removes psi). It is also possible to set the temperature and energy units by `temp` and `eunit`, respectively.
Now we will use this free energy surface (calculated from 10 ns metadynamics) as the "ground truth" and we will try how this free energy surface was flooded by 1 ns metadynamics with a single CV, either phi or psi.
Hills file from the other two simulations can be loaded by:

```
myhills1 <- read.hills("HILLS.cv1", per=_FILL_)
myhills2 <- read.hills("HILLS.cv2", per=_FILL_)
```

Replace `_FILL_` by the correct expression.
Next, for time 10 ps to 1 ns (hills 10 to 1000, with an increment of 10), calculate the free energy surface, convert it to a bias potential by multiplication by minus (biasfactor - 1)/biasfactor and add it to the ground truth free energy surface. The bias factor was 15 in this simulation. Plot the result (in black) together with the ground truth free energy surface in red:

```
myfes12.1d <- myfes12.1d - min(myfes12.1d)
flooded <- myfes12.1d
flooded$hills <- c()
png("snap%04d.png")
plot(myfes12.1d, ylim=c(0,100), col="red")
for(i in _FILL_) {
  plot(myfes12.1d, ylim=c(0,100), col="red")
  myfes1 <- fes(myhills1, imin=_FILL_, imax=_FILL_)
  flooded <- flooded + _FILL_*myfes1
  lines(flooded)
}
dev.off()
```

Again, replace each `_FILL_` by a suitable expression. Keep in mind that it is possible to compose the plot from multiple plots. The function `plot` creates the first plot and functions `lines` or `points` add to this plot.
The meaning of the line `flooded$hills <- c()` is following. Objects in R may contain different instances stored as `variable$instance` The object of the free energy surface contains the free energy surface, information about the axes and their periodicity and other information, but also the original hills from which the free energy surface was calculated. Something similar applies also to the object of free energy minima. The reason for this is

that the function `feprof` needs hills data. Summation of free energy surfaces is equivalent to concatenation of hills, so the sum of two free energy surfaces contains concatenated hills as its `$hills` instance. However, in the line `flooded <- flooded + _FILL_*myfes1` it is not possible to concatenate 1D and 2D hills. For this reason we erased hills from the variable `flooded`.
If possible, make a movie or visually inspect the resulting files. Evaluate the flooding by the 1D potential in terms of convergence.

11.4.5.4 Exercise 4: Reweighing

It would be also interesting to see the performance of metadynamics with psi torsion as the only CV. Unfortunately, it is not possible to calculate the free energy surface in the space of phi torsion from this simulation simply by summing hills. This can be done by reweighing. This will also demonstrate the advanced features of metadynminer.
Reweighing predicts what would have been the distribution of a CV or CVs in the biased simulation, if it had not been biased. Umbrella sampling reweighing weights the distribution of the states by exp(+bias/kT). This can be applied to metadynamics bias potential with a correction to the time-dependence of the bias potential. We will use the correction introduced by Tiwari and Parrinello. The main idea is the bias potential is divided into time-independent and the time-dependent component. The time-dependent component is calculated by the following procedure:

```
library(metadynminer)
bf <- 15
nframes <- 50
temp <- 300
myhills2 <- read.hills("HILLS.cv2", per=c(T))
s1<-rep(0, nframes)
s2<-rep(0, nframes)
for(i in 1:nframes) {
  step <- i*length(myhills2$time)/nframes
  onefes <- fes(myhills2, imax=step)
  s1[i] <- sum(exp(-1000*onefes$fes/8.314/temp))
  s2[i] <- sum(exp(-1000*onefes$fes/8.314/temp/bf))
}
ebetac <- s1/s2
```

The variable `bf` contains the bias factor. The R function `rep` generates a vector by repeating an element. Here it generates a vector with `nframes` (50) elements equal to zero. The variable `step` is equal to the number of hills divided by 50 in the first step, twice as many in the second step, etc. The free energy surface is calculated for this number of hills. Next, exp(-G/kT) and exp(-G/kDeltaT) is calculated and stored in vectors `s1` and `s2`, respectively. Finally, `s1` and `s2` are divided elementwise and stored to `ebetac`.
Next, we open the colvar file and calculate the probabilities of the phi torsion. This can be done by:

```
mycolvar2 <- read.table("COLVAR.cv2")
cv1 <- floor(32*(mycolvar2[,2]+pi)/pi) + 1
bias <- 1000*mycolvar2[,4]
probs <- rep(0, 64)
for(i in 1:nrow(mycolvar2)) {
  step <- (i-1)*nframes/nrow(mycolvar2)+1
  probs[cv1[i]] <- probs[cv1[i]] + exp(bias[i]/temp/8.314)/ebetac[step]
}
fes <- -8.314*temp*log(probs)/1000
fes <- fes - min(fes)
fes[fes>100] <- 100
myfes <- fes(myhills2, npoints=64, imax=1)
myfes$fes <- fes
plot(myfes, ylim=c(0,60))
```

The function `read.table` is a standard function of R for reading tabular data. The first CV is extracted from the table and converted to bin number (with 64 bins). Next, the probability vector `prob` is generated. The program then runs across all lines of the colvar file and adds `exp(bias[i]/temp/8.314)/ebetac[step]` to the probability in the bin of phi angle in i-th line. The resulting probabilities are converted to free energy and the free energy of unsampled bins is converted to 100 kJ/mol (instead of infinity, for better handling). Finally, a free energy surface with 64 bins is calculated from the first hill and the free energy surface object is replaced by our `fes`. This free energy surface is plotted. Compare free energy surfaces calculated with phi CV and psi CV.

11.5 PLUMED Masterclass 22.3: OPES method

Authors

Michele Invernizzi

Date

February 28, 2022

11.5.1 Aims

This Masterclass is an introduction to the OPES (On-the-fly Probability Enhanced Sampling) method and its PLU↩ MED implementation.

11.5.2 Objectives

Once this Masterclass is completed, you will be able to:

- Perform OPES simulations and analyse the results.
- Use OPES_EXPANDED to sample different generalized ensembles
- Test the effect of various OPES_METAD input options
- Understand the typical use-case for OPES_METAD_EXPLORE

11.5.3 Prerequisites

We assume that you are familiar with PLUMED, metadynamics, umbrella sampling, and replica exchange. If you are not, the 2021 PLUMED Masterclass is a great place to start. In particular we suggest PLUMED Masterclass 21.1: PLUMED syntax and analysis, PLUMED Masterclass 21.3: Umbrella sampling, and PLUMED Masterclass 21.4: Metadynamics. Reading the OPES papers is also recommended [78] [128] [79].

11.5.4 Overview of the theory

The OPES method is an evolution of Metadynamics that also incorporates some of the ideas of the Variationally Enhanced Sampling (VES code) method. OPES is designed to be simple to use and robust with respect to suboptimal collective variables. Compared to metadynamics, it is faster in converging to a quasi-static bias and it handles multi-dimensional collective variables more efficiently.
The theory for the OPES method is presented in three separate papers [78] [128] [79]. A short overview can be found in the PLUMED documentation at OPES (On-the-fly Probability Enhanced Sampling), OPES_EXPANDED, OPES_METAD, and OPES_METAD_EXPLORE.
The OPES method aims at sampling a given target distribution over the configuration space, $p^{tg}(\mathbf{x})$, different from the equilibrium Boltzmann distribution, $P(\mathbf{x}) \propto e^{-\beta U(\mathbf{x})}$. To do so, it iteratively builds a bias potential $V(\mathbf{x})$, by estimating on-the-fly the needed probability distributions:

$$V(\mathbf{x}) = -\frac{1}{\beta}\log\frac{p^{tg}(\mathbf{x})}{P(\mathbf{x})}\,.$$

The bias quickly becomes quasi-static and the desired properties, such as the free energy, can be calculated with a simple reweighting. For any given observable $O = O(\mathbf{x})$ one can write the expectation value as:

$$\langle O\rangle_P = \frac{\langle Oe^{\beta V}\rangle_{p^{tg}}}{\langle e^{\beta V}\rangle_{p^{tg}}}\,.$$

There are two ways to define an OPES target distribution, a metadynamics-like and a replica-exchange-like. A replica-exchange-like target distribution is an expanded ensemble defined as a sum of overlapping probability distributions:

$$p^{tg}_{\{\lambda\}}(\mathbf{x}) = \frac{1}{N_{\{\lambda\}}}\sum_{\lambda} P_\lambda(\mathbf{x})\,.$$

A typical example is the temperature expanded ensemble, where each $P_\lambda(\mathbf{x})$ is the same system, but at a different temperature. This kind of target distribution can be defined via a set of expansion collective variables (ECVs). The action OPES_EXPANDED can be used to sample this kind of target distributions.

A metadynamics-like target distribution, is defined by its marginal over a set of collective variables (CVs), $\mathbf{s} = \mathbf{s}(\mathbf{x})$. A typical choice for this marginal distribution is the well-tempered one:

$$p^{WT}(\mathbf{s}) = [P(\mathbf{s})]^{1/\gamma} ,$$

where $P(\mathbf{s})$ is the marginal over the CVs of the Boltzmann distribution, and $\gamma > 1$ is the bias factor. The well-tempered distribution is a smoother version of the original one, and in the limit of $\gamma = \infty$ it become a uniform distribution. The actions OPES_METAD and OPES_METAD_EXPLORE can be used to sample this kind of target distributions.

11.5.5 Setting up the software

For this Masterclass we will use GROMACS 2020.6 patched with PLUMED 2.8, with the OPES module and MPI enabled. The easiest way to install all the software needed is to use the `plumed-masterclass-2022` conda environment, as described here.

11.5.6 Resources

The data needed to complete the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine with the usual command:

```
git clone https://github.com/invemichele/masterclass-22-03.git
```

The repository contains all the data to run the simulations, together with the scripts to analyse them. It also contains some annotated Jupyter notebooks, that will guide us through this tutorial. Most of the information can be found there, and this page is mainly meant as an overview of the content of the notebooks.
As done in some previous masterclass, we will play with a toy system, the small alanine dipeptide molecule in vacuum (ala2). It has the advantage of being a well-known system, cheap to simulate and with a higher level of complexity compared to a 2D model potential as the one used in this other OPES_METAD Tutorial: Running and post-processing. Ala2 has two main metastable basins which can be identified by a very efficient collective variable (CV), the ϕ torsional angle. Here is its free energy surface (FES) as a function of the ϕ and ψ angles:

Note

All the exercises have been tested with PLUMED version 2.8.0 and GROMACS 2020.6

11.5.7 Exercise 1: Sampling expanded ensembles with OPES

We start by using OPES to sample expanded ensembles [79]. This probably is not a familiar application of adaptive-bias enhanced sampling for most of PLUMED users, so we will start with a little

11.5.7.1 1.1 Multithermal simulations

The first example we consider is a multithermal simulation of alanine dipeptide. One can use replica exchange to perform a parallel tempering simulation and sample the two metastable states of alanine dipeptide, as explained for instance in Belfast tutorial: Replica exchange I. Here instead, we want to sample the same multithermal distribution not by combining simulations at different temperatures, but by adding a bias potential to a single simulation.
We choose 3 temperatures T_0, T_1, T_2, ranging from 300 K to 1000 K, and setup our GROMACS simulation to run at $T_0 = 300$ K. Our target distribution is then:

$$p^{tg}(\mathbf{x}) = \frac{1}{3}\left[\frac{e^{-\frac{1}{k_B T_0}U(\mathbf{x})}}{Z_0} + \frac{e^{-\frac{1}{k_B T_1}U(\mathbf{x})}}{Z_1} + \frac{e^{-\frac{1}{k_B T_2}U(\mathbf{x})}}{Z_2}\right]$$

This can be rewritten highlighting $P(\mathbf{x})$

$$p^{tg}(\mathbf{x}) = P(\mathbf{x})\frac{1}{3}\left[1 + e^{-\left(\frac{1}{k_B T_1} - \frac{1}{k_B T_0}\right)U(\mathbf{x})}\frac{Z_0}{Z_1} + e^{-\left(\frac{1}{k_B T_2} - \frac{1}{k_B T_0}\right)U(\mathbf{x})}\frac{Z_0}{Z_2}\right]$$

and then using $\Delta F_i = -k_B T_0 \log \frac{Z_i}{Z_0}$:

$$p^{tg}(\mathbf{x}) = P(\mathbf{x})\frac{1}{3}\left[1 + e^{-\frac{1-T_0/T_1}{k_B T_0}U(\mathbf{x}) + \frac{1}{k_B T_0}\Delta F_1} + e^{-\frac{1-T_0/T_2}{k_B T_0}U(\mathbf{x}) + \frac{1}{k_B T_0}\Delta F_2}\right]$$

so we just need to know ΔF_1 and ΔF_2, and we can sample the target multithermal distribution by using the following bias:

$$V(U) = -k_B T_0 \log \frac{1}{3}\left[1 + e^{-\frac{1-T_0/T_1}{k_B T_0}U(\mathbf{x})+\frac{1}{k_B T_0}\Delta F_1} + e^{-\frac{1-T_0/T_2}{k_B T_0}U(\mathbf{x})+\frac{1}{k_B T_0}\Delta F_2}\right]$$

that is a function only of the potential energy $U = U(\mathbf{x})$. In the OPES spirit, we estimate on-the-fly these free energy differences using a simple reweighting, and iteratively reach the target distribution.
The final bias expression is determined by these quantities $\frac{1-T_0/T_i}{k_B T_0}U(\mathbf{x})$, that we call expansion_cv (ECVs). There are several types of expanded ensembles that might be of interest, and in order to sample them with OPES one simply needs to find the explicit form of the corresponding ECVs. To sample the same expanded ensemble with replica exchange, one has to run a different replica of the system for each of the ECVs.
Enough equations, here is the plumed input file we need to run this multithermal simulation:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-03.txt
ene: ENERGY
ecv: ECV_MULTITHERMAL ARG=ene TEMP_MIN=300 TEMP_MAX=1000 TEMP_STEPS=3
opes: OPES_EXPANDED ARG=ecv.ene PACE=500
```

We use the OPES_EXPANDED bias action, with a multithermal expansion defined by the ECV_MULTITHERMAL action. The resulting bias is a function of the potential energy, and is updated every 500 MD steps.
In the notebook we show the shape of the resulting bias, how to check for convergence and how to reweight for estimating the free energy at different temperatures. We also explain a simple procedure that is used to automatically set a reasonable number of temperature steps, if TEMP_STEPS is not provided.

11.5.7.2 1.2 Multiumbrella simulations

Next, we present another expanded ensemble that can also be sampled by replica exchange. We call it multiumbrella and it is composed by all the distributions sampled by a typical multiple windows umbrella sampling simulation. In this kind of umbrella sampling, one chooses a CV and runs several simulations each one with a parabolic bias potential at a different CV location (see PLUMED Masterclass 21.3: Umbrella sampling). We can use OPES to sample an expanded target distribution that is composed of all of these combined.
In multiple windows umbrella sampling, the free energy difference between the windows is estimated during post-processing (e.g. with WHAM), in OPES this is done on-the-fly at simulation time. The main advantages are that choosing the number and location of the umbrellas becomes easier, and having a single longer simulation can also help with sampling orthogonal slow degrees of freedom.
We choose as CV the ϕ angle. The ECVs we need for this type of expansion are the following:

$$\Delta u_i = \frac{(\phi - \phi_i)^2}{2\sigma^2},$$

where ϕ_i are the fixed centers of the umbrellas and σ determines their width. Usually in umbrella sampling another parameter is used, $k = 1/\sqrt{\sigma}$, but we will see in the notebook why we made a different choice.
The plumed input we need might look like this:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-03.txt
phi: TORSION ATOMS=5,7,9,15
ecv: ECV_UMBRELLAS_LINE ARG=phi CV_MIN=-pi CV_MAX=pi SIGMA=0.2
opes: OPES_EXPANDED ARG=ecv.phi PACE=500
```

This places umbrellas uniformly along ϕ, each one separated by a distance of one σ. We can make them less close to each with the keyword SPACING. We can also provide a guess of the maximum free energy BARRIER we need to overcome, it can help speed up convergence.
In the notebook we show what kind of sampling this input provides, and invite you to play around with the parameters, to get a sense of what each option does. We also perform reweighting to obtain a FES estimate. This reweighting is done with kernel density estimation (similarly to plumed histogram function), and is interesting to see the effect of changing the bandwidth parameter.

11.5.7.3 1.3 Combined multithermal and multiumbrella simulations

Another advantage of using OPES to sample expanded ensembles is that it makes it very easy to combine them. We show this by running a combined multithermal and multiumbrella simulation of alanine dipeptide.
Using the multithermal expansion, we were able to sample both metastable states, but not very efficiently. With the multiumbrella target distribution we have many transitions, but we sample only at 300 K. By combining these two

expansions we obtain the best of both, with frequent transitions and an efficient reweighting over a whole range of temperatures.
The drawback? We might have to deal with a lot of ECVs! However, while in a replica exchange scheme each ECV requires its own replica, with OPES we are free to use just one, or choose any number of parallel replica, in a multiple walkers scheme.
The way we combine expansions is to create a grid with all possible combinations of temperature ECVs and umbrella ECVs. So, if we have 10 ECVs for the temperature expansion and 20 for the multiumbrella one, we end up with 200 combined ECVs. This will become clearer once we look at the notebook and the simulation output.
From the point of view of the plumed input things are quite straightforward. For a combined multithermal-multiumbrella simulation it should look similar to this one:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-03.txt
# temperature expansion, based on energy
ene: ENERGY
ecv_mt: ECV_MULTITHERMAL ARG=ene TEMP_MIN=300 TEMP_MAX=1000
# multiumbrella expansion, based on phi
phi: TORSION ATOMS=5,7,9,15
ecv_mu: ECV_UMBRELLAS_LINE ARG=phi CV_MIN=-pi CV_MAX=pi SIGMA=0.2
# the OPES bias combines them
opes: OPES_EXPANDED ARG=ecv_mt.ene,ecv_mu.phi PACE=500
```

To fully appreciate the strength of combining these two different expansions, one can run the same simulations but using the ψ angle instead of ϕ, and see what happens.

11.5.8 Exercise 2: OPES for metadynamics-like simulations

We now use OPES to perform biased simulations very similar to those of metadynamics [78] [128]. We will sample a well-tempered distribution along a chosen set of collective variables.

11.5.8.1 2.1 Biasing a good CV

We start by biasing the ϕ angle, which is known to be a good CV for alanine dipeptide. Here is how the FES along this CV looks like:
We use this simple example to get familiar with OPES_METAD, its input parameter and output quantities. We also compare it with well-tempered metadynamics, and we start to see how this two methods differ.
This also gives us the opportunity of using some of the postprocessing scripts for OPES. For OPES, there is not yet a tool similar to the sum_hills, but we can obtain the same result with simple python scripts and by dumping the state of the simulation through the STATE_WFILE keyword. This STATE file contains the compressed kernels used by OPES at a given time to represent the probability distribution, and thus the bias.

11.5.8.2 2.2 Biasing two good CVs

Next, we move to biasing both the Ramachandran angles of ala2. A minimal plumed input to do this is the following:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-03.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
opes: OPES_METAD ARG=phi,psi PACE=500 BARRIER=40
```

This example gives us a nice opportunity to visualize the kernel compression algorithm in action. We can also get an idea of the differences between OPES_METAD and OPES_METAD_EXPLORE methods. The plumed inputs are identical:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-03.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
opes: OPES_METAD_EXPLORE ARG=phi,psi PACE=500 BARRIER=40
```

From here it should be easy to compare both OPES variants to metadynamics, maybe by running some longer simulations. We can also check the way the bias converges by focusing on the estimate of the free energy difference between the basins.

11.5.8.3 2.3 Biasing more CVs: alanine tetrapeptide

In order to experiment with more than two CVs, we have to introduce a new system, alanine tetrapeptide in vacuum (ala4). It is a bigger brother of ala2, with not one but three ϕ and ψ torsional angles, that can be used to sample its 8 metastable basins.
We can try to sample ala4 with each of the tree methods, METAD, OPES_METAD, and OPES_METAD_EXPLORE. How will the respective input files look like?
We run only short simulations to see how quickly the CV space is sampled by the various methods, and not for converging a FES. This example can help to appreciate the effect of the Z term in OPES_METAD. In fact, the more CVs there are, the more it plays an important role in speeding up exploration.
We can also see the effect of changing the bias factor γ on the sampling efficiency.

11.5.8.4 2.4 Biasing a suboptimal CV

Finally, what is probably the most interesting scenario. We bias again ala2, but this time using a suboptimal collective variable. We don't know much about this CV, it could be for instance the output of a machine learning procedure. Will we be able to calculate the FES as a function of this CV? How should the plumed input files look like? Here it would be nice to also have some error bars to our estimates.
We will discuss together the results of these simulations, and also reveal how this CV was constructed.
If you prefer a more straightforward example of suboptimal CV, you can check out this OPES_METAD Tutorial: Running and post-proce

11.5.9 Conclusion

Hopefully this masterclass helped you to start playing around with the OPES method and get a sense of what it can be useful for. If you are still interested in testing it out and understanding more about it, then you should check out the PLUMED-NEST. Just search for "opes" and you will find several papers that use the method, together with all the input files necessary to reproduce their simulations.

11.6 PLUMED Masterclass 22.5: Machine learning collective variables with Pytorch

Authors

Luigi Bonati

Date

March 28, 2022

11.6.1 Aims

This Masterclass describes how to design data-driven collective variables using machine learning (ML) methods.

11.6.2 Objectives

Once this Masterclass is completed, you will know how to:

- Design data-driven CVs in Python using Pytorch
- Deploy them in PLUMED via the PYTORCH_MODEL interface

11.6.3 Prerequisites

We assume that you are familiar with PLUMED and enhanced sampling calculations. If you are not, the 2021 PL↩ UMED Masterclass is a great place to start. We will also use OPES (On-the-fly Probability Enhanced Sampling) to enhance sampling, which has been covered on the PLUMED Masterclass 22.3: OPES method.
Furthermore, Python knowledge is required for this masterclass.

11.6.4 Overview

The identification of suitable collective variables (CVs) is crucial for the success of several enhanced sampling methods. Typically, the quest for CVs has been driven by physical intuition, however, the complexity of the systems that can be simulated nowadays calls for new approaches. For this reason, here we want to learn the CVs directly from molecular dynamics data. As in any machine learning approach, the three key ingredients will be (1) the data, (2) the model, and (3) the objective function. We note that these are tightly connected to each other. Indeed, depending on the kind of data that is available we can ask different questions. Furthermore, the ability to answer such questions (as well as to interpret the results) will depend on the choice of appropriate models.
Broadly speaking, the CVs should represent the collective atomic movement as the system translocates from one metastable state to another. As a general requirement, CVs should be continuous and differentiable functions of the atomic positions able to (1) distinguish the different states and (2) be connected to the slow modes of the system. However, this results in a chicken and egg problem: if we have reactive trajectories that already explored the relevant phase space we might analyze such simulations and extract good CVs. But more often than not, for obtaining such reactive simulations we already need good CVs. Here we describe two possible ways out of this chicken-and-egg situation, which reflects two scenarios that in the practice occur very often:
(1) Only the metastable states are known. These could be for instance the reactant and product of a chemical reaction, crystalline and liquid configurations of a material, the bound and unbound states of a ligand into a protein, etc... Given this limited data, we can guess the CVs by optimizing variables with a classification criterion [66]. This consists of enforcing only the first CV requirement stated above. Of course, one should not expect this to work as well as if one could add dynamic information on the transition paths, but it should be seen as a way to make the best out of limited knowledge. Very often the only information available is what the states of interest are, and so it is necessary to develop methods that can construct approximate CVs from this information.
(2) We have an enhanced sampling simulation in which we have biased some CV, but with suboptimal results. Instead of proceeding by trial and error by changing the CVs until we find the right one(s), we can analyze such simulations and identify the slow degrees of freedom. If we can express them in a functional form and subsequently bias them we will be able to significantly improve sampling quality [110].

11.6.5 Software and data

The data needed to complete the exercises of this Masterclass can be found on `GitHub`. To complete the exercises, we will go through a set of **Jupyter notebooks**, from which all the simulations can be performed and analyzed.
Participants of the masterclass will receive a link to a project on `DeepNote`, a collaborative data science platform. There you will find all the necessary software already installed and you will be able to go through the notebook and perform MD simulations.
If instead you want to run the exercises on your computer you can follow the instructions available on `GitHub`. In particular, you will need to use the development version of PLUMED. Furthermore, you will need to download LibTorch and configure PLUMED against it, as well as enable the PYTORCH and OPES modules. To configure PLUMED with Pytorch you can follow the instructions in the PYTORCH (Machine Learning Collective Variables) page. To run the simulations we will use GROMACS 2020.6 patched with PLUMED.

11.6.5.1 Training CVs with Pytorch

In this tutorial, we will use `mlcvs`, a Python-based package designed for building data-driven collective variables for enhanced sampling simulations. In `mlcvs` we have implemented different kinds of data-driven CVs proposed in the literature, including Deep-LDA [66] and Deep-TICA [110] which we describe in this masterclass.

11.6.5.2 Using the CVs in PLUMED

Once the models have been trained on the data, we can export them using the jit (just in time) compiler of Pytorch. This creates a Python-independent model, which can be loaded in PLUMED using Pytorch C++ APIs (LibTorch). Of particular relevance is that we can exploit the automatic differentiation feature of Pytorch to compute derivatives of the model with respect to the inputs. Note that in this way we can load the models exported from `mlcvs` as well as any other function built using Pytorch methods and serialized with jit.
A typical PLUMED input might be the following, in which we first calculate some descriptors and feed them as inputs of the model.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-05.txt
#SETTINGS AUXFILE=regtest/pytorch/rt-pytorch_model_2d/torch_model.ptc
```

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
model: PYTORCH_MODEL FILE=torch_model.ptc ARG=phi,psi
PRINT FILE=COLVAR ARG=model.node-0,model.node-1
```

11.6.6 Exercises

The exercises are presented in the Jupyter notebooks, from which all the simulations can be performed and analyzed. For this reason, this page only presents a high level overview of the contents of the notebooks.

11.6.6.1 Toy system: alanine dipeptide

As done in some previous masterclass, we will play with a toy system, the small alanine dipeptide molecule in a vacuum (ala2). This has the advantage of being a well-known system and it is cheap to simulate. The typical use case is to test enhanced sampling methods using as CVs the torsional angle ϕ and ψ. Since the former is an almost ideal CV for describing the transition between its two metastable states, this will typically result in a very efficient sampling. Here we want to approach this problem in a more blind way and try to design efficient C↩ Vs with as little knowledge as possible, learning the CVs directly from the output of molecular dynamics simulations.

In both exercises, we will characterize the molecule with a general set of physical descriptors, which are the interatomic distances between heavy atoms. The reason for not working directly with atomic positions is twofold: on one hand, invariant CVs can be easily obtained under the relevant symmetries, while on the other, we can use different kind of descriptors to focus the sampling on the relevant processes.

11.6.6.2 Tutorial 1: `Collective variables from equilibrium fluctuations

You can find the first tutorial in the jupyter notebook `tutorial1_DeepLDA.ipynb`. In this exercise, we will design the CVs starting only from a realization of the metastable states. The main steps will be the following:

1. Perform short unbiased MD simulations in the metastable states and evaluate a set of physical descriptors (e.g. interatomic distances between heavy atoms)
2. We train a neural-network based CV to discriminate between the states, using Fisher's discriminant as the objective function (DeepLDA)
3. Finally, we apply a bias potential to enhance the fluctuations of the DeepLDA CV and drive the system back and forth between the two states.

11.6.6.3 Tutorial 2: Collective variables as slow modes of biased simulations

In the second exercise, which you can find in the notebook `tutorial2_DeepTICA.ipynb` , we will extract the CVs from biased simulations. As before, we will go through three main steps:

1. Perform an enhanced sampling calculation using some approximate CVs as well as generalized ensembles simulations (e.g. multicanonical).
2. Use the trajectory to train a neural-network based CV to find the maximally autocorrelated modes, which correspond to the slowest degrees of freedom (DeepTICA)
3. Bias these slow degrees of freedom to improve sampling

11.7 PLUMED Masterclass 22.6: EDS module + Coarse-Grained directed simulations

Authors

Glen Hocky, Andrew White

Date

April 26, 2022

11.7.1 Aims

This Masterclass describes how to bias simulations to agree with experimental data using experiment directed simulation.

11.7.2 Objectives

Once this Masterclass is completed, you will know how to:

- How to bias collective variables to agree with set values
- Develop a deeper feeling how parameters in method change learning efficiency

11.7.3 Prerequisites

We assume that you are familiar with PLUMED and enhanced sampling calculations. If you are not, the 2021 PL↩ UMED Masterclass is a great place to start. In particular, you should be familiar with specifying collective variables. Furthermore, some Python or other programming knowledge is required for this masterclass to generate plots and perform some analysis calculations.

11.7.4 Overview

Experiment directed simulation (EDS) is a maximum entropy method for biasing specific collective variables (CVs) to agree with set values. These are typically from experimental data, like a known radius of gyration or NMR chemical shift. Biasing a CV is an under-determined problem because there are many ways to change the systems' potential energy to agree with a set point. If we further maximize ensemble entropy while matching the set point, the problem has a unique solution of

$$U'(x) = U(x) + \lambda f(x)$$

where $U(x)$ is the potential energy of the system, $f(x)$ is the CV, and λ is a fit parameter. EDS is a time-dependent method that finds λ while the simulation is running. It typically converges much faster than free-energy methods, but comes with the same caveats that insufficient sampling or rare events can affect the method. Another important detail is that EDS/maximum entropy biasing is for matching the set point on average (in expectation), rather than at every frame.

11.7.5 Software and data

The only data needed to complete the exercises of this Masterclass can be found on an the following github page data, with alanine inputs being borrowed from an earlier one GitHub-22-03. Simulations will be performed using PLUMED's pesmd module, and gromacs.

11.7.6 Exercises

The exercises are presented below.

11.7.6.1 Setup 1-dimensional system.

Set up a plumed file to use with pesmd that has a harmonic potential. Then run using the 1d input provided in the github. For example:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-06.txt
d1: DISTANCE ATOMS=1,2 COMPONENTS
ff: MATHEVAL ARG=d1.x PERIODIC=NO FUNC=0.5*10*(x^2)
bb: BIASVALUE ARG=ff

PRINT ARG=d1.x STRIDE=25 FILE=_PREFIX_.colvars.dat

plumed pesmd < harmonic_1d.in
```

Histogram your data, and see if it fits the ideal Boltzmann distribution for this harmonic oscillator. Make sure to normalize your distribution properly!
Now try adding a harmonic bias using the RESTRAINT function, so that the data becomes centered near $x = 1$.
Finally, we will add an EDS bias. Make your EDS bias also be centered at 1.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-masterclass-22-06.txt
d1: DISTANCE ATOMS=1,2 COMPONENTS
ff: MATHEVAL ARG=d1.x PERIODIC=NO FUNC=0.5*10*(x^2)
bb: BIASVALUE ARG=ff

eds: EDS ARG=d1.x CENTER=__FILL__ PERIOD=__FILL__ OUT_RESTART=__FILL__ TEMP=1.0 BIAS_SCALE=1
PRINT ARG=* STRIDE=100 FILE=_PREFIX_.colvars.dat
```

Note, we have to add TEMP=1 because pesmd does not provide the temperature to the EDS module.
Now look at how the bias factor changes with time. Can you also compute the bias from this value, and the value of the position versus time? Look in the colvar file and the restart file to see what values are there.

11.7.6.2 Setup and bias a 2-dimensional system.

Now we will move on to a 2d harmonic oscillator. This way we can see the effect of biasing x, y, or a combination. Note, there is a different pesmd input file for this.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-masterclass-22-06.txt
d1: DISTANCE ATOMS=1,2 COMPONENTS
ff: MATHEVAL ARG=__FILL__ PERIODIC=NO FUNC=__FILL__
bb: BIASVALUE ARG=ff

PRINT ARG=__FILL__ STRIDE=25 FILE=_PREFIX_.colvars.dat
```

Try to bias either the x or y directions separately. Try biasing a combination of them. We biased the average of x$*$y to be = 1. Use the CUSTOM command to try that out. What do you think it should look like? Write a python program to histogram the x and y data and see. It should look like below.

11.7.6.3 Alanine dipeptide

Bias alanine dipeptide using gromacs and the input files provided in the github.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-masterclass-22-06.txt
MOLINFO STRUCTURE=./input.ala2.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2

eds: EDS ARG=__FILL__ CENTER=__FILL__ PERIOD=__FILL__ OUT_RESTART=_PREFIX_.restart.dat BIAS_SCALE=1

PRINT ARG=* STRIDE=500 FILE=_PREFIX_.colvars.dat
```

Try biasing phi and psi independently. Try biasing them at the same time. What is a good PERIOD to choose?

11.7.6.4 Free explore

Take a protein of interest, or alanine 2, and bias more than one variable. To try CGDS, choose the center of mass of some different parts of your protein. Can you bias the distance between these COMs and the angle between them at the same time? Does using COVAR or LM affect the speed of convergence?

11.8 PLUMED Masterclass 22.8: Modelling Concentration-driven processes with PLUMED

Author

Matteo Salvalaglio

Date

May 20, 2022

11.8.1 Aims

This Masterclass aims to introduce users to the tools available via PLUMED to analyze and perform simulations of concentration-driven processes such as nucleation, growth, and diffusion.

11.8.2 Objectives

Once this Masterclass is completed, users will be able to:

- Write a PLUMED input file to analyze a trajectory with multicolvars.
- Use the Depth-First-Search tools available in PLUMED to characterize clusters of atoms in an existing vapour condensation trajectory.
- Write a PLUMED input file to analyze the condensation of a simple liquid phase from vapour.
- Write a PLUMED input file for CmuMD, and perform a CmuMD simulation of liquid condensation at constant driving force.
- Write a PLUMED input file for CmuMD, and perform a CmuMD simulation to model steady-state diffusive flux across a simulation box.

11.8.3 Prerequisites

We assume that you are familiar with PLUMED. However, the 2021 PLUMED Masterclass is a great place to start if you are not.

11.8.4 Setting up the software

Follow the instructions provided here to install gromacs+plumed2.8 compiled with CmuMD.

11.8.5 Background Overview

PLUMED facilitates the calculation of functions of atom coordinates and the introduction of bias potentials in atomistic simulations.
In this tutorial, we review how to use PLUMED functionalities for two complementary tasks:

- Calculating collective variables that capture processes of assembly/disassembly such as the nucleation, condensation, dissolution etc. (Ex. 1, 2)
- Introducing biasing forces through CmuMD, a method that mimics open-boundary conditions and allows for mitigation of finite-size effects. (Ex. 3, 4)

11.8.6 Resources

The data needed to complete the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/mme-ucl/PLUMED_MClass_22-08
```

11.8.7 Solution

The solution of this masterclass is available on GitHub as a jupyter notebook.

11.8.8 Exercise 1: analysis of a nucleation trajectory I

We can start by using plumed to analyze a trajectory and identify the total number of atoms belonging to a liquid phase condensing from a supersaturated vapour in a long unbiased MD simulation.
To this aim, a typical choice is the implementation of a criterion inspired by the work of Ten Wolde and Frenkel: all the atoms with a threshold number of nearest neighbours larger than a threshold is considered as part of the liquid phase.
In PLUMED this criterion can be readily implemented using the multicolvar COORDINATIONNUMBER:

```
BEGIN_PLUMED_FILE
# LIQUID-like atoms
lq: COORDINATIONNUMBER SPECIES=1-10000 SWITCH={CUBIC D_0=0.45 D_MAX=0.55} MORE_THAN={RATIONAL R_0=5.0 D_MAX=10
PRINT ARG=lq.morethan STRIDE=1 FILE=nliquid.dat
```

The time evolution of the number of atoms in the liquid phase can be computed by analyzing the trajectory using the driver functionality (where liquid.dat is the plumed file for this analysis).

```
plumed driver --mf_xtc LJvap_to_liq_red.xtc --plumed liquid.dat
```

11.8.8.1 Try on your own

Using other MultiColvar keywords can you count the number of atoms on the surface of the clusters forming in this trajectory? Can you estimate how the surface to volume ratio of the droplets forming in the system changes directly within PLUMED?

11.8.8.2 Available data

The MD trajectory that we will analyze can be found in the folder called `data`:

- `LJvap.gro`: reference conformation of 10000 LJ particles in a metastable vapour phase.
- `LJvap_to_liq.xtc`: trajectory.

11.8.9 Exercise 2: analysis of a nucleation trajectory II

In certain contexts, it can be helpful to characterize the phase transition by following analyzing the population of clusters that form during the nucleation process. In PLUMED, this is possible by taking advantage of the DFSCLUSTERING algorithm, which builds on the construction of a CONTACTMATRIX to identify clusters of atoms with specific properties.
Here we analyse the clusters distribution emerging in the trajectory `LJvap_to_liq.xtc`, focussing on the number of clusters, and following the evolution of the four largest clusters in the system:

```
BEGIN_PLUMED_FILE
# Identify liquid-like atoms
lq: COORDINATIONNUMBER SPECIES=1-10000 SWITCH={CUBIC D_0=0.45  D_MAX=0.55} MORE_THAN={RATIONAL R_0=5.0 D_MAX=1

# Define a contact matrix & perfom DFS clustering
cm: CONTACT_MATRIX ATOMS=lq  SWITCH={CUBIC D_0=0.45  D_MAX=0.55}
dfs: DFSCLUSTERING MATRIX=cm

# Compute the size of the four largest clusters
cluster_1: CLUSTER_NATOMS CLUSTERS=dfs CLUSTER=1
cluster_2: CLUSTER_NATOMS CLUSTERS=dfs CLUSTER=2
cluster_3: CLUSTER_NATOMS CLUSTERS=dfs CLUSTER=3
cluster_4: CLUSTER_NATOMS CLUSTERS=dfs CLUSTER=4

# Compute the number of clusters
nclust: CLUSTER_DISTRIBUTION CLUSTERS=dfs MORE_THAN={GAUSSIAN D_0=D_0=1.95 R_0=0.01 D_MAX=1.99}

# PRINT to file
PRINT ARG=lq.morethan,nclust.*,cluster_1,cluster_2,cluster_3,cluster_4 STRIDE=1  FILE=clusters.dat

FLUSH STRIDE=1
```

11.8.9.1 Try on your own

Modifying the setup above, can you compute the number of clusters with a size larger than 20 LJ particles?

11.8.9.2 Available data

The MD trajectory that we will analyze can be found in the folder called `data`:

- `LJvap.gro`: reference conformation of 10000 LJ particles in a metastable vapour phase.
- `LJvap_to_liq.xtc`: trajectory.

11.8.10 Exercise 3: Steady-state diffusive flux with CmuMD

In the two examples discussed in Exercises 1 and 2 it is apparent that the process of condensation reaches steady state due to finite-size effects. However, looking at the process more carefully, one can notice that the driving force leading to phase separation is far from constant. CμMD allows using PLUMED to apply ad-hoc forces and keep constant the composition of spatial regions of the simulation box called control regions. This feature allows to model processes driven by concentration in steady, out-of-equilibrium conditions.
The first example we will focus on is a purely diffusive process in a LJ vapour.
A PLUMED file that allows imposing a steady concentration difference with CmuMD looks like:

```
BEGIN_PLUMED_FILE
# Define groups of atoms
LJ: GROUP ATOMS=1-1000:1

# Provide parameters for the CV
left:  CMUMD GROUP=lj NSV=1 FIXED=0.5 DCR=0.25 CRSIZE=0.1 WF=0.0001  ASYMM=-1 NINT=0.1 NZ=291
right:  CMUMD GROUP=lj NSV=1 FIXED=0.5 DCR=0.25 CRSIZE=0.1 WF=0.0001  ASYMM=1 NINT=0.1 NZ=291

# CmuMD is implemented as a restraint on the densities of species in CR
rleft:  RESTRAINT ARG=left AT=1.4 KAPPA=1000.0
rright: RESTRAINT ARG=right AT=0.05 KAPPA=2000.0

# Report the densities and bias
PRINT ...
ARG=left,right,rleft.bias,rright.bias
STRIDE=10
FILE=CMUMD_log
... PRINT
```

We can run a CMUMD simulation under these conditions as:

```
gmx_mpi mdrun --plumed cmumd_diff.dat
```

11.8.10.1 Try on your own

Perform simulations at varying concentration differences. Compute the concentration gradient across the simulation box as a function of the concentration difference between left and right controlled volumes.

11.8.10.2 Available data

In the folder `data`:

- `LJ_diffusion.gro`: reference conformation of 1000 LJ particles.
- `LJ_diffusion.xtc`: trajectory evolving under the effect of the stationary concentration gradient.

11.8.11 Exercise 4: Steady-state condensation process with CμMD

The fourth exercise combines aspects of the previous three. We will use CmuMD to control the driving force associated with the growth of a dense phase represented by a slab at the center of a simulation box.
Similarly to the diffusion case the plumed file that can be used to perform this type of simulation reads:

```
BEGIN_PLUMED_FILE
# Define groups of atoms
LJ: GROUP ATOMS=1001-2000:1

# Provide parameters for the CV
left:  CMUMD GROUP=lj NSV=1 FIXED=0.4 DCR=0.25 CRSIZE=0.1 WF=0.0001  ASYMM=-1 NINT=0.1 NZ=291
right:  CMUMD GROUP=lj NSV=1 FIXED=0.6 DCR=0.25 CRSIZE=0.1 WF=0.0001  ASYMM=1 NINT=0.1 NZ=291

# CmuMD is implemented as a restraint on the densities of species in CR
left:  RESTRAINT ARG=left AT=3. KAPPA=2000.0
right: RESTRAINT ARG=right AT=3. KAPPA=2000.0

# Report the densities and bias
PRINT ...
ARG=left,right,rleft.bias,rright.bias
STRIDE=10
FILE=CMUMD_log
... PRINT
```

11.8.11.1 Available data

In the folder `data`, a CmuMD trajectory ready for analysis can be found together with the gromacs input files necessary to setup and run it.

- `LJ_slab.gro`: reference conformation of 1000 LJ particles.
- `LJ_slab.xtc`: trajectory evolving under the effect of the stationary concentration gradient.
- `md_input_LJ_slab`: MD input files

11.8.11.2 Try on your own

- Setup and run CMUMD simulations for the LJ slab system at varying CR concentrations.
- Use the multicolvar-based approach discussed in Ex1 to follow the condensation process.
- Use the graph-based approach discussed in Ex2 to follow the condensation process.
- Monitor the density profile across the simulation box.

11.9 PLUMED Masterclass 22.9: Using path collective variables to find reaction mechanisms in complex free energy landscapes

Authors

Bernd Ensing

Date

June 6, 2022

Much of this exercise was originally created by Alberto Pérez de Alba Ortíz for the Cecam School "Understanding Molecular Simulation", held annually in Amsterdam.

11.9.1 Aims

The aim of this Masterclass is to demonstrate the advantages of path collective variables for describing and simulating activated molecular processes, and to provide hands-on experience with setting up, running, and analyzing biased path-CV simulations, such as path-metadynamics.

11.9.2 Objectives

Once this Masterclass is completed, users will be able to:

- Use the Plumed PesMD engine to run test simulations of enhanced sampling methods.
- Write Plumed input files to run metadynamics and path-metadynamics simulations.
- Tune the adaptive path-CV parameters for optimal efficiency.
- Analyze the evolution and convergence of the transition path and the free energy profile.
- Use the multiple walker parallelization of metadynamics and path-metadynamics.
- Restart a PMD simulation from a previous guess path.

11.9.3 Prerequisites

We assume that you are familiar with PLUMED. However, the 2021 PLUMED Masterclass is a great place to start if you are not.

11.9.4 Setting up the software

The data needed to complete the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/Ensing-Laboratory/masterclass-22-09
```

We will use Plumed version 2.3.0 for this exercise together with personal versions of the PathCV and PESMD features. With the above `git clone` command, you have already downloaded them, and with the following commands we compile this version of plumed:

```
cd masterclass-22-09/plumed
chmod +x install
./install
```

This takes about 5-10 minutes on a modern laptop or desktop computer. After that, the plumed executable is installed in the folder `masterclass-22-09/plumed/bin`.
Information on these versions of the PathCV option and the PESMD option may be slightly different from that in the latest online plumed manual. Instead, open your local version in a browser, which you can find in the following locations on your computer:

- `masterclass-22-09/plumed/plumed-2.3.0/user-doc/html/index.html`
- `masterclass-22-09/plumed/plumed-2.3.0/user-doc/html/pesmd.html`
- `masterclass-22-09/plumed/plumed-2.3.0/user-doc/html/_p_a_t_h_c_v.html`

The exercises are found in the folder `masterclass-22-09/exercises`.

11.9.4.1 Introduction

In this exercise, we will perform metadynamics [1] and path-metadynamics (PMD) [2-4] simulations of a system that can undergo a transition between two free energy minima. These two stable states are separated by a transition state barrier, so that the transition process is a *rare event* for ``normal'' (i.e. unbiased) molecular dynamics simulations.
PMD is able to simultaneously and efficiently converge the transition path (which can be either an average transition path or a minimum free energy path (MFEP)) and the free energy profile of the process along this path. This is achieved by performing metadynamics on an adaptive path connecting the two known stable states.
In order to understand the principles and advantages of PMD, we will first perform standard metadynamics and get familiar with its parameters and functionalities. Subsequently, we will add the features corresponding to the adaptive path. To ease computation and visualization, we will employ a simple model system.
This exercise makes use of PLUMED's [5] own internal engine: *plumed pesmd*, i.e. without being coupled to an MD engine that runs an actual molecular system. The advantages are that the simulations will be very fast, and you do not need to know anything about the MD engine of our choice, which might be different from the one that you are used to.
Still, the enhanced sampling techniques that you will use in this exercise, and the plumed input files that we will write for that, can also be used (with very little changes) with plumed patched to any MD software, which means that you can potentially use it for your own research!
In order to overcome the timescale limitations of MD and be able to observe relevant molecular transitions, the community has developed a plethora of enhanced sampling methods. Among them, we can identify the so-called biasing methods, which work by exerting an external potential (or force) on a few key descriptive degrees of freedom of the transition, commonly called *collective variables* (CVs).
The choice and design of CVs is a task of great importance. In the following, we will assume that the CVs are known. We will consider a molecular system whose free energy surface (FES) is fully described by a set of N CVs, $\{z_i(\mathbf{q})\}$ with $i = 1...N$, which are all functions of the system's particle positions $\mathbf{q}(t)$. We will also assume that these particle positions $\mathbf{q}(t)$ evolve in time with a canonical equilibrium distribution at temperature T under a potential $V(\mathbf{q})$.

11.9.4.2 Metadynamics

We are interested in exploring the space spanned by the CVs, $\{z_i\}$, and in quantifying the free energy. In order to do so, in metadynamics we exert a history-dependent repulsive potential by summing Gaussian kernels over time:

$$V_{bias}(\mathbf{z},t) = \sum_{\kappa\tau<t} H(\kappa\tau)\exp\Bigg(-\sum_{i=1}^{N}\frac{(z_i - z_i(q(\kappa\tau)))^2}{2W_i^2}\Bigg),$$

with Gaussian height H, widths along each CV W_i, and depositing frequency τ. This bias drives the system away from already visited configurations and into new regions of CV-space. More importantly, in the long-time limit, the bias potential converges to the minus free energy as function of the CVs: $V_{bias}(\mathbf{z}, t\to\infty) = -F(\mathbf{z})$.
As a well-established sampling technique, metadynamics also has several extensions to accelerate convergence and improve computational performance, such as the multiple-walker and well-tempered versions [6,7].
Metadynamics can handle a few CVs simultaneously in a trivial manner, which spares us from having to find a single perfect CV. Instead, an appropriate set of CVs allows us to converge an insightful multidimensional free energy landscape, in which (meta)stable states and connecting MFEPs can be identified. In practice however, especially when investigating complex transitions, the number of CVs is limited to $N \approx 3$, because of the exponential growth of computational cost with CV dimensionality.

11.9.4.3 Path-metadynamics

In order to tackle complex transitions that require many CVs, path-based methods were developed. In these schemes, a path-CV is introduced. That is, a parametrized curve that connects two FES basins — the known stable states A and B — in the space spanned by the CVs: $\mathbf{s}(\sigma) : \mathbb{R} \to \mathbb{R}^N$, where the parameter $\sigma(\mathbf{z})|_s : \mathbb{R}^N \to [0,1]$ yields the progress along the path from A to B, such that $\mathbf{s}(0) \in A$ and $\mathbf{s}(1) \in B$.
This progress parameter emulates the reaction coordinate and can in principle be connected to the committor value. Since the transition path curve is not known a priori, the path-CV must be adaptive. If we assume that, in the vicinity of the path, the iso-commitor planes S_σ are perpendicular to $\mathbf{s}(\sigma)$, and that the configurational probability is a good indicator of the transition flux density, then we can converge the average transition path by iteratively adapting a guess path to the cumulative average density: $\mathbf{s}_g(\sigma_g) = \langle \mathbf{z} \rangle_{\sigma_g}$ [3].
The key advantage of PMD is that the metadynamics biasing can be performed on the 1D progress parameter along the adaptive path-CV, instead of the N-dimensional CV-space. PMD is able to converge a path, and the free energy along it, with a sublinear rise in computational cost with respect to the growth in CV dimensionality (instead of the exponential relation when not using a path-CV)[4].
Numerically, the adaptive path-CV is implemented as a set of M ordered nodes (coordinates in CV-space) $\mathbf{s}_g(\sigma_g, t) \to \mathbf{s}_j^{t_i}$ with $j = 1, ..., M$, where t_i is the discrete time of path updates. The projection of any point $\mathbf{z}$ onto the path is done considering the closest two nodes, and σ is obtained by interpolation. This approach requires equidistant nodes, which are imposed by a reparametrization step after each path update. The update step for the path nodes is done by:

$$\mathbf{s}_j^{t_{i+1}} = \mathbf{s}_j^{t_i} + \frac{\sum_k w_k \cdot (\mathbf{z}_k - \mathbf{s}^{t_i}(\sigma(\mathbf{z}_k)))}{\sum_k \xi^{t_i - k} w_{j,k}}, with w_k = \max\Bigg[0, \Bigg(1 - \frac{||\mathbf{s}_j^{t_i} - \mathbf{s}^{t_i}(\sigma(\mathbf{z}_k))||}{||\mathbf{s}_j^{t_i} - \mathbf{s}_{j+1}^{t_i}||}\Bigg)\Bigg],$$

where k is the current MD step and w is the weight of the adjustment, which is non-zero only for the two nodes closest to the average CV density. We also introduce the fade factor $\xi = \exp(-\ln(2)/\tau_{1/2})$, with $\tau_{1/2}$ as a half-life: the number of MD steps after which the distances measured from the path weight only 50% of its original value. A short half-life increases path flexibility by allowing it to rapidly `‘forget’’ a bad initial guess, while a long half-life restricts the path fluctuations and leads eventually to optimal convergence. The first and last nodes, located at the stable states A and B respectively, remain fixed. Extra trailing nodes can be placed at both ends of the path to better capture the free energy valleys. Wall potentials can be used to keep the sampling near the relevant [0,1] σ-interval.
Additionally, a harmonic restraint set on $||\mathbf{z}_k - \mathbf{s}^{t_i}(\sigma(\mathbf{z}_k))||$ can help in converging to a specific transition path in scenarios with multiple or ill-defined transition channels. We refer to this restraint as a `‘tube’’ potential. In the limit of an infinitesimally narrow tube potential, the path update step follows the local free energy gradient and PMD converges to the MFEP instead of the average transition path. This control over the behavior of the algorithm is an advantageous feature, as switching from a density-driven to a gradient-driven path optimization leads either to the average transition path or the MFEP.
Metadynamics, the method of choice to sample the path-CV, drives the system back and forth along the path, providing sufficient sampling of the CV density to localize the transition path. At the same time, metadynamics continuously self-corrects the free energy by overwriting the previously deposited Gaussians with new ones. This leads to a converged free energy along the localized path: $V_{bias}(\sigma, t\to\infty) = -F(\sigma)$. Most of the algorithmic

extensions developed for metadynamics (well-tempered, transition-tempered, multiple walkers, etc.) can be straightforwardly combined with the adaptive path-CV. Moreover, other biasing techniques (umbrella sampling, constrained MD, steered MD, etc.) can also be used to sample along the path, or even in the direction perpendicular to it.

11.9.4.4 Model system

In this exercise, we will sample the well-known Müller-Brown (MB) potential energy surface (PES). You can find more about it in Ref. [8]. We add harmonic walls to limit the sampling within the interesting region of the PES.

11.9.4.5 Using PLUMED

To install PLUMED, if you have not already done this above, go to the folder `plumed` in the folder for this exercise. In the script `install`, change the prefix to the installation folder of your choice, and then run the script. If you have issues with the installation you can check: https://plumed.github.io/doc-v2.3/user-doc/html/_installation.html
To visualize the results of this exercise, plotting script examples are provided for the `gnuplot` plotting software (see the script files names starting with `plot_` or `anim_`). Make sure to change the corresponding variables according to your choices of the metadynamics and path-CV parameters. Of course, feel free to use other plotting tools, for example pythons matplotlib.
To run the PLUMED pesmd on the MB PES, use the command `plumed pesmd < path/to/input`. The `input` file is located in the `Exercise` folder. Review the PLUMED pesmd documentation: https://plumed.github.io/doc-v2.4/user-doc/html/pesmd.html. To perform metadynamics, you can review the keywords on: https://plumed.github.io/doc-v2.3/user-doc/html/_m_e_t_a_d.html. To use the adaptive path-CV, you can review the keywords in the header of the file `PathCV.cpp` in the folder `plumed` in the folder for this exercise. To see the html documentation on the path-CV in a browser, open <PMD_exercise/plumed/plumed-2.3.0/user-doc/html/_p_a_t_h_c_v.html> in a browser. To construct a FES from the metadynamics hills, use the `plumed sum\_hills` functionality. Consult its features on: https://plumed.github.io/doc-v2.3/user-doc/html/sum_hills.html. We recommend to use `--stride` 1 and `--mintozero`.

11.9.4.6 Exercise 1

The first exercise is found in the folder: `masterclass-22-09/exercises/0_md`.
This first exercise is meant to get acquainted with running the `plumed pesmd` engine, understanding the input files, and plotting the MD trajectory in the MB potential from the colvar file using gnuplot (or other plotting software). To use the `./run.me` script to start the simulation, first make the script executable using:

```
chmod +x ./run.me
```

Use PLUMED pesmd to run regular MD on the MB PES.
What do you observe?
How would you change the behaviour of the system without using an enhanced sampling method (just modifying the MD run parameters)?

11.9.4.7 Exercise 2

This exercise is found in the folder: `masterclass-22-09/exercises/1_metadynamics`.
Before you can run the metadynamics simulation, the `plumed.dat` should be completed:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-09.txt
UNITS LENGTH=nm TIME=fs

cv: DISTANCE ATOMS=1,2 COMPONENTS

lwall: LOWER_WALLS ARG=cv.x,cv.y AT=-1.5,-0.5 KAPPA=1000.0,1000.0
uwall: UPPER_WALLS ARG=cv.x,cv.y AT=1.5,2.5 KAPPA=1000.0,1000.0

metad: ### ADD YOUR METADYNAMICS INSTRUCTIONS HERE ###

PRINT ARG=cv.x,cv.y,metad.bias STRIDE=100 FILE=colvar.out

Use PLUMED pesmd to run metadynamics on the MB PES.
```

Here are some hints for defining the METAD rule:

- the CVs are here the DISTANCE components cv.x and cv.y, which are actually the coordinates of a "2D particle" (with respect to the origin) undergoing a Langevin dynamics on the MB PES. Use these components as the ARG arguments in the METAD rule.
- reasonable widths of the Gaussians (SIGMA) are ca. 0.1.
- 0.1 is also a reasonable Gaussian height
- a good interval for depositing Gaussians is probably around 500
- don't forget to define an output FILE, that can afterward be read using plumed sum_hills to generate and plot the free energy landscape.

Try to obtain a good FES reflecting the two minima.
Vary the width of the Gaussians for each CV in a systematic manner, what effect do you observe?
Vary the height and deposition frequency of the Gaussians, what trends do you observe?
How many crossings and recrossings do you sample in a single run for each choice of parameters?

11.9.4.8 Exercise 3

This exercise is found in the folder: masterclass-22-09/exercises/2_fixedpathmetadynamics.
Run metadynamics along a fixed path-CV on the MB PES. We recommend using 20 transition nodes, with 20 trailing nodes at each end (60 nodes in total). Use harmonic walls to keep the sampling within a relevant [-0.2,1.2] σ-range.
Now there's only one dimension for the Gaussian width, how do you interpret it?
Does the diffusion of the system along σ reflect anything particular? What could be the reason?
What happens if you add and strengthen a tube potential?
What is the final path and free energy that you obtain?

11.9.4.9 Exercise 4

This exercise is found in the folder: masterclass-22-09/exercises/3_adaptivepathmetadynamics.
First, change the plumed.dat input file to make the path adaptive by changing PACE, STRIDE, and HALFLIFE.
Run metadynamics along an adaptive path-CV on the MB PES. For simplification, start with the same pace for the metadynamics Gaussians and the path update.
For simplification, you can keep only 20 transition nodes, and not use trailing nodes.
Use harmonic walls to keep the sampling within a relevant [-0.2,1.2] σ-range.
What do you expect if you vary the ratio between the metadynamics and the path update frequency?
Vary the tube potential strength and the half-life parameter, what trend do you observe?
A small value for the HALFLIFE (say 1000 MD steps), keeps the path flexible so that the path evolves efficiently away from a poor initial guess path. However, once it has reached the transition channel, it would be better to set it to a very large value, in order to converge to the average transition path. Run first a short PMD simulation using a flexible path and relatively large Gaussians, until the system has filled both basins and has recrossed back to the initial state. Then, restart the simulation, using smaller Gaussians and a large halflife. Use the INFILE directive (instead of GENPATH) to read the last updated path, which can be taken from the tail of the path.out file of the first run.

11.9.4.10 Exercise 5

This exercise is found in the folder: masterclass-22-09/exercises/4_multiplewalkers.
In this final exercise, you will run the parallel "multiple walker" version of path-metadynamics.
Note that with more walkers building the metadynamics simulation, the total run time can be shorter, and possibly also the Gaussians can be smaller.
Try several configurations of walkers and try to answer the question if there is an optimal number of walkers.

11.9.4.11 Bibliography

[1] A. Laio and M. Parrinello, “Escaping free-energy minima,” *Proc. Natl. Acad. Sci. USA*, vol. 99, no. 20, pp. 12562–12566, 2002.

[2] D. Branduardi, F. Gervasio, and M. Parrinello, “From A to B in free energy space,” *J. Chem. Phys*, vol. 126, p. 054103, 2007.
[3] G. Díaz Leines and B. Ensing, “Path finding on high-dimensional free energy landscapes,” *Phys. Rev. Lett.*, vol. 109, no. 2, p. 020601, 2012.
[4] A. Pérez de Alba Ortíz, A. Tiwari, R. C. Puthenkalathil, and B.Ensing, “Advances in enhanced sampling along adaptive paths of collective variables,” _ J. Chem. Phys._, vol. 149, no. 7, p. 072320, 2018.
[5] G. A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, and G. Bussi, “Plumed2:Newfeathers for an old bird,” *Comp. Phys. Comm.*, vol. 185, no. 2, pp. 604–613, 2014.
[6] P. Raiteri, A. Laio, F. L. Gervasio, C. Micheletti, and M. Parrinello, “Efficient reconstruction of complex free energy landscapes by multiple walkers metadynamics,” *J. Phys. Chem. B.*, vol. 110, no. 8, pp. 3533–3539, 2006.
[7] A. Barducci, G. Bussi, and M. Parrinello, “Well-tempered metadynamics: a smoothly converging and tunable free-energy method,” *Phys. Rev. Lett.*, vol. 100, no. 2, p. 020603, 2008.
[8] K. Müller and L. D. Brown, “Location of saddle points and minimum energy paths by a constrained simplex optimization procedure,” *Theor. Chim. Acta*, vol. 53, no. 1, pp. 75–93, 1979.

11.10 PLUMED Masterclass 22.10: Hamiltonian replica exchange with PLUMED and GROMACS

Authors

Giovanni Bussi

Date

June 21, 2022

11.10.1 Aims

In this Masterclass, we will discuss how to run Hamiltonian replica exchange using PLUMED and GROMACS. We will also understand how to analyze the resulting trajectories. It is highly recommended to follow PLUMED Masterclass 21.5: Simulations with multiple replicas in advance.

11.10.2 Objectives

Once you have completed this Masterclass you will be able to:

- Use PLUMED to manipulate GROMACS topologies and prepare a solute tempering simulation.
- Use PLUMED and GROMACS to run replica-exchange simulations with multiple topologies.
- Use WHAM to combine multiple simulations performed with different topologies and/or bias potentials.

11.10.3 Setting up PLUMED

For this masterclass you will need versions of PLUMED and GROMACS that are compiled using the MPI library. In order to obtain the correct versions, please follow the instructions at this link.

11.10.4 Resources

The data needed to execute the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-22-10.git
```

Note

All the exercises were tested with PLUMED version 2.8.0 and GROMACS 2020.6

Additional documentation about the replica exchange implementation discussed in this Masterclass can be found at this page: Using Hamiltonian replica exchange with GROMACS .

11.10.5 Exercises

Throughout this Masterclass we will run simulations of **alanine dipeptide in water** using GROMACS and PLUMED. Whereas this system is too simple to be considered a proper benchmark for enhanced sampling methods, it is complex enough to be used in learning them. This Masterclass regards a specific implementation of Hamiltonian replica exchange that is only available when combining GROMACS and PLUMED.
Notice that simulations might take up to a couple of hours, depending on the hardware you have access to. The version of GROMACS that we provide on our conda channel is not optimized. Compiling GROMACS directly on your machine might lead to much better performance. In addition, using a GPU will also make your simulations significantly faster.

11.10.5.1 Introduction to Hamiltonian replica exchange

In PLUMED Masterclass 21.5: Simulations with multiple replicas we have learned how to run replica-exchange simulations where each replica was possibly feeling different biasing forces. A typical example is umbrella sampling with replica exchange, where each replica has a restraint located in a different position. Another example is bias-exchange metadynamics, where each replica is biased along a different collective variable. We have also seen parallel-tempering metadynamics, where replicas are kept at a different temperature and, at the same time, keep track of different history-dependent potentials. All these examples are a form of Hamiltonian replica exchange, since different replicas are subject to different potential energy functions, and thus different Hamiltonians.
In this masterclass we will consider a conceptually similar but technically different implementation of Hamiltonian replica exchange. Here, the different replicas will be simulated using different force field parameters. For most of the exercise, we will thus use an empty `plumed.dat` file (no bias added!). You will however need to include this file to ensure PLUMED can be enabled. We will also spend some time in learning how to edit the GROMACS topologies, so as to generate the modified force fields.
In order to use multiple-replica methods, you should run your simulation using MPI. This can be done prefixing your command with `mpiexec -np N --oversubscribe`, where `N` is the number of processes that you want to use and the `--oversubscribe` option is an OpenMPI option that is required to use more processes than the number of available processors. This is typically suboptimal, but we will need it in our lectures to run, e.g., simulations with 16 replicas even if we have a computer with 4 cores.
In brief, to run a GROMACS simulation where the individual replicas are in directories names `dir0`, `dir1`, etc and the (possibly empty) `plumed.dat` file is in the parent directory you will need a command such as

```
mpiexec -np 16 --oversubscribe gmx_mpi mdrun -multidir dir? dir?? -plumed ../plumed.dat -replex 200 -hrex
```

The option `-replex 200` enables replica exchange and ensures exchanges are attempted every 200 steps. The option `-hrex` is specific for the implemenation discussed in this Masterclass, and informs GROMACS and PLUMED that there might be different force fields used in different replicas.

Warning

> If you forget the `-hrex` flag, no error will be issued, but the acceptance for exchanges will be incorrectly calculated.

If you have random crashes on MacOS, try to set this environemnt variable:

```
export OMPI_MCA_btl="self,tcp"
```

In the root directory of this Masterclass you will find a `conf.gro` file, that can be used as a starting configuration for your simulations, a `topol.top` file, which contains the topology information, and a `grompp.mdp` file with reasonable simulation parameters. You have also a `conf.pdb` file, that is basically the `conf.gro` file converted to PDB format, and can be used for the MOLINFO keyword so as to facilitate atom selections in analysis.

11.10.5.2 Exercise 1a: Test with different temperatures

Before swithing to solute tempering, we will play a bit with parallel tempering. Parallel tempering simulations require the highest replica to be hot enough for important energy barriers to be crossed in the simulation. We will first estimate how much we should raise the temperature so as to see forward and backward transitions between the two metastable states of our system.
To do so:

- Prepare an array of simulations, with temperatures equal to 300, 400, 500, 600, 700, 800, 900, 1000
- Run each of them for 1 ns and check at which temperature you have at least one transition from the initial basin to the other one and one backward transition

These are serial simulations, so you can run them with this command:

```
# first edit grompp.mdp setting the temperature (both solute and solvent groups!)
# then create the topol.tpr file:
gmx_mpi grompp
# then run your simulation:
gmx_mpi mdrun -nsteps 500000
```

though it is recommended to use a script to run your simulations.
To analyze the resulting trajectories you can use PLUMED driver

```
plumed driver --ixtc traj_comp.xtc --plumed plumed.dat
```

with the following `plumed.dat` file:

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=conf.pdb
phi: TORSION ATOMS=@phi-2
PRINT ARG=phi FILE=COLVAR
```

You can then plot the second column of the resulting `COLVAR` file and see if there is a transition from the first basin (phi in range (-3,-1)) to the second basin (phi in range (0.5,1.5)), and back. Answer the following question:

- How much should you increase the temperature to see a transition back-and-forth between the two basins?

The result will depend on stochastic factors, but also on the length of the simulation. I recommend running a 1 ns long simulation (500000 steps), but you can try with a longer or shorter trajectory.

11.10.5.3 Exercise 1b: Run a parallel tempering simulation

Once you have identified this temperature, you can run a parallel tempering simulation. You will need a number of replicas to bridge from T=300 to the temperature you have identified. As a first guess, you can place them in a geometric series (that's the allocation that leads to uniform acceptance in a system with temperature-independent specific heat).
Let's say that you want to try with 16 replicas. You should create 16 directories named `dir0`, `dir1`, ... `dir15` and place a file named `topol.tpr` in each. Each `topol.tpr` files will be created setting a different temperature in the `grompp.mdp` file. We will not need PLUMED for this exercise, so the simulation can be run with a command like this one:

```
mpiexec -np 32 --oversubscribe gmx_mpi mdrun -multidir dir? dir?? -replex 200 -nsteps 500000
```

The `-replex` option tells to GROMACS how frequently exchanges should be attempted. The average acceptance will be reported at the end of the `md.log` file.
The hottest replica will display transitions between the two metastable states. Thanks to the exchanges, both states will be observed also in the replica that is kept as T=300K.
Now answer the following questions:

- How many replicas do you need to have an acceptance that is at least 30%? (the answer will depend on the maximum temperature you have chosen)
- How much is the relative population of the two metastable states at T=300? Plot this population as a function of the temperature (you can just extract results from the different replicas).

11.10.5.4 Exercise 2a: Setting up scaled Hamiltonians

In order to run Hamiltonian replica exchange simulations with multiple topologies, we will have first to learn how to generate the multiple topologies. Different tools could be used (including editing the `topol.top` file by hand!) but we will now learn how to use the partial_tempering tool available in PLUMED. This tool basically allows you to generate topologies where the energy of a subset of the atom have be scaled by a chosen factor. Remember that dividing the energy by a factor 2 is equivalent to multiplying the temperature by a factor 2.
Let's first have a look at the `topol.top` file. In this file there are a number of lines that look like `#include ...`, so that this file is not self-contained. The first thing that we have to do is to generate a self-contained topology file:

```
gmx_mpi grompp -p topol.top -pp processed.top
```

Have a look at the resulting `processed.top` file with a text editor. This file does not contain only information about alanine dipeptide, but also about the generic force-field parameters. It is thus self-contained.
Then you should edit the `processed.top` file to indicate which atoms you want to scale. To do so you have to add an underscore (_) to the atom name of the selected atoms. For instance, this line:

```
1         HC     1    ACE   HH31      1     0.1123      1.008   ; qtot 0.1123
```

should be modified to this line:

```
1         HC_    1    ACE   HH31      1     0.1123      1.008   ; qtot 0.1123
```

To perform a solute tempering simulation, you should add the underscore to all the solute atoms (look in the `[ atoms ]` section). Once this is done you can use the following command:

```
plumed partial_tempering 1.0 < processed.top  > scaled.top
```

This will scale the Hamiltonian of the selected atoms by a factor 1.0 (which means: no change!). Have a look at the resulting `scaled.top` file and find out what has changed. You can also try to apply two different scaling factors and check the difference:

```
plumed partial_tempering 1.0 < processed.top  > scaled0.top
plumed partial_tempering 0.5 < processed.top  > scaled1.top
diff scaled0.top scaled1.top
```

11.10.5.5 Exercise 2b: Sanity check on generated topologies

Notice that these scaled topologies can be used to run GROMACS simulations. The `parial_tempering` script is far from perfect. We will now make some sanity check. To do these checks, we will either generate a new trajectory or simply take one of the trajectories that we generated in the previous exercise. We will then used a GROMACS tool named `rerun`, which allows recomputing the energy along a trajectory using a new topology.

First, create an energy file corresponding to the original topology:

```
# it is better to do this in a separate directory and rename the
# trajectory file, or gromacs will complain:
gmx_mpi mdrun -rerun traj.xtc -s topol.tpr
```

The resulting `ener.edr` file can be converted to a text file with the `gmx_mpi energy` command.

Then, generate a new topology with scaling factor 1.0 and compute energies again:

```
plumed partial_tempering 1.0 < topol_selected.top > topol_scaled.top
gmx_mpi grompp -p topol_scaled.top -o topol_scaled.tpr
gmx_mpi mdrun -rerun traj.xtc -s topol_scaled.tpr -e ener_scaled.edr
```

The resulting energies should be identical!

As a second check, generate a new topology where you scaled *all* atoms (make sure to also select water!) by a factor 0.5:

```
plumed partial_tempering 0.5 < topol_all_selected.top > topol_all_scaled.top
gmx_mpi grompp -p topol_all_scaled.top -o topol_all_scaled.tpr
gmx_mpi mdrun -rerun traj.xtc -s topol_all_scaled.tpr -e ener_all_scaled.edr
```

This time, dihedral angles, LJ, and Coulomb energies should be multiplied by 0.5.

This is a very important check. The `partial_tempering` script is not compatible with some specific functional forms, e.g., CHARMM CMAP (search PLUMED mailing list for a fix). In case you are using an incompatible force field, you will find inconsistencies in these sanity checks. For the provided `topol.top` file, everything should work.

These tests are very important, please only proceed if you managed to pass them.

11.10.5.6 Exercise 2c: Sanity check on replica-exchange implementation

Another thing that we will have to check now is if acceptance is computed correctly. The code inserted in GROMACS to implement this calculation is non trivial and quite fragile. To do this check, you should run a short Hamiltonian replica exchange with two equivalent topology files. For instance, you can use the original `topol.tpr` file and the one that you obtained with scaling factor 1.0. Make sure that the two tpr files are using a different seed for randomizing the initial velocities or, even better, use two different `conf.gro` files to initialize the two simulations. For everything else, use the same settings you will use in production (ideally, same number of processes per replica, same GPU settings, etc).

Now run a short replica exchange simulation:

```
mpiexec -np 2 --oversubscribe gmx_mpi mdrun -multidir dir0 dir1 -replex 200 -nsteps 10000 -hrex -plumed
      plumed.dat
```

As written above, `plumed.dat` can be just an empty file. Check the resulting acceptance. Since the Hamiltonians are identical, the acceptance **should be exactly 1.0**.

Notice that this might be expected to fail if you use a GPU.**

11.10.5.7 Exercise 2d: Find minimum scaling factor needed to cross the barrier

You are now able to generate tpr files where the energy of a subset of the atoms is scaled. We will do something similar to Exercise 1a: Test with different temperatures, but playing with scaling factor instead of temperature. Make sure that you select all (and only) the atoms belonging to the solute (alanine dipeptide), so as to implement solute tempering. Try to run a set of 1 ns long simulations, with scaling factors decreasing (e.g., 1.0, 0.9, 0.8, etc). Now answer the following question:

- How much should you decrease the scaling factor to see transitions between the two metastable states in the first nanosecond?

11.10.5.8 Exercise 3: Run Hamiltonian replica exchange simulations

Now run a Hamiltonian replica exchange simulation with multiple replicas, bridging from lambda=1.0 to the minimun value identified in the previous point. I would suggest using a linear distribution in lambda rather than a geometric one, but you can experiment. Similarly to parallel tempering, we can for now just analyze the reference replica (at lambda=1.0).
Now answer the following questions:

- How many replicas (and which is the optimal distribution) you need to have an acceptance that is at least 30%? (the answer will depend on the minimum lambda you have chosen). Is this smaller or larger than the number of replicas that you used for Exercise 1b: Run a parallel tempering simulation?
- How much is the relative population of the two metastable states at T=300? Plot this population as a function of lambda (you can just extract results from the different replicas). Is there any relationship between the dependence on lambda and the dependence on T that we have seen in Exercise 1b: Run a parallel tempering simulation?

11.10.5.9 Exercise 4: Analyze Hamiltonian replica exchange simulations with WHAM

So far we only analized the reference replica (lambda=1.0). We can however do better and combine all replicas with WHAM. To this aim you should:

- Concatenate all trajectories in a single file (`gmx_mpi trjcat -cat -f dir?/traj_comp.xtc dir??/traj_comp.xtc -o traj_multi.xtc`).
- Recompute the potential energy according to each of the tpr files (for each replica) (`gmx_mpi mdrun -rerun traj_multi.xtc -s dir0/topol.tpr -e energy0.edr`, `gmx_mpi mdrun -rerun traj_multi.xtc -s dir1/topol.tpr -e ener1.edr`, etc.).
- Convert the energies to text files (`echo 11 | gmx_mpi energy -f ener0.edr -xvg no -e energy0.xvg -xvg no`, etc.). 11 should select the potential energy.

Then you can use the provided python wham script to compute the weight of all frames with the following python commands:

```
import wham
print(wham.__file__) # make sure you are using the wham script provided with this masterclass
energies=[]
with cd("3/6reps"):
    for i in range(len(lambdas)):
        energies.append(np.loadtxt("energy{}.xvg".format(i),usecols=1))
energies=np.array(energies).T
# energies[i,j] is the energy of frame i according to the j-th Hamiltonian
kBT=0.00831446261815324*300
w=wham.wham(energies,T=kBT)
# w["logW"] are the Boltzmann factors
# notice that we did not specify yet to which ensemble we are reweighting
# this can be done with the following line:
logW=w["logW"]-energies[:,0]/kBT
# logW are the logaritm weights to obtain properties corresponding to replica 0
logW-=np.max(logW) # avoid numerical errors in exp
weights=np.exp(logW)
weights/=np.sum(weights) # normalize weights
# these weights can be used to compute weighted averages
```

Now plot the `weights` array and answer the following questions:

- How does the weight depend on the frame index? Remember that the replica exchange trajectories were concatenated. Can you recognize those frames coming from the first, second, etc replica?
- Compute the population of the two states by summing the weight of all states in the two basins.
- Modify the script above to compute weights corresponding to replicas other than the first one, and compute the population of the two states as a function of lambda.

11.10.5.10 Exercise 5: Optimize the lambda ladder

The aim of this exercise is to optimize the list of values of lambda. As a target, we will try to make the product of the acceptances as large as possible. We will do this *without* running new simulations, i.e. just analyzing the simulation above.
This is a difficult exercise. It will be solved in the solution, but it is **not** necessary to complete it so as to proceed to the next step. I write some hints here.

- In solute tempering, the energy is a quadratic function of `sqrt(lambda)`. In other words, for each frame, you could write the energy as `A*lambda+B*sqrt(lambda)+C`, where `A`, `B`, and `C` are to be recomputed at each frame. Analyze the `ene` array that we generated in the last exercise so as to obtain these coefficients. The goal is to have a function that, for any frame, and for any value of lambda (including values not yet simulated!), returns the energy of the system.
- Using these functions, you will be able to compute weights corresponding to arbitrary values of lambda. You could for instance use these weights to generate a smooth version of the population vs lambda plot that we did in the last exercise.
- Given two values of lambda, you can compute the average acceptance by doing ensemble averages. You can easily test this function: use it to predict the acceptance associated to the neighboring replicas simulated above. Can you predict the acceptances seen during the simulation?

Once you have this function (let's call it `predict_acceptance(lambda1,lambda2)`) you can use the following script to find the ladder that maximizes the product of the acceptances. Let's say that the largest lambda is fixed to 1.0 and the lowest is fixed to `lambda_min`

```
# lambda is the array with the replicas we used for out simulation (a starting point(
# first and last replicas are fixed (1.0 and lambda_min)
# we thus optimize all elements excluding the first and the last (lambda[1:-1])
# this function compute the acceptances given the three intermediate values of
# lambda
def predict_all_acceptances(x):
    x=np.array(x)
    acc=[]
    # lambdas should not be negative
    if np.any(x<=0.0):
        return np.zeros(len(x)+1)
    # nor larger than 1
    if np.any(x>=1.0):
        return np.zeros(len(x)+1)
    # first replica has lambda=1.0
    acc.append(predict_acceptance(1.0,x[0]))
    for i in range(len(x)-1):
        acc.append(predict_acceptance(x[i],x[i+1]))
    # last replica has lambda=lambda_min
    acc.append(predict_acceptance(x[-1],lambda_min))
    return acc
# this is function to be minimized
def func(x):
    acc=predict_all_acceptances(x)
    return -np.prod(acc) # negative, since we minimize the result
from scipy.optimize import minimize
res=minimize(func,lambdas[1:-1]) # starting values
print("optimal lambdas: ",1.0,res.x,lambda_min)
print(predict_all_acceptances(res.x))
```

You can then run a new simulation with the optimal lambdas and check if the predicted acceptances correspond to the observed ones!

11.10.5.11 Exercise 6: Combine with metadynamics on psi

As a very last point, repeat exercise Exercise 5: Parallel-tempering metadynamics using alanine dipeptide in explicit solvent and solute tempering with the lambda values optimized above. No further guidance is provided for this exercise, but it should not be difficult following the suggestions given for Exercise 5: Parallel-tempering metadynamics.

11.11 PLUMED Masterclass 22.11: Variationally enhanced sampling with PLUMED

Authors

Omar Valsson

Date

July 4, 2022

11.11.1 Aims

In this Masterclass, we will discuss how to run variationally enhanced sampling simulations with PLUMED. We will also understand how to analyze the results.

11.11.2 Objectives

Once you have completed this Masterclass you will be able to:

- Run variationally enhanced sampling simulations biasing one and two CVs
- Assess the convergence of the simulation
- Perform reweighting from a variationally enhanced sampling simulations

Use PLUMED to run and analyze

- Use PLUMED to reweight from

11.11.3 Setting up PLUMED

For this masterclass you will need versions of PLUMED (with the VES module enabled) and GROMACS that are compiled using the MPI library. All the exercises were tested with PLUMED version 2.8.0 and GROMACS 2020.6. In order to obtain the correct versions, please follow the instructions at this link.
The data needed to execute the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/valsson-group/masterclass-22-11.git
```

11.11.4 Summary of theory

Here, we will briefly summarize the theory behind variationally enhanced sampling (VES). For an full overview of VES, you should read the original paper, a recent review book chapter on VES, or a recent enhanced sampling review that includes discussion about VES.
VES is based on the the following functional of the bias potential:

$$\Omega[V] = \frac{1}{\beta} \log \frac{\int d\mathbf{s}\, e^{-\beta[F(\mathbf{s})+V(\mathbf{s})]}}{\int d\mathbf{s}\, e^{-\beta F(\mathbf{s})}} + \int d\mathbf{s}\, p_{\mathrm{tg}}(\mathbf{s}) V(\mathbf{s}),$$

where $\mathbf{s}$ are the CVs that we are biasing, $p_{\mathrm{tg}}(\mathbf{s})$ is a predefined probability distribution that we will refer to as the target distribution, and $F(\mathbf{s})$ is the free energy surface. This functional can be shown to be convex and to have a minimum at:

$$V(\mathbf{s}) = -F(\mathbf{s}) - \frac{1}{\beta} \log p_{\mathrm{tg}}(\mathbf{s}).$$

The last equation states that when we minimize the functional $\Omega[V]$, we can obtain the free energy surface from the bias potential (and the target distribution) We can choose the target distribution $p_{\mathrm{tg}}(\mathbf{s})$ at will and it is the CV distribution that we obtain when minimizing $\Omega[V]$.
We put the variational principle to practice by expanding $V(\mathbf{s})$ in some basis set:

$$V(\mathbf{s}) = \sum_i \alpha_i\, f_i(\mathbf{s}),$$

where $f_i(\mathbf{s})$ are the basis functions and the α are the coefficients in the expansion. We then need to find the coefficients α that minimize $\Omega[V]$. In principle one could use any optimization algorithm. In practice the algorithm that has become the default choice for VES is the so-called averaged stochastic gradient descent algorithm [122]. In this algorithm the α are evolved iteratively according to:

$$\alpha^{(n+1)} = \alpha^{(n)} - \mu \left[\nabla\Omega(\bar{\alpha}^{(n)}) + H(\bar{\alpha}^{(n)})[\alpha^{(n)} - \bar{\alpha}^{(n)}] \right]$$

where μ is the step size, $\bar{\alpha}^{(n)}$ is the running average of $\alpha^{(n)}$ at iteration n, and $\nabla\Omega(\bar{\alpha}^{(n)})$ and $H(\bar{\alpha}^{(n)})$ are the gradient and Hessian of $\Omega[V]$ evaluated at the running average at iteration n, respectively. The behavior of the coefficients will become clear in the examples below.
As said above, we can choose the target distribution $p_{\mathrm{tg}}(\mathbf{s})$ at will. The most simple choice would be a uniform target distribution. However, it has found more optimal to employ the so-called well-tempered distribution [121] :

$$p_{\mathrm{tg}}(\mathbf{s}) = \frac{[P(\mathbf{s})]^{1/\gamma}}{\int d\mathbf{s}\, [P(\mathbf{s})]^{1/\gamma}}$$

where γ is the so-called bias factor and $P(\mathbf{s})$ is the unbiased CV distribution. Therefore the well-tempered distribution is the unbiased distribution with enhanced fluctuations and lowered barriers. This is the same distribution as sampled in well-tempered metadynamics. The advantages of this distribution are that the features of the FES (metastable states) are preserved and that the system is not forced to sample regions of high free energy (that can represent un-physical configurations) as it would if we had chosen the uniform target distribution.
There is a caveat though, the well-tempered $p_{\mathrm{tg}}(\mathbf{s})$ depends on $F(\mathbf{s})$ that is the function that we are trying to calculate. One way to approach this problem is to calculate $p_{\mathrm{tg}}(\mathbf{s})$ self-consistently [121], for instance at iteration k:

$$p^{(k+1)}(\mathbf{s}) = \frac{e^{-(\beta/\gamma)F^{(k+1)}(\mathbf{s})}}{\int d\mathbf{s}\, e^{-(\beta/\gamma)F^{(k+1)}(\mathbf{s})}}$$

where:

$$F^{(k+1)}(\mathbf{s}) = -V^{(k)}(\mathbf{s}) - \frac{1}{\beta}\log p^{(k)}(\mathbf{s})$$

Normally $p^{(0)}(\mathbf{s})$ is taken to be uniform. Therefore the target distribution evolves in time until it becomes stationary when the simulation has converged. It has been shown that in some cases the convergence is faster using the well-tempered target distribution than using the uniform $p(\mathbf{s})$ [121].

11.11.5 The system

In this tutorial, we will consider the association/dissociation of NaCl in aqueous solution. The system consists of 1 Na atom, 1 Cl atom, and 107 water molecules for a total of 323 atoms. In an effort to speed up the simulations, we employ a rather small water box, and thus need to employ smaller cutoffs than usually used. Therefore, this simulation setup should not be used in production runs. Typically, the run should take around 15-20 minutes to run on a laptop using two MPI processes. By running the simulations on a cluster, you reduce the simulation time.
The most relevant CV for this system is the distance between the Na and Cl atoms that is defined in PLUMED as

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-11.txt
dist:  DISTANCE ATOMS=322,323
```

Furthermore, the NaCl association/dissociation is coupled to the collective motion of the solvent. To measure that, we will use a CV that measures the solvation of the Na atom. For this, we employ the coordination number of the Na atom with respect to the oxygens of the water molecules that we define in PLUMED as

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-11.txt
COORDINATION ...
 GROUPA=322
 GROUPB=1-321:3
 SWITCH={RATIONAL R_0=0.315 D_MAX=0.5 NN=12 MM=24}
 NLIST
 NL_CUTOFF=0.55
 NL_STRIDE=10
 LABEL=coord
... COORDINATION
```

We will also limit CV space exploration by employing an upper wall on the distance between Na and Cl atoms that is defined in PLUMED as

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-11.txt
UPPER_WALLS ...
   ARG=dist
   AT=0.6
   KAPPA=4000.0
   LABEL=uwall
... UPPER_WALLS
```

11.11.6 Exercise 1: Biasing with one collective variable

We will start by performing a simulation where we bias the Na-Cl distance.
Every VES simulation has three key ingredients

- Basis set
- Target distribution
- Optimization algorithm

For the basis set, we will use the recently introduced wavelet-based basis set that are localized basis functions that have been shown to perform better than the previously used Chebyshev or Legendre polynomials. We will employ the least asymmetric variant of these wavelets or so-called symlets (as indicated by the TYPE=SYMLETS keyword). We will use an order 10 of the symlets or Sym10 (as indicated by the ORDER=10 keyword). Furthermore information about the wavelets can be found in the reference above.
We need to select the range on which the bias potential is expanded. Here we will use the range from 0.2 nm to 0.7 nm (as indicated by the MINIMUM and MAXIMUM keywords). We also need to select the number of basis functions, and here we will use 26 basis functions (as indicated by the NUM_BF keyword). The PLUMED action corresponding to this setup is given by

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-11.txt
# Basisset for Na-Cl distance
BF_WAVELETS ...
  LABEL=bf1
  TYPE=SYMLETS
  ORDER=10
  NUM_BF=26
  MINIMUM=0.2
  MAXIMUM=0.7
  TAILS_THRESHOLD=0.01
... BF_WAVELETS
```

For the target distribution, we employ a well-tempered distribution with a bias factor of 10.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-11.txt
# Target distribution
td: TD_WELLTEMPERED BIASFACTOR=10
```

Then we define the VES bias potential using the VES_LINEAR_EXPANSION action

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-11.txt
VES_LINEAR_EXPANSION ...
  LABEL=ves
  ARG=dist
  BASIS_FUNCTIONS=bf1
  TEMP=300.0
  GRID_BINS=300
  OPTIMIZATION_THRESHOLD=0.000001
  TARGET_DISTRIBUTION=td
... VES_LINEAR_EXPANSION
```

Finally, we need to define the optimization algorithm. The standard is the averaged stochastic gradient descent (OPT_AVERAGED_SGD). We need to define two parameters: the stride and the step size. The stride (given by the keyword STRIDE) is the number of MD steps in which samples are collected to calculate the gradient and hessian of $\Omega[V]$. The step size (given by the keyword STEPSIZE) is the step by which the coefficients are evolved at every optimization steps, given by μ in the equation above. It has become traditional to choose a stride of around 500-2000 MD steps. It must be noted that we are not looking for an accurate estimation of the gradient, since for this we would need to sample all the CV space. The step size in the optimization has a strong connection with the height of typical barriers in the system. The larger the barriers, the larger the step size needed such that the bias can grow fast enough to overcome them. For this example we have chosen a stride of 500 steps (i.e., 1 ps) and a step size of 5.0 kJ/mol. We also need to choose how often we update the target distribution (given by the keyword TARGETDIST_STRIDE) and we do this every 100 bias potential updates (i.e., every 100 ps in the current case).

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-11.txt
OPT_AVERAGED_SGD ...
  LABEL=opt
  BIAS=ves
  STRIDE=500
  STEPSIZE=5.0
  FES_OUTPUT=100
  BIAS_OUTPUT=500
  COEFFS_OUTPUT=10
  TARGETDIST_STRIDE=100
  TARGETDIST_OUTPUT=500
... OPT_AVERAGED_SGD
```

The other parameters are related to the outputting frequency of various output files.
Finally, we need to define the PRINT action to output all the variables

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-11.txt
PRINT ARG=dist,coord,ves.*,uwall.* FILE=colvar.data STRIDE=125
```

The full PLUMED input file can be found in the Exercise-1 folder. There you also find all the relevant GROMACS input file. First, you need to run the generate-tpr-file.sh script that generates the GROMACS TPR file using the parameters defined in MD-NPT.mdp and the initial geometry defined using the StartingGeometry variable. You can then run the simulation using the run.sh script

```
./generate-tpr-file.sh
./run.sh
```

The run might take around 15-20 minutes to run using two MPI processes. You can adjust the number of MIP processes used for the simulation using the NumProcs variable in the run.sh script.
At the end of simulation, you will get several files:

- colvar.data: Colvar file
- coeffs.data : Values of the coefficients α and $\bar{\alpha}$ at different iterations.
- bias.<bias-label>.iter-<iteration-number>.data : Bias potential at iteration <iteration-number>.
- fes.<bias-name>.iter-<iteration-number>.data : FES at iteration <iteration-number>.
- targetdistribution.<bias-name>.iter-<iteration-number>.data : Target distribution at iteration <iteration-number>.

To assess the simulation and its convergence, you should first look at the time evolution of the biased CV and check that it is diffusive in CV space. Second, you should look at how the free energy surfaces behave as a function of time by looking at the fes.<bias-name>.iter-<iteration-number>.data files at different number of iterations (the minimum of the FES is always align to zero to facilitate comparison). To do this, you need to use your favorite way to plot datafiles (e.g., Matplotlib or Gnuplot).
You can also visualize the trajectory by opening it with VMD by using the command

```
vmd NaCl_StartingStructure-1.gro NaCl_VES_NPT-300K.pbc-whole.xtc
```

The NaCl_VES_NPT-300K.pbc-whole.xtc is the trajectory file with the periodic boundary conditions made whole. This is done with the run.sh script.
The coeffs.data file includes the values of coefficients α and $\bar{\alpha}$ at different iterations. To extract the time evolution of a given coefficient, you can use the ExtractCoeff.sh script, for example to extract coefficient 3:

```
./ExtractCoeff.sh 3 > coeff.3.data
```

This will create a file with the first column the iteration number, the second column the averaged coefficient $\bar{\alpha}$, and the third column the instantaneous coefficients α. You should create files for different coefficient and visualize both the second and third column to understand how the coefficients converge.

11.11.7 Exercise 2: Reweighting a VES simulation

Apart from estimating the FES directly from the bias potential, you can also estimate the FES through reweighting by histogramming where each configuration is weighted by the bias acting on it, $e^{\beta V(\mathbf{s})}$. The VES bias acting at each time step is given by the ves.bias variable in the colvar.dat file.
When doing performing reweighting, it is better to ignore the initial part of the simulation where the bias potential is changing more rapidly. You can use the trim-colvar-file.py python script in the Exercise-2 folder to do this

```
./trim-colvar-file.py --colvar-file ../Exercise-1/colvar.data --output-file colvar_reweight.data --time-min 40
```

where here we ignore the first 400 ps of the colvar.data file from the Exercise-1 and create a new file called colvar_reweight.data.
We can then perform the reweighting for the distance using the following PLUMED input (plumed_reweight.dat in the Exercise-2 folder)

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-11.txt
dist:   READ FILE=colvar_reweight.data IGNORE_TIME VALUES=dist
coord:  READ FILE=colvar_reweight.data IGNORE_TIME VALUES=coord
ves:    READ FILE=colvar_reweight.data IGNORE_TIME VALUES=ves.bias

weights: REWEIGHT_BIAS TEMP=300 ARG=ves.bias

HISTOGRAM ...
  ARG=dist
  GRID_MIN=0.2
  GRID_MAX=0.7
  GRID_BIN=60
  KERNEL=DISCRETE
  LOGWEIGHTS=weights
  LABEL=hg_dist
... HISTOGRAM

fes_dist: CONVERT_TO_FES GRID=hg_dist TEMP=300 MINTOZERO
DUMPGRID GRID=fes_dist FILE=fes-reweight.dist.data FMT=%24.16e
```

You can run this input by using the PLUMED driver

```
plumed driver --plumed plumed_reweight.dat --noatoms
```

You should compare the resulting FES (the fes-reweight.dist.data file) to the results obtained directly from the bias potential in Exercise 1.
We can also obtained the FES for CVs that are not biased in the VES simulation. For example, we can obtain the two-dimensional FES for the distance and the solvation CV given by the coordination number CV. For this, you will need to add the following to the plumed_reweight.dat file and repeat the PLUEMD driver run

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-11.txt
HISTOGRAM ...
  ARG=dist,coord
  GRID_MIN=0.2,2.5
  GRID_MAX=0.7,7.5
  GRID_BIN=200,200
  BANDWIDTH=0.004,0.04
  LOGWEIGHTS=weights
  LABEL=hg_dist_coord
... HISTOGRAM
fes_dist_coord: CONVERT_TO_FES GRID=hg_dist_coord TEMP=300 MINTOZERO
DUMPGRID GRID=fes_dist_coord FILE=fes-reweight.dist-coord.data FMT=%24.16e
```

Note that here we use kernel density estimation with Gaussian kernels to obtain smoother results.
You can also try to obtain the one-dimensional FES for the solvation CV by adjusting the input above.
This will generate a two-dimensional FES that you can visualize.

11.11.8 Exercise 3: Run another independent simulation

To check the results, it is a good practice to run another independent simulation using different initial conditions. You can achieve this here by changing the initial geometry in the generate-tpr-file.sh script

```
StartingGeometry=NaCl_StartingStructure-2.gro
```

and regenerating the GROMACS tpr file. Do this and rerun the simulation, check the convergence, and perform reweighting as before. Make sure that you do this in a new clean folder that is separate from the run in Exercise 1.

11.11.9 Exercise 4: biasing with two collective variables

We will now bias also the solvation CV. To achieve this, we need first to setup a separate basis set for the solvation CV, where again we use the symlets but with a different range from 2.0 to 8.0

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-11.txt
# Basisset for coordination number
BF_WAVELETS ...
  LABEL=bf2
  TYPE=SYMLETS
  ORDER=10
  NUM_BF=22
  MINIMUM=2.5
  MAXIMUM=7.5
  TAILS_THRESHOLD=0.01
... BF_WAVELETS
```

We also need to change the relevant keywords in the VES_LINEAR_EXPANSION action, namely the ARG, BAS↩
IS_FUNCTIONS, and GRID_BINS keywords

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-11.txt
VES_LINEAR_EXPANSION ...
  LABEL=ves
  ARG=dist,coord
  BASIS_FUNCTIONS=bf1,bf2
  TEMP=300.0
  GRID_BINS=300,300
  OPTIMIZATION_THRESHOLD=0.000001
  TARGET_DISTRIBUTION=td
  PROJ_ARG1=dist
  PROJ_ARG2=coord
... VES_LINEAR_EXPANSION
```

Additionally, we have turned on the calculation of the one-dimensional projection of the FES on the two CVs (we also need to set the keyword FES_PROJ_OUTPUT=100 in OPT_AVERAGED_SGD). This is useful to assess the

convergence as this is easier in one-dimension. We can also compare the projection to the results from Exercise 1. These changes should be sufficient to do the simulations using two CVs. You can see full input file in the Exercise-4 folder.
Once you have performed this simulation, you should also try to reweight from this simulations. Furthermore, if you have time, you should also try to perform another independent simulation.

11.11.10 Optional exercises

The following three exercises are optional, but they will show you how different parameters effect the results. You should base these exercises on the files from the Exercise-1 folder and make the necessary changes. Make sure that you run these simulations in separate folders and start from clean files from the Exercise-1 folder.

11.11.11 Optional exercise 5: Play with the optimization parameters

The main parameter in the optimization algorithm is the step size and it is not always easy to choose this parameter. Luckily, the algorithm is quite robust and will work for different step sizes.
Run different simulations using step sizes 0.5 and 50.0 and try to rationalize the behavior. Normally, when the step size is too large, the system gets stuck in CV space and coefficients oscillate wildly. When the step size is too small, the algorithm runs out of "steam" too fast and the simulation converges slowly. These two extreme cases should be avoided.

11.11.12 Optional exercise 6: Uniform target distribution

Perform a simulation using an uniform target distribution and see how this changes the results.
In this case, you need to change the target distribution to

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-11.txt
# Target distribution
td: TD_UNIFORM
```

and remove the TARGETDIST_STRIDE and TARGETDIST_OUTPUT keywords from the OPT_AVERAGED_SGD action.

11.11.13 Optional exercise 7: Legendre polynomials basis function

Perform a simulation using Legendre polynomials (BF_LEGENDRE) basis functions instead of the wavelets and see how this will affect the result. As the Legendre polynomials are delocalized basis functions, this should lead to more fluctuations in the bias potential as has been observed in the paper introducing the wavelets.
In this case, you need to change the basis set action in the PLUMED input to

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-11.txt
# Basisset
BF_LEGENDRE ...
  LABEL=bf1
  ORDER=20
  MINIMUM=0.2
  MAXIMUM=0.7
... BF_LEGENDRE
```

11.11.14 Additional comments

This should cover the basics of running VES simulations in PLUMED, but the following comments might be of interest to some.

11.11.14.1 Restarting

VES simulations can be easily restarted. The code will automatically output all the file needed at the end of the simulation. To restart, you just need to reset the simulation with your MD code in the traditional way and add a RESET keyword to the top of the PLUMED input (for some codes like GROMACS, a restart is automatically detected by PLUMED and thus this keyword is not needed).

11.11.14.2 Multiple Walkers

VES simulations supports the usage of multiple walkers where different copies of the system share the same bias potential (i.e. coefficients) and cooperatively sample the averages needed for the gradient and Hessian. This can significantly help with convergence in difficult cases. It is of course best to start the different copies from different positions in CV space. To activate this option you just need to add the MULTIPLE_WALKERS keyword to the OPT_AVERAGED_SGD action. Note that this is only supported if the MD code support running multiple replicas connected via MPI (e.g., GROMACS or LAMMPS).

11.12 PLUMED Masterclass 22.12: Liquid-solid chemical potential differences with the environment similarity CV

Authors

Pablo Piaggi

Date

July 18, 2022

11.12.1 Aims

This Masterclass is an introduction to the use of the environment similarity CV available in PLUMED to the calculation of liquid-solid free energy differences (chemical potentials).

11.12.2 Objectives

The objectives of this Masterclass are:

- Learn to define reference environments for a crystal structure and choose appropriate parameters
- Learn how to run **bulk interconversion** simulations in which the bulk liquid and the bulk solid are reversibly interconverted
- Learn how to run **biased coexistence** simulations (similar to interface pinning)
- Learn how to perform **thermodynamic integration** along isobars
- Understand the importance of finite size effects
- Understand the advantages and limitations of each method
- Understand how these methods complement each other

11.12.3 Prerequisites

We assume that the person that will follow this tutorial is familiar with the Linux terminal, LAMMPS, and basic functionality of PLUMED. Knowledge of one enhanced sampling technique, such as metadynamics, is recommended. Familiarity with at least one plotting software is required, for instance, gnuplot, xmgrace, or python with matplotlib.

11.12.4 Setting up PLUMED

We will use LAMMPS and PLUMED to perform the calculations. Conda packages with the software required for this class have been prepared and you can install them following the instructions in this link. Make sure to install the conda package for LAMMPS.
The environment similarity CV is a part of the crystallization module and you will need to enable this module if you are compiling PLUMED on your own. Also, LAMMPS has to be compiled with the following optional packages PLUMED, MANYBODY, EXTRA-DUMP, and EXTRA-FIX.
The data needed to run the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/PabloPiaggi/masterclass-22-12
```

I suggest that you install the trajectory visualization software Ovito before starting the tutorial.

11.12.5 Summary of theory

We would like to define a per-atom quantity that tells us how similar the environments around atoms in the simulation box are to the environments found in some reference crystal structure. Crystal structures are usually defined by a Bravais lattice with an associated basis of atoms. An important consideration, that follows from the symmetry properties of lattices, is that the number of distinct environments that exist in a crystal structure is equal to the number of atoms in the basis M. To judge if an environment is compatible with a given crystal structure, we will have to compare it against M reference environments.
The environment similarity kernel is a way to quantify the similarity between environments. This idea was introduced in this article [27] and is based on the SOAP descriptors of Bartok et al. Unlike SOAPs, this kernel will be non rotationally invariant and thus will break the SO(3) symmetry of space. The starting point for the definition of our CV is the local atomic density around a given atom. We consider an environment χ around this atom and we define the density by,

$$\rho_\chi(\mathbf{r}) = \sum_{i\in\chi} \exp\left(-\frac{|\mathbf{r}_i - \mathbf{r}|^2}{2\sigma^2}\right),$$

where i runs over the neighbors in the environment χ, σ is a broadening parameter, and $\mathbf{r}_i$ are the coordinates of the neighbors relative to the central atom. We will see later how the best σ can be rigorously determined. We now define a single reference environment χ_0 that contains n reference positions $\{\mathbf{r}_1^0, ..., \mathbf{r}_n^0\}$ that describe, for instance, the nearest neighbors in a given crystal structure.
The environments χ and χ_0 are compared using the kernel,

$$k_{\chi_0}(\chi) = \int d\mathbf{r}\rho_\chi(\mathbf{r})\rho_{\chi_0}(\mathbf{r}).$$

Combining the two equations above and performing the integration analytically we obtain,

$$k_{\chi_0}(\chi) = \sum_{i\in\chi}\sum_{j\in\chi_0} \pi^{3/2}\sigma^3 \exp\left(-\frac{|\mathbf{r}_i - \mathbf{r}_j^0|^2}{4\sigma^2}\right).$$

The kernel is finally normalized,

$$\tilde{k}_{\chi_0}(\chi) = \frac{1}{n}\sum_{i\in\chi}\sum_{j\in\chi_0} \exp\left(-\frac{|\mathbf{r}_i - \mathbf{r}_j^0|^2}{4\sigma^2}\right),$$

such that $\tilde{k}_{\chi_0}(\chi_0) = 1$. We note that using the normalization above, the measure looses the symmetry property $k_{\chi_0}(\chi) = k_\chi(\chi_0)$.
The definition above is only useful for Bravais lattices since these have a single, unique atomic environment. The kernel can be generalized to crystal structures described as a lattice with a basis of more than one atom. As discussed above, in this case there is more than one type of environment. We consider the case of M environments $X = \chi_1, \chi_2, ..., \chi_M$ and we define the kernel through a best match strategy:

$$\tilde{k}_X(\chi) = \frac{1}{\lambda}\log\left(\sum_{l=1}^{M}\exp\left(\lambda\,\tilde{k}_{\chi_l}(\chi)\right)\right).$$

For a large enough λ this expression will select the largest $\tilde{k}_{\chi_l}(\chi)$ with $\chi_l \in X$.
$\tilde{k}_X(\chi)$ is a per-atom quantity and we will have to compute global functions of these quantities, for instance, the mean or the number of atoms with at $\tilde{k}_X(\chi)$ larger than some value.

11.12.6 The system: Sodium as the alanine dipeptide of solids

Alanine dipeptide is the prototypical system on which enhanced sampling methods are tested (and sometimes benchmarked) on. The reasons behind this choice is that alanine dipeptide is a small and relatively simple molecule that nonetheless captures well the phenomenon of conformational isomerism with relatively large free energy barriers. I argue that a similar system in the context of materials is sodium. It crystallizes in the bcc structure which is one of the simplest crystal structures. Also, it does not show the competition between fcc and hcp structures typical of systems that crystallize in these two compact structures. This fact greatly simplifies the task of characterizing the structure: all that we can see is liquid or bcc environments. We will use an embedded atom model (EAM) for sodium that is described in this paper. The melting temperature is supposed to be 366 K. Let's see if we can reproduce that result.

11.12.7 Exercise 1: Choosing reference environments and appropriate sigma values

In the case of the bcc structure, the reference environment for the environment similarity CV can be chosen automatically using the following PLUMED input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-12.txt
ENVIRONMENTSIMILARITY ...
 SPECIES=1-1024
 CRYSTAL_STRUCTURE=BCC
 LATTICE_CONSTANTS=0.423
 SIGMA=0.07
...
```

Here we have chosen the bcc crystal structure with a lattice constant of 0.423 nm which is appropriate for bcc sodium. This choice will automatically build an environment with 14 nearest neighbors. There are other structures for which environments can be built automatically, see the manual page of ENVIRONMENTSIMILARITY for further information. Another option is to use CRYSTAL_STRUCTURE=CUSTOM and provide environments manually as PDB files using the REFERENCE keyword. In this case, I suggest using the `Environment Finder` app to determine the appropriate environments for your structure.
In the PLUMED input above, we also have to determine the value for the SIGMA keyword. The first exercise of this class is to determine this value. Here is the rationale behind the choice. We are trying to determine a per-atom quantity that is able to distinguish liquid from solid environments. We should find a SIGMA such that the probability of mislabeling liquid atoms as solid, and vice versa, is minimized. This can be done by computing the probability of finding a given value of $\tilde{k}_X(\chi)$ in the bulk solid and the bulk liquid. So, first things first, let's do a simulation of the bulk liquid and the bulk bcc solid!
If you are using the conda environment for the masterclass, activate it now,

```
conda activate plumed-masterclass-2022
```

Now, cd to the folder 1-distributions/bcc and run the command:

```
mpiexec lmp -in start.lmp > /dev/null &
```

Then cd to the folder 1-distributions/liquid and run again the same command. These commands will run simulations of the bulk liquid and bulk solid at 1 bar and 375 K. I have chosen this temperature because it is close to coexistence for these phases. The files start.lmp are the LAMMPS input and they specify that we are doing NPT simulations. The output of these runs are in the files log.lammps, dump.na, and out.dcd . log.lammps is a log file that also contains some thermodynamic output. dump.na is the trajectory in LAMMPS dump atom format and it is useful for visualization with `Ovito`. out.dcd is the trajectory in DCD format, which is useful to post-process directly using PLUMED's driver, as we shall see.
Once that these simulations have completed, we will compute the distributions of the environment similarity kernel using this input

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-12.txt
s: ENVIRONMENTSIMILARITY ...
 SPECIES=1-1024
 SIGMA=replace
 CRYSTAL_STRUCTURE=BCC
 LATTICE_CONSTANTS=0.423
 MEAN
...

hh: HISTOGRAM ...
   DATA=s
   GRID_MIN=-0.5
   GRID_MAX=2
   GRID_BIN=1000
   BANDWIDTH=0.01
...

DUMPGRID GRID=hh FILE=histo
```

Here, SIGMA=replace has to be replaced by an appropriate value. The action HISTOGRAM will compute the normalized histogram using all per-atom values of the kernel and the action DUMPGRID will write it to a file named histo. We have to compute these histograms for different values of SIGMA and this can be done with the script run.sh in the folder 1-distributions (execute the command ./run.sh $>$ results.txt). This script will use the PLUMED driver and the DCD trajectory files to calculate the distributions of the environment similarity kernel. Here's the command we are using inside run.sh to do this,

```
mpiexec plumed driver --plumed plumed.dat --mf_dcd out.dcd > plumed.out
```

but you don't need to execute this yourself because it is done in the script run.sh. The script run.sh will also compute the overlap between the liquid and bcc distributions for different SIGMA. The overlap $O(p,q)$ between two distributions $p(x)$ and $q(x)$ can be defined in a variety of ways. Here we use,

$$O(p,q) = \int dx\, min[p(x), q(x)]$$

The output of the script will have two columns, the sigma value used and the overlap between the liquid and solid distributions of the kernel for that sigma. Plot sigma vs overlap. Nice results? Upload them to Slack! The best choice of SIGMA will be the one that minimizes the overlap which in this case should be around 0.07 nm. Did you get that result? Great! Then, let's move to the next task.
Now let's plot the distributions of the kernel in the liquid and in the solid. Plot simultaneously the histograms in `1-distributions/liquid/histo-0.07` and `1-distributions/bcc/histo-0.07` (column 1 vs column 2). How do the distributions look like? Do they have overlap? Determine 1) the maximum for each distribution, and 2) the kernel value for which both distributions have equal values. We will need these values for the next exercise! Feel free to share these values and your plots on Slack as you get them.

11.12.8 Exercise 2: Bulk interconversion

We now move to the first method to calculate chemical potentials, the bulk interconversion method. This method is based on simulating the reversible interconversion of the liquid to the solid and calculating the difference in chemical potential using:

$$\Delta\mu = -\frac{1}{\beta N}\ln\left(Z_\beta / Z_\alpha\right)$$

with Z_α and Z_β the partition functions restricted to each phase. These are computed with the formulae,

$$Z_\alpha = \int_{-\infty}^{s^*} ds \int d\mathbf{R} e^{-\beta[U(\mathbf{R},V)+PV]}\delta(s - s(\mathbf{R},V))$$

$$Z_\beta = \int_{s^*}^{\infty} ds \int d\mathbf{R} e^{-\beta[U(\mathbf{R},V)+PV]}\delta(s - s(\mathbf{R},V))$$

assuming that the collective variable s can separate well the phases at threshold s^* . The ratio of the partition functions is easy to calculate if we have a simulation in which both phases are visited, as we shall see below.
In a standard MD simulation we would only observe a single phase, and in order to see interconversion we need to apply a bias potential. Here we will use OPES (On-the-fly Probability Enhanced Sampling) to construct a bias potential as a function of the environment similarity kernels. If you prefer to use METAD or Variationally Enhanced Sampling (VES code) , you can do so, adapting the input accordingly. Otherwise, you may want to have a look at the PLUMED Masterclass 22.3: OPES method.
The environment similarity kernel, which are per-atom quantities, can be combined into a global quantity by evaluating how many have a value larger than some threshold. In file 2-bulk-interconversion/250atoms/400K/plumed.dat we can see the input for a simulation with 250 atoms at 400 K,

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-12.txt
cv: ENVIRONMENTSIMILARITY ...
 SPECIES=1-250
 SIGMA=0.07
 CRYSTAL_STRUCTURE=BCC
 LATTICE_CONSTANTS=0.423
 MORE_THAN={CUBIC D_0=0.38 D_MAX=0.80}
 MEAN
...
```

Here you see that we are calculating the MEAN and the number of atoms that have a value of the kernel MORE_↩ THAN something. Here, instead of having a strict threshold at some value, we have a continuous and differentiable switching function that is 0 below 0.38, 1 above 0.80, and changes smoothly from 0 to 1 between these values. The values for D_0 and D_MAX have been chosen based on the maxima of the distributions determined in the previous exercise. Did you get those values right? I hope you did.
The input for OPES is very simple,

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-12.txt
opes: OPES_METAD ARG=cv.morethan PACE=500 BARRIER=100 TEMP=400 STRIDE=2
```

and uses the morethan value described above as CV. We estimate a barrier of less than 100 kJ/mol and that's the rationale behind the choice of BARRIER. If you are using well tempered METAD you might want to use a biasfactor of around 40. The other keywords are fairly standard.
We will also use a trick to avoid forming structures that have orientations different from the one that we are trying to form. This will be discussed during the lecture and the PLUMED input is:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-12.txt
q6: Q6 SPECIES=1-250 SWITCH={CUBIC D_0=0.4 D_MAX=0.5} VMEAN
diff: MATHEVAL ARG=q6.vmean,cv.mean FUNC=(x-0.0604)/(0.397-0.0604)-(y-0.387)/(0.756-0.387) PERIODIC=NO
UPPER_WALLS ARG=diff AT=0.1 KAPPA=100000 EXP=2 LABEL=uwall STRIDE=2
```

In a nutshell, we are limiting the increase of Q6 (rotationally invariant) not followed by an increase in ENVIRONM↩ ENTSIMILARITY (non rotationally invariant).
The first task of this exercise is to run a few simulations at different temperatures (say, 380 K, 400 K, 420K, or more if you have a computer cluster available). In order to run the simulation you have to execute on the folder 2-bulk-interconversion/250atoms/400K the command:

```
mpiexec lmp -in start.lmp > /dev/null &
```

10 or 20 ns of simulation should suffice, but you may run them for (much) longer if you are using a computer cluster. To run simulations at other temperatures, copy the folder 400K, for example,

```
cp -r 400K 380K
```

And edit the files plumed.dat and start.lmp to change the temperature from 400 K to 380 K. The lines where you should change the temperatures are highlighted! You can run as many simulations as you want to, but at least two are required. Each simulation should take about 2 hours to get $\sim$10 ns if you are running on $\sim$8 cores.
Now that the simulations are running, let's start analyzing them. You don't have to wait until they finish to start with the analysis. The most useful output of these simulations is in the file COLVAR. For instance, the first column is the time, the third column is the environment similarity morethan value, and the fourth column is the OPES bias. Let's plot time vs environment similarity morethan (column 1 vs column 3). What is happening during the simulation? Are there transitions between liquid and solid phases? I suggest that you also visualize the trajectory in file dump.na using Ovito. Try the modifiers Smooth Trajectory (with window size 5) and Polyhedral Template Matching. Is the bcc solid forming? Is it well aligned with the simulation box? It is also interesting to compute the free energy surface (FES). I provide a simple script fes.py to do this with python, numpy and matplotlib and you can run it using,

```
python fes.py
```

This script is based in constructing a histogram $h(s)$ with weights $e^{\beta V}$ where V the bias potential. The FES is then obtained using,

$$G(s) = -\frac{1}{\beta} \ln h(s).$$

You can plot the FES at all the temperatures you have run simulations at. How does the FES look like? What does each basin represent? How do they change with temperature? Don't forget to upload your plots to Slack!
Now let's move to the most important task of this section, the calculation of differences in chemical potentials. I prepared a basic python script that you can use to computed the difference in chemical potential between the liquid and the solid. You can find it in 2-bulk-interconversion/250atoms/chemical_potentials.py . You have to edit the highlighted lines to include more temperatures in the calculation and in the plot that it will generate. If you have at least two temperatures, it will fit a straight line and estimate the coexistence temperature from the condition that the chemical potential difference is zero at coexistence. Got nice results? I'll **buy a coffee** to the first person that uploads a plot of the chemical potentials vs temperature to Slack! Of course, I'll buy it next time we meet in person, if ever. Do the chemical potential differences depend linearly on the temperature? What melting temperature do you find? How does it compare with the reference value (366 K) that has been reported?
If you have enough computer resources and time, I suggest that you repeat the exercise with a larger system, for instance, with 1024 atoms. I prepared input files in the folder 2-bulk-interconversion/1024atoms/400K. Do you find the same chemical potentials and coexistence temperature? Remember that the difference in chemical potential and coexistence temperature depend linearly on 1/N with N the system size. You can use this fact to extrapolate to the thermodynamic limit. Did you get a better coexistence temperature now? Which system sizes are big enough to make an error of about 1 K in the coexistence temperature?

11.12.9 Exercise 3: Biased coexistence

In this section we will use a different method to calculate the difference in chemical potential between the liquid and the solid. I call it **biased coexistence** and it is a generalization of the `interface pinning` method. The idea

of the method is to simulate the liquid and the solid (or any pair of phases) in direct coexistence. Then, we add a bias potential such that we sample reversibly the growth and melting of a portion of the system, for instance, one layer of the solid. From this simulation we can compute the free energy as a function of the number of solid-like atoms in the system $G(N)$. This quantity is connected to the chemical potential in the following way,

$$G(N) \approx \Delta\mu\, N + const$$

This equation is an approximation and rests on several assumptions, yet it tells us that the chemical potential is the slope of the FES with respect to N.
The files to perform a biased coexistence simulation with 2048 atoms can be found in the folder 3-biased-coexistence/2048atoms/400K. The LAMMPS input script start.lmp will do a lot of things automatically for us:

- Create a bcc crystal and equilibrate it in NPT
- Determine the equilibrium box dimensions and fix the lengths in x and y to their equilibrium value at the simulated temperature
- Replicate the crystal along the z axis and melt half of the crystal in the box
- Equilibrate the liquid-solid surface under the constraint of having around half the atoms with solid-like environments in the NP_zT ensemble
- Production run with a bias potential in the NP_zT ensemble

The equilibration under a constraint is done using the following PLUMED input found on plumed.equil.dat,

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-12.txt
uwall: UPPER_WALLS ARG=cv.morethan AT=1152. KAPPA=0.1 EXP=2 STRIDE=2
lwall: LOWER_WALLS ARG=cv.morethan AT=896. KAPPA=0.1 EXP=2 STRIDE=2
```

And in the production run, the bias is defined in file plumed.dat and is,

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-masterclass-22-12.txt
opes: OPES_METAD ARG=cv.morethan PACE=500 BARRIER=100 TEMP=400 STRIDE=2

uwall: UPPER_WALLS ARG=cv.morethan AT=1152. KAPPA=0.1 EXP=2 STRIDE=2
lwall: LOWER_WALLS ARG=cv.morethan AT=896. KAPPA=0.1 EXP=2 STRIDE=2
```

Run the simulation at 400 K using the command,

```
mpiexec lmp -in start.lmp > /dev/null &
```

I estimate that you need at least 3 ns of simulation to see reasonable results and this should take a couple of hours on $\sim$8 cores. If you are running on a computer cluster, run it for longer. I suggest that you repeat the simulations at different temperatures, for instance, 360 K, 380 K, and 400 K, and you may do more temperatures if you have the computational resources. The lines of the input scripts plumed.dat and start.lmp where the temperature has to be changed are clearly highlighted.
Let's see the progress of our simulation by plotting time vs environment similarity morethan (column 1 vs 3) on the COLVAR file. How does it look like? What is happening to the system under the action of the OPES bias and the walls? You can visualize the trajectory dump.na using Ovito. Now we shall calculate the chemical potential from the slope of the FES with respect to the number of solid-like atoms. In order to do this, we will use the script chemical_potential_bc.py found on folder 3-biased-coexistence/2048atoms. This script will compute the F↩ ES, fit a straight line to get the slope (chemical potentials), plot the chemical potentials vs temperature, and get the coexistence temperature. Depending on how many temperatures you have performed, go into the script and add more temperatures. The line that you need to change has been highlighted. The FES is computed from the histogram with weights $e^{\beta V}$ where V is the bias potential. Are the FES linear with the number of solid-like atoms? What melting temperature did you find?
Note that we are computing a "strict" morethan value for the environment similarity kernel that we use for post-processing. This definition can't be used for biasing but it is a better way to count solid-like atoms and we use it to construct the FES. The definition of the "strict" version is,

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-12.txt
ENVIRONMENTSIMILARITY ...
 SPECIES=1-2048
 SIGMA=0.07
 CRYSTAL_STRUCTURE=BCC
 LATTICE_CONSTANTS=0.423
 MORE_THAN={CUBIC D_0=0.5525 D_MAX=0.55251}
...
```

and is in column 13 of the COLVAR file.
Discuss the advantages and disadvantages of the two methods presented above. Which one did you like best? Why?

11.12.10 Exercise 4: Thermodynamic integration along isobars

Perhaps the simplest method to compute chemical potential differences is thermodynamic integration. Here we will discuss the version of the method in which the integration is performed along isobars. It is very cheap and easy. What's the catch? You have to know the chemical potential difference for some thermodynamic condition in order to use it. It is therefore a perfect complement for the methods described above. PLUMED is not required for this exercise, but we will use the results obtained here to validate the methods described above.
The method relies on the following thermodynamic identity,

$$\frac{\Delta\mu(T,P)}{T} = \frac{\Delta\mu(T_0,P)}{T_0} - \int_{T_0}^{T} \frac{\Delta h(T',P)}{T'^2} dT'$$

where T_0 is the reference temperature at which $\Delta\mu$ is known, and Δh is the per-atom enthalpy difference. If T_0 is the coexistence temperature, the first term on the RHS vanishes.
In order to use this method, we shall compute the enthalpy for the liquid and the solid at several temperatures and pressures. Example scripts are given in folders 4-thermodynamic-integration/liquid and 4-thermodynamic-integration/bcc . You can use the script newtemp.sh to create folders to run simulations at different temperatures between 330 K and 410 K in steps of 10 K. It is used in the following way,

```
./newtemp.sh 400
```

to create the folder 400K with inputs for the simulation at a temperature of 400 K. The command to run the LAMM↩ PS input scripts start.lmp is the same as the one above. The simulations should be fast, a couple of minutes each one, and they will output a file thermo.txt that has the enthalpy. I prepared a python script chemical_potentials_ti.py located in folder 4-thermodynamic-integration that plots the enthalpies vs temperature, performs the integration, and then plots the chemical potential differences vs temperature. How do these chemical potentials compare with the ones obtained with other methods?

11.12.11 Final thoughts

We have discussed three methods to calculate chemical potential differences between the liquid and a solid. There are other methods that are not described here because they are not based on enhanced sampling techniques. Also, an important tool to study phase equilibria is the integration of the Clausius-Clapeyron equation to trace coexistence lines. The latter method is an important complement to the methods discussed in this tutorial and extremely useful when computing phase diagrams in the temperature-pressure plane.

11.13 PLUMED Masterclass 22.13: SASA module and the application of PLUMED for implicit solvent simulations

Authors

Andrea Arsiccio

Date

September 12, 2022

11.13.1 Aims

This Masterclass is an introduction to the use of the SASA module of Plumed for the execution of implicit solvent simulations.

11.13.2 Objectives

The objectives of this Masterclass are:

- Learn how the SASA module of PLUMED works
- Learn how to run an implicit solvent simulation of a protein using PLUMED
- Learn how to introduce the effect of temperature, pressure and osmolytes on protein stability in implicit solvent simulations
- Understand the advantages and limitations of implicit solvent simulations using PLUMED

11.13.3 Prerequisites

We assume that the person that will follow this tutorial is familiar with the Linux terminal, AMBER and basic functionality of PLUMED. Knowledge of the metadynamics enhanced sampling technique is recommended. Familiarity with gnuplot and xmgrace (or python with matplotlib) is recommended.

11.13.4 Setting up PLUMED

We will use AMBER and PLUMED to perform the calculations. Conda packages with the software required for this class have been prepared and you can install them following the instructions in this link. Make sure to install the conda package for AMBER.
If you are compiling PLUMED on your own, you will need to install the SASA module manually by adding '–enable-modules=sasa' to your './configure' command when building PLUMED.
The data needed to run the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/andrea-arsiccio/masterclass-22-13
```

You will also need a python script for the calculation of free energies of transfer, available on GitHub.

11.13.5 Summary of theory

The SASA module contains methods for the calculation of the solvent accessible surface area (SASA) of proteins. It can be used to include the SASA as a collective variable in metadynamics simulations, and for implicit solvent simulations.
There are two SASA functions that could be used:
SASA_HASEL: employs the algorithm described in [83] for the computation of the SASA.
SASA_LCPO: employs the algorithm described in [84] for the computation of the SASA.
The algorithm by Hasel et al. is about 2 to 3 times faster than the LCPO (linear combination of pairwise overlaps) algorithm, but slightly less accurate. Both algorithms have the advantage that apply simple analytical functions for the computation of the SASA, and because of this it is easy to compute their derivative, which is a necessary step to apply a bias based on the SASA to a molecular dynamics simulation.
The atoms for which the SASA is desired should be indicated with the keyword ATOMS, and a pdb file of the protein must be provided in input with the MOLINFO keyword. Two types of calculations are possible:
TOTAL: the total SASA is computed, which is the desired type of calculation if one wants to use the SASA as a CV in metadynamics simulations.
TRANSFER: in this case the free energy of transfer for the protein is computed according to the transfer model (T↩RANSFER). This keyword can be used, for instance, to compute the transfer of a protein to different temperatures, as detailed in [86], or to different pressures, as detailed in [87], or to different osmolyte solutions, as detailed in [85].
Using the protein SASA as a CV in metadynamics simulations, or monitoring the SASA on the fly during the simulation or while postprocessing a trajectory, is straightforward. The user, after having enabled the SASA module, can use the SASA functions mentioned above similarly to any other CV in PLUMED.
For instance, the following input tells plumed to print the total SASA for atoms 10 to 20 in a protein chain:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-13.txt
SASA_HASEL TYPE=TOTAL ATOMS=10-20 NL_STRIDE=10 LABEL=sasa
PRINT ARG=sasa STRIDE=1 FILE=colvar
```

This tutorial will focus on the use of the SASA module for running implicit solvent simulations with PLUMED. In implicit solvent simulations, the solvent is not any more described at atomistic/coarse-grained level, but is instead treated as a continuum and described as a mean field. The advantages are that the absence of the solvent degrees of freedom alleviates the computational cost, and the absence of viscous friction accelerates the exploration of the protein conformational space. However, the solvent effects are described less accurately. The degree of accuracy can be improved by using the concept of free energy of transfer, which is at the basis of the SASA module.
The common implementation of implicit solvent simulations relies on the computation of the free energy of a protein as the sum of three contributions:

$$G^{tot} = E^{vac} + G^{np} + G^{el}$$

where E^{vac} is the molecule's energy in vacuum, which is the sum of internal contributions (bond and angle stretching, dihedral angles interactions) and van der Waals energy terms. G^{np} is the non-polar solvation contribution, i.e., the free energy of solvation for a molecule from which all charges have been removed. G^{el} is the electrostatic part,

calculated as the free energy for turning on the partial charges in solution. This approach has been developed to describe (implicitly) the solvation of a protein in pure water, at ambient temperature (298 K) and pressure (1 atm).
The idea behind the SASA module is the addition of a free energy of transfer term $G^{tr}(T, P, c)$:

$$G^{tot} = E^{vac} + G^{np} + G^{el} + G^{tr}(T, P, c)$$

that describes the transfer of the protein to any given temperature T, pressure P and concentration c of an osmolyte.
The free energy of transfer term has the following functional form:

$$G^{tr} = \sum_{j=1}^{n} g^{tr}_{j,sc} \alpha_{j,sc} + g^{tr}_{bb} \sum_{j=1}^{n} \alpha_{j,bb}$$

where n is the number of residues in the protein and the global transfer free energy is obtained by summing the contributions given by the amino acid side chains ($g^{tr}_{j,sc}$) and by the peptide backbone (g^{tr}_{bb}).
Each contribution is weighed by the fractional solvent accessible surface area $SASA_j$ of residue j,

$$\alpha_j = \frac{SASA_j}{SASA_{j,Gly-X-Gly}}$$

where $SASA_{j,Gly-X-Gly}$ is the solvent accessibility of amino acid X in the tripeptide Gly-X-Gly, and X is the amino acid residue type j.
The amino acid side chains ($g^{tr}_{j,sc}$) and peptide backbone (g^{tr}_{bb}) contributions to the transfer free energy are computed according to the mathematical derivation described in [86] (for the effect of temperature), [87] (for the effect of pressure), or [85] (for the effect of osmolyte concentration).
Briefly, the tranfer free energy contributions describing the effect of temperatures have been derived by downloading a large set of Protein Data Bank (pdb) files resolved by nuclear magnetic resonance at different temperatures, and by computing the probability of different side chains/backbone groups to be surface exposed at different temperatures. The relation between energy and probability has then been exploited to compute the free energy of transfer contributions as a function of temperature.
The transfer free energy terms describing the effect of pressure have been obtained by computing three different contributions for each side chain/backbone group: 1) the elimination of a consistent fraction of the void volumes that are present within the native state upon unfolding, 2) the volume reduction of bound water molecules due to the increased SASA of a protein upon unfolding, and 3) the stabilizing excluded volume effect of a denser solvent.
Finally, the transfer free energy contributions as function of osmolyte type/concentrations have been obtained from experimental works, where solubility measurements of protein amino acids/peptide backbone models have been conducted in different osmolyte solutions and exploited to extract the difference in chemical potential between water and the osmolyte solution itself.
The interested user is invited to read the original works on this topic ([86], [87] and [85]) for a better understanding of the theory behind the free energy of transfer contributions.
When the TRANSFER keyword is used, a file with the free energy of transfer values for the sidechains ($g^{tr}_{j,sc}$) and backbone (g^{tr}_{bb}) atoms should be provided (using the keyword DELTAGFILE). Such file should have the following format:

```
----------------Sample DeltaG.dat file----------------------
ALA     0.711019999999962
ARG     -2.24832799999996
ASN     -2.74838799999999
ASP     -2.5626376
CYS     3.89864000000006
GLN     -1.76192
GLU     -2.38664400000002
GLY     0
HIS     -3.58152799999999
ILE     2.42634399999986
LEU     1.77233599999988
LYS     -1.92576400000002
MET     -0.262827999999956
PHE     1.62028800000007
PRO     -2.15598800000001
SER     -1.60934800000004
THR     -0.591559999999987
TRP     1.22936000000027
TYR     0.775547999999958
VAL     2.12779200000011
BACKBONE        1.00066920000002
------------------------------------------------------------
```

where the second column is the free energy of transfer for each sidechain/backbone, in kJ/mol.
A Python script for the computation of free energy of transfer values to describe the effect of osmolyte concentration, temperature and pressure (according to [85], [86] and [87]) is freely available on GitHub. The script automatically outputs a DeltaG.dat file compatible with the SASA module. Please have a look at the README file of this script to better understand its usage.
For instance, the following input tells plumed to compute the transfer free energy for the protein chain containing atoms 10 to 20. Such transfer free energy is then used as a bias in the simulation (e.g., implicit solvent simulations). The free energy of transfer values are read from a file called DeltaG.dat:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-13.txt
SASA_HASEL TYPE=TRANSFER ATOMS=10-20 NL_STRIDE=10 DELTAGFILE=DeltaG.dat LABEL=sasa

bias: BIASVALUE ARG=sasa

PRINT ARG=sasa,bias.* STRIDE=1 FILE=colvar
```

If the DELTAGFILE is not provided, the SASA module computes the free energy of transfer values as if they had to take into account the effect of temperature according to approaches 2 or 3 (they differ in the mathematical model employed to extract free energies of transfer) in the paper [86]. Please read and cite this paper if using the transfer model for computing the effect of temperature in implicit solvent simulations. For this purpose, the keyword APP↩ ROACH should be added, and set to either 2 or 3, as exemplified in the following input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-masterclass-22-13.txt
SASA_HASEL TYPE=TRANSFER ATOMS=10-20 NL_STRIDE=10 APPROACH=2 LABEL=sasa

bias: BIASVALUE ARG=sasa

PRINT ARG=sasa,bias.* STRIDE=1 FILE=colvar
```

11.13.6 The system: Refolding of a model peptide

For this tutorial we will work on a model peptide called (AAQAA) $_3$. (AAQAA) $_3$ is a short peptide with known α-helix structure. We will simulate this peptide starting from an unfolded conformation, and we will see in which conditions (of temperature, pressure, and solution composition) it folds back to a α-helix, and which conditions promote instead an unfolded conformation.
We will perform the simulations in implicit solvent, using the AMBER ff03 force field.
For this tutorial, the following conditions will be explored:
1) 298 K, 1 bar, 0 M
2) 228 K (approach 2 in [86]), 1 bar, 0 M
3) 348 K (approach 2 in [86]), 1 bar, 0 M
4) 298 K, 1 bar, 10 M urea
5) 298 K, 3 kbar, 0 M
Each simulation will be performed for 5 ns.

11.13.7 Exercise 1: Simulation 1 at 298 K, 1 bar and 0 M osmolyte concentration

For the exercises, we will need PLUMED and AMBER. For this purpose, you should first proceed to their installation using the provided conda packages:

```
#install the PLUMED environment
conda create --name plumed-masterclass-2022
conda activate plumed-masterclass-2022
conda install -c conda-forge plumed py-plumed numpy pandas matplotlib notebook mdtraj mdanalysis git


#install the AMBER environment
conda create --name plumed-masterclass-2022-amber
conda activate plumed-masterclass-2022-amber
conda install -c conda-forge ambertools


#stack the two environments together
conda activate plumed-masterclass-2022
conda activate --stack plumed-masterclass-2022-amber
export PLUMED_KERNEL=/your_path_here/plumed-masterclass-2022/lib/libplumedKernel.so
```

Then, you should download the folder with the input files for the exercises using the following command:

```
git clone https://github.com/andrea-arsiccio/masterclass-22-13
```

In order to perform exercise 1, cd to the folder SASA_module/01_298K. There you will find a number of files:
-aaqaa.prmtop: a topology file of the protein, described according to the AMBER ff03 force field.
-AAQAA_298.in: an input file for running the simulation through AMBER. I will go more in detail over this file in the following.
-aaqaa_min.ncrst/aaqaa_min.pdb: configuration files for the (unfolded) protein, which we will use as starting point for our simulations.
-histograms.py/picture_rg_alpha.gnu/script_rg_alpha.bash: files that we will use for the analyses and postprocessing of our trajectories.
-plumed.dat: PLUMED input file. I will go more in detail over this in the following.
The AAQAA_298.in file is an input AMBER file for carrying out our implicit solvent simulation, and it has the following aspect:

```
MD Generalise Born, no cut off
 &cntrl
  imin = 0,
  igb = 5, gbsa = 1, extdiel = 78.4, ntpr = 1000, ntwx = 1000, ntwr = 1000,
  ntt = 3, gamma_ln = 1.0, ig = -1, nscm = 500,
  tempi = 298.0, temp0 = 298.0,
  nstlim = 2500000, dt = 0.002, ntc = 2, ntf = 2,
  cut = 9999.0, plumed = 1, plumedfile = 'plumed.dat'
 /
```

where the different commands have the following meaning:
imin = 0: we are running an MD simulation without performing energy minimization
igb = 5: we are using the model described in https://doi.org/10.1002/prot.20033 for computing the electrostatic interactions G^{el}
gbsa = 1: the G^{np} term is also computed
extdiel = 78.4: the dielectric constant of the solvent
ntpr = 1000: we print energy info every 1000 steps
ntwx = 1000: we print coordinates every 1000 steps
ntwr = 1000: we print the restart file every 1000 steps
ntt = 3: we apply Langevin dynamics
gamma_ln = 1.0: we set the collision frequency for Langevin dynamics
ig = -1: seed for pseudo-random number generator. -1 means that the random seed will be based on the current date and time
nscm = 500: translational/rotational COM motions will be removed every 500 steps
tempi = 298.0: initial temperature
temp0 = 298.0: reference temperature at which the system is to be kept
nstlim = 2500000: we will run the system for 2500000 steps
dt = 0.002: we will apply a 2 fs time step
ntc = 2: we will constrain bonds linking to a hydrogen
ntf = 2: bond interactions involving H-atoms will be omitted in force evaluations
cut = 9999.0: no cut off will be applied for nonbonded interactions
plumed = 1: we will run PLUMED together with AMBER
plumedfile = 'plumed.dat': the PLUMED file to read is called plumed.dat
The PLUMED input file plumed.dat has, instead, the following aspect:

```
MOLINFO MOLTYPE=protein STRUCTURE=aaqaa_min.pdb

# radius of gyration
rgyr: GYRATION TYPE=RADIUS ATOMS=1-174

# antiparallel beta
ab: ANTIBETARMSD RESIDUES=all LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} STRANDS_CUTOFF=1

# parallel beta
pb: PARABETARMSD RESIDUES=all LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} STRANDS_CUTOFF=1

# alpha helix
alfa: ALPHARMSD RESIDUES=all LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12}
```

```
PRINT ...
 ARG=rgyr,ab.lessthan,pb.lessthan,alfa.lessthan
 STRIDE=1000
 FILE=COLVAR
... PRINT
```

We are providing PLUMED with a protein structure (aaqaa_min.pdb) in input using the MOLINFO keyword. We are then computing the radius of gyration, parallel/antiparallel beta-sheet content and alpha-helix content of the protein, and we are printing this info to a file called COLVAR every 1000 integration steps. (AAQAA) $_3$ should fold as a alpha-helix, but we are monitoring also the parallel/antiparallel beta-sheet content in order to identify potential cases of misfolding.
In this first example, we are simulating (AAQAA) $_3$ in pure water at 298 K and ambient pressure, so we do not need to add the free energy of transfer term $G^{tr}(T, P, c)$, and the SASA module is not even called in the PLUMED file. However, for exercises 2-5, you will need to use the SASA module to include the effect of temperature/pressure/urea concentration on protein stability.
To run the simulation, just type the command:

```
sander -O -i AAQAA_298.in -o aaqaa-MD.out -c aaqaa_min.ncrst -p aaqaa.prmtop -r aaqaa-MD.ncrst -x aaqaa-MD.nc
```

The simulation should take 1-2 hours on a common laptop, so you should not need a cluster for this masterclass.
At the end of the simulation, you will have a COLVAR file that lists the evolution of radius of gyration and secondary structure content of the protein as function of time. Try to plot the columns of the COLVAR file versus time. Does the protein fold? If yes, how fast is the process? Do you think the process would be equally fast in explicit solvent?
Here is the evolution of the helical content of (AAQAA) $_3$ over time during the simulation:
You can also use the script_rg_alpha.bash file provided to have a 2D representation of the conformational space explored by the protein during the simulation. Just type:

```
chmod +x script_rg_alpha.bash
./script_rg_alpha.bash COLVAR
```

The script generates a file called normFEL.dat that can be visualized in gnuplot using the picture_rg_alpha.gnu script provided. You should obtain something similar to:

11.13.8 Exercises 2-5: Introducing the free energy of transfer contribution to implicit solvent simulations

Using the learnings from the theoretical part of this masterclass and exercise 1, you should now be able to autonomously run exercises 2-5. The difference now is that you need to employ free energy of transfer contributions to simulate the solution conditions of exercises 2-5. Exercises 2 and 3 explore extreme (cold and hot) values of temperature. Exercise 4 includes the presence of a potent denaturant (urea), and exercise 5 is performed at a high pressure value.
This means that you will need to use the SASA module, as described in the thoretical summary, to introduce the effect of temperature, pressure and osmolyte concentration within the implicit solvent simulation.
The folder that you downloaded from GitHub contains 5 subfolders, one for each exercise. In each subfolder, every file is ready to use, with the only exception of the plumed.dat file, that you will need to write autonomously using the template employed in exercise 1. Specifically, please monitor for each exercise the radius of gyration and secondary structure content (alpha-helix, parallel and antiparalle beta-sheet) of the protein, and print this information to a file called COLVAR. You will also need to add a SASA_HASEL (hint: use SASA_HASEL instead of SASA_↩ LCPO, as the algorithm by Hasel et al. is faster) and a BIASVALUE section to the plumed.dat file, as discussed in the theoretical section.
You will need to perform the following steps:
-install the software (PLUMED, AMBER)
-clone the GitHub folder with input files: git clone https://github.com/andrea-arsiccio/masterclass-22-13
-write a PLUMED file in each subfolder according to the example of Exercise 1 (hint: use the DeltaG-calculation.py script if needed! The README for the script is available on GitHub, please have a look into it)
-Run the simulations using the command: sander -O -i AAQAA_%%%.in -o aaqaa-MD.out -c aaqaa_min.ncrst -p aaqaa.prmtop -r aaqaa-MD.ncrst -x aaqaa-MD.nc >& logfile
-Analyze the trajectories as already done during exercise 1 (plot the COLVAR columns versus time, and use the script_rg_alpha.bash file to get a representation of the conformational space explored)
You should ask yourself the following questions:

-what would you expect to occur at the protein structure at extreme values of temperature/pressure/urea concentration? Are your expections reflected in the simulation outputs?
-would you observe similar phenomena also in 5-ns-long explicit solvent calculations?

11.13.9 Final thoughts

Through this tutorial we have learnt that the SASA module of PLUMED implements two algorithms for the computation of the SASA (the faster algorithm by Hasel et al., and the more accurate LCPO algorithm).
We have found out that the SASA module can be used for introducing the SASA as a collective variable or for monitoring the SASA, but it can furthermore be employed to perform implicit solvent MD simulations.
Implicit solvent simulations with the SASA module are based on the concept of free energy of transfer (where the 'transfer' occurs from pure water at ambient temperature and pressure to an osmolyte solution at any desired value of temperature and pressure).
Implicit solvent simulations have the advantage, compared to their explicit solvent counterpart, to speed up conformational transitions, and allow a faster exploration of conformational space.
NOTE: The Masterclass files with the solutions to the exercises are also avilable on GitHub, but please do the exercises on your own before checking on the results!

11.14 PLUMED Masterclass 22.15: FISST module and application of mechanical forces with PLUMED

Authors

Glen M. Hocky

Date

October 17, 2022

11.14.1 Aims

This Masterclass explains how mechanical forces can be modeled using PLUMED, and the application of the FISST module for applying multiple forces simultaneously.

11.14.2 Objectives

The objectives of this Masterclass are:

- Learn how to apply a constant force in PLUMED
- Learn how to perform steered-MD in PLUMED, and approximately get a free energy surface from repeating this calculation
- Learn how to apply constant forces with FISST, and reweight to different intermediate forces

11.14.3 Prerequisites

We assume that the person that will follow this tutorial is familiar with the Linux terminal, Gromacs and basic functionality of PLUMED.
Familiarity with python and matplotlib is recommended for FISST reweighting and plotting

11.14.4 Setting up PLUMED

We will use GROMACS, LAMMPS, PLUMED, and PLUMED's pesmd function to perform the calculations. Conda packages with the software required for this class have been prepared and you can install them following the instructions in this link.
If you are compiling PLUMED on your own, you will need to install the FISST module manually by adding '–enable-modules=fisst' to your './configure' command when building PLUMED.
The data needed to run the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/hockyg/masterclass-22-15
```

11.14.5 Background

A force along some direction here is defined as the negative gradient of the potential energy along that direction.
A constant force F on a scalar collective variable $Q(\vec{X})$ therefore is a simple addition to the system's energy function.

$$U(\vec{X}, F) = U(\vec{X}) - FQ(\vec{X})$$

Notice that, because of the negative sign, a postive value of F results in a lower energy for large values of Q, meaning $F0$ corresponds to a "pulling" force.
A mechanical force would often in reality would correspond to pulling apart two atoms, and so Q would often be a simple distance coordinate.
Note however, that other quantities could be used, such as an area which would mean F corresponds to a pressure.
Dimensional analysis implies that the units of F must be [Energy]/[Q].
The effect of constant force can be assessed using any enhanced sampling method.
The work done by the bias in this case is very simple

$$W = \int_a^b F \cdot dQ = F(Q_b - Q_a)$$

Constant forces can be applied in PLUMED with the SLOPE keyword of the RESTRAINT bias.

11.14.6 Steered MD

Steered molecular dynamics (SMD) is one way of pulling on a molecular coordinate, and has a connection to experiments done where a molecule is attached via a "spring" to an object such as an optical tweezer or AFM tip.
To represent this in simulation, instead of applying a constant force, we impose a Harmonic restraint on Q with a center that moves:

$$U(\vec{X}, F) = U(\vec{X}) + \frac{1}{2}k(Q - Q_0(t))^2$$

Typically $Q_0(t)$ would move linearly, with $Q_0(t) = Q_0(0) - \lambda t$ although that is not a requirement.
At any given time, the force along Q from the moving bias is given as:

$$F(t) = -\frac{\partial U}{\partial Q} = -k(Q - Q_0(t))$$

This force is positive (pulling) when $Q_0(t)$ is bigger than Q, and it can get very large if the spring moves quickly to larger values.
SMD is implemented in PLUMED using the MOVINGRESTRAINT bias, and the work is computed automatically.

$$W = \int_a^B F dQ \approx \sum_{i=1}^{N_{steps}} \bar{F}_i(Q_0(t_i) - Q_0(t_{i-1})) = \lambda dt \sum_{i=1}^{N_{steps}} \bar{F}_i,$$

$$\bar{F}_i = \frac{1}{2}(F_i + F_{i-1}) = -\frac{k}{2}(Q(t_i) - Q_0(t_i) - Q(t_{i-1}) + Q_0(t_{i-1})) = -\frac{k}{2}(\Delta Q_i - \lambda dt)$$

11.14.7 FISST

Infinite Switch Simulated Tempering in Force (FISST) is a method implemented in Ref. [64]
This method takes advantage of the limit of changing a force from F_{min} to F_{max} and back infinitely quickly.
In this limit, the system feels an average force $\bar{F}(Q)$ that depends on the current value of the pulling coordinate.

$$U(\vec{X}, F) = U(\vec{X}) - \bar{F}(Q)Q(\vec{X})$$

In practice, this $\bar{F}(Q)$ is computed using weights $w(F)$ for each force in the force range that are learned on the fly, and separate "observable weights" $W_F(Q_i)$ are used to compute the average of any quantity A at a given force.

$$\langle A \rangle_F = \frac{1}{N} \sum_{i=1}^{N_{steps}} A(X(t_i)) W_F(Q_i)$$

where the observable weights are computed in a way that

$$W_F(Q_i) \propto \left(\int_{F_{min}}^{F_{max}} dF' w(F') e^{\beta(F'-F)Q_i} \right)^{-1}$$

The module writes out both the force weights and the observable weights to files.

11.14.8 Exercises

The exercises are presented below.

11.14.8.1 Effect of force on a 1-dimensional potential

Use the RESTRAINT function to add a constant force of different magnitudes (e.g. -5 to 5 in these units) and look at how the force changes the resulting free energy surface.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-masterclass-22-15.txt
UNITS ENERGY=kcal/mol

d1: DISTANCE ATOMS=1,2
ff: MATHEVAL ARG=d1 PERIODIC=NO FUNC=0.2*(((x-10)^2)*((x-20)^2))
bb: BIASVALUE ARG=ff

metad: METAD ARG=d1 PACE=500 HEIGHT=0.1 SIGMA=2.5 FILE=__FILL__ BIASFACTOR=10 TEMP=300.0 GRID_WFILE=__FILL__ G

RESTRAINT __FILL__

PRINT ARG=* FILE=__FILL__ STRIDE=100
```

Then run the simulation using the command:

```
plumed pesmd < doublewell_prod.pesmd.input
```

Plot the free energy surface from the GRID or after using sum_hills to compute the surface, and zero the potential at the left minimum. What do you notice about the other minimum and barrier?

11.14.8.2 Effect of force on a 2-dimensional potential

The following implements a "V-shaped" potential which has 2 minima at small y values and 1 minima at large y value.
Try plotting the potential to see.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-masterclass-22-15.txt
d1: DISTANCE ATOMS=1,2 COMPONENTS
ff: MATHEVAL ARG=d1.x,d1.y PERIODIC=NO FUNC=-8*log((exp(-(y-exp(-x))^2/2)+exp(-(y+exp(-x))^2/2))*exp(-x^2/2))
bb: BIASVALUE ARG=ff

PRINT ARG=* FILE=__FILL__ STRIDE=100
```

First run a simulation using the command, and make a 2d histogram of x and y to show that it is trapped on one side.

```
plumed pesmd < doublewell_prod.pesmd.input
```

If you have time, use RESTRAINT to add a constant force in the Y direction that favors the higher vertical state.
Now add FISST in order to sample forces all at once:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-masterclass-22-15.txt
f: FISST MIN_FORCE=-15 MAX_FORCE=15.0 PERIOD=200 NINTERPOLATE=41 ARG=d1.x KBT=1.0 OUT_RESTART=__FILL__ OUT_OBS
```

We used forces -15 to 15 in the paper, but feel free to experiment with your own force range. Inspect the observable file and restart file to see what is in there.
Now generate a histogram at each force using the following command, which will render it as an animated gif if you have the proper libraries installed.

```
python reweight_vshape.py observable_file cv_file
```

11.14.8.3 Model bistable helix

In this section, we will study a toy model of a a helix using a bead spring model in lammps. This model has lowest energy when it is in a left or right handed helix, but it is not chiral, so they must have the same free energy.
First, run the system without any pulling, if you fisualize the trajectory, you will see that it is stuck in one configuration (epsilon sets the strength of the bonds, it can be studied at lower or higher stabilities by changing eps).

```
lmp -log helix_sf_example.log \
          -var plumed_file helix_sf_f-4.5_eps7.5_10000000.plumed.dat  \
          -var outprefix helix_sf_pull0 \
          -var steps 10000000  \
          -var eps 7.5 \
          -in run_helix_plumed.lmp
```

Now, run the system using FISST, suitibly modifying the plumed file to impose a force from -2 to 8 (as studied in the paper)

```
lmp -log helix_sf_example.log \
          -var plumed_file helix_fisst_fmin-2.0_fmax8.0_eps7.5_50000000.plumed.dat  \
          -var outprefix helix_fisst_fmin-2.0_fmax8.0_eps7.5_50000000 \
          -var steps 50000000  \
          -var eps 7.5 \
          -in run_helix_plumed.lmp
```

The included script 'compute_handedness.py' has an ad-hoc handedness calculation which is computed using R↩
MSD to a left and right handed helix. The end-end distance is already computed in the colvars file.
Use the handedness calculation in this script to create a 2d FES from the unbiased simulation, and from the unweighted FISST data.
Note that both sides are equally explored in this relatively long simulation. Then compute plots reweighted to small/negative forces and large forces, and notice how the result mirrors that from the V-shaped potential.

11.14.8.4 Solvated alanine-10

We will now study the effect of force on a model peptide, alanine 10 in water. You can set this up to run yourself, but the colvar files are also included for a 200ns simulation of alanine 10 run with FISST from forces -10 to 10 pN and from -10 to 100 pN.
Note, now we are using real units, a conversion factor is needed: 69.4786 pN = 1 kcal/mol/Angstrom
To run yourself, modify ala10_fisst.plumed.dat

```
gmx_mpi -s ala10.tpr -plumed ala10_fisst.plumed.dat -nsteps 100000000 --deffnm OUTPUT_PREFIX
```

Then analyze your files, or my files ala10_pull_fRange_fmin-10_fmax100.∗ to compute the end-end distance of Alanine 10 at different forces. Do your results from different FISST ranges agree?
Finally, use the observable weights to compute the Ramachandran plot (averaged over residues) at different forces (mine is from analyzing the gromacs structural output, but you may want to put the phi-psi calculation in the plumed file as in previous tutorials!)

11.15 PLUMED Masterclass 21.1: PLUMED syntax and analysis

Authors

Max Bonomi

Date

January 18, 2021

11.15.1 Aims

The aim of this Masterclass is to introduce users to the PLUMED syntax and to illustrate how PLUMED can be used to analyze pre-existing molecular dynamics trajectories.

11.15.2 Objectives

Once this Masterclass is completed, users will be able to:

- Write a simple PLUMED input file and use it with the driver utility to analyze a trajectory.
- Use advanced selection tools with MOLINFO.
- Define and use virtual atoms, such as CENTER.
- Deal with discontinuities in the trajectory due to periodic boundary conditions.
- Use RMSD to measure protein conformational changes.
- Align the system to a template with FIT_TO_TEMPLATE.

11.15.3 Setting up the software

11.15.3.1 Installation

We will use `conda` to install all the software needed in this Masterclass:

- `PLUMED` version 2.7.0
- `NumPy`
- `pandas`
- `matplotlib`
- `jupyter`
- `MDTraj`
- `MDAnalysis`
- `Git`

First, make sure conda is installed by typing:

```
conda
```

If the command is not found, please refer to these `instructions` to install conda on your machine. Alternatively, if you use the `Homebrew` package manager, you can install conda with:

```
brew install --cask anaconda
# add this line to your .bashrc
export PATH="/usr/local/anaconda3/bin:$PATH"
```

Now we can create a conda environment for the PLUMED Masterclass:

```
conda create --name plumed-masterclass
```

and activate it with:

```
conda activate plumed-masterclass
```

Finally, we can proceed with the installation of the required software:

```
conda install -c conda-forge plumed py-plumed numpy pandas matplotlib notebook mdtraj mdanalysis git
```

Note

Do not forget to activate the `plumed-masterclass` environment every time you open a new terminal/shell.

11.15.3.2 PLUMED overview

PLUMED is a library that can be incorporated into many molecular dynamics (MD) codes by adding a relatively simple and well documented interface. Once it is incorporated you can use PLUMED to perform on-the-fly a variety of different analyses and to bias the sampling in MD simulations. Additionally, PLUMED can be used as a standalone code for analyzing trajectories. If you want to use the code in this way, you can run the PLUMED executable by issuing the command:

```
plumed <instructions>
```

Let's start by getting a feel for the range of calculations that PLUMED can do. Issue the following command now:

```
plumed --help
```

The output of this command is the list of Command Line Tools included in PLUMED. Among these, there are commands that allow you to patch an MD code, postprocess metadynamics simulations, and build the manual. In this class we will use PLUMED to analyze trajectories. In order to do so, we will learn how to use the driver command line tool. Let's look at the options of PLUMED driver by issuing the following command:

```
plumed driver --help
```

As you can see we can do a number of things with driver. For all of these options, however, we are going to need to write a PLUMED input file.

11.15.4 Resources

The data needed to complete the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-21-1.git
```

The MD trajectory that we will analyze can be found in the folder called `data`:

- `5-HT1B.pdb`: reference conformation of the 5-HT1B receptor with the serotonin ligand.
- `5-HT1B.xtc`: trajectory of the 5-HT1B receptor with the serotonin ligand.

A Jupyter notebook to be used as template for the analysis of the PLUMED output can be found in the folder called `notebooks`. In this directory, you will also find the complete solution to all exercises in the notebook `solution.ipynb`.
Please note that:

- This is a simulation of a membrane receptor, but water, lipids, and ions have been stripped out of the trajectory.
- This is the raw trajectory generated by GROMACS. Therefore it is discontinuous due to periodic boundaries conditions (PBCs).
- The students are invited to solve the exercises by themselves after completing the PLUMED input file templates provided below. In case of problems, students can rely on the solution notebook provided in the GitHub repository.

To keep things clean, it is recommended to run each exercise in a separate sub-directory (i.e. Exercise-1, Exercise-2, ...), which you can create inside the root directory `masterclass-21-1`.

Note

All the exercises have been tested with PLUMED version 2.7.0.

11.15.5 A sample PLUMED input file

The main goal of PLUMED is to compute collective variables (or CVs), which are complex descriptors of the system that can be used to describe the conformational change of a protein or a chemical reaction. This can be done either *on-the-fly* during a molecular dynamics simulations or *a posteriori* on a pre-calculated trajectory using PLUMED as a post-processing tool. In both cases, you should create an input file with a specific PLUMED syntax. Have a look at the sample input file below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aa-masterclass-21-1.txt
# Compute distance between atoms 1 and 10.
# Atoms are ordered as in the trajectory files and their numbering starts from 1.
# The distance is called "d" for future reference.
d: DISTANCE ATOMS=1,10

# Compute the torsional angle between atoms 1, 10, 20, and 30.
# The angle is called "phi1" for future reference.
phi1: TORSION ATOMS=1,10,20,30

# The same CV defined above can be split into multiple lines
# The angle is called "phi2" for future reference.
TORSION ...
LABEL=phi2
ATOMS=1,10,20,30
...

# Print "d" on a file named "COLVAR1" every 10 steps.
PRINT ARG=d FILE=COLVAR1 STRIDE=10

# Print "phi1" and "phi2" on another file named "COLVAR2" every 100 steps.
PRINT ARG=phi1,phi2 FILE=COLVAR2 STRIDE=100
```

In the input file above, each line defines a so-called action. In this simple example, actions are used to compute a distance, a dihedral angle, or print some values on a file. Each action supports a number of keywords, whose value is specified. Action names are highlighted in green and, by clicking on them, you can go to the corresponding page in the manual that contains a detailed description of each keyword. Actions that support the keyword `STRIDE` are those that determine how frequently things are done. Notice that the default value for `STRIDE` is always 1. In the example above, omitting `STRIDE` keywords the corresponding `COLVAR` files would have been written for every frame of the analyzed trajectory. All the other actions in the example above, i.e. DISTANCE and TORSION, do not support the `STRIDE` keyword and are only calculated when requested. That is, `d` will be computed every 10 frames, and `phi1` and `phi2` every 100 frames.
Variables should be given a name (in the example above, `d`, `phi1`, and `phi2`), which is then used to refer to these variables in subsequent actions, such as the PRINT command. A lists of atoms should be provided as comma separated numbers, with no space.
You can find more information on the PLUMED syntax at Getting Started page of the manual. The complete documentation for all the supported collective variables can be found at the Collective Variables page.

11.15.5.1 The PLUMED internal units

By default the PLUMED inputs and outputs quantities in the following units:

- Length: nanometers
- Energy: kJ/mol
- Time: picoseconds
- Mass: amu
- Charge: e

If you want to change these units, you can do this using the UNITS keyword.

11.15.6 Exercises

11.15.6.1 Exercise 1: Computing and printing simple collective variables

In this exercise, we will learn how to compute and print collective variables on a pre-calculated MD trajectory. To analyze the trajectory provided here, we will:

- create a PLUMED input file with a text editor (typically called `plumed.dat`);
- run the PLUMED driver utility;
- visualize the output with the aid of a Jupyter notebook.

Notice that you can also visualize trajectories with VMD (always a good idea!). For example, the trajectory `5-HT1B.xtc` can be visualized with the command:

```
vmd 5-HT1B.pdb 5-HT1B.xtc
```

When you try this, you will notice that this trajectory is discontinuous due to PBCs. We need to keep this in mind in our analysis.
Let's now prepare a PLUMED input file to calculate:

- the gyration radius of the CA protein atoms (GYRATION) of the first 40 N-terminal residues;
- the distance (DISTANCE) between CA atoms of residues 1 and 40.

The first 40 residues of the 5-HT1B receptor correspond to an extracellular flexible loop of which we want to characterize the dynamics during our MD simulation. Below you can find a sample `plumed.dat` file that you can use as a template. Whenever you see an highlightedFILL string, be careful because this is a string that you must replace. To retrieve the atom indexes that you need to include in the input file, you can have a look at `5-HT1B.pdb`. The atoms indexes are contained in the second column. Keep in mind that numbering scheme in PLUMED starts from 1.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-1.txt
# Compute gyration radius on CA atoms of the first 40 N-terminal residues:
r: GYRATION ATOMS=__FILL__

# Compute distance between CA atoms of residues 1 and 40
d: DISTANCE ATOMS=__FILL__

# Print the two collective variables on COLVAR file every step
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

Once your `plumed.dat` file is complete, you can run the PLUMED driver as follows:

```
plumed driver --plumed plumed.dat --mf_xtc 5-HT1B.xtc
```

Scroll in your terminal to read the PLUMED log. As you can see, PLUMED gives a lot of feedback about the input that it is reading and the actions that it will execute. Please take your time to inspect the log file and check if PLUMED is actually doing what you intend to do.
The command above will create a `COLVAR` file. The first lines should be identical to these ones:

```
#! FIELDS time r d
 0.000000 1.271069 2.782998
 1.000000 1.263125 2.388697
 2.000000 1.348965 2.606062
 3.000000 1.291011 2.204363
 4.000000 1.280714 2.411836
 5.000000 1.257692 2.334839
```

Notice that the first line informs you about the content of each column.
In case you obtain different numbers, check your input, you might have made some mistakes!
This `COLVAR` file can be analyzed using the Jupyter notebook `plumed-pandas.ipynb` provided in the folder `notebooks`. You can use the following command to open the notebook:

```
jupyter notebook plumed-pandas.ipynb
```

This notebook allows you to import the `COLVAR` file produced by PLUMED and to generate the desired figures using the `matplotlib` library:

```
# import python modules
import plumed
import matplotlib.pyplot as plt
# import COLVAR file as pandas dataset
# set the right path to the COLVAR file
data=plumed.read_as_pandas("../Exercise-1/COLVAR")
# print pandas dataset
data
# plot time serie of gyration radius (r) and distance (d)
plt.plot(data.time,data.r, label="gyration radius")
plt.plot(data.time,data.d, label="distance")
# x-y axis labels
plt.xlabel("MD frame")
plt.ylabel("r/d [nm]")
plt.legend()
```

```
# plot gyration radius vs distance
plt.plot(data.r,data.d, 'o')
# x-y axis labels
plt.xlabel("gyration radius [nm]")
plt.ylabel("distance [nm]")
```

What can you deduce about the dynamics of this region of the 5-HT1B receptors? Are the two CVs both providing useful information or are they quite correlated? To answer to the latter question, you can inspect the plot of one CV against the other.

11.15.6.2 Exercise 2: Mastering advanced selection tools

PLUMED provides some shortcuts to select atoms with specific properties. To use this feature, you should specify the MOLINFO action along with a reference PDB file. This command is used to provide information on the molecules that are present in your system.
Let's try to use this functionality to calculate the backbone dihedral angle ϕ (phi) of residue 2 of the 5-HT1B receptor. This CV is defined by the action TORSION and a set of 4 atoms. For residue `i`, the dihedral ϕ is defined by these atoms: C(i-1),N(i),CA(i),C(i) (see Fig. masterclass-21-1-dih-fig).
After consulting the manual and inspecting `5-HT1B.pdb`, let's define the dihedral angle ϕ of residue 2 in two different ways:

1. specifying an explicit list of 4 atoms (`t1`).
2. using the MOLINFO shortcut to select quadruplets for dihedral angles (`t2`);

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-1.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__

# Define the dihedral phi of residue 2 as an explicit list of 4 atoms
t1: TORSION ATOMS=__FILL__
# Define the same dihedral using MOLINFO shortcuts
t2: TORSION ATOMS=__FILL__

# Print the two collective variables on COLVAR file every step
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

After completing the PLUMED input file above, let's use it to analyze the trajectory `5-HT1B.xtc` using the driver tool:

```
plumed driver --plumed plumed.dat --mf_xtc 5-HT1B.xtc
```

You can modify the Jupyter notebook used in Exercise 1: Computing and printing simple collective variables to visualize the trajectory of the two CVs calculated with the PLUMED input file above and written in the `COLVAR` file. If you executed this exercise correctly, these two trajectories should be identical.
As a second example of MOLINFO capabilities, we will use the advanced atom selection tools provided by the `MDAnalysis` and `MDTraj` libraries. Let's redo Exercise 1: Computing and printing simple collective variables, this time using MOLINFO shortcuts to select CA atoms. You need to complete the following template PLUMED input file using the appropriate selection syntax for the corresponding library used. Please consult the `MDAnalysis` and `MDTraj` documentations if you are not familiar with these libraries and their syntax.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-1.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__

# To use MDAnalysis selection tools:
r1: GYRATION ATOMS={@mda:{resid __FILL__ and name __FILL__}}
d1: DISTANCE ATOMS={@mda:{resid __FILL__ and name __FILL__}}

# To use MDTraj selection tools:
r2: GYRATION ATOMS={@mdt:{resid __FILL__ and name __FILL__}}
d2: DISTANCE ATOMS={@mdt:{resid __FILL__ and name __FILL__}}

# Print all the collective variables on COLVAR file every step
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

Now, you can compare the `COLVAR` file obtained with the one of Exercise 1: Computing and printing simple collective variables↩ : they should be identical!

11.15.6.3 Exercise 3: Using virtual atoms

Sometimes, when calculating a CV, you may not want to use the positions of a number of atoms directly. Instead you may want to define a virtual atom whose position is generated based on the positions of a collection of other atoms. For example you might want to use the center of mass (COM) or the geometric center (CENTER) of a group of atoms.
In this exercise, you will learn how to specify virtual atoms and later use them to define a CV. The objective is to calculate the distances between the geometric center of the serotonin ligand (indicated as residue `LIG` in `5-HT1B.pdb`) and the geometric centers of the two glycans located at position N24 and N32. Glycans are carbohydrate-based polymers that are sometimes linked to certain protein aminoacids. If you examine `5-HT1B.pdb`, you will find the two glycans defined after the end of the protein, i.e. after residue SER-390. These two glycans have different length:

- the glycan attached at position N24 ranges from residue BGLC-1 to AFUC-9
- the glycan attached at position N32 ranges from residue BGLC-1 to AFUC-10

Let's complete the PLUMED input file below. You can use the advanced selection tools learned in Exercise 2: Mastering advanced selection tools to specify the atoms belonging to the ligand and to the two glycans:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-1.txt
# Geometric center of the ligand
lig: CENTER ATOMS=__FILL__
# Geometric center of the first glycan
g1: CENTER ATOMS=__FILL__
# Geometric center of the second glycan
g2: CENTER ATOMS=__FILL__

# Distance between ligand and first glycan
d1: DISTANCE ATOMS=__FILL__
# Distance between ligand and second glycan
d2: DISTANCE ATOMS=__FILL__

# Print the two distances on COLVAR file every step
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

Once you have prepared a PLUMED input file containing the above instructions, you can execute it on the trajectory `5-HT1B.xtc` by making use of the following command:

```
plumed driver --mf_xtc 5-HT1B.xtc --plumed plumed.dat
```

Let's now analyze the output of the calculation by plotting the time series of the two CVs. Can you say if the ligand is overall staying closer to the first or second glycan?

11.15.6.4 Exercise 4: Fixing PBCs discontinuities

As mentioned above, `5-HT1B.xtc` is the raw trajectory generated by the GROMACS MD code. Therefore, it typically presents discontinuities due to PBCs. Many of the CVs used so far, such as CENTER or DISTANCE, take care of these discontinuities automatically. However, other CVs need a special command, called WHOLEMOLECULES, to fix PBCs discontinuities before the calculation of the CV. In this exercise, you will learn how to use this action.
We have seen that the first 40 N-terminal residues of the 5-HT1B receptor are quite flexibile. In this exercise, we want to estimate the secondary structure content (alpha-helix and beta-sheet) of this fragment during the course of the MD simulations. In order to do so, we can use the following 3 CVs:

- ALPHARMSD to measure the alpha-helical content of a protein structure.
- PARABETARMSD to measure the parallel beta-sheet content.
- ANTIBETARMSD to measure the antiparallel beta-sheet content.

Let's first try to complete the following PLUMED input file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-1.txt
# Activate MOLINFO functionalities
MOLINFO __FILL__
# make the first 40 N-terminal residues whole
```

```
WHOLEMOLECULES __FILL__
# alpha-helix content of residues 1-40
h: ALPHARMSD __FILL__ TYPE=OPTIMAL
# parallel beta-sheet content of residues 1-40
pb: PARABETARMSD __FILL__ TYPE=OPTIMAL
# antiparallel beta sheet-content of residues 1-40
ab: ANTIBETARMSD __FILL__ TYPE=OPTIMAL
# now we create a new CV that sums parallel and antiparallel beta-sheet contents
b: COMBINE __FILL__ PERIODIC=NO
# Print the alpha-helical content and the *total* beta-sheet content on COLVAR file every step
PRINT __FILL__
```

and use it to analyze the trajectory `5-HT1B.xtc`. Can you say if the first 40 N-terminal residues tend to populate more alpha-helical or beta-sheet conformations?

11.15.6.5 Exercise 5: Using RMSD to measure conformational changes

As previously mentioned, the first 40 N-terminal residues of the 5-HT1B receptor are quite flexible, while the rest of the protein remains more stable during the course of the simulation. In this exercise, you will learn how to use RMSD to measure deviations from a reference structure. To use this CV, you need to keep in mind that you must specify in the PLUMED input a PDB file in which you mark the atoms that you want to use to:

- optimally align a conformation to the reference;
- calculate the displacement from the reference conformation after optimal alignment.

Keep in mind that these two sets of atoms might be different! In fact, the objective of this exercise is to calculate:

- the RMSD of the backbone atoms of the first 40 N-terminal residues after aligning the system on the backbone atoms of residues 41 to 390;
- the RMSD of the backbone atoms of residues 41 to 390 after aligning the system on the same set of atoms.

To create the two PDB files needed to define the two RMSD CVs, you can start from the provided PDB file `5-HT1B.pdb`. Please consult the manual at the RMSD page, in order to:

- create the PLUMED input file to calculate the two CVs defined above (use `TYPE=OPTIMAL`);
- learn how to mark atoms for alignment and displacement in the PDB files;
- check whether PBCs are automatically taken care of or you need to use the WHOLEMOLECULES action.

After analyzing `5-HT1B.xtc` with the driver tool, can you say which part of the receptor is more flexible and deviates more from the starting conformation during the course of the simulation?

11.15.6.6 Exercise 6: Aligning conformations to a template

In this exercise, we will learn how to align a MD trajectory to a reference conformation, after fixing possible discontinuities due to PBCs. The goal is to compute the vertical position of the serotonin ligand with respect to the lipid bilayer. In the simulations of membrane proteins, typically the initial conformation is oriented so that the lipid bilayer is parallel to the xy plane (look for example at `5-HT1B.pdb`). Therefore, initially one could use for example the coordinate `z` of the geometric center of the ligand to measure how far it is from the membrane bilayer. However, during the simulation:

- the system can translate from its original position;
- the system can be broken by PBCs;

therefore one could not use an absolute position to keep track of the location of the ligand. To solve this problem, there are several PLUMED actions that can be used to make sure the system is not broken by PBCs, to re-align it to a reference conformation, and thus to use absolute positions safely.
To complete this exercise, the users will need to make heavy use of the PLUMED `manual` to prepare the input file on their own. In the following, some suggestions will be given:

- first, make sure the entire protein is not broken by PBCs using WHOLEMOLECULES;

- then, make sure the ligand is not broken by PBCs and in the same cell as the protein, using the WRAPAROUND action and the `GROUPBY` option;
- to align the *stable* protein residues (as defined in Exercise 5: Using RMSD to measure conformational changes, i.e. residues 41 to 390) to the template `5-HT1B.pdb`, you can use FIT_TO_TEMPLATE;
- at this point, you can safely define the position of the geometric center of the ligand using the POSITION CV with the option `NOPBC`;
- the requested CV is the `z` component of the POSITION CV.

You can check that your PLUMED input file is correct in two ways. First, you can print out the conformations of the system after the transformations done by WHOLEMOLECULES, WRAPAROUND, and FIT_TO_TEMPLATE by modifying your input file as follows:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-1.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__
# Here put your PLUMED input file
#
#
# Write coordinates of all the atoms to file after PLUMED transformations
DUMPATOMS FILE=5-HT1B_aligned.gro ATOMS=__FILL__
```

Now, you can visualize the `5-HT1B_aligned.gro` file using VMD. Second, if you inspect the time series of your CV, this should be a continuous trajectory that spans the range from 9 nm to 18 nm.

11.15.6.7 Exercise 7: Estimating binding propensity

In this last exercise, we want to determine the propensity of the serotonin ligand to bind the first 40 N-terminal flexible residues of the 5-HT1B receptor and if there are hot-spots where binding is more favorite. In order to answer to these questions, the user will:

- compute the fraction of bound conformations over the total number of frames in the MD trajectory;
- for each individual residue and glycan, compute the fraction of bound conformation per residue/glycan;
- dump all bound conformations to a `gro` file, after fixing PBCs as in Exercise 6: Aligning conformations to a template;
- for each individual residue and glycan, dump all bound conformations to a separate `gro` file, after fixing PBCs.

To solve this exercise, no template PLUMED input file nor any indication of the procedure to follow will be given. The users should only keep in mind that:

- we arbitrarily define as *bound* a conformation in which at least one pair of atoms of the ligand and of the protein/residue/glycan is closer than 0.4 nm;
- any pre-exisiting CV defined in the PLUMED `manual` can be used;
- any CV defined directly by the user in the PLUMED input file via the CUSTOM action can be used.

11.16 PLUMED Masterclass 21.2: Statistical errors in MD

11.16.1 Aims

In this Masterclass, we will discuss how to report the results from molecular simulations.
We will emphasize that any result that we get from any simulation is a random variable. To make the result reproducible, we must characterize the distribution that has been sampled. It is not sufficient to report the averages.

11.16.2 Objectives

Once you have completed this Masterclass you will be able to:

- Use PLUMED to calculate time averages and histograms from biased and unbiased simulation data.
- Use PLUMED to perform block averaging.
- Calculate error bars on-time averages computed form biased and unbiased simulation data using the central limit theorem and non-parametric bootstrap.

11.16.3 Setting up PLUMED

If you have not yet set up PLUMED, you can find information about installing it in the section Setting up the software of PLUMED Masterclass 21.1: PLUMED syntax and analysis. Please ensure that you have setup PLUMED on your machine before starting the exercises.

11.16.4 Resources

The data needed to execute the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-21-2.git
```

The data you need is in the folder called `data`. You will find the following files in that folder:

- `uncorrelated_data`: you will analyze this data set in the first six exercises that follow
- `correlated_data`: you will analyze this data set in exercise 7
- `weighted_data`: you will analyze this data in exercise 8

In exercise 9, you will pull everything together by generating a metadynamics trajectory. This exercise draws together all the ideas from exercises 1-8 by getting you to run a metadynamics simulation and extract the free energy surface. In the `data` folder, you will thus also find the following files, which are the input for the metadynamics simulation:

- in: The input file for simplemd that contains the parameters for the MD simulation.
- input.xyz: An initial configuration for the cluster that we are studying in this tutorial.

Notice that PLUMED input files have not been provided in the GitHub repository. You must prepare these input files yourself using the templates below.
We would recommend that you run each exercise in separate sub-directories inside the root directory `masterclass-21-2`.

Note

All the exercises were tested with PLUMED version 2.7.0.

11.16.5 Background

PLUMED Masterclass 21.1: PLUMED syntax and analysis showed you how PLUMED can be used to calculate the value of a collective variables, $s(\mathbf{x})$, from the positions of the atoms, $\mathbf{x}$. You should also have seen that doing so allows you to describe the conformational changes or chemical reactions that have occured during your molecular dynamics trajectory. We will build on this idea in this tutorial by recalling that the values for collective variable we calculate for the frames of a constant-temperature molecular dynamics trajectory are samples from the probability distribution for the canonical (NVT) ensemble:

$$P(s') = \frac{\int \mathrm{d}x\mathrm{d}p\delta(s(x) - s')e^{-\frac{H(x,p)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}}$$

where k_B is Boltzmann's constant, T is the temperature $H(x,p)$ is the Hamiltonian and δ is a Dirac delta function. Notice also that the integrals in the numerator and denominator are integrals over all of phase space. If N

is the number of atoms in our system we must, therefore, integrate over each of the $3N$ momentum (p) and $3N$ position (x) coordinates when calculating these integrals.

It is only possible to calculate the integrals in the quotient above exactly for elementary physical systems. For more complex systems we thus assume that we can extract information on $P(s')$ by sampling from this distribution multiple times (using molecular dynamics or Monte Carlo) and using the tools of statistics. Critically, however, any result we get from such simulations is a random variable. **To make our results reproducible we thus need to characterize the distribution sampled in our simulation.**
Reporting only the average value we get is not sufficient as any averages we take are random.
Links to background information on statistical mechanics and statistics are provided throughout this tutorial. You can complete the tutorial without looking at the information at these links. However, we hope that the information we have provided through these links will prove useful if you want to do a more in-depth study of the topic.

11.16.6 Exercises

11.16.6.1 Exercise 1: Calculating the average value of a CV

We will start our study of averaging by estimating the ensemble average of the CV. The ensemble average for $s(x)$ is given by:

$$\langle s \rangle = \frac{\int \mathrm{d}x\mathrm{d}p s(x) e^{-\frac{H(x,p)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}}$$

This equation is the same quotient that was introduced for $P(s')$ in Background but with the Dirac delta replaced by $s(x)$.
In statistics, the quantity known as the ensemble average in statistical mechanics is referred to as the expectation of the random variable's distribution. The expectation of a random variable is often esimated by taking multiple identical samples from the distribution, X_i and computing a sample mean as follows:

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

We can do the same with the data from our MD trajectory. We replace the X_i in the equation above with the CV values calculated for each of our trajectory frames.
To calculate averages using PLUMED, you can use the input file below. This input calculates averages for the data in the `uncorrelated_data` file you downloaded when you collected the GitHub repository.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
data: READ FILE=__FILL__ VALUES=__FILL__
av: AVERAGE ARG=__FILL__ STRIDE=1
PRINT ARG=av FILE=colvar
```

Copy the input to a file called plumed.dat, fill in the blanks and run the calculation by executing the following command:

```
> plumed driver --noatoms
```

The `colvar` file that is output by PLUMED contains the average computed value from progressively larger and larger numbers of CV values. You should thus be able to use the data in `colvar` to produce a graph that shows the average as a function of the number of variables it is computed from as shown below.
The fluctuations in the average get smaller as this quantity is computed from larger numbers of random variables. We say that the average thus converges to the ensemble average, which is zero for the graph above.

11.16.6.2 Exercise 2: Calculating the free energy

We can estimate the distribution for our CV, $P(s')$ (see Background), by calculating a histogram. The histogram we obtain is a sample from a multinomial distribution so we can estimate parameters for the multinomial by using likelihood maximisation. Once we have the marginal distribution, $P(s')$ we can then calculate the free energy, $F(s')$ as a function of $s(x)$ as $F(s')$ is related to $P(s')$ (see Background) by:

$$F(s') = -k_B T \ln P(s')$$

If we estimate $P(s')$ using likelihood maximisation we can thus get an estimate of the free energy surface. To estimate the free energy surface for the data in `uncorrelated_data` using PLUMED in this way we can use the input file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
# We use natural units here so that kBT is set to 1
UNITS NATURAL
data: READ FILE=__FILL__ VALUES=__FILL__
hhh: HISTOGRAM ARG=__FILL__ STRIDE=1 __FILL__=-4.5 __FILL__=4.5 __FILL__=100 __FILL__=DISCRETE
fes: CONVERT_TO_FES GRID=__FILL__ TEMP=1  # This sets k_B T = 1
DUMPGRID GRID=__FILL__ FILE=fes.dat
```

Copy this input to a file called plumed.dat, fill in the blanks so that you have a grid that runs from -4 to +4 with 100 bins. Run the calculation by executing the following command:

```
> plumed driver --noatoms
```

The `--noatoms` flag here is needed since `plumed driver` is commonly used to analize a trajectory of atomic coordinates, but here we are using it to directly analyze a collective variable.
You should see that the free energy curve for this data set looks like this:
As you can see, there is a single minimum in this free energy surface.

11.16.6.3 Exercise 3: Calculating the fluctuations for a CV

Physical systems spend the majority of their time fluctuating around minima in the free energy landscape. Let's suppose that this minima is at μ and lets use the Taylor series to write an expression for the free energy at s' as follows:

$$F(s) = F(\mu) + F'(\mu)(s-\mu) + \frac{F''(\mu)(s-\mu)^2}{2!} + \ldots + \frac{F^{(n)}(\mu)(s-\mu)^n}{n!} + \ldots$$

In this expression $F'(\mu)$, $F''(\mu)$ and $F^{(n)}(\mu)$ are the first, second and nth derivatives of the free energy at μ. We know there is a minimum at μ so $F'(\mu) = 0$. If we truncate the expansion at second order we can, therefore, write:

$$F(s) \approx F(\mu) + \frac{F''(\mu)(s-\mu)^2}{2}$$

We now recall that $F(s) = -k_B T \ln P(s')$ and thus write:

$$P(s) = \exp\left(-\frac{F(s)}{k_B T}\right) \approx \exp\left(-\frac{F(\mu) + \frac{F''(\mu)(s-\mu)^2}{2}}{k_B T}\right) = \exp\left(-\frac{F(\mu)}{k_B T}\right) \exp\left(-\frac{F''(\mu)(s-\mu)^2}{2k_B T}\right)$$

The first term in the final product here is a constant that does not depend on s, while the second is a Gaussian centered on μ with $\sigma^2 = \frac{k_B T}{F''(\mu)}$.
We can assume that the constant term in the product above normalizes the distribution. The derivation above, therefore, suggests that our CV values are all samples from a `normal distribution`. We thus no longer need to estimate the histogram to get information on $P(s')$. If $P(s')$ is indeed a normal distribution, it is fully characterized if we have the two parameters μ and σ.
Statistics tells us that if we have N identical normal random variables, X_i we can estimate μ and σ using:

$$\mu = \frac{1}{N}\sum X_i \qquad \sigma = \sqrt{\frac{N}{N-1}\left[\frac{1}{N}\sum_{i=1}^{N} X_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} X_i\right)^2\right]}$$

We learned how to estimate μ using these expressions in Exercise 1: Calculating the average value of a CV. To estimate σ^2 for the data in `uncorrelated_data` using PLUMED and the expression above we can use the following input file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
data: READ FILE=__FILL__ VALUES=__FILL__
# This line should calculate the square of the quantity read in from the file above
d2: CUSTOM ARG=__FILL__ FUNC=__FILL__ PERIODIC=NO
# Calculate the average from the read-in data
av: AVERAGE ARG=__FILL__ STRIDE=1
# Calculate the average of the squares of the read in data
av2: AVERAGE ARG=__FILL__ STRIDE=1
# Evaluate the variance using the expression above
var: CUSTOM ARG=__FILL__ FUNC=y-x*x PERIODIC=NO
# Print the variance
PRINT ARG=__FILL__ FILE=colvar
```

Copy this input to a file called plumed.dat, fill in the blanks and run the calculation by executing the following command:

```
> plumed driver --noatoms
```

Once you have run this calculation, you should be able to draw a graph showing how the estimate of σ changes as a function of the number of CVs used in its calculation. As you can see from the graph below, σ quantity behaves similarly to the mean that we studied in Exercise 1: Calculating the average value of a CV
Truncation the Taylor series of the free energy at second order as we have done in this section is equivalent to assuming that a Harmonic Oscillator can be used to describe the fluctuations along our CV. The partition function, ensemble average and distribution for such systems can be calculated exactly, and there is no need for simulation. Even when the system is not harmonic, calculating the quantity we have called σ^2 in this section is still useful as this quantity is an estimator for the variance of the distribution. For anharmonic systems, there is not a simple closed-form expression between the variance (σ^2) and the second derivative of the free energy at μ though.

11.16.6.4 Exercise 4: Calculating block averages

The following PLUMED input splits the CV values into blocks and calculates an average from each block of data separately. We can thus use it to get information on the distribution that is being sampled when we calculate an average from sets of 500 random variables using the ideas discussed in Exercise 1: Calculating the average value of a CV.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
data: READ FILE=__FILL__ VALUES=__FILL__
av: AVERAGE ARG=__FILL__ STRIDE=1 CLEAR=500
PRINT ARG=__FILL__ STRIDE=__FILL__ FILE=colvar
```

Copy this input to a file called plumed.dat, fill in the blanks and run the calculation by executing the following command:

```
> plumed driver --noatoms
```

Plot a graph showing the original data from `uncorrelated_data` and the averages in the `colvar` file. You should be able to see something similar to this graph.
Notice how the distribution for both the black (original data) and yellow points (averages) in this graph are centred on the same quantity. Both of these quantities are thus accurate estimators for the expectation of the distribution. However, the block average that is shown in yellow is a more precise estimator for this quantity.

The reason the block average is a more precise estimator is connected to a well known result in statistics. If we compute a mean as follows:

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

where the X_i are all independent and identical random variables it is straightforward to show that the expectation and variance of this random quantity are given by:

$$\mathbb{E}(\overline{X}) = \mathbb{E}(X) \qquad \text{and} \qquad \mathrm{var}(\overline{X}) = \frac{\mathrm{var}(X)}{N}$$

where $\mathbb{E}(X)$ and $\mathrm{var}(X)$ are the expectation and variance of the random variable X_i from which the mean was computed.

11.16.6.5 Exercise 5: Free energy from block averages

We can use the block averaging method introduced in Exercise 4: Calculating block averages to calculate error bars on the estimates of free energy. To generate 10 histograms from the data in `uncorrelated_data` with 100 bins starting at -4 and finishing at +4 using PLUMED we can use the input file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
data: READ FILE=__FILL__ VALUES=__FILL__
hhh: HISTOGRAM ARG=__FILL__STRIDE=1 __FILL__=-4.5 __FILL__=4.5 __FILL__=100 CLEAR=__FILL__ _FILL__=DISCRETE
DUMPGRID GRID=__FILL__ FILE=hist.dat STRIDE=1000
```

Copy this input to a file called plumed.dat, fill in the blanks and run the calculation by executing the following command:

```
> plumed driver --noatoms
```

Running this command should generate several files containing histograms that will be called: analysis.0.hist.↩ dat, analysis.1.hist.dat etc. These files contain the histograms constructed from each of the blocks of data in your trajectory. You can merge them all to get the final free energy surface, which can be calculated using the well-known relation between the histogram, $P(s)$, and the free energy surface, $F(s)$, by using the following python script:

```
import matplotlib.pyplot as plt
import numpy as np
import glob
hist1 = np.loadtxt("../Exercises/Exercise_5/hist.dat")
N, average, average2 = 1, hist1[:,1], hist1[:,1]*hist1[:,1]
for filen in glob.glob( "../Exercises/Exercise_5/analysis.*.hist.dat") :
    histn = np.loadtxt(filen)
    N, average, average2 = N + 1, average + histn[:,1], average2 + histn[:,1]*histn[:,1]

# Final averages
average = average / N
# Final variances
var = (N/(N-1))*( average2 / N - average*average )
# Errors
error = np.sqrt( var / N )
# Convert to free energy
fes = -np.log( average )
# Convert to error in fes
ferr = error / average
# And draw graph of free energy surface
plt.fill_between( hist1[:,0], fes-ferr, fes+ferr )
plt.xlabel("CV value")
plt.ylabel('Free energy')
plt.show()
```

Copy this script to a cell in a python notebook and then run it on your data.
You may need to adjust the names of the files that are being read to suit your machine's setup. The graph shown in the figure below shows the free energy surface generated from the python script.

The value of the free energy in the ith bin is calculated using:

$$F_i = -k_B T \ln \left(\frac{1}{N} \sum_{j=1}^{N} H_i^{(j)} \right)$$

The sum here runs over the N histograms and $H_i^{(j)}$ is the value of ith bin in the jth estimate of the histogram. The above expression is thus calculating the logarithm of the histogram's average value in bin i.
The error on the free energy, which is illustrated using the shaded region in the figure above, is calculated using:

$$\sigma_{F_i} = \frac{k_B T}{\frac{1}{N}\sum_{j=1}^{N} H_i^{(j)}} \sqrt{\frac{1}{N-1}\left[\frac{1}{N}\sum_{j=1}^{N} (H_i^{(j)})^2 - \left(\frac{1}{N}\sum_{j=1}^{N} H_i^{(j)} \right)^2 \right]}$$

The term in the square root here is the error on the average value of the histogram in bin i. The error on this `average` can be calculated using the formulas in Exercise 4: Calculating block averages. To get the error on the free energy, we then have to propagate the errors through the expression:

$$F(s) = -k_B T \ln \left(P(s) \right)$$

to get the expression above.

Lastly, notice that it is often useful to average the error over the grid using:

$$\sigma = \frac{1}{M}\sum_{i=1}^{M} F_i$$

where the sum runs over the M bins in the histogram.

11.16.6.6 Exercise 6: Calculating bootstrap averages

Exercise 4: Calculating block averages demonstrated one way of sampling the mean's distribution (see Exercise 1: Calculating the average value of a CV). We assumed that the M estimates of the mean were all normal random variables in the previous section but we did not need to do that. We could instead have used a non-parametric method. When using such methods we have to generate more data, which we can either do by running longer simulations, which is expensive, or by bootstrapping, which is cheaper.

We can demonstrate how bootstrapping works in practice by using the following script, which works with the first 500 points in `uncorrelated_data`. As you can see, we first calculate the average from all the data points. We then take new means by repeatedly sampling sets of 500 points with replacement from the data and calculating new means.

```
import numpy as np
ddd = np.loadtxt("../data/uncorrelated_data")
data = ddd[0:500,1]
bootstraps = np.zeros(200)
for i in range(200) :
    av = 0
    for j in range(500) : av = av + data[np.random.randint(0,500)]
    bootstraps[i] = av / 500
f = open("bootstraps", "w")
f.write("#! FIELDS time boot \n")
for i in range(0,200):
    f.write(str(i) + " "  + str(bootstraps[i) + "\n" )
f.close()
```

When you run the script above, it generates a file called `bootstraps` containing the averages that have been calculated by bootstrapping.
If you now calculate the variance from all your bootstrapped averages you should see that it close to the value you got from the expression below which was introduced in Exercise 4: Calculating block averages:

$$\mathrm{var}(\overline{X}) = \frac{\mathrm{var}(X)}{N}$$

To use this expression you can insert the value of $\mathrm{var}(X)$ you computed in Exercise 3: Calculating the fluctuations for a CV with $N = 500$.

You can also use bootstrapping to estimate the errors in the free energy surface. As an additional exercise, you can try to do this form of analysis on the data in `uncorrelated_data` You should get a result that is similar to the result you got in Exercise 5: Free energy from block averages

11.16.6.7 Exercise 7: Dealing with correlated data

In this exercise, you will review everything you have done in the previous two sections. **You should**:

- Calculate block averages for the data in `correlated_data`. Calculate the error on the average of the block average.

- Calculate bootstrap averages for the data in `correlated_data`. Calculate the error from the bootstrap averages.

You will see that the variance you obtain from the bootstrap averages is less than the variance you get by block averaging.
These variances are different because there are correlations between the data points in `correlated_data`. Such correlations were not present in the data set you have examined in all the exercises that appeared previously to this one. The reason this matters is that the expression:

$$\mathrm{var}(\overline{X}) = \frac{\mathrm{var}(X)}{N}$$

is only valid if the random variables that $\overline{X}$ is computed from are both identical and **independent**. This expression is thus not valid for correlated data. To be clear, however, we can still write:

$$\mathbb{E}(\overline{X}) = \mathbb{E}(X)$$

as this expression holds as long as the random variables from which $\overline{X}$ are identical. When we use this expression, the random variables only need to be independent and can be correlated.

Any data we get by computing CVs from a molecular dynamics trajectory is almost certain to contain correlations. It is thus essential to know how to handle correlated data. The block averaging technique that was introduced in Exercise 4: Calculating block averages resolves this problem. You can show that if the blocks are long enough, the averages you obtain are uncorrelated.

For the remainder of this exercise, you should use the data in `correlated_data` and what you have learned in the previous exercises to calculate block averages for different block sizes.

For each block size

- Estimate the mean and variance for your N block averages (the X_i) using $\mu = \frac{1}{N}\sum X_i$ and $\sigma^2 = \frac{N}{N-1}\left[\frac{1}{N}\sum_{i=1}^{N} X_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} X_i\right)^2\right]$
- Insert your estimate of the variance into the following expression for the error bar on your estimate for μ: $\epsilon = \sqrt{\frac{\sigma^2}{N}}$

You should be able to use your data to draw a graph showing the value of the average and the associated error barm ϵ, as a function of the size of the blocks similar to the one shown below:

This graph shows that, when the data is correlated, the error bar is underestimated if each block average is computed from a small number of data points. When sufficient data points are used to calculate each block average, however, the error bar settles on a constant value that is independent of the block size. When this has happened you can be confident that your block averages are no longer correlated and the expression $\mathrm{var}(\overline{X}) = \frac{\mathrm{var}(X)}{N}$ is thus valid.

Notice that you can also calculate the error bar using bootstrapping by selecting samples from your block averages. If you have time, you should try this. You should try to confirm that this method gives similar estimates for the error.

11.16.6.8 Exercise 8: Weighted averages

PLUMED is routinely used to run simulations using methods such as umbrella sampling and metadynamics. In these methods, a bias potential, $V(x)$, is added to the Hamiltonian to enhance the sampling of phase space. CVs calculated from the frames of such biased MD simulations thus samples from the following distribution:

$$P(s') = \frac{\int \mathrm{d}x\mathrm{d}p\delta(s(x) - s')e^{-\frac{H(x,p)}{k_B T}}e^{-\frac{V(x)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}e^{-\frac{V(x)}{k_B T}}}$$

To get information on the unbiased distribution:

$$P(s') = \frac{\int \mathrm{d}x\mathrm{d}p\delta(s(x) - s')e^{-\frac{H(x,p)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}}$$

from such simulations we thus have to calculate weighted averages using:

$$\overline{X}_w = \frac{\sum_{i=1}^{N} w_i X_i}{\sum_{i=1}^{N} w_i}$$

where the w_i are (random) weights that counteract the effect of the bias. The way the weights are calculated is described in PLUMED Masterclass 21.3: Umbrella sampling)

In this exercise we are going to reweight the data in `weighted_data`, which consists of samples from the following distribution:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

with $\mu = 0.6$ and $\sigma = 0.5$ with and additional constrant that $0 < x < 1$. Given this information the following weighted average:

$$\overline{X}_w = \frac{\sum_{i=1}^N w_i X_i}{\sum_{i=1}^N w_i} \qquad \text{where} \qquad w_i = \frac{1}{P(X_i)}$$

should be an estimator for the expectation of a uniform random variable between 0 and 1. Furthermore, the expression below:

$$\sigma_X^2 = \frac{V_1}{V_1 - 1} \sum_{i=1}^N \frac{w_i}{V_1} (x_i - \overline{X}_w)^2 \qquad \text{where} \qquad V_1 = \sum_{i=1}^N w_i$$

gives an estimate for the variance of **the distribution** after reweighting (i.e. the variance of the uniform random variable).
Notice, finally, that the tools of statistics gives us expressions for the expectation and variance of the **weighted average**:

$$\mathbb{E}(\overline{X}_w) = \overline{X} \qquad \text{and} \qquad \text{var}(\overline{X}_w) = \frac{\sum_{i=1}^N w_i^2 (X_i - \overline{X}_w)^2}{(\sum_{i=1}^N w_i)^2}$$

These expressions hold if all the X_i from which the weighted average is computed are independent and identically distributed random variables. Notice, furthermore, how the expression above for the the variance of the weighted average **is not** a function of the variance for the variable as was the case for the unweighted averages in Exercise 4: Calculating block averages.
In what follows we are thus going to try to extract the average and the fluctuations for the CV in the unbiased (in this case uniform) distribution as well as the unbiased free energy. We will calculate these quantities by computing weighted averages and weighted histograms from our simulation data. For the histogram we are also going to extract error bars by reweighting. To calculate these quantities using PLUMED we will use an input like this:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
UNITS NATURAL # This ensures that Boltzmann's constant is one
data: READ FILE=__FILL__ VALUES=__FILL__
# This restraint and the REWEIGHT_BIAS command after computes the weights in the formulas above.
mm: RESTRAINT ARG=data AT=0.6 KAPPA=4
rw: REWEIGHT_BIAS TEMP=1
wav: AVERAGE ARG=__FILL__ STRIDE=1 LOGWEIGHTS=__FILL__
# These lines compute the variance of the random variable
dd: CUSTOM ARG=data,wav FUNC=(x-y)*(x-y) PERIODIC=NO
uvar: AVERAGE ARG=dd STRIDE=1 LOGWEIGHTS=rw NORMALIZATION=false
one: CONSTANT VALUE=1
wsum: AVERAGE ARG=one STRIDE=1 LOGWEIGHTS=rw NORMALIZATION=false
var: CUSTOM ARG=uvar,wsum FUNC=x/(y-1) PERIODIC=NO
# Print out the average and variance of the uniform random variable
PRINT ARG=__FILL__ STRIDE=1 FILE=colvar
# Construct the histogram
hhh: HISTOGRAM ARG=__FILL__ LOGWEIGHTS=__FILL__ __FILL__=0 __FILL__=1 __FILL__=20 CLEAR=__FILL__ NORMALIZATION
DUMPGRID GRID=__FILL__ FILE=hist.dat STRIDE=1000
```

Copy this input to a file called plumed.dat, fill in the blanks and run the calculation by executing the following command:

```
> plumed driver --noatoms
```

I obtained the following graphs showing how the mean and variance change with sample size.

The free energy with errors from this data looks like this:
It is up to you to work out how to adapt the script given in Exercise 5: Free energy from block averages, so it works here. The script above works for unweighted means. Here, however, you have to calculate weighted means using the formulas above.

11.16.6.9 Exercise 9: The free energy from a biased simulation

We can now bring all this together by running a metadynamics simulation and extracting the free energy surface by reweighting. The other exercises in this tutorial have shown us that when we do this, it is essential to:

- Quote error bars on the estimates of the free energy we obtain as the values we get from our simulation will be random variables. Quoting error bars is thus essential in terms of making our results reproducible.
- Calculate weighted averages using the ideas discussed in Exercise 8: Weighted averages as we need to account for the effect the bias has on the sampling of phase space.
- Use the block averaging method discussed in Exercise 4: Calculating block averages to obtain multiple estimates for the free energy. Notice that we need to use block averaging because, as discussed in Exercise 7: Dealing with correlated data, there are correlations between the CV values the system visits during the trajectory.

As these three issues have been the focus of this tutorial, we focus on them in the exercise that follows. The theory behind the metadynamics method that we are using is discussed in detail in PLUMED Masterclass 21.4: Metadynamics if you are interested.

We will study a system of 7 Lennard Jones atoms in two dimensions in what follows using the MD code **simplemd** that is part of PLUMED. You can run this code by issuing the command:

```
plumed simplemd < in
```

where **in** here is the input file from the GitHub repository for this tutorial. This input file instructs PLUMED to perform 200000 steps of MD at a temperature of $k_B T = 0.1\epsilon$ starting from the configuration in input.xyz.
The timestep in this simulation is 0.005 $\sqrt{\epsilon m \sigma^2}$ and the temperature is kept fixed using a Langevin thermostat with a relaxation time of $0.1\sqrt{\epsilon m \sigma^2}$. Trajectory frames are output every 1000 MD steps to a file called trajectory.xyz. Notice also that to run the calculation above you need to provide a PLUMED input file called plumed.dat.

We want to investigate transitions between the four structures of Lennard Jones 7 that are shown below using metadynamics.
However, when we run the metadynamics, we will often find that the cluster evaporates and the seven atoms separate. To prevent this, we will thus add restraints to prevent the cluster from evaporating. The particular restraint we are going to use will prevent all the atoms from moving more than 2σ from the centre of mass of the cluster. As the masses of all the atoms in the cluster are the same, we can compute the position of the centre of mass using:

$$\mathbf{x}_{\rm com} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$$

where $\mathbf{x}_i$ is the position of the atom with the index i. The distance between the atom with index i and the position of this centre of mass, d_i, can be computed using Pythagoras' theorem. These distances are then restrained by using the following potential:

$$V(d_i) = \{ 1\ 00 * (d_i - 2.0)^2 \text{if} \quad d_i > 20 \text{otherwise}$$

as you can see, this potential does not affect the dynamics when these distances are less than 2 ϵ. If an atom is more than 2 ϵ from the centre of mass, however, this potential will drive it back towards the centre of mass.
A metadynamics bias will be used to force the system to move between the four configurations shown in masterclass-21-1-4-lj7-minima. This bias will act on the second and third central moments of the distribution of coordination numbers. The nth central moment of a set of numbers, $\{X_i\}$ can be calculated using:

$$\mu^n = \frac{1}{N}\sum_{i=1}^{N}(X_i - \langle X \rangle)^n \qquad \text{where} \qquad \langle X \rangle = \frac{1}{N}\sum_{i=1}^{N} X_i$$

Furthermore, we can compute the coordination number of our Lennard Jones atoms using:

$$c_i = \sum_{i \neq j} \frac{1 - \left(\frac{r_{ij}}{1.5}\right)^8}{1 - \left(\frac{r_{ij}}{1.5}\right)^{16}}$$

where r_{ij} is the distance between atom i and atom j. With all this information in mind the following cell contains a skeleton input file for PLUMED that gets it to perform metadynamics using the second and third central moments of the distribution of coordination numbers as a CV.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
# this optional command tells VIM that this is a PLUMED file and to colour the text accordingly
# vim: ft=plumed

# This tells PLUMED we are using Lennard Jones units
UNITS NATURAL

# Calculate the position of the centre of mass.  We can then refer to this position later in the input using t
com: COM __FILL__

# Add the restraint on the distance between com and the first atom
d1: DISTANCE __FILL__
UPPER_WALLS ARG=d1 __FILL__

# Add the restraint on the distance between com and the second atom
d2: DISTANCE __FILL__
UPPER_WALLS  __FILL__

# Add the restraint on the distance between com and the third atom
d3: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fourth atom
d4: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fifth atom
d5: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the sixth atom
d6: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the seventh atom
d7: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Calculate the collective variables
c1: COORDINATIONNUMBER SPECIES=__FILL__ MOMENTS=__FILL__ SWITCH={RATIONAL __FILL__ }

# Do metadynamics
METAD ARG=__FILL__ HEIGHT=__FILL__ PACE=__FILL__ SIGMA=__FILL__ GRID_MIN=-1.5,-1.5 GRID_MAX=2.5,2.5 GRID_BIN=5
```

Copy this input to file called plumed.dat and modify it so that it instructs PLUMED to add Gaussian kernels with a bandwidth of 0.1 in both the second and third moment of the distribution of coordination numbers and a height of 0.05 ϵ every 500 MD steps. You can then use this input together with the input.xyz, and in files, you obtained from the GitHub repository to generate a metadynamics trajectory at $k_BT = 0.1\epsilon$ by running the command:

```
plumed simplemd < in
```

Once you have run the metadynamics calculations, you can post-process the output trajectory using **driver** to extract the free energy by `reweighting`. Notice that to do block averaging, you will need to extract multiple estimates for the (weighted) histogram. You should thus use the following input file to extract estimates of the histogram:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-2.txt
# this optional command tells VIM that this is a PLUMED file and to colour the text accordingly
# vim: ft=plumed

UNITS NATURAL

# We can delete the parts of the input that specified the walls and disregard these in our analysis
# It is OK to do this as we are only interested in the value of the free energy in parts of phase space
# where the bias due to these walls is not acting.

c1: COORDINATIONNUMBER SPECIES=__FILL__ MOMENTS=__FILL__ SWITCH={RATIONAL __FILL__}

# The metadynamics bias is restarted here so we consider the final bias as a static bias in our calculations
METAD ARG=__FILL__ HEIGHT=0.05 PACE=50000000 SIGMA=0.1,0.1 GRID_MIN=-1.5,-1.5 GRID_MAX=2.5,2.5 GRID_BIN=500,50
```

```
# This adjusts the weights of the sampled configurations and thereby accounts for the effect of the bias poten
rw: REWEIGHT_BIAS TEMP=0.1

# Calculate the histogram and output it to a file
hh: HISTOGRAM ARG=c1.* GRID_MIN=-1.5,-1.5 GRID_MAX=2.5,2.5 GRID_BIN=200,200 BANDWIDTH=0.02,0.02 LOGWEIGHTS=__F
DUMPGRID GRID=hh FILE=my_histogram.dat STRIDE=2500
```

Once you have filled in the blanks in this input, you can then run the calculation by using the command:

```
> plumed driver --ixyz trajectory.xyz  --initial-step 1
```

You must make sure that the HILLS file that was output by your metadynamics simulation is available in the directory where you run the above command.
For the rest of the exercise, you are on your own. You must use what you have learned in the other parts of the tutorial to generate plots similar to those shown below. The leftmost plot here is the free energy surface computed by taking the (weighted) average of the blocks, the right panel shows the size of the error bar on the free energy at each point of the free energy surface.

Notice that the data is correlated here so you should investigate how the error size depends on the lengths of the blocks as was discussed in Exercise 7: Dealing with correlated data. When I did this analysis I found that the error does not have a strong dependence on the size of the blocks.
Finally, if you are struggling to plot the 2D free energy surface, you can generate free energy as a function of one CV only using the ideas from earlier exercises.
Hint: You are now calculating weighted averages so you will need to use the code you wrote for Exercise 8: Weighted averages to merge the histograms

11.16.7 Further reading

If you want to know more about good practise using PLUMED you can read https://arxiv.org/abs/1812.08213. We would also recommend learning about kernel density estimation, which will often give you smoother histograms. You can start learning about kernel density estimation by reading https://en.wikipedia.org/wiki/Kernel_density_estimation. My full sets of notes are available here:

- Probability theory notes
- Thermodynamics notes
- Statistical Mechanics notes

11.17 PLUMED Masterclass 21.3: Umbrella sampling

Authors

Giovanni Bussi

Date

February 15, 2021

11.17.1 Aims

In this Masterclass, we will discuss how to perform and analyze umbrella sampling simulations. We will learn how to introduce a bias potential with PLUMED, how to compute free energy landscapes, and how to reweight the resulting ensembles. We will also understand how to compute statistical errors on the computed quantities.

11.17.2 Objectives

Once you have completed this Masterclass you will be able to:

- Use PLUMED to run simulations using static bias potentials with different functional forms.

- Use WHAM to combine multiple simulations performed with different bias potentials.
- Reweight the resulting ensembles so as to obtain the free-energy profile as a function of a different variable.
- Calculate error bars on free energies and populations.

11.17.3 Setting up PLUMED

If you have not yet set up PLUMED, you can find information about installing it in the section Setting up the software of PLUMED Masterclass 21.1: PLUMED syntax and analysis.
Once you have installed PLUMED, you will need to install GROMACS as well. In particular, you will need a special version of GROMACS that has been patched with PLUMED. You can obtain it using conda with the following command

```
conda install --strict-channel-priority -c plumed/label/masterclass -c conda-forge gromacs
```

The `--strict-channel-priority` might be necessary in case your conda install is configured to download packages from the `bioconda` channel. Indeed, `bioconda` contains a version of GROMACS that is **not** patched with PLUMED and would thus not work here.
On Linux, the command above should install the following packages:

```
  gromacs            plumed/label/masterclass/linux-64::gromacs-2019.6-h3fd9d12_0
  libclang           conda-forge/linux-64::libclang-11.0.1-default_ha53f305_1
  libevent           conda-forge/linux-64::libevent-2.1.10-hcdb4288_3
  libhwloc           conda-forge/linux-64::libhwloc-1.11.13-h3c4fd83_0
  libllvm11          conda-forge/linux-64::libllvm11-11.0.1-hf817b99_0
  libpq              conda-forge/linux-64::libpq-12.3-h255efa7_3
  [ etc ... ]
```

The exact versions might be different. Notice however that GROMACS comes from the `plumed/label/masterclass` channel, whereas the required libraries come from the `conda-forge` channel. To be sure the installed GROMACS is patched with PLUMED, try the following shell command:

```
gmx mdrun -h 2> /dev/null | grep -q plumed && echo ok
```

It should print `ok`.
Please ensure that you have setup PLUMED and GROMACS on your machine before starting the exercises. Also notice that in order to obtain good performances it is better to compile GROMACS from source on the machine you are running your simulations. You can find out in the PLUMED documention how to patch GROMACS with PLUMED so as to be able to install it from source. For this tutorial, the conda precompiled binaries will be sufficient.

11.17.4 Resources

The data needed to execute the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-21-3.git
```

The data you need is in the folder called `data`. You will find the following files in that folder:

- `topolA.tpr`: an input file that can be used to run a GROMACS simulation of alanine dipeptide starting from one of the two main free-energy minima
- `topolB.tpr`: same as `topolA.tpr`, but starting from the other minimum.
- `wham.py`: a python script that can be used to perform binless WHAM analysis

Notice that PLUMED input files have not been provided in the GitHub repository. You must prepare these input files yourself using the templates below.
We would recommend that you run each exercise in separate sub-directories inside the root directory `masterclass-21-3`.

Note

All the exercises were tested with PLUMED version 2.7.0 and GROMACS 2019.6

11.17.5 Exercises

Throughout this tutorial we will run simulations of alanine dipeptide in vacuum using GROMACS and PLUMED. Whereas this system is too simple to be considered a proper benchmark for enhanced sampling methods, it is complex enough to be used in learning them. Some of the commands below are specific for GROMACS, but all the PLUMED input files are compatible with other MD engines as well.

11.17.5.1 Exercise 1: Running an unbiased simulation with PLUMED

In PLUMED Masterclass 21.1: PLUMED syntax and analysis we learned how to analyze trajectories a posteriori. One of the nice features of PLUMED is that the very same analysis can be done on the fly. In other words, you might compute your collective variables while GROMACS is running instead of waiting for the simulation to end. This can could be convenient if you want to run a large number of simulations, you already know what to compute, and you do not want to use too much disk space.
To run a simulation with GROMACS you have to type this command in the shell:

```
gmx mdrun -plumed plumed.dat -s topolA.tpr -nsteps 200000 -x traj_unbiased.xtc
```

Notice that the file `topolA.tpr` contains all the relevant information (simulation parameters, initial conditions, etc.). In this tutorial we will just need to play with the number of steps (200000 in the example above) and we will tune the name of the trajectory saved by GROMACS (traj_unbiased.xtc in the example above).
Also consider that GROMACS and PLUMED are not going to delete your files but are taking backups when you try to overwrite them. GROMCAS backup files start with `#`, whereas PLUMED backup files start with `bck.`. Both GROMACS and PLUMED will complain if you try to create too many backups of the same file. It is thus recommended to regularly clean your directory with a command such as `rm -f \#* bck.*`
The command above will only succeed if a file named `plumed.dat` exists in the current directory. We know already how to create such file from PLUMED Masterclass 21.1: PLUMED syntax and analysis. In PLUMED Masterclass 21.2: Statistical errors in MD we also learned how to compute histograms. You should be thus able to complete the template below and put it in a file named `plumed.dat`.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-3.txt
# vim:ft=plumed
MOLINFO STRUCTURE=reference.pdb
phi: TORSION ATOMS=__FILL__ # use MOLINFO shortcuts to identify the phi angle of the second residue
psi: TORSION ATOMS=__FILL__ # use MOLINFO shortcuts to identify the psi angle of the second residue

# use the command below to compute the histogram of phi
# we use a smooth kernel to produce a nicer graph here
# notice that when asking for numbers PLUMED is happy to accept strings such as "pi" meaning 3.14...
# also arithmetics is allowed (e.g. 2*pi could be used if necessary)
hhphi: HISTOGRAM ARG=__FILL__ STRIDE=100 GRID_MIN=-pi GRID_MAX=__FILL__ GRID_BIN=600 BANDWIDTH=0.05
ffphi: CONVERT_TO_FES GRID=hhphi # no need to set TEMP here, PLUMED will obtain it from GROMACS
DUMPGRID GRID=__FILL__ FILE=fes_phi.dat STRIDE=200000 # stride is needed here since PLUMED does not know when

# now add three more lines to compute and dump the free energy as a function of **psi** on a file names fes_ps
__FILL__

PRINT __FILL__ # use this command to write phi and psi on a file named colvar.dat, every 100 steps
```

You can then monitor what happened during the simulation using the python script above.

```
import plumed
import matplotlib.pyplot as plt
import numpy as np
# plot the time series of phi and psi
colvar=plumed.read_as_pandas("colvar.dat")
plt.plot(colvar.time,colvar.phi,"x",label="phi")
plt.plot(colvar.time,colvar.psi,"x",label="psi")
plt.xlabel("time")
plt.ylabel("$\phi$")
plt.legend()
plt.show()
# scatter plot with phi and psi
plt.plot(colvar.phi,colvar.psi,"x")
plt.xlabel("$\phi$")
plt.ylabel("$\psi$")
plt.xlim((-np.pi,np.pi))
plt.ylim((-np.pi,np.pi))
plt.show()
# FES as a function of phi
# we remove infinite and nans here
fes_phi=plumed.read_as_pandas("fes_phi").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_phi.phi,fes_phi.ffphi)
plt.xlim((-np.pi,np.pi))
```

```
plt.xlabel("$\phi$")
plt.ylabel("$F(\phi)$")
plt.show()
# FES as a function of psi
# we remove infinite and nans here
fes_psi=plumed.read_as_pandas("fes_psi").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_psi.psi,fes_psi.ffpsi)
plt.xlim((-np.pi,np.pi))
plt.xlabel("$\psi$")
plt.ylabel("$F(\psi)$")
plt.show()
```

In this manner you will be able to see (a) the time series of ϕ and ψ, (b) which portion of the (ϕ, ψ) space has been explored (c) the free energy as a function of ϕ and (d) the free energy as a function of ψ. In particular, you should be able to identify two minima separated by a moderate barrier, and several transitions between these minima will be visible in the simulated time scale.

11.17.5.2 Exercise 2: Running biased simulations with PLUMED

We are now ready to use PLUMED to perform the task it was originally designed for: biasing a simulation on the fly. We will first try to add simple bias potentials that change the balance between the two minima we have obtained in the previous exercise. Not particularly useful here since both minima were sampled anyways. However, it is instructive since we will be able to test the same type of analysis we did in PLUMED Masterclass 21.2: Statistical errors in MD. The two minima observed in the previous exercise are located at ϕ approximately equal to -2.5 and -1.5, separated by a barrier located at ϕ approximately equal to -2. We can predict that adding a bias potential in the form `-A*sin(x+2)`, with A positive and large enough, should favor the minimum at -1.5 at the expense of the other. We can try with A=10 using an input file like the following one:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-3.txt
# vim:ft=plumed
__FILL__ # compute phi and psi here, as in the previous input file

# fill in with the required function (i.e. -10*sin(phi+2)):
f: CUSTOM ARG=phi FUNC=__FILL__ PERIODIC=NO

# this command allows to add a bias potential equal to f
BIASVALUE ARG=f

# here you can paste the same HISTOGRAM/CONVERT_TO_FES/DUMPGRID commands that you used in
# the previous exercise. let's just write the results on different files
hhphi: __FILL__
ffphi: __FILL__
DUMPGRID FILE=fes_phi_biased1.dat __FILL__
hhpsi: __FILL__
ffpsi: __FILL__
DUMPGRID FILE=fes_psi_biased1.dat __FILL__

lw: REWEIGHT_BIAS

# and here we do the same again, but this time using LOGWEIGHTS.
# these free energies will be printed on files fes_phi_biased1r.dat and
# fes_psi_biased1r.dat and will be reweighted so as to be unbiased
hhphir: __FILL__ LOGWEIGHTS=lw
ffphir: __FILL__
DUMPGRID FILE=fes_phi_biased1r.dat __FILL__

hhpsir: __FILL__ LOGWEIGHTS=lw
ffpsir: __FILL__
DUMPGRID FILE=fes_psi_biased1r.dat __FILL__

PRINT __FILL__ # monitor what's happening, as before, writing on file plumed_colvar1.dat
```

Call this file `plumed_biased1.dat` and run the simulation using this command:

```
gmx mdrun -plumed plumed_biased1.dat -s topolA.tpr -nsteps 200000 -x traj_comp_biased1.xtc
```

Notice that we are storing the trajectory on a separate file. We will need both the unbiased and the biased trajectory later.

You can then monitor what happened during the simulation using these commands

```
colvar=plumed.read_as_pandas("colvar_biased1.dat")
plt.plot(colvar.time,colvar.phi,"x",label="phi")
plt.plot(colvar.time,colvar.psi,"x",label="psi")
plt.xlabel("time")
plt.ylabel("$\phi$")
plt.legend()
plt.show()
```

```
plt.plot(colvar.phi,colvar.psi,"x")
plt.xlabel("$\phi$")
plt.ylabel("$\psi$")
plt.xlim((-np.pi,np.pi))
plt.ylim((-np.pi,np.pi))
plt.show()
fes_phi=plumed.read_as_pandas("fes_phi.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_phi.phi,fes_phi.ffphi,label="original")
fes_phib=plumed.read_as_pandas("fes_phi_biased1.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_phib.phi,fes_phib.ffphi,label="biased")
fes_phir=plumed.read_as_pandas("fes_phi_biased1r.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_phir.phi,fes_phir.ffphir,label="reweighted")
plt.legend()
plt.xlim((-np.pi,np.pi))
plt.xlabel("$\phi$")
plt.ylabel("$F(\phi)$")
plt.show()
fes_psi=plumed.read_as_pandas("fes_psi.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_psi.psi,fes_psi.ffpsi,label="original")
fes_psib=plumed.read_as_pandas("fes_psi_biased1.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_psib.psi,fes_psib.ffpsi,label="biased")
fes_psir=plumed.read_as_pandas("fes_psi_biased1r.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_psir.psi,fes_psir.ffpsir,label="reweighted")
plt.legend()
plt.xlim((-np.pi,np.pi))
plt.xlabel("$\psi$")
plt.ylabel("$F(\psi)$")
plt.show()
```

The free-energy plots here will include three lines: (a) the original one (as obtained from the previous exercise), (b) the one sampled in this second simulation, where the minimum at higher phi appears more stable, and (c) the reweighted free-energy from the second simulation, that should look closer to the original one. Notice that the relationship between plots (b) and (c) is straightforward when the analyzed variable is phi: the third line could have been obtained by just adding $-10\sin(x+2)$ to the second line. However, when analyzing psi the effect is less obvious.

11.17.5.3 Exercise 3: Combining statistics from biased and unbiased simulations

We will now run a new simulation that is even more biased, and precisely choosing the prefactor A=20. This will favor even more the minimum at higher values of phi. Please prepare a `plumed_biased2.dat` input file that is identical to `plumed_biased1.dat` except for:

- it applies a bias -20∗sin(phi+2)
- all the written files are named with a `2` suffix (e.g. `fes_phi_biased2.dat`).

The run the new simulation with the following command:

```
gmx mdrun -plumed plumed_biased2.dat -s topolA.tpr -nsteps 200000 -x traj_comp_biased2.xtc
```

You can once more analyze this simulation as you did for the previous one. You should notice that the peak at higher ϕ is even more favored now.
We will now use WHAM to combine the three simulations we have done so far, namely: (a) unbiased (`traj_comp_unbiased.xtc`), (b) biased (`traj_comp_biased1.xtc`), and (c) more biased (`traj_comp_biased2.xtc`).
The first thing we have to do is to concatenate the three files:

```
gmx trjcat -cat -f traj_comp_unbiased.xtc traj_comp_biased1.xtc traj_comp_biased2.xtc -o traj_comp_cat.xtc
```

Notice that this new trajectory contains all the simulated frames, and we **do not** explicitly tracking from which of the simulation each frame originated. We then have to compute the bias that would have been felt in each of the three runs above by each of the frames in the concatenated trajectory. We could do this reusing the three plumed input files we used above, but it is perhaps clearer to do it with a single input file in this case. You can use an input like this one:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-3.txt
# vim:ft=plumed
MOLINFO STRUCTURE=reference.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2

# here we list the bias potentials that we used in the three simulations we want to combine
b0: CUSTOM ARG=phi FUNC=0.0 PERIODIC=NO
b1: CUSTOM ARG=phi FUNC=__FILL__ PERIODIC=NO # fill here with the bias you used in the first biased simulation
b2: CUSTOM ARG=phi FUNC=__FILL__ PERIODIC=NO # fill here with the bias you used in the second biased simulatio
PRINT ARG=phi,psi,b0,b1,b2 FILE=biases.dat
```

If you call it `plumed_wham.dat` you can then use the following command

```
plumed driver --plumed plumed_wham.dat --ixtc traj_comp_cat.xtc
```

We are now ready to use binless WHAM to compute the weights associated to the concatenated trajectory. This can be done with the following script

```
import wham
# this is to check that you actually imported the module located in this directory
# it should print /current/directory/wham.py
print(wham.__file__)
bias=plumed.read_as_pandas("biases.dat")
kBT=300*8.314462618*0.001 # use kJ/mol here
w=wham.wham(np.stack((bias.b0,bias.b1,bias.b2)).T,T=kBT)
print(w)
# you can then add a new column to the bias dataframe:
print(bias)
bias["logweights"]=w["logW"]
print(bias)
# and write it on disk
plumed.write_pandas(bias,"bias_wham.dat")
```

With the procedure above we computed the weights that can be associated to the concatenated trajectory so as to recover an unbiased distribution. Notice that, although we did not explicitly enforced it, the weights only depends on the value of ϕ. This is because only ϕ was biased here. Thus, if we plot the weight as a function of ϕ all the points will collapse to a line. This will not happen if we plot the weights as a function of ψ:

```
plt.plot(bias.phi,bias.logweights,"x")
plt.show()
plt.plot(bias.psi,bias.logweights,"x")
plt.show()
```

Also notice the form of the weights as a function of ϕ. This graph is telling us that points at larger ϕ are weighted less. This makes sense, since in two simulations out of three we were biasing the ensemble so as to visit those points **more** frequently than in the Boltzmann ensemble. However, the curve is not exactly a sin function. This precise curve can only be obtained by solving self-consistently the WHAM equations.
We can then read the obtained weights and use them in an analysis similar to the one we have done above

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-3.txt
# vim:ft=plumed
phi: READ FILE=bias_wham.dat VALUES=__FILL__
psi: READ FILE=bias_wham.dat VALUES=__FILL__
lw: READ FILE=bias_wham.dat VALUES=__FILL__

hhphi: HISTOGRAM ARG=phi GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05
ffphi: CONVERT_TO_FES GRID=hhphi
DUMPGRID GRID=ffphi FILE=fes_phi_cat.dat

hhpsi: HISTOGRAM ARG=psi GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05
ffpsi: CONVERT_TO_FES GRID=hhpsi
DUMPGRID GRID=ffpsi FILE=fes_psi_cat.dat

# we use a smooth kernel to produce a nicer graph here
hhphir: HISTOGRAM ARG=phi GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05 LOGWEIGHTS=lw
ffphir: CONVERT_TO_FES GRID=hhphir
DUMPGRID GRID=ffphir FILE=fes_phi_catr.dat

hhpsir: HISTOGRAM ARG=psi GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05 LOGWEIGHTS=lw
ffpsir: CONVERT_TO_FES GRID=hhpsir
DUMPGRID GRID=ffpsir FILE=fes_psi_catr.dat
```

Use the template above to produce a file names `plumed_wham.dat` and run the following command:

```
plumed driver --noatoms --plumed plumed_wham.dat --kt 2.4943387854
```

You can now show the result with the following script

```
colvar=plumed.read_as_pandas("bias_wham.dat")
plt.plot(colvar.time,colvar.phi,"x",label="phi")
plt.plot(colvar.time,colvar.psi,"x",label="psi")
plt.xlabel("time")
plt.ylabel("$\phi$")
plt.legend()
plt.show()
plt.plot(colvar.phi,colvar.psi,"x")
plt.xlabel("$\phi$")
plt.ylabel("$\psi$")
plt.xlim((-np.pi,np.pi))
plt.ylim((-np.pi,np.pi))
plt.show()
fes_phi=plumed.read_as_pandas("fes_phi.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_phi.phi,fes_phi.ffphi,label="original")
```

```
fes_phib=plumed.read_as_pandas("fes_phi_cat.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_phib.phi,fes_phib.ffphi,label="biased")
fes_phir=plumed.read_as_pandas("fes_phi_catr.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_phir.phi,fes_phir.ffphir,label="reweighted")
plt.legend()
plt.xlim((-np.pi,np.pi))
plt.xlabel("$\phi$")
plt.ylabel("$F(\phi)$")
plt.show()
fes_psi=plumed.read_as_pandas("fes_psi.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_psi.psi,fes_psi.ffpsi,label="original")
fes_psib=plumed.read_as_pandas("fes_psi_cat.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_psib.psi,fes_psib.ffpsi,label="biased")
fes_psir=plumed.read_as_pandas("fes_psi_catr.dat").replace([np.inf, -np.inf], np.nan).dropna()
plt.plot(fes_psir.psi,fes_psir.ffpsir,label="reweighted")
plt.legend()
plt.xlim((-np.pi,np.pi))
plt.xlabel("$\psi$")
plt.ylabel("$F(\psi)$")
plt.show()
```

When you look at the first graph (i.e., the concatenated time series) you should notice that as you proceed towards the end the frames are coming from the most biased simulation, and thus will have a systematically larger value of ϕ. The reweighted ensemble will be in the end quite similar to the original one.

11.17.5.4 Exercise 4: Enhancing conformational transitions with multiple-windows umbrella sampling

So far we only considered the fast transition between the two minima at negative values of ϕ. However, alanine dipeptide has another metastable state at positive ϕ, that is separated by the minima we have seen by a large barrier. In line of principle, one could use the trick above to discount the effect of any arbitrary biasing potential. For instance, if one had an estimate of a relevant free-energy barrier, a bias potential approximately equal to the negative of the barrier would make the residual free energy basically barrierless. However, this procedure requires knowing the free energy profile in advance. As we will see in the next Masterclass, metadynamics provides exactly a self-healing method that adjusts the bias until dynamics becomes diffusive.

In the context of the static potentials that we are using here, this is a very difficult approach. In the tutorial Lugano tutorial: Using restraints you can find an attempt to derive a potential cancelling the large barrier and leading to transition between the important metastable states. In this class we will take a more pragmatic approach and directly introduce the method that is commonly used in these situations, namely "multiple windows umbrella sampling".

To do so we will have to run a number of separate simulations, each of them with a bias potential designed to maintain the ϕ variable in a narrow range. If the consecutive windows are sufficiently overlapping, and if the windows span the relevant region in the ϕ variable, we will be able to reconstruct the full free-energy profile.

We will use 32 simulations, with `RESTRAINT` potentials centered at uniformly spaced values of ϕ in the range between -pi and pi. In order to avoid the repetition of the first potential, recalling the ϕ is periodic, we can generate the list with the command

```
import numpy as np
at=np.linspace(-np.pi,np.pi,32,endpoint=False)
print(at)
```

Now you should write a python script that generate 32 plumed input files (you can call them `plumed_0.dat`, `plumed_1.dat`, etc) that look like this one

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-3.txt
# vim:ft=plumed
MOLINFO STRUCTURE=reference.pdb
phi: TORSION ATOMS=__FILL__
psi: TORSION ATOMS=__FILL__
bb: RESTRAINT ARG=phi KAPPA=200.0 AT=__FILL__
PRINT ARG=phi,psi,bb.bias FILE=__FILE__ STRIDE=100
# make sure that each simulation writes on a different file
# e.g. colvar_multi_0.dat, colvar_multi_1.dat, ...
```

Now run the 32 simulations with commands like these ones

```
gmx mdrun -plumed plumed_0.dat -s topolA.tpr -nsteps 200000 -x traj_comp_0.xtc
gmx mdrun -plumed plumed_1.dat -s topolA.tpr -nsteps 200000 -x traj_comp_1.xtc
# etc.
```

As we did before, you should now concatenate the resulting trajectories

```
gmx trjcat -cat -f traj_comp_[0-9].xtc traj_comp_[0-9][0-9].xtc -o traj_multi_cat.xtc
```

and analyze them with plumed driver:

```
plumed driver --plumed plumed_0.dat --ixtc traj_multi_cat.xtc --trajectory-stride 100
plumed driver --plumed plumed_1.dat --ixtc traj_multi_cat.xtc --trajectory-stride 100
# etc.
```

Notice the `--trajectory-stride` option: we are telling the driver that the trajectory was saved every 100 frames. In this manner, we will be able to recycle the same plumed input file that we used while running the simulations. This approach is more robust than the one we used above, where a single plumed file was used to reproduce all the biases, and is thus recommended in production cases. Also notice that it is common practice to ignore the initial part of each simulation, since it typically contains a transient that would make the final result less robust. For simplicitly, we are not doing it here. Finally, notice that in Exercise 3: Combining statistics from biased and unbiased simulations we were writing a new input file just for writing the three biases. Here instead we recycled the input files used to run the 32 simulations.
We can now read the produced files and process them with our WHAM script

```
col=[]
for i in range(32):
    col.append(plumed.read_as_pandas("colvar_multi_" + str(i)+".dat"))
# notice that this is the concatenation of 32 trajectories with 2001 frames each
    plt.plot(col[i].phi[2001*i:2001*(i+1)],col[i].psi[2001*i:2001*(i+1)],"x")
plt.xlabel("$\phi$")
plt.ylabel("$\psi$")
plt.show()
# in this graph you can appreciate which region was sampled by each simulation
bias=np.zeros((len(col[0]["bb.bias"]),32))
for i in range(32):
    bias[:,i]=col[i]["bb.bias"][-len(bias):]
w=wham.wham(bias,T=kBT)
plt.plot(w["logW"])
plt.show()
colvar=col[0]
colvar["logweights"]=w["logW"]
plumed.write_pandas(colvar,"bias_multi.dat")
```

Now you can do something similar to what we did before, namely:

- write a file to be read by PLUMED.
- process it with plumed driver so as to compute the free energy with respect to ϕ and psi
- plot the free energy as a function of ϕ and psi

How do the resulting free-energy profiles compares with the one we obtained before?

11.17.5.5 Exercise 5: Computing populations and errors

Now that you have a simulation that sampled all the relevant metastable states, you can analyze it in many different manners. In particular, you can compute the average of any quantity just using the weights that we obtained. This can be done directly with PLUMED, as we have seen in PLUMED Masterclass 21.2: Statistical errors in MD, using the AVERAGE command. However, we will here use Python to compute averages, since this will make it easier to compute their statistical errors with bootstrap.
To complete this exercise you should compute the average value and the statistical error of the following quantities:

- Population of the metastable state B defined as 0$<\phi<$2. Notice that the exact definition is not very important if the minimum is clearly delimited by high free-energy barriers (i.e., low probability regions).
- Free-energy difference between the state B defined as 0$<\phi<$2 and the state A defined as -pi$<\phi<$0. Notice that the free-energy difference is defined as -kBT log PB/PA, where PB is the population of state B and PA the population of state A.
- Average value of ϕ in state A and in state B. Warning: how to compute the average of an angle?

For each of these quantities, you should also compute the standard error. It is here recommended to compute the error using boostrap. You can do so by adjusting the following script

```
# number of blocks
NB=10
# we reshape the bias so that it appears as NB times frames of each traj times number of biases
bb=bias.reshape((32,-1,32))[:,-2000:,:].reshape((32,NB,2000//NB,32))
# we reshape the trajectory so that it appears as NB times frames of each traj times number of biases
cc=np.array(col[0].phi).reshape((32,-1))[:,-2000:].reshape((32,NB,2000//NB))
# we first analyse the complete trajectory:
tr=cc.flatten()
is_in_B=np.int_(np.logical_and(tr>0,tr<2))
w0=wham.wham(bb.reshape((-1,32)),T=kBT)
print("population:",np.average(is_in_B,weights=np.exp(w0["logW"])))
```

```
pop=[]
for i in range(200):
    # we then analyze the bootstrapped trajectories
    print(i)
    c=np.random.choice(NB,NB)
    w=wham.wham(bb[:,c,:,:].reshape((-1,32)),T=kBT)
    tr=cc[:,c,:].flatten()
    is_in_B=np.int_(np.logical_and(tr>0,tr<2))
    pop.append(np.average(is_in_B,weights=np.exp(w["logW"])))
```

11.17.5.6 Exercise 6: Effect of initial conditions

Repeat exercise Exercise 4: Enhancing conformational transitions with multiple-windows umbrella sampling using as an input file `topolB.tpr`. This will make the simulations start from the minimum in B. Plot the free energy as a function of ϕ and as a function of psi. How does it differ from the one obtained in solving Exercise 4: Enhancing conformational transitions with multiple-windows umbrella sampling?

11.18 PLUMED Masterclass 21.4: Metadynamics

Authors

Max Bonomi

Date

March 1, 2021

11.18.1 Aims

The aim of this Masterclass is to train users to perform and analyze a metadynamics simulation with PLUMED.

11.18.2 Objectives

Once this Masterclass is completed, users will be able to:

- Write the PLUMED input file to perform metadynamics simulations.
- Calculate the free energy as a function of the metadynamics collective variables.
- Unbias metadynamics simulations.
- Estimate the error in the reconstructed free energies using block analysis.
- Assess the convergence of metadynamics simulations.
- Recognize good from bad collective variables.

11.18.3 Overview of the theory

PLUMED can be used not only to analyze a pre-existing MD trajectory, but also to add forces on the CVs during a MD simulation, for example, in order to accelerate sampling. To this aim, we have implemented a variety of possible biases acting on CVs. The complete documentation for all the biasing methods available in PLUMED can be found at the Bias page. In the following we will learn how to use PLUMED to perform and analyze a metadynamics simulation. The users are invited to familiarize with the theory behind this method by looking at one of the many reviews available, such as [133] [134] [135] [136]. Below you can find a brief recap of the basic metadynamics theory.
In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t \to \infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t \to \infty) = -\frac{\Delta T}{T + \Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard MD, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T + \Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135] [136].

11.18.4 Setting up the software

The users can refer to the procedure introduce in PLUMED Masterclass 21.1: PLUMED syntax and analysis and PLUMED Masterclass 21.3: Umbrella sampling to install the required software. In this class, we will perform the simulations ourselves, so make sure that the `GROMACS` code is properly installed.

11.18.5 Resources

The data needed to complete the exercises of this Masterclass can be found on `GitHub`. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-21-4.git
```

This repositoy contains 3 folders:

- `data`: GROMACS topology/PDB files for the two systems that we are going to simulate and python script for error analysis;
- `notebooks`: Jupyter notebook to be used as template for the analysis of the PLUMED output;
- `slides`: a brief presentation of the metadynamics theory.

To keep things clean, it is recommended to run each exercise in a separate sub-directory (i.e. Exercise-1, Exercise-2, ...), which you can create inside the root directory `masterclass-21-4`. The students are invited to solve the exercises by themselves after completing the PLUMED input file templates provided below. In case of problems, students can rely on the solution notebook `solution.ipynb` provided in the `GitHub` repository.

In many exercises, we will play with a toy system, alanine dipeptide simulated in vacuo (see Fig. master-ISDD-2-ala-fig). This rather simple molecule is useful to understand data analysis and free-energy methods. This system is a nice example because it presents two metastable states separated by a high free-energy barrier. It is conventional use to characterize the two states in terms of Ramachandran dihedral angles, which are denoted with ϕ (phi) and ψ (psi) in Fig. master-ISDD-2-transition-fig. In the `data/diala` directory of the `GitHub` repository for this Masterclass, you will find two GROMACS `.tpr` files (`topolA.tpr` and `topolB.tpr`), which contain all the necessary information to perform a simulation starting from either one of the two metastable states of alanine dipeptide.

Note

All the exercises have been tested with PLUMED version 2.7.0 and GROMACS 2019.6.

11.18.6 Exercises

11.18.6.1 Exercise 1: Familiarizing with alanine dipeptide

In this brief exercise, we will perform two 20 ns long standard MD simulations of alanine dipeptide starting from the two main metastable states of this system. To keep things clean, the users are invited to run these two simulations in two separate sub-folders. To run these simulations with GROMACS, please use the following commands:

```
# run this command in one directory
gmx mdrun -s topolA.tpr -nsteps 10000000
# and this in another one
gmx mdrun -s topolB.tpr -nsteps 10000000
```

After the simulations are completed, we can use PLUMED to monitor the behavior of the system. As learnt in PLUMED Masterclass 21.1: PLUMED syntax and analysis, PLUMED can compute and print collective variables (CVs) on a pre-calculated MD trajectory. Here, we will:

- create a PLUMED input file with a text editor;
- run the PLUMED driver utility;
- visualize the output with the aid of a Jupyter notebook.

Let's now prepare a PLUMED input file to calculate:

- the value of the backbone dihedral ϕ;
- the value of the backbone dihedral ψ.

by completing the template below (whenever you see an highlightedFILL string, this is a string that you must replace!):

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-4.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__
# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
# you should use MOLINFO shortcuts
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
# here also you should to use MOLINFO shortcuts
psi: TORSION ATOMS=__FILL__
# Print the two collective variables on COLVAR file every step
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

Once your `plumed.dat` file is complete, you can run the PLUMED driver on the two MD trajectories as follows:

```
plumed driver --plumed plumed.dat --mf_xtc traj_comp.xtc
```

The two `COLVAR` files can be analyzed using the Jupyter notebook `plumed-pandas.ipynb` provided in the folder `notebooks`. This notebook allows you to import a `COLVAR` file produced by PLUMED and to generate the desired figures using the `matplotlib` library. The users are invited to:

- inspect the dynamics of the two backbone dihedrals in the two separate simulations;
- calculate the fluctuations (standard deviation) of the two CVs in the different basins visited.

Are both simulations long enough to visit all the relevant conformations or instead they remain trapped in different regions of the ϕ / ψ space?

11.18.6.2 Exercise 2: My first metadynamics simulation

In this exercise we will setup and perform a well-tempered metadynamics run using the backbone dihedral ϕ as collective variable. During the calculation, we will also monitor the behavior of the other backbone dihedral ψ. Here you can find a sample `plumed.dat` file that you can use as a template.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-4.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__
# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
# you should use MOLINFO shortcuts
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
# here also you should to use MOLINFO shortcuts
psi: TORSION ATOMS=__FILL__
# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 500 time steps, with initial height
# equal to 1.2 kJ/mol and bias factor equal to 8
  PACE=500 HEIGHT=1.2 BIASFACTOR=8
# Gaussian width (sigma) should be chosen based on the CV fluctuations in unbiased run
# try 1/2 or 1/3 of the estimated fluctuations
  SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
...
# Print both collective variables on COLVAR file every 10 steps
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

Once your `plumed.dat` file is complete, you can run a 20 ns long metadynamics simulations starting from either of the two provided conformations, for example `topolA.tpr`. All you need to do is execute the following command:

```
gmx mdrun -s topolA.tpr -nsteps 10000000 -plumed plumed.dat
```

During the metadynamics simulation, PLUMED will create two files, named `COLVAR` and `HILLS`. The `COLVAR` file contains all the information specified by the PRINT command, in this case the value of the backbone dihedrals ϕ and ψ every 10 steps of simulation. The `HILLS` file contains a list of the Gaussian kernels deposited along the simulation.
Let's visualize the time series of the two collective variables. Take your time to inspect the behavior of the two CVs. **What are the main differences with respect to the trajectory produced in Exercise 1: Familiarizing with alanine dipeptide ?**
At this point, we can estimate the free energy as a function of the metadynamics CV directly from the metadynamics bias potential. In order to do so, the utility sum_hills can be used to sum the Gaussian kernels deposited during the simulation and stored in the `HILLS` file.
To calculate the free energy as a function of ϕ, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called `fes.dat` in which the free-energy surface as function of ϕ is calculated on a regular grid. One can modify the default name for the free-energy file, as well as the boundaries and bin size of the grid, by using the following sum_hills options:

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

To give a preliminary assessment of the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The sum_hills option `--stride` should be used to give an estimate of the free energy every `N` Gaussian kernels deposited, and the option `--mintozero` can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --stride 200 --mintozero
```

one free energy is calculated every 200 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. Now, you can visualize the free-energy estimate as a function of simulation time and assess how it changed

during the course of the simulation. In the last part of this 20 ns long metadynamics simulation, the free-energy estimate should not change significantly.
Looking at the time-evolution of the entire free-energy profile might not be straightforward. Therefore, what we usually do is focusing on a few metastable states, or local free-energy minima, and calculating their estimated free-energy difference as a function of time. In case of alanine dipeptide, this is rather easy since there are only two major states in the free-energy profile as a function of the CV ϕ.
The users should now:

- calculate from the estimate of the free energy $F(\phi)$ at a given simulation time, the difference in free energy between the two basins. In order to do this, you should define a reasonable interval around the two local free-energy minima and recall that the probability of this state is the integral of the probability $P(\phi) = exp(-F(\phi)/k_BT)$ in the chosen interval;
- plot the estimated free-energy difference as a function of simulation time.

These two observations:

1. the system is diffusing rapidly in the entire CV space
2. the estimated free energy does not significantly change as a function of time

are two indications that the simulation **might** have converged.

Warning

The two conditions listed above are necessary, but not sufficient to declare convergence. We will learn below how to perform a quantitative analysis of the convergence of a metadynamics simulation.

11.18.6.3 Exercise 3: Reweighting (unbiasing) a metadynamics simulation

In the previous exercise we biased ϕ and computed the free energy as a function of the same variable directly from the metadynamics bias potential using the sum_hills utility. However, in many cases you might decide which variable should be analyzed *after* having performed a metadynamics simulation. For example, you might want to calculate the free energy as a function of CVs other than those biased during the metadynamics simulation, such as the dihedral ψ. At variance with standard MD simulations, you cannot simply calculate histograms of other variables directly from your metadynamics trajectory, because the presence of the metadynamics bias potential has altered the statistical weight of each frame. To remove the effect of this bias and thus be able to calculate properties of the system in the unbiased ensemble, you must reweight (unbias) your simulation.
There are multiple ways to calculate the correct statistical weight of each frame in your metadynamics trajectory and thus to reweight your simulation. For example:

1. weights can be calculated by considering the time-dependence of the metadynamics bias potential [3];
2. weights can be calculated using the metadynamics bias potential obtained at the end of the simulation and assuming a constant bias during the entire course of the simulation [50].

In this exercise we will use the second method, which is identical to the umbrella-sampling reweighting approach that you have already used in PLUMED Masterclass 21.2: Statistical errors in MD and PLUMED Masterclass 21.3: Umbrella sampling.
In order to compute the weights we will use the driver tool.
First of all, you need to prepare a `plumed_reweight.dat` file that is identical to the one you used for running your metadynamics simulation except for a few modifications:

- you need to add the keyword `RESTART=YES` to the METAD command. This will make this action behave as if PLUMED was restarting, i.e. PLUMED will read from the `HILLS` file the Gaussians that have previously been accumulated;
- you need to set the Gaussian `HEIGHT` to zero and the `PACE` to a large number. This will actually avoid adding new Gaussians (and even if they are added they will have zero height);
- you need to modify the PRINT statement so that you write every frame and that, in addition to `phi` and `psi`, you also write `metad.bias`;
- you might also want to change the name of the output file to `COLVAR_REWEIGHT`.

Here how the `plumed_reweight.dat` should look like:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-4.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__

__FILL__ # here goes the definitions of the phi and psi CVs

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 10000000 time steps (never!), with initial height equal to 0.0 kJ/mol
  PACE=10000000 HEIGHT=0.0 # <- this is the new stuff!
# The bias factor and Gaussian width are the same as before
  BIASFACTOR=__FILL__ SIGMA=__FILL__
# Gaussians will be read from file and stored on grid
# Make sure you specify the path the HILLS file produced in Exercise 2!
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
# Say that METAD should be restarting (= reading an existing HILLS file)
  RESTART=YES # <- this is the new stuff!
...

# Print out the values of phi, psi and the metadynamics bias potential
# Make sure you print out the 3 variables in the specified order at every step
PRINT ARG=__FILL__ FILE=COLVAR_REWEIGHT STRIDE=__FILL__  # <- also change this one!
```

Now you can run the driver tool using this command:

```
plumed driver --mf_xtc traj_comp.xtc --plumed plumed_reweight.dat --kt 2.494339
```

where `traj_comp.xtc` is the metadynamics trajectory produced in Exercise 2: My first metadynamics simulation. Notice that you have to specify the value of k_BT in energy units. While running your simulation this information was communicated by the MD code.
As a result, PLUMED will produce a new `COLVAR_REWEIGHT` file with one additional column containing the metadynamics bias potential $V(s)$ calculated using all the Gaussians deposited along the entire trajectory. You can easily obtain the weight w of each frame using the expression $w \propto \exp\left(\frac{V(s)}{k_BT}\right)$ (umbrella-sampling-like reweighting). At this point, you can read the `COLVAR_REWEIGHT` file using a python notebook and compute a weighted histogram or, alternatively, if you want PLUMED to do the weighted histograms for you, you can add the following lines at the end of the `plumed_reweight.dat` file and re-run PLUMED driver:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-4.txt
# Use the metadynamics bias as argument
as: REWEIGHT_BIAS ARG=__FILL__

# Calculate histograms of phi and psi dihedrals every 50 steps
# using the weights obtained from the metadynamics bias potentials (umbrella-sampling-like reweighting)
# Look at the manual to understand the parameters of the HISTOGRAM action!
hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=50 BANDWIDTH=0.05 LOGWEIGHTS=as
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=50 BANDWIDTH=0.05 LOGWEIGHTS=as

# Convert histograms h(s) to free energies F(s) = -kBT * log(h(s))
ffphi: CONVERT_TO_FES GRID=hhphi
ffpsi: CONVERT_TO_FES GRID=hhpsi

# Print out the free energies F(s) to file once the entire trajectory is processed
DUMPGRID GRID=ffphi FILE=ffphi.dat
DUMPGRID GRID=ffpsi FILE=ffpsi.dat
```

You can now compare the free energies as a function of ϕ calculated:

1. directly from the metadynamics bias potential using sum_hills as done in Exercise 2: My first metadynamics simulation;
2. using the reweighting procedure introduced in this exercise.

Are the two free energies identical?

11.18.6.4 Exercise 4: Estimating the error in free energies using block-analysis

In the previous exercise, we calculated the *final* bias $V(s)$ on the entire metadynamics trajectory and we used this quantity to calculate the correct statistical weight of each frame, which is needed to reweight the biased simulation.

In this exercise, the user will learn how this information can be used to calculate the error in the reconstructed free energies and assess whether the simulation is converged or not.
Let's first:

- calculate the un-biasing weights $w \propto \exp\left(\frac{V(s)}{k_B T}\right)$ from the `COLVAR_REWEIGHT` file obtained at the end of Exercise 3: Reweighting (unbiasing) a metadynamics simulation;
- print them in a file (called for example `phi.weight`) containing the value of the dihedral ϕ and the corresponding (un-normalized) weight w for each frame of the metadynamics trajectory.

At this point we can apply the block-analysis technique (for more info about the theory, have a look at PLUMED Masterclass 21.2: Statistical errors in MD) to calculate the average free energy across the blocks and the error as a function of block size. For your convenience, the `do_block_fes.py` python script provided in the `data` directory of the `GitHub` repository of this Masterclass can be used to read the `phi.weight` file and produce the desired output. The users should properly choose the following input parameters:

```
# Arguments of do_block_fes.py
# - FILE: input file with 2 colums containing the CV value and weight for each frame of the trajectory
# - NCV: number of CVs
# - MIN/MAX: CV range
# - NBIN: # points in output free energy
# - KBT: temperature in energy units (kJoule/mol)
# - N: Block size
#
python3 do_block_fes.py FILE NCV MIN MAX NBIN KBT N
```

and run the above script for different block sizes ranging from 1 to 1000. For each choice of block size, the script will produce an output file, which contains 3 colums:

- the value of the ϕ variable on a grid;
- the free energy averaged across the blocks for each point of the grid;
- the associated error for each point of the grid.

At this point, the users should:

- calculate the average error across the free-energy profile, i.e. the grid points, for each block size chosen;
- visualize this average error as a function of the block size.

The users should verify that the error increases with the block size (why?) until it reaches a plateau when the dimension of the block exceeds the correlation time between data points. If a plateau is not observed, then the simulation is not converged yet and should be run a bit longer.
From this analysis, what can we say about the convergence of the metadynamics simulation performed in Exercise 2: My first metadynamics simulation ?

11.18.6.5 Exercise 5: Recognizing good from bad CVs

In the previous exercises, we have performed a metadynamics simulation on alanine dipeptide using the backbone dihedral ϕ as CV. We have analyzed the simulation and noticed that:

- under the effect of the bias potential, the system rapidly diffuses in the entire CV space;
- the error in the reconstructed free energy calculated with block analysis rapidly reaches a plateau.

These two observations indicate that our simulation is converged and that the dihedral ϕ can be considered a *good* CV. A good set of CVs for metadynamics should indeed:

1. discriminate between the relevant metastable states of the system;
2. include all the slow modes that characterize transitions between such states;
3. be as small as possible.

Identifying a priori a good set of CVs to describe a specific problem is far from trivial. In this exercise, the users will learn how to detect potential problems in their choice of CV(s). Let's first perform additional 1D metadynamics simulations using one of the following CVs:

- the dihedral ψ;
- the radius of gyration, defined on all the heavy atoms of the system;
- the distance between Oxigen atom 6 and Hydrogen atom 8.

The users are invited to prepare, for each of the CV listed above, a PLUMED input file to perform a 1D well-tempered metadynamics simulation, execute the simulation, and analyse the results in terms of:

- diffusion of the system in the space of the dihedral ϕ
- error analysis in the free energy as a function of ϕ

and compare the results with the previous simulation in which ϕ was used as the metadynamics CV. **Based on this analysis, can you discriminate between *good* from *bad* CVs?**
To complete this exercise, the users should define and test a new CV that is either:

- as good as the dihedral ϕ;
- the worst possible CV to accelerate sampling and describe the two main conformational states of alanine dipeptide.

Please post the results of your simulations on Slack!

11.18.6.6 Exercise 6: A 'real-life' application

In this last exercise, we will tackle a real-life biological problem: studying the conformational transition of a complex biological system. The system that we are going to study is the C-terminal domain (CTD) of the RfaH virulence factor from *Escherichia coli*. This part of the system, which we refer to as RfaH-CTD, undergoes a dramatic conformational transformation from -barrel to -helical, which is stabilized by the N-terminal domain of the RfaH virulence factor (see Fig. masterclass-21-4-RfaH-CTD-fig).
In the `data/RfaH-CTD` folder of the `GitHub` repository of this Masterclass, you will find:

- two PDB files of RfaH-CTD in the -helical and -barrel states;
- a `topol.tpr` file, which is needed to perform a MD simulation of this sytem with GROMACS.

The objective of this exercises are to:

1. compute the free-energy difference between the -helical and -barrel states of RfaH-CTD (with error estimate);
2. (optional) determine the structure and population of other metastable states, if present.

In order to complete the exercise, the students should:

- choose their own CVs (maximum 3) and perform a well-tempered metadynamics simulation. Any CV natively implemented in PLUMED (see Collective Variables) or defined by the users directly in the input file (see CUSTOM) can be used;
- monitor the RMSD of the system from the two reference conformations during the course of the simulation;
- analyze the results as done in the previous exercises (assessment of convergence and error analysis);
- report the free energies (with error bar) as a function of the two RMSD CVs calculated with respect to the reference PDBs;
- (optional) report structure and population of the most signicant, i.e. populated, states.

Please keep in mind that:

- we are simulating the system using a simplified, structure-based potential, called `SMOG`. SMOG is significantly less computational demanding than all-atoms, explicit solvent force fields. However, the simulation of this system might take a few hours, so allocate enough time to complete this exercise;
- some of the CVs or PLUMED functionalities might not work as hydrogen atoms are not present in the system. However, there is always a way around this, so be creative;

- due to the nature of the force field, we are simulating at an unphysical temperature of 60K. Be ready to test large values of the `BIASFACTOR`.

Finally, due to the special nature of the force field, please execute GROMACS using the following command:

```
gmx mdrun -plumed plumed.dat -ntomp 4 -noddcheck
```

You can adjust the number of CPU cores you want to use (here 4, OpenMP parallelization), based on the available resources. The system is not particularly big, therefore using a large number of cores might be inefficient.
Please post the results of your simulations on Slack!

11.19 PLUMED Masterclass 21.5: Simulations with multiple replicas

Authors

Giovanni Bussi

Date

March 15, 2021

11.19.1 Aims

In this Masterclass, we will discuss how to perform and analyze multi-replica simulations where different replicas feel a different bias potential. We will also understand how to compute statistical errors on the computed quantities.

11.19.2 Objectives

Once you have completed this Masterclass you will be able to:

- Use PLUMED and GROMACS to run multiple-replica simulations.
- Use WHAM to combine multiple simulations performed with different bias potentials.
- Calculate error bars on free energies and populations, taking into account correlations induced by replica exchanges.

11.19.3 Setting up PLUMED

For this masterclass you will need versions of PLUMED and GROMACS that are compiled using the MPI library. The versions used in the previous masterclasses will thus not work properly. In order to obtain the correct versions, please use the following commands:

```
conda install --strict-channel-priority -c plumed/label/masterclass-mpi -c conda-forge plumed
conda install --strict-channel-priority -c plumed/label/masterclass-mpi -c conda-forge gromacs
```

The `--strict-channel-priority` might be necessary in case your conda install is configured to download packages from the `bioconda` channel. Indeed, `bioconda` contains a version of GROMACS that is **not** patched with PLUMED and would thus not work here. Similarly, the channel `plumed/label/masterclass-mpi` should receive a priority higher than `conda-forge`, so as to install the MPI version of PLUMED.
On Linux, the command above should install the following packages:

```
  gromacs            plumed/label/masterclass-mpi/linux-64::gromacs-2019.6-h3fd9d12_100
  plumed             plumed/label/masterclass-mpi/linux-64::plumed-2.7.0-h3fd9d12_100
  mpi                conda-forge/linux-64::mpi-1.0-openmpi
  openmpi            conda-forge/linux-64::openmpi-4.1.0-h9b22176_1
  [ etc ... ]
```

The exact versions might be different. Notice however that GROMACS and PLUMED come from the `plumed/label/masterclass-mpi` channel, whereas the required libraries come from the `conda-forge` channel. To be sure the installed GROMACS is compiled with MPI patched with PLUMED, try the following shell command:

```
gmx_mpi mdrun -h 2> /dev/null | grep -q plumed && echo ok
```

It should print `ok`. To be sure that PLUMED has been compiled with MPI, try the following shell command:

```
plumed --has-mpi && echo ok
```

It should print `ok`.
Please ensure that you have setup PLUMED and GROMACS on your machine before starting the exercises. Also notice that in order to obtain good performances it is better to compile GROMACS from source on the machine you are running your simulations. You can find out in the PLUMED documention how to patch GROMACS with PLUMED so as to be able to install it from source. For this tutorial, the conda precompiled binaries will be sufficient.

11.19.4 Resources

The data needed to execute the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-21-5.git
```

Note

All the exercises were tested with PLUMED version 2.7.0 and GROMACS 2019.6

11.19.5 Exercises

Throughout this tutorial we will run simulations of alanine dipeptide in vacuum using GROMACS and PLUMED. Whereas this system is too simple to be considered a proper benchmark for enhanced sampling methods, it is complex enough to be used in learning them. Notice that, although PLUMED has a portable interface, the support for replica-exchange simulations is limited depending on the specific molecular dynamics engine. You should check the documentation of the MD code you are using to know if replica exchange simulations will work correctly with PLUMED.

Warning

At the time of this writing there is a bug in the rendering of the manual for PLUMED 2.7. In particular, all pages containing an example that requires multiple replicas are truncated. Since there is no new features in v2.7 in this sense, you are recommended to switch to the v2.6 manual. To do so, just replace the string `doc-v2.7/user-doc` with the string `doc-v2.6/user-doc` in the address bar.

11.19.5.1 Introduction to replica simulations

Many methods are based on the simultaneous simulation of multiple replicas. In some cases, all the replicas will use the same input file, whereas in other cases a separate input file should be provided for each replica. Notice that using a single input file does not imply that all the replicas will feel the same biasing potential. Indeed, since biasing potentials in PLUMED might be history dependent, and the history of each replica might different from the history of other replicas, the potentials might in the end be different.
PLUMED has been designed so that multiple-replica simulations can be run even if all the replicas are acting in the same directory. In order to avoid clashes in output files, thus, PLUMED will append a suffix corresponding to the index of the replica to the name of each output file (for instance, the command `PRINT FILE=colvar.dat` will print on a file names `colvar.0.dat` in the first replica, etc.). Suffixes will be added also to input files, so that if you run a simulation where the input file is `plumed.dat`, the first replica will open a file named `plumed.0.dat`, and so on. However, for input files, if the file including the suffix does not exist, PLUMED will look for the file without the suffix (in the example, `plumed.dat`). This provides maximum flexibility and allows to manage both cases where the input file is the same and cases where specific input files should be used.
In addition to this, it is possible to use a Special replica syntax that allows to differentiate the input of different replicas, even if they are all reading the same `plumed.dat` file. For instance, the command `RESTRAINT ARG=dAT=@replicas:1.0,1.1,1.2 KAPPA=1.0` will apply restraints at different positions for three replicas.
Notice however that starting with GROMACS 2019 replica simulations are forced to run in separate directories. To exploit the possibility to use a single input file, one should put it in the parent directory and refer to is as `-plumed ../plumed.dat`. Output files will be produced in separate directories by default, but their names will be suffixed.

If you want the PLUMED output files to be in the parent directory, just prepend their name with `../` (as in `PRINT FILE=../colvar.dat`).
In order to use multiple-replica methods, you should run your simulation using MPI. This can be done prefixing your command with `mpiexec -np N --oversubscribe`, where `N` is the number of processes that you want to use and the `--oversubscribe` option is an OpenMPI option that is required to use more processes than the number of available processors. This is typically suboptimal, but we will need it in our lectures to run, e.g., simulations with 32 replicas even if we have a computer with 4 cores.
In brief, to run a GROMACS simulation where the individual replicas are in directories names `dir0`, `dir1`, etc and the `plumed.dat` file is in the parent directory you will need a command such as

```
mpiexec -np 16 --oversubscribe gmx_mpi mdrun -multidir dir? dir?? -plumed ../plumed.dat
```

To run the PLUMED driver processing a trajectory with multiple processes you will need a command such as

```
mpiexec -np 16 --oversubscribe plumed driver -multi 16 -plumed plumed.dat --ixtc traj.xtc
```

If you have random crashes on MacOS, try to set this environemnt variable:

```
export OMPI_MCA_btl="self,tcp"
```

11.19.5.2 Exercise 1: Multiple-windows umbrella sampling with replica exchange

In Exercise 4: Enhancing conformational transitions with multiple-windows umbrella sampling we have seen how to run a multiple-windows umbrella sampling simulation with independent simulations. Here we will run it using replica exchange. The only differences are that:

- Simulations should be run at the same time using `mpiexec`
- You will have to specify a stride for GROMACS to attempt coordinate exchanges, using the `-replex` option.

It will be sufficient to use a single `plumed.dat` file that looks like this:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-5.txt
# vim:ft=plumed
MOLINFO STRUCTURE=../reference.pdb
phi: TORSION ATOMS=__FILL__
psi: TORSION ATOMS=__FILL__
bb: RESTRAINT ARG=phi KAPPA=200.0 AT=@replicas:__FILL__
PRINT ARG=phi,psi,bb.bias FILE=../colvar_multi.dat STRIDE=100
```

According to the instructions above, you should create 32 directories (one per replica), place the tpr file (for this exercise: `topolA.tpr`) in each of them, and run the following command

```
mpiexec -np 32 --oversubscribe gmx_mpi mdrun -multidir dir? dir?? -plumed ../plumed.dat -s topolA.tpr -replex
```

Notice that by omitting the `-replex` option you will be able to run a non-replica-exchange umbrella sampling simulation, identical to the one you performed in Exercise 4: Enhancing conformational transitions with multiple-windows umbrella sampling
We will now repeat exercises Exercise 4: Enhancing conformational transitions with multiple-windows umbrella sampling and Exercise 6: Effect of initial conditions using replica exchange. We will also test different initial conditions, as in Exercise 6: Effect of initial conditions. Please run the following four simulations:

- Starting from state A, without `-replex` (will be identical to Exercise 4: Enhancing conformational transitions with multiple-wind
- Starting from state B, without `-replex` (will be identical to Exercise 6: Effect of initial conditions)
- Starting from state A, with `-replex 200`
- Starting from state B, with `-replex 200`

For the four simulations, perform a WHAM analysis to compute the weights of each frame, and then compute the relative stability of the two minima (as in Exercise 5: Computing populations and errors). To compute weights you need to do the following steps:

1. Concatenate the trajectories (`gmx_mpi trjcat -cat -f dir?/traj_comp.xtc dir??/traj_comp.xtc -o traj_multi.xtc`).
2. Run plumed driver on the concatenated trajectory (`mpiexec -np 32 --oversubscribe plumed driver --ixtc traj_multi.xtc --plumed plumed.dat --multi 32 --trajectory-stride 100`).
3. Read the resulting trajectories, perform WHAM, and compute relative population of the two states adapting this script:

```
import wham
kb=0.008314462618
T=300
col=[]
for i in range(32):
    col.append(plumed.read_as_pandas("colvar_multi." + str(i)+".dat"))
bias=np.zeros((len(col[0]["bb.bias"]),32))
for i in range(32):
        bias[:,i]=traj[i]["bb.bias"]
w=wham.wham(bias,T=kBT)
tr=col[0].phi
is_in_B=np.int_(np.logical_and(tr>0,tr<2))
is_in_A=np.int_(tr<0)
print(np.average(is_in_B,weights=np.exp(w["logW"]))/np.average(is_in_A,weights=np.exp(w["logW"])))
```

Now answer the following questions:

- Is the population different in the four runs?
- Is the dependence on initial condition that we have seen in PLUMED Masterclass 21.3: Umbrella sampling also present when you are using replica exchange?

11.19.5.3 Exercise 2: Demuxing your trajectories

Close to the end of one of the `md.log` files produced by your simulation you will find a short report of the accepted exchanges. For instance

```
Repl  average probabilities:
Repl     0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20
Repl      .30  .32  .29  .24  .21  .17  .16  .21  .31  .27  .28  .26  .23  .21  .16  .08  .11  .23  .28  .27
```

A result like this one will already warn you that there is a bottleneck between replicas 27 and 28 (only 1 percent of the attempted exchanges have been accepted). Anyway, bottlenecks might be not visible in this representation. The full path in the replica-index space of the continuous trajectories ("demuxed") is much more informative.
We will now "demux" our trajectories. For these short trajectories we can use the `demux.pl` script provided by GROMACS. Notice that for long trajectories and frequent exchanges it could have problems to process correctly the output file. In particular, since the value of time is written by GROMACS with a limited number of digits, the original script might be confused regarding when exchanges happened. At this link you can find a modified script that solves the problem by asking you the value of the time step and computing the value of time from the step number, that is stored as an integer and does not suffer roundoff problems: https://github.com/srnas/demux .
The demux script can be used to produce files named `replica_temp.xvg` and `replica_index.xvg` as follows

```
demux.pl dir0/md.log
```

The `replica_temp.xvg` provides, as a function of time, the number of the replica on which each of the continuous simulations is located. For instance, you can follow the migration in the replica ladder of the first replica as follows:

```
replica_temp=np.loadtxt("replica_temp.xvg")
plt.plot(replica_temp[:,0],replica_temp[:,1])
```

The column 1 contains time, whereas the number on column i+1 says which is the current replica index (that is: temperature, in a temperature replica exchange simulation; position of the restraint in a replica-exchange umbrella sampling simulation) of the continuous simulations that started at position i. This file is called "replica_temp" because it has been implemented with temperature replica exchange in mind, but here the index actually refers to the position of the restraint.
Now answer the following two questions:

- Is there any replica that is able to explore the full range of indexes?
- Are all the continuous replicas able to explore the full range of indexes?

Notice that each row of the `replica_temp.xvg` file contains a permutation. The `replica_index.xvg` file just contains the inverse of this permutation. The `replica_index.xvg` can be used to generate the "demuxed" (continuous) trajectories with the following command:

```
import subprocess
subprocess.run("gmx_mpi trjcat -cat -f dir?/traj_comp.xtc dir??/traj_comp.xtc -demux replica_index.xvg -o " +
```

The resulting trajectories can be visualized or analyzed as usual and, at variance with the original trajectories, will have no jump or discontinuity but will rather be continuous functions of time. For instance, you could use `plumed driver` to compute phi on the demuxed trajectories.
Now answer the following two questions:

- Is there any replica that is able to jump from the metastable state at negative phi to the one at positive phi (or viceversa)?
- Are all the continuous replicas able to do so?
- Are these two questions related to the two questions above?

Notice that, if you run your simulation long enough, each "demuxed" trajectory is expected to cover uniformly the whole range of replica indexes. Due to the location of the restraints, this will imply that each "demuxed" trajectory is expected to cover in an approximately uniform manner the range of the biased CV. Thus, to some extent, each of these trajectories should behave similarly to a metadynamics simulation (see ref masterclass-21-4). The flatness of the distribution on the biased CV depends however on the specific parameters of the restraints (stiffness and locations).

11.19.5.4 Exercise 3: Block analysis from demuxed trajectories

Notice that the WHAM analysis does not need to know where each of the frames come from. This implies that when you run WHAM you can do it equivalently using the concatenation of the original trajectories or the concatenation of the "demuxed" trajectories. The advantage of the latter choice is that you can then perform a block analysis on the resulting trajectory where the number of blocks is exactly equal to the number of replicas. These blocks will be independent simulation, with only two small exceptions:

- the paths in replica space are partly constrained, since when a replica goes up another replica goes down.
- replicas might be initialized from correlated conformations (e.g., all of them in A), inducing a correlation.

The second factor can be decreased by improving the way replicas are initialized. The first factor is usually impacting correlation much less than the actual exchanges. These blocks are thus optimally suited to perform a bootstrap analysis of the error without incurring in underestimation due to correlations between blocks.
There is a small tricky issue here. In particular, when we perform the bootstrap analysis, we are going to pick each block a different number of times. Since each block (that is: each "demuxed" trajectory) has been spanning the replica indexes by spending a different time at each replica, the bootstrap trajectory will not satisfy anymore the property that it was generated spending the same time in each replica. The included wham script allows to use this information passing an additional option `traj_weight`. You can adjust the script below to perform the bootstrap analysis:

```
bias=np.zeros((2001*32,32))
! demux.pl dir0/md.log
replica_temp=np.loadtxt("replica_temp.xvg")
replica_temp=np.int_(replica_temp[:,1:]) # ignore first column (time) and convert to int
for i in range(32):
    col=plumed.read_as_pandas("colvar.{}.dat".format(i))
    bias[:,i]=col["bb.bias"]
# here is the calculation done using the full trajectory
w0=wham.wham(bias,T=kb*T)
tr=col.phi
is_in_B=np.int_(np.logical_and(tr>0,tr<2))
is_in_A=np.int_(tr<0)
# here is the resulting ratio in the population of the two minima:
print(np.average(is_in_B,weights=np.exp(w0["logW"]))/np.average(is_in_A,weights=np.exp(w0["logW"])))
# now we run the bootstrap analysis
pop=[]
for i in range(200): # will require some time, you can first play with less than 200 iterations
    # here we pick the blocks
    c=np.random.choice(32,32)
    # here we count how much time was spent in each replica for the resulting trajectory
    tr_w=np.zeros(32)
    for k in range(32):
        tr_w+=np.bincount(replica_temp[:,c[k]],minlength=32)
    # we then use wham. The traj_weight option can be used to tell to the script
    # how much time was spent at each replica
    w=wham.wham(bias.reshape((32,-1,32))[c].reshape((-1,32)),T=kb*T,traj_weight=tr_w)
    tr=np.array(col.phi).reshape((32,-1))[c].flatten()
    is_in_B=np.int_(np.logical_and(tr>0,tr<2))
    is_in_A=np.int_(tr<0)
    pop.append(np.average(is_in_B,weights=np.exp(w["logW"]))/np.average(is_in_A,weights=np.exp(w["logW"])))
# and here we print average and standard deviation
print(np.average(pop),np.std(pop))
```

Notice that this approach is not really standard, so use it with care. There are a few papers in the literature discussing similar ideas, but they usually require estimating the autocorrelation time in advance.

11.19.5.5 Exercise 4: Bias-exchange metadynamics

We will now run a bias-exchange simulation of alanine dipeptide. In bias-exchange simulations, each replica biases a different collective variable. This is a very practical way to enhance sampling for a large number of variables (as many as the replicas that you can afford!). Notice that they will be biased one at a time. As a matter of fact only those that are useful in identifying a transition state will help, but the other ones will not hurt.
Prepare the input file for a simulation with 3 replicas where the following variables are biased:

- `phi`
- `psi`
- none of them

Initialize two of them in structure A (using topolA.tpr) and one of them is structure B (using topolB.tpr). You can use a single input file that looks like this:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-5.txt
# vim:ft=plumed
MOLINFO STRUCTURE=../reference.pdb

# this is needed to allow arbitrary pairs to try exchanges
# in this case, 0<->1, 0<->2, and 1<->2
RANDOM_EXCHANGES

phi: TORSION ATOMS=__FILL__
psi: TORSION ATOMS=__FILL__
# You can use the same parameters that you used in masterclass 21.4
m: METAD ...
  ARG=@replicas:phi,psi,phi
  SIGMA=@replicas:__FILL__
  HEIGHT=__FILL__ # make sure that there is no bias on the third replica!
  BIASFACTOR=__FILL__
  PACE=__FILL__
  GRID_MIN=-pi
  GRID_MAX=+pi
...
```

Now run two separate simulations for 1000000 steps per replica. In one of them you will propose exchanges between replicas with a pace 200, in the other you will not propose any exchange (just omit `-replex 200` from the command line). The second run will thus be equivalent to running three simulations (free, metadynamics on phi, metadynamics on psi) that you already ran in PLUMED Masterclass 21.4: Metadynamics .
We will now use WHAM to combine the resulting trajectories. We can proceed as we did above, but taking into account that when analyzing a metadynamics simulations the way to compute the weight is slightly different. As discussed in Exercise 3: Reweighting (unbiasing) a metadynamics simulation, one of the possible manners to obtain the weight is to use the final potential computed along the trajectory. This required a further processing step in a simple metadynamics simulation. Here we can compute the final potential while processing the concatenated trajectory. In practice, the only difference with respect to the analysis done in Exercise 1: Multiple-windows umbrella sampling with replica exchange is that here we will have to process our trajectories using a different input file, where `PACE` has been set to a large number and `HEIGHT` set to zero. You can then perform WHAM as in Exercise 1: Multiple-windows umbrella sampling with replica exchange and compute the population of the two metastable states.
After you have calculated the relative populations in the two runs (with and without exchanges), answer the following questions:

- Is the relative population consistent with what you obtained in Exercise 1: Multiple-windows umbrella sampling with replica excha
- Are the two simulations (with exchanges and without exchanges) consistent with each other?

Notice that the third replica has been simulated without any metadynamics. This is a so-called neutral replica, that is used sometime in bias-exchange simulations. You can compute the relative population of the two metastable states directly using the populations in that replica (no post-processing needed!).

- Is the result the same as when using WHAM with all replicas?

Now imagine to perform the bias-exchange simulation again usign only two replicas: one of them biasing psi and the other one with no bias. In other words, you would on purpose forget a variable that is very important:

- How do you expect the resulting population to be?

11.19.5.6 Exercise 5: Parallel-tempering metadynamics

We will finally learn how to use parallel-tempering metadynamics. In parallel-tempering metadynamics, sampling is enhanced using parallel-tempering (which enhances all degrees of freedom), whereas metadynamics is used to flatten their histogram. If the biased CV contains a relevant bottleneck and is capable to approximately single out the corresponding transition state, the corresponding transition will be enhanced as well. Notice however that if the parallel-tempering side of the algorithm is sufficient to enhance sampling, it is not necessary to bias a CV that can identify the transition state.
First we will need to prepare our input files. We will use 4 replicas, with temperatures taken from a geometric distribution ranging between 300 and 800K. You should be able to generate the corresponding tpr files using the following script

```
import numpy as np
import re
T=np.geomspace(300,800,4)
for i in range(len(T)):
    with open("top/grompp.mdp") as f:
        l=f.read()
    with open("top/grompp{}.mdp".format(i),"w") as f:
        # if you use this script on your input files, make sure that 300 only appears
        # on the temperature line! or better replace it with a placeholder string such as __TEMP__
        print(re.sub("300",str(T[i]),l),file=f)
    subprocess.run("mkdir -p ptmetad_{}".format(i),shell=True)
    # we will initialize some replica in A and some replica in B
    if i%2==0:
        conf="A"
    else:
        conf="B"
    # we use -maxwarn 1 here since the grompp file has been adapted from an old gromacs version.
    # in general, only use this option after you have understood that the warning is harmless
    subprocess.run("cd top/; gmx_mpi grompp -f grompp{}.mdp -c conf{}.gro -maxwarn 1 -o
       ../ptmetad_{}/topol.tpr".format(i,conf,i), shell=True)
```

You will then be able to run a parallel tempering simulation with the following command

```
mpiexec -np 4 --oversubscribe gmx_mpi mdrun -multi ptmetad_? -replex 200
```

Notice that the acceptance will be compute by GROMACS taking into account the fact that simulations are running at different temperatures. Also notice that in order to obtain a large enough acceptance given the temperature span, you will need a number of replicas that grows with the square root of the number of atoms in the system. For solvated molecules, you would typically need tens of replicas at least.
We will now add the metadynamics ingredient, by preparing a suitable PLUMED input file. Since parallel-tempering metadynamics is designed to cope with cases where you do not have a good CV available, we will directly use `psi` rather than `phi`!

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-5.txt
# vim:ft=plumed
MOLINFO STRUCTURE=../reference.pdb
phi: TORSION ATOMS=__FILL__
psi: TORSION ATOMS=__FILL__
# You can use the same parameters that you used in masterclass 21.4
# However, it is recommended to scale HEIGHT with temperature.
# You can do it either using replicas: syntax in HEIGHT or specifying TAU
# instead of HEIGHT
m: METAD ...
  ARG=psi
  SIGMA=__FILL__
  HEIGHT=__FILL__
  BIASFACTOR=__FILL__
  PACE=__FILL__
  GRID_MIN=-pi
  GRID_MAX=+pi
...
```

The analysis will be simpler here. We will just analyze the first replica (in `ptmetad_0`) as if it was generated using a simple metadynamics simulation. Now compute the usual ratio between the populations of the two metastable minima and answer the following questions:

- Is the result compatible with what you obtained using umbrella sampling?

- Was biasing psi useful in this case (you can also try to compute the populations from a parallel tempering simulation without metadynamics to answer this question)?

11.19.5.7 Exercise 6: Parallel-tempering: pathological case

Repeat exercise Exercise 5: Parallel-tempering metadynamics, but now place your replicas in the range between 300 and 310.

- Is the population of the two states compatible with what you obtained in the other exercises above.
- If not, which is the correct answer? Draw some conclusion on how to detect this type of problem in a realistic situation.

11.20 PLUMED Masterclass 21.6: Dimensionality reduction

Authors

Gareth Tribello

Date

12th April 2021

11.20.1 Aims

The primary aim of this Masterclass is to show you how you might use PLUMED in your work. We will see how to call PLUMED from a python notebook and discuss some strategies for selecting collective variables.

11.20.2 Objectives

Once this Masterclass is completed, users will be able to:

- Calculate CVs and import them into a python notebook.
- Generate visualizations of data using chemiscope
- Use the projection of a vector as a CV.
- Use path collective variables.
- Run dimensionality reduction algorithms with PLUMED
- Use Gibbs dividing surfaces to determine the extent of a phase.

11.20.3 Acknowledgements

Throughout this exercise, we use the atomistic simulation environment:
https://wiki.fysik.dtu.dk/ase/
and chemiscope:
https://chemiscope.org/
Please look at the websites for these codes that are given above for more information.

11.20.4 Setting up the software

For this Masterclass, you will need to set up the plumed and gromacs as you did for PLUMED Masterclass 21.5: Simulations with multip
You thus install plumed and gromacs using:

```
conda install --strict-channel-priority -c plumed/label/masterclass-mpi -c conda-forge plumed
conda install --strict-channel-priority -c plumed/label/masterclass-mpi -c conda-forge gromacs
```

You can install the other software you need using:

```
conda install -c conda-forge py-plumed Numpy pandas matplotlib notebook mdtraj mdanalysis git ase
```

Notice that you need a package called ase (the atomic simulation environment) for these exercises and the other packages you have been using. You also need to install chemiscope, which you can do by using the following command:

```
pip install chemiscope
```

11.20.5 Resources

The data needed to complete this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-21-6.git
```

I recommend that you run each exercise in a separate sub-directory (i.e. Exercise-1, Exercise-2, ...), which you can create inside the root directory `masterclass-21-6`. Organizing your data this way will help you to keep things clean.

Note

All the exercises have been tested with PLUMED version 2.7.0.

11.20.6 Exercises

11.20.6.1 Exercise 1: Running PLUMED from a python notebook

I like working with Python notebooks. Using notebooks allows me to keep the codes that I am working on close to the explanations of how these codes work. I can also make all my figures within a notebook. By using notebooks, I can thus have a single file that contains:

- Discussion of the work done.
- The analysis code.
- The final figures that were generated.

In previous masterclasses we have run plumed driver through bash from within a notebook by using commands like the one shown below:

```
!cd ../Exercises/Exercise_1 && plumed driver --noatoms > /dev/null
# Read in colvar file produced
data = np.loadtxt("../Exercises/Exercise_1/colvar")
```

We have then read in the colvar file produced and analyzed it from within the notebook. We can avoid using plumed driver and can call plumed directly from python using the python interface. The code below, for instance, generates the plot underneath the code. The code reads in the trajectory from the file traj.pdb that you obtained from the GitHub repository. The plot then shows how the ϕ angle on the second residue of the protein changes with time.

```
import matplotlib.pyplot as plt
import numpy as np
import plumed
import ase
import ase.io
# Read in trajectory using ase
traj = ase.io.read('../data/traj.pdb',':')
# Setup plumed object to do calculation
p = plumed.Plumed()
p.cmd("setMDEngine","python")
# Read PDB so need to multiply by 0.1 to convert to nm
p.cmd("setMDLengthUnits", 0.1)
p.cmd("setTimestep", 1.)
p.cmd("setKbT", 1.)
natoms = len(traj[0].positions)
p.cmd("setNatoms",natoms)
p.cmd("setLogFile","test.log")
p.cmd("init")
# Read plumed input
p.cmd("readInputLine","MOLINFO STRUCTURE=../data/bhp.pdb")
# If you are doing many variables I would represent putting these
# next three PLUMED commands into a function
p.cmd("readInputLine", "t1: TORSION ATOMS=@phi-2" )
# Now setup some memory to hold the variable that is shared
# between plumed and the underlying code
shape = np.zeros( 1, dtype=np.int_ )
p.cmd("getDataRank t1 ", shape )
```

```
t1 = np.zeros((1))
p.cmd("setMemoryForData t1", t1)
# Loop over trajectory and get data from plumed
nfram, tt, v1, box = 0, [], [], np.array([[100.,0,0],[0,100.,0],[0,0,100]])
charges, forces, virial = np.zeros(natoms,dtype=np.float64), np.zeros([natoms,3]),
       np.zeros((3,3),dtype=np.float64)
for ts in traj :
    # Set all the input variables to PLUMED
    p.cmd("setStep",nfram)
    p.cmd("setBox",box )
    p.cmd("setMasses", ts.get_masses() )
    p.cmd("setCharges", charges )
    pos = np.array(ts.get_positions(), dtype=np.float64 )
    p.cmd("setPositions", pos )
    p.cmd("setForces", forces )
    p.cmd("setVirial", virial )
    # Run the plumed calculation
    p.cmd("calc")
    tt.append(nfram)
    # We can now extract the value of the torsion by accessing the shared memory we set up earlier
    v1.append(t1[0])
    nfram = nfram + 1
# Plot teh graph of the torsional angle as a function of time
plt.plot( tt, v1, 'ko')
plt.xlabel("Simulation step")
plt.ylabel("Torsion angle / radians")
plt.show()
```

Your task in this first exercise is to modify the code above and to produce a figure similar to the one shown below. This figure shows all the values of the ϕ and ψ angles in the second residue of the protein during the simulation.

11.20.6.2 Exercise 2: Generating a chemiscope representation

Plots showing the trajectory in CV space similar to those you generated at the end of the previous exercise are helpful. What would be more useful, however, is some way of understanding the relationship between the positions in CV space and the structures of the various atoms. In other words, what we would like is something like this:
You can see the frame in the trajectory that each point in the plot corresponds to from the above figure. The snapshots on the right correspond to the structures the system had at the points highlighted in red, yellow, green and blue respectively in the plot on the left.

The figure above was generated using chemiscope.
This server allows you to generate and interact with plots like the one shown above. **Your task in this exercise is to generate your own chemiscope representation of the data in traj.pdb.** To create a chemiscope representation of the ϕ angles that we generated using the first python script from the previous exercise, you would add the following python code:

```
from ase.data import atomic_masses
from chemiscope import write_input
# This ensures that the atomic masses are used in place of the symbols
# when constructing the atomic configurations' chemiscope representations.
# Using the symbols will not work because ase is written by chemists and not
# biologists.  For a chemist, HG1 is mercury as opposed to the first hydrogen
# on a guanine residue.
for frame in traj:
    frame.numbers = np.array(
        [
            np.argmin(np.subtract(atomic_masses, float(am)) ** 2)
            for am in frame.arrays["occupancy"]
        ]
    )
# This constructs the dictionary of properties for chemiscope
properties = {
    "time": {
        "target": "structure",
        "values": tt,
        "description": "Simulation step number",
    },
    "t2": {
        "target": "structure",
        "values": v1,
        "description": "Phi angle of second residue",
    },
}
# This generates our chemiscope output
write_input("torsion_chemiscope.json.gz", frames=traj, properties=properties )
```

You would then upload the torsion_chemiscope.json.gz file that is generated by this script at https://chemiscope.org.
See if you can generate your own chemiscope representation of the data in traj.pdb. I would recommend

calculating and uploading a chemiscope representation of all the protein's torsional angles.
At the very least, you need to do at least two backbone torsional angles. However, if you do more than two torsions, you can generate a plot like the one shown below.

11.20.6.3 Exercise 3: Dimensionality reduction

Chemiscope comes into its own when you are working with a machine learning algorithm. These algorithms can (in theory) learn the collective variables you need to use from the trajectory data. To make sense of the coordinates that have been learned, you have to carefully visualize where structures are projected in the low dimensional space. You can use chemiscope to complete this process of visualizing the representation the computer has found for you. In this next set of exercises, we will apply various dimensionality reduction algorithms to the data contained in the file traj.pdb. If you visualize the contents of that file using VMD, you will see that this file contains a short protein trajectory. You are perhaps unsure what CV to analyze this data and thus want to see if you can shed any light on the contents of the trajectory by using machine learning.
Typically, PLUMED analyses one set of atomic coordinates at a time. To run a machine learning algorithm, however, you need to gather information on multiple configurations. Therefore, the first thing you need to learn to use these algorithms is how to store configurations for later analysis with a machine learning algorithm. The following input illustrates how to complete this task using PLUMED.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
# This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=__FILL__ MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES __FILL__=@nonhydrogens

# This should output the atomic positions for the frames that were collected to a pdb file called traj.pdb
OUTPUT_ANALYSIS_DATA_TO_PDB USE_OUTPUT_DATA_FROM=__FILL__ FILE=traj.pdb
```

Copy the input above into a plumed file and fill in the blanks. You should then be able to run the command using:
Then, once all the blanks are filled in, run the command using:

```
plumed driver --mf_pdb traj.pdb
```

You can also store the values of collective variables for later analysis with these algorithms. **Modify the input above so that all Thirty backbone dihedral angles in the protein are stored, and output using OUTPUT_ANALYSIS_DATA_TO_COLVAR and rerun the calculation.**
For more information on the dimensionality reduction algorithms that we are using in this section see:
https://arxiv.org/abs/1907.04170

11.20.6.4 PCA

Having learned how to store data for later analysis with a dimensionality reduction algorithm, lets now apply principal component analysis (PCA) upon our stored data. In principle component analysis, low dimensional projections for our trajectory are constructed by:

- Computing a covariance matrix from the trajectory data
- Diagonalizing the covariance matrix.
- Calculating the projection of each trajectory frame on a subset of the eigenvectors of the covariance matrix.

To perform PCA using PLUMED, we are going to use the following input with the blanks filled in:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
# This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=__FILL__ MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES __FILL__=@nonhydrogens
# This diagonalizes the covariance matrix
pca: PCA USE_OUTPUT_DATA_FROM=__FILL__ METRIC=OPTIMAL NLOW_DIM=2
```

```
# This projects each of the trajectory frames onto the low dimensional space that was
# identified by the PCA command
dat: PROJECT_ALL_ANALYSIS_DATA USE_OUTPUT_DATA_FROM=__FILL__ PROJECTION=__FILL__

# This should output the PCA projections of all the coordinates
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=__FILL__ ARG=dat.* FILE=pca_data

# These next three commands calculate the secondary structure variables.  These
# variables measure how much of the structure resembles an alpha helix, an antiparallel beta-sheet
# and a parallel beta-sheet.  Configurations that have different secondary structures should be projected
# in different parts of the low dimensional space.
alpha: ALPHARMSD RESIDUES=all
abeta: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1.0
pbeta: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1.0

# These commands collect and output the secondary structure variables so that we can use this information to
# determine how good our projection of the trajectory data is.
cc2: COLLECT_FRAMES ARG=alpha,abeta,pbeta
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=cc2 ARG=cc2.* FILE=secondary_structure_data
```

To generate the projection, you run the command:

```
plumed driver --mf_pdb traj.pdb
```

You can generate a projection of the data above using chemiscope by using the following script:

```
# This ase command should read in the traj.pdb file that was analyzed.  Notice that the analysis
# actions below ignore the first frame in this trajectory, so we need to take that into account
# when we generated the chemiscope
traj = ase.io.read('../data/traj.pdb',':')
# This reads in the PCA projection that are output by by the OUTPUT_ANALYSIS_DATA_TO_COLVAR command
# above
projection = np.loadtxt("pca_data")
# We also read in the secondary structure data by colouring points following the secondary
# structure. We can get a sense of how good our projection is.
structure = np.loadtxt("secondary_structure_data")
# This ensures that the atomic masses are used in place of the symbols
# when constructing the atomic configurations' chemiscope representations.
# Using the symbols will not work because ase is written by chemists and not
# biologists.  For a chemist, HG1 is mercury as opposed to the first hydrogen
# on a guanine residue.
for frame in traj:
    frame.numbers = np.array(
        [
            np.argmin(np.subtract(atomic_masses, float(am)) ** 2)
            for am in frame.arrays["occupancy"]
        ]
    )
# This constructs the dictionary of properties for chemiscope
properties = {
    "pca1": {
        "target": "structure",
        "values": projection[:,0],
        "description": "First principle component",
    },
    "pca2": {
        "target": "structure",
        "values": projection[:,1],
        "description": "Second principle component",
    },
    "alpha": {
        "target": "structure",
        "values": structure[:,0],
        "description": "Alpha helical content",
    },
    "antibeta": {
        "target": "structure",
        "values": structure[:,1],
        "description": "Anti parallel beta sheet content",
    },
    "parabeta": {
        "target": "structure",
        "values": structure[:,2],
        "description": "Parallel beta sheet content",
    },
}
# This generates our chemiscope output
write_input("pca_chemiscope.json.gz", frames=traj[1:], properties=properties )
```

When the output from this set of commands is loaded into chemiscope, we can construct figures like the one shown below. On the axes here, we have plotted the PCA coordinates. The points are then coloured according to the alpha-helical content.

See if you can use PLUMED and chemiscope to generate a figure similar to the one above. Try to experiment

with the way the points are coloured. Look at the beta-sheet content as well.

11.20.6.5 MDS

In the previous section, we performed PCA on the atomic positions directly. In the section before last, however, we also saw how we could store high-dimensional vectors of collective variables and then use them as input to a dimensionality reduction algorithm. Therefore, we might legitimately ask if we can do PCA using these high-dimensional vectors as input rather than atomic positions. The answer to this question is yes as long as the CV is not periodic. If any of our CVs are not periodic, we cannot analyze them using the PCA action. We can, however, formulate the PCA algorithm differently. In this alternative formulation, which is known as classical multidimensional scaling (MDS), we do the following:

- We calculate the matrix of distances between configurations
- We perform an operation known as centring the matrix.
- We diagonalize the centred matrix The eigenvectors multiplied by the corresponding eigenvalue's square root can then be used as a set of projections for our input points.

This method is used less often than PCA as the matrix that we have to diagonalize here in the third step can be considerably larger than the matrix that we have to diagonalize when we perform PCA.
To avoid this expensive diagonalization step, we often select a subset of so-called landmark points on which to run the algorithm directly. Projections for the remaining points are then found by using a so-called out-of-sample procedure. This is what has been done in the following input:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
 This command reads in the template pdb file and thus allows us to use the @nonhydrogens
# group later in the input
MOLINFO STRUCTURE=beta-hairpin.pdb MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES ATOMS=@nonhydrogens

# The following commands compute all the Ramachandran angles of the protein for you
r2-phi: TORSION ATOMS=@phi-2
r2-psi: TORSION ATOMS=@psi-2
r3-phi: TORSION ATOMS=@phi-3
r3-psi: TORSION ATOMS=@psi-3
r4-phi: TORSION __FILL__
r4-psi: TORSION __FILL__
r5-phi: TORSION __FILL__
r5-psi: TORSION __FILL__
r6-phi: TORSION __FILL__
r6-psi: TORSION __FILL__
r7-phi: TORSION __FILL__
r7-psi: TORSION __FILL__
r8-phi: TORSION __FILL__
r8-psi: TORSION __FILL__
r9-phi: TORSION __FILL__
r9-psi: TORSION __FILL__
r10-phi: TORSION __FILL__
r10-psi: TORSION __FILL__
r11-phi: TORSION __FILL__
r11-psi: TORSION __FILL__
r12-phi: TORSION __FILL__
r12-psi: TORSION __FILL__
r13-phi: TORSION __FILL__
r13-psi: TORSION __FILL__
r14-phi: TORSION __FILL__
r14-psi: TORSION __FILL__
r15-phi: TORSION __FILL__
r15-psi: TORSION __FILL__
r16-phi: TORSION __FILL__
r16-psi: TORSION __FILL__

# This command stores all the Ramachandran angles that were computed
angles: COLLECT_FRAMES __FILL__=r2-phi,r2-psi,r3-phi,r3-psi,r4-phi,r4-psi,r5-phi,r5-psi,r6-phi,r6-psi,r7-phi,r
# Lets now compute the matrix of distances between the frames in the space of the Ramachandran angles
distmat: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=__FILL__ METRIC=EUCLIDEAN
```

```
# Now select 500 landmark points to analyze
fps: LANDMARK_SELECT_FPS USE_OUTPUT_DATA_FROM=__FILL__ NLANDMARKS=500
# Run MDS on the landmarks
mds: CLASSICAL_MDS __FILL__=fps NLOW_DIM=2
# Project the remaining trajectory data
osample: PROJECT_ALL_ANALYSIS_DATA USE_OUTPUT_DATA_FROM=__FILL__ PROJECTION=__FILL__

# This command outputs all the projections of all the points in the low dimensional space
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=__FILL__ ARG=osample.* FILE=mds_data

# These next three commands calculate the secondary structure variables.  These
# variables measure how much of the structure resembles an alpha helix, an antiparallel beta-sheet
# and a parallel beta-sheet.  Configurations that have different secondary structures should be projected
# in different parts of the low dimensional space.
alpha: ALPHARMSD RESIDUES=all
abeta: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1.0
pbeta: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1.0

# These commands collect and output the secondary structure variables so that we can use this information to
# determine how good our projection of the trajectory data is.
cc2: COLLECT_FRAMES ARG=alpha,abeta,pbeta
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=cc2 ARG=cc2.* FILE=secondary_structure_data
```

This input collects all the torsional angles for the configurations in the trajectory. Then, at the end of the calculation, the matrix of distances between these points is computed, and a set of landmark points is selected using a method known as farthest point sampling. A matrix that contains only those distances between the landmarks is then constructed and diagonalized by the CLASSICAL_MDS action so that projections of the landmarks can be built. The final step is then to project the remainder of the trajectory using the PROJECT_ALL_ANALYSIS_DATA action. Try to fill in the blanks in the input above and run this calculation now using the command:

```
plumed driver --mf_pdb traj.pdb
```

Once the calculation has completed, you can, once again, visualize the data using chemiscope by using a suitably modified version of the script from the previous exercise. The image below shows the MDS coordinates coloured according to the alpha-helical content.
Try to generate an image that looks like this one yourself by completing the input above and then using what you learned about generating chemiscope representations in the previous exercise.

11.20.6.6 Sketch-map

The two algorithms (PCA and MDS) that we have looked at thus far are linear dimensionality reduction algorithms. In addition to these, there are a whole class of non-linear dimensionality reduction reduction algorithms which work by transforming the matrix of dissimilarities between configurations, calculating geodesic rather than Euclidean distances between configurations or by changing the form of the loss function that is optimized. In this final exercise, we will use an algorithm that uses the last of these three strategies to construct a non-linear projection. The algorithm is known as sketch-map and input for sketch-map is provided below:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
# This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=__FILL__ MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES __FILL__=@nonhydrogens
# This should output the atomic positions for the frames that were collected and analyzed using MDS
OUTPUT_ANALYSIS_DATA_TO_PDB USE_OUTPUT_DATA_FROM=__FILL__ FILE=traj.pdb

# The following commands compute all the Ramachandran angles of the protein for you
r2-phi: TORSION ATOMS=@phi-2
r2-psi: TORSION ATOMS=@psi-2
r3-phi: TORSION ATOMS=@phi-3
r3-psi: TORSION ATOMS=@psi-3
r4-phi: TORSION __FILL__
r4-psi: TORSION __FILL__
r5-phi: TORSION __FILL__
r5-psi: TORSION __FILL__
r6-phi: TORSION __FILL__
r6-psi: TORSION __FILL__
r7-phi: TORSION __FILL__
```

```
r7-psi: TORSION __FILL__
r8-phi: TORSION __FILL__
r8-psi: TORSION __FILL__
r9-phi: TORSION __FILL__
r9-psi: TORSION __FILL__
r10-phi: TORSION __FILL__
r10-psi: TORSION __FILL__
r11-phi: TORSION __FILL__
r11-psi: TORSION __FILL__
r12-phi: TORSION __FILL__
r12-psi: TORSION __FILL__
r13-phi: TORSION __FILL__
r13-psi: TORSION __FILL__
r14-phi: TORSION __FILL__
r14-psi: TORSION __FILL__
r15-phi: TORSION __FILL__
r15-psi: TORSION __FILL__
r16-phi: TORSION __FILL__
r16-psi: TORSION __FILL__

# This command stores all the Ramachandran angles that were computed
angles: COLLECT_FRAMES __FILL__=r2-phi,r2-psi,r3-phi,r3-psi,r4-phi,r4-psi,r5-phi,r5-psi,r6-phi,r6-psi,r7-phi,r
# Lets now compute the matrix of distances between the frames in the space of the Ramachandran angles
distmat: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=__FILL__ METRIC=EUCLIDEAN
# Now select 500 landmark points to analyze
fps: LANDMARK_SELECT_FPS USE_OUTPUT_DATA_FROM=__FILL__ NLANDMARKS=500
# Run sketch-map on the landmarks
smap: SKETCH_MAP __FILL__=fps NLOW_DIM=2 HIGH_DIM_FUNCTION={SMAP R_0=6 A=8 B=2} LOW_DIM_FUNCTION={SMAP R_0=6 A
# Project the remaining trajectory data
osample: PROJECT_ALL_ANALYSIS_DATA USE_OUTPUT_DATA_FROM=__FILL__ PROJECTION=__FILL__

# This command outputs all the projections of all the points in the low dimensional space
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=__FILL__ ARG=osample.* FILE=smap_data

# These next three commands calculate the secondary structure variables.  These
# variables measure how much of the structure resembles an alpha helix, an antiparallel beta-sheet
# and a parallel beta-sheet.  Configurations that have different secondary structures should be projected
# in different parts of the low dimensional space.
alpha: ALPHARMSD RESIDUES=all
abeta: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1.0
pbeta: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1.0

# These commands collect and output the secondary structure variables so that we can use this information to
# determine how good our projection of the trajectory data is.
cc2: COLLECT_FRAMES ARG=alpha,abeta,pbeta
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=cc2 ARG=cc2.* FILE=secondary_structure_data
```

This input collects all the torsional angles for the configurations in the trajectory. Then, at the end of the calculation, the matrix of distances between these points is computed, and a set of landmark points is selected using a method known as farthest point sampling. A matrix that contains only those distances between the landmarks is then constructed and diagonalized by the CLASSICAL_MDS action, and this set of projections is used as the initial configuration for the various minimization algorithms that are then used to optimize the sketch-map stress function. As in the previous exercise, once the projections of the landmarks are found, the projections for the remainder of the points in the trajectory are found by using the PROJECT_ALL_ANALYSIS_DATA action. Try to fill in the blanks in the input above and run this calculation now using the command:

```
plumed driver --mf_pdb traj.pdb
```

Once the calculation has completed, you can, once again, visualize the data generated using chemiscope by using the scripts from earlier exercises. My projection is shown in the figure below. Points are, once again, coloured following the alpha-helical content of the corresponding structure.
Try to see if you can reproduce an image that looks like the one above

11.20.6.6.1 Summary This exercise has shown you that running dimensionality reduction algorithms using P↩ LUMED involves the following stages:

- Data is collected from the trajectory using COLLECT_FRAMES.
- Landmark points are selected using a Landmark Selection algorithm

- The distances between the trajectory frames are computed using EUCLIDEAN_DISSIMILARITIES
- A loss function is optimized to generate projections of the landmarks.
- Projections of the non-landmark points are found using PROJECT_ALL_ANALYSIS_DATA.

There are multiple choices to be made in each of the various stages described above. For example, you can change the particular sort of data this is collected from the trajectory. There are multiple different ways to select landmarks. You can use the distances directly, or you can transform them. You can use various loss functions and optimize the loss function using a variety of different algorithms. When you tackle your own research problems using these methods, you can experiment with the various choices that can be made.

Generating a low-dimensional representation using these algorithms is not enough to publish a paper. We want to get some physical understanding from our simulation. Gaining this understanding is the hard part.

11.20.6.7 Exercise 4: Path collective variables

In many papers in this area, you will hear people talk about the distinction between collective variables and the reaction coordinate. The distinction these authors are trying emphasize when they use this language is between a descriptor that takes different values for the various important basins in the energy landscape (the collective variable) and the actual pathway the reaction proceeds along (the reaction coordinate). Furthermore, the authors of such papers will often argue that the reaction coordinate is simply some linear/non-linear combination of collective variables. In this exercise, we will study alanine dipeptide with path collective variables to show you one way of thinking about this distinction between collective variables and reaction coordinates.
By studying a system that is this simple, you will also hopefully come to understand how we can interpret the coordinates that we extract using the dimensionality reduction algorithms that were discussed in the previous exercise.
To remind you, the free energy surface as a function of the ϕ and ψ angles for alanine dipeptide is shown below:
In other masterclasses, we have discussed how there are two critical states in the above energy landscape. These states are labelled C7eq and C7ax above.
The two Ramachandran angles plotted on the x and y axes of the free energy surface above are examples of what we have called collective variables. Both of these angles can be used to distinguish between C7eq and C7ax configurations. The reaction coordinate is the path that the system actually takes as it moves from the C7ax to the C7eq configuration. Based on the shape of the free energy surface, we might suppose that this reaction coordinate looks something like the black line shown below:
The file called alanine-transformation.pdb that you can find in the data directory of the GitHub repository contains a series of configurations that lie close to the transition path that is illustrated in black in the figure above. Below are plots that show how ϕ and ψ change as the system moves along this path. **Try to see if you can use what you have learned in previous masterclasses to reproduce the figure above before continuing.**

11.20.6.7.1 RMSD distances

We know what structures correspond to the various stable states of our system. We might, therefore, be tempted to ask ourselves if we can not just use the RMSD distance from a structure as the reaction coordinate. This approach will work if there is a single important configuration in the energy landscape. We could use the RMSD distance from this lowest energy structure as a CV to extract this configuration's free energy relative to everything else. How well does this work if we have two distinct states, though, as we have for alanine dipeptide? To investigate this question, fill in the PLUMED input below that calculates the RMSD distances from the C7ax and C7eq configurations:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
c7ax: RMSD __FILL__=../data/C7ax.pdb TYPE=__FILL__
c7eq: RMSD __FILL__=../data/C7eq.pdb TYPE=__FILL__
PRINT ARG=c7ax,c7eq FILE=colvar STRIDE=100'''
```

You can run an MD simulation to monitor these distances using the python script below.

```
def generate_gromacs_input( directory, plumed_input, nsteps, temp ) :
    # Setup the initial configuration by picking a frame at random from the startpoints file
    allc = io.read('../data/startpoints.pdb',':')
    nframes = len(allc)
    io.write( directory + '/conf.pdb', allc[int(np.random.randint(0,nframes))] )
    # Copy the topology to the appropriate directory
    shutil.copyfile("../data/topol.top", directory + "/topol.top")
    # Setup the mdp file
    mdp = open("../data/md.mdp","r")
    contents = mdp.read()
```

```
    mdp.close()
    new_content = contents.replace("SEED",
       str(np.random.randint(0,1000000))).replace("NSTEPS",str(nsteps)).replace("TEMP",str(temp))
    mdpout = open(directory + "/md.mdp", "w")
    mdpout.write(new_content)
    mdpout.close()
    # And write the plumed input
    pout = open(directory + "/plumed.dat", "w")
    pout.write( plumed_input )
    pout.close()
    return
plm = '''
INSERT PLUMED INNPUT HERE
'''
# Create a directory to run the calculation in
!rm -rf ../Unbiased_MD && mkdir ../Unbiased_MD
# Generate gromacs input for MD simulation at 1000 K
generate_gromacs_input( '../Unbiased_MD', plm, 500000, 1000 )
# Now run gromacs
!cd ../Unbiased_MD/ && gmx_mpi grompp -f md.mdp -c conf.pdb -p topol.top -maxwarn 2 &> /dev/null
!cd ../Unbiased_MD/ && gmx_mpi mdrun --plumed plumed.dat &> /dev/null
# And read in the colvar files
unbiased_data = np.loadtxt('../Unbiased_MD/colvar')
```

I ran simulations at 300K and 1000K using the above script. When the simulations completed, I was able to construct the figures below:

The black points above are the RMSD values for the trajectories. I have also shown the RMSD values for the frames in alanine-transformation in red. Notice that all the configurations explored in the MD are very distant from the C7ax and C7eq states. **Try now to reproduce the above figure yourself.**

You can see that at 300K, the simulation we ran did not visit the C7eq state. Furthermore, you can also clearly see this from the projections of the trajectories on the RMSD distances from these two configurations. Notice in these figures, however, that the distances from these two reference configurations were often considerably larger than the distance between these reference configurations and the configurations on the reference path in alanine-transformation.pdb.

11.20.6.7.2 Using dot products to calculate CVs Instead of calculating two distances, we might ask ourselves if the linear combination of ϕ and ψ that is illustrated in the figure below:

gives a better description for the transition. We can define this CV as follows:

$$s = \frac{(\phi_2 - \phi_1).(\phi_3 - \phi_1) + (\psi_2 - \psi_1).(\psi_3 - \psi_1)}{\sqrt{(\phi_2 - \phi_1)^2 + (\psi_2 - \psi_1)^2}}$$

In this expression (ϕ_1, ψ_1) are the Ramachandran angles in the C_7eq configuration and (ϕ_2, ψ_2) are the Ramachandran angles in the C_7ax. (ϕ_3, ψ_3) is the instantaneous configuration. You should thus be able to see that we have arrived at the above expression by using our knowledge of the dot product between two vectors.

See if you can write an input to re-analyse the data in alanine-transformation.pdb and the MD simulations from the previous section using this CV.

You should be able to get plots of the value of this CV as a function of step number that looks like the ones shown below:

I implemented this CV using a combination of TORSION and COMBINE. I also ignored the fact that the torsions are periodic variables when calculating the linear combination.

You can't use the sort of linear algebra above with periodic variables, but it is OK for these illustrative purposes.

Notice that the first frame in alanine-transformation.pdb has the molecule in the C_7ax configuration. The last frame has the molecule in the C_7eq state.

11.20.6.7.3 PCA Collective Variables Figure masterclass-21-6-pca-figure showed how we can construct a new collective variables by taking a linear combination of two other variables. This idea can be extended to higher dimensions, however. As long as we can find the vector that connectes the C_7eq and C_7ax states we can project our current coordinates on that particular vector. We can even define this vector in the space of the coordinates of the atoms. In other words, if the 3 N coordinate of atomic positions is $\mathbf{x}^{(1)}$ for the C_7eq configuration and $\mathbf{x}^{(2)}$ for the C_7ax configuration and if the instantaneous configuration of the atoms is $\mathbf{x}^{(3)}$ we can use the following as a CV:

$$s = \frac{\sum_{i=1}^{3N}(x_i^{(2)} - x_i^{(1)})(x_i^{(3)} - x_i^{(1)})}{\sqrt{\sum_{i=1}^{3N}(x_i^{(2)} - x_i^{(1)})^2}}$$

as long as rotational and translational movement is removed before calculating the two displacement vectors above. We can also get some sense of how far the system has moved from this vector by computing:

$$z = \sqrt{\sum_{i=1}^{3N} (x_i^{(3)} - x_i^{(1)})^2 - s^2}$$

which follows by applying Pythagoras theorem.
You can calculate this collective variable using PLUMED by using the input below:

```
p: PCAVARS __FILL__=pca-reference.pdb __FILL__=OPTIMAL
PRINT ARG=p.* FILE=colvar
```

Use this input to re-analyse the data in alanine-transformation.pdb and your MD simulations before continuing. The figure below shows the results that I obtained by analyzing the data using the PCAVARS command above.
The figure above shows that this coordinate is good. We move quite far from this initial vector when we move to the C7ax state, however.

11.20.6.7.4 Two PCA Collective variables Instead of using a single PCA variable and calculating the projection on the vector connecting these two points, we can instead use the projection of the instantaneous coordinates in the plane that contains three reference configurations. You can calculate these collective variables using PLUMED by using the input below:

```
p: PCAVARS __FILL__=pca2-reference.pdb __FILL__=OPTIMAL
PRINT ARG=p.* FILE=colvar
```

Notice that there are now three structures within pca2-reference.pdb, and thus two vectors are defined. **Use this input to re-analyse the data in alanine-transformation.pdb and your MD simulations before continuing.**
The figure below shows the results that I obtained by analyzing the data using the PCAVARS command above.

11.20.6.7.5 Path collective variables Instead of using the projection on more than one vector, we can extend the ideas in the previous section and use curvilinear paths rather than linear paths. This trick is what is done with the PATH CVs that are illustrated in the figure below:
As you can see, there are two path collective variables, $S(X)$ measures your position on a path and is calculated using:

$$S(X) = \frac{\sum_{i=1}^{N} i \ \exp^{-\lambda|X-X_i|}}{\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|}}$$

$Z(x)$, meanwhile, measures the distance from the path using:

$$Z(X) = -\frac{1}{\lambda} \ln \left[\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|} \right]$$

We can calculate these CVs using a filled-in version of the input that is shown below:

```
path: PATH __FILL__=../data/alanine-path.pdb TYPE=__FILL__ LAMBDA=15100.
PRINT ARG=* FILE=colvar
```

The figure below shows the ϕ and ψ angles for the configurations that define the path as red dots. Meanwhile, the black dots are the ϕ and ψ angles that are visited during a high temperature, unbiased MD trajectory. The green dots are then the configurations in alanine-transformation.pdb You can clearly see that the red dots lie along the transition path that connects the C7eq and C7ax configurations and that the green dots follow this pathway:
When we calculate the path collective variables for the black and red data points in the above figure, we get the result shown on the right. The panel on the left shows the output from a 300 K simulation during which the C7eq configuration was not visited:
You can see that the majority of configurations have very small z values. It would thus seem that we have identified our reaction coordinate. We can use the S-path coordinate to tell us where we are on the transition pathway between the two states. Furthermore, in defining this coordinate, we have used the positions of all the heavy atoms in the protein.

11.20.6.7.6 The isocommittor Notice that when we talk about the reaction coordinate, we are not talking about a single set of configurations that connect the two states. The previous sections have shown that there are always multiple pathways that connect the two states. The task of identifying a single reaction coordinate thus appears impossible. How, after all, can there be a single reaction path if we know that there are multiple pathways that connect the two states?

One way of answering this question is to look at how far the transitions you observe deviate from the reference transition path using the Z-coordinate. An alternative answer can be found by considering so-called isocommittor surfaces. When using this method, you suppose there is a saddle point between the two states of interest (the C_7ax and C_7eq configurations in our alanine dipeptide example).
Let's suppose that we now start a simulation with the system balanced precariously on this saddle point. The system will, very-rapidly, fall off the maximum and move towards either the left or the right basin. If we are exactly at the saddle, 50% of the trajectories will fall to the right, and 50% will fall to the left.
We can use the idea of the isocommittor to identify whether any collective variable is giving a good description of the transition state ensemble's location. The idea in such simulations is to define regions of space that correspond to the various stable states in the free energy landscape. We then launch lots of simulations from configurations that are not within these basins and identify the basin that these calculations visit first. We can then make graphs like those shown below. These graphs show the fraction of times a particular CV value was visited in a trajectory that visited the c7eq basin before it visited the c7ax basin.
The PLUMED input that I used when making the figure above is a filled in version of the following:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
phi: TORSION __FILL__=5,7,9,15
psi: TORSION __FILL__=7,9,15,17
pca: PCAVARS __FILL__=../data/pca-reference.pdb TYPE=__FILL__
PRINT __FILL__=phi,psi,pca.eig-1,path.spath FILE=colvar STRIDE=1
__FILL__ ARG=phi STRIDE=1 BASIN_LL1=-3 BASIN_UL1=-1 BASIN_LL2=1 BASIN_UL2=2 FILE=basin
```

I deployed the input above within the following python script that launches a large number of gromacs simulations:

```
def gen_trajectories( ntraj, plm ) :
    nc7eq, c7eq_colv, all_data = 0, np.zeros([0,5]), np.zeros([0,5])
    for i in range(ntraj) :
        !rm -rf ../Test && mkdir ../Test
        generate_gromacs_input( '../Test', plm, 10000000, 300 )
        !cd ../Test/ && gmx_mpi grompp -f md.mdp -c conf.pdb -p topol.top -maxwarn 2 &> /dev/null
        !cd ../Test/ && gmx_mpi mdrun --plumed plumed.dat &> /dev/null
        bfile = open('../Test/basin','r')
        with open( '../Test/basin', "r" ) as myfile :
            for line in myfile :
                if line.startswith("#! SET COMMITTED TO BASIN") : basin = line.split()[5]
        colv_data = np.loadtxt("../Test/colvar")
        if len(colv_data.shape)==1 : colv_data = np.reshape( colv_data, (1,5) )
        all_data = np.concatenate( (all_data, colv_data), axis=0 )
        if basin=="1" :
            c7eq_colv = np.concatenate( (c7eq_colv, colv_data), axis=0 )
            nc7eq = nc7eq + 1
    print( "NUMBER c7ax", ntraj-nc7eq, "c7eq", nc7eq )
    return c7eq_colv, all_data
p="'
YOUR PLUMED INPUT GOES HERE
"'
bas_data, all_data = gen_trajectories( ntraj, p )
```

Notice how the above script stores the CV values from trajectories that finished in the C7eq basin in the NumPy array called bas_data. We can use the data in this file to construct the (unnormalized) histograms of CV value that finished in C7eq and the total histogram. The script I used to construct these histograms is shown below. This script also shows how we can arrive at the final conditional probability distributions from masterclass21-6-rmsd-committor by dividing by the total number of configurations that moves to C7eq first by the total number of configuration that visited a point.

```
def histo( data, nbins, xmin, xmax ) :
    delr = ( xmax - xmin ) / nbins
    hist = np.zeros(nbins)
    for d in data :
        xbin = int( np.floor( (d-xmin) / delr ) )
        hist[xbin] = hist[xbin] + 1
    return hist / delr
def get_isocommitor( bas_data, full_data, nbins, xmin, xmax ) :
    bas_histo = histo( bas_data, nbins, xmin, xmax )
    full_histo = histo( full_data, nbins, xmin, xmax )
    for i in range(nbins) :
        if np.abs(full_histo[i])<1E-4 : bas_histo[i] = 0
        else : bas_histo[i] = bas_histo[i] / full_histo[i]
    return bas_histo
# This makes the isocommittor as a function of \f$\phi\f$:
```

```
commit = get_isocommitor( bas_data[:,1],  all_data[:,1], 30, -np.pi, np.pi )
```

Notice that the script above does not compute the error bars I included in masterclass21-6-rmsd-committor. **Your task in this exercise is to reproduce this figure with the error bars.** I generated the error bars in masterclass21-6-rmsd-committor by running 10 sets of 500 simulations. I constructed separate histograms from each of these batches of simulations and calculated the mean and variance from these ten sets of samples.

11.20.6.7.7 Conclusions The exercises in this section aimed to discuss the difference between collective variables and the reaction coordinate. I have argued that collective variables are developed by researchers thinking about coordinates that can distinguish between the important states. Reaction coordinates, meanwhile, are the actual pathways that reactions proceed along. These reaction coordinates are found by probing the data we get from simulations.

I am not convinced that the distinction between reaction coordinates and collective variables is very important. Every coordinate we have tried in the previous sections has allowed us to determine whether we are in the C7eq or the C7ax basin. My view is that when we do science, we impose structure on our results by asking questions. Our question has been to determine the relative free energies of the C7ax and C7eq basins in this exercise. To do this, we have to define what configurations are in the C7ax basin and what configurations are in the C7eq basin. The distinction between these two structures is an arbitrary figment of our imagination. If we use different variables, we might be defining these states differently, and we might get slightly different answers. We should not be surprised by the fact that we get different answers if we ask the question differently. The important thing is to develop an interesting question and to tell a good story about whatever you find out.

11.20.6.8 Exercise 4: Dealing with indistinguishability

Many researchers have used these rare event methods to study nucleation and crystal growth. Studying such problems introduces an additional challenge when designing collective variables as all the atoms are indistinguishable. You thus know before even running the simulation that there are multiple paths that connect the reactant (liquid) state and the product (solid) state. The nucleation, after all, can start with any atom in the simulation box. The first step in developing CVs for studying nucleation processes is thus to identify some order parameter that allows us to distinguish atoms that are in the solid state from atoms that are in the liquid state. The following input, once it is filled in will calculate some suitable order parameters for all the atoms in the system:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
UNITS NATURAL
coord: COORDINATIONNUMBER __FILL__=1-5184 __FILL__={CUBIC D_0=1.2 D_MAX=1.5}
cub: FCCUBIC __FILL__=1-5184 __FILL__={CUBIC D_0=1.2 D_MAX=1.5} ALPHA=27
fcub: MTRANSFORM_MORE DATA=__FILL__ __FILL__={SMAP R_0=0.45 D_0=0.0 A=8 B=8}
DUMPMULTICOLVAR __FILL__=coord STRIDE=1000 __FILL_=coord.xyz
DUMPMULTICOLVAR __FILL__=cub STRIDE=1000 __FILL__=cub.xyz
DUMPMULTICOLVAR __FILL__=fcub STRIDE=1000 __FILL__=fcub.xyz
```

You can run a short MD simulation on Lennard-Jonesium to compute those order parameter by using the following input (in) file for simplemd:

```
inputfile interface.xyz
outputfile output.xyz
temperature 0.6
tstep 0.005
friction 1
forcecutoff 2.5
listcutoff  3.0
nstep 10000
nconfig 100 trajectory.xyz
nstat   100 energies.dat
```

You can run this from a python notebook and read in the values of the order parameters by using the following script:

```
p = '''
INSERT YOUR PLUMED INPUT HERE
'''
inp = '''
INSERT YOUR SIMPLEMD INPUT HERE
'''
# Make a directory to run in
!rm -rf ../LJ-trajectories/First-run && mkdir ../LJ-trajectories/First-run
# Copy the initial configuration
```

```
!cp ../data/interface.xyz ../LJ-trajectories/First-run
# Output the plumed file
f = open("../LJ-trajectories/First-run/plumed.dat", 'w')
f.write(p)
f.close()
# Output the input to simplemd
f = open("../LJ-trajectories/First-run/in", 'w')
f.write(inp)
f.close()
# Now run PLUMED
!cd ../LJ-trajectories/First-run/ && plumed simplemd < in &> /dev/null
# Read in the various order parameters (N.B. you can almost certainly write better python here)
!grep X ../LJ-trajectories/First-run/coord.xyz | awk '{print $5}' > ../LJ-trajectories/First-run/coord.dat
coord = np.loadtxt("../LJ-trajectories/First-run/coord.dat")
!grep X ../LJ-trajectories/First-run/cub.xyz | awk '{print $5}' > ../LJ-trajectories/First-run/cub.dat
cub = np.loadtxt("../LJ-trajectories/First-run/cub.dat")
!grep X ../LJ-trajectories/First-run/fcub.xyz | awk '{print $6}' > ../LJ-trajectories/First-run/fcub.dat
fcub = np.loadtxt("../LJ-trajectories/First-run/fcub.dat")
```

A visualization of the order parameters for all the atoms can then be produced using chemiscope, which allows you to see the value of all the individual atom order parameters.

```
import ase
import ase.io
from chemiscope import write_input
# Read in the trajectory
traj = ase.io.read('../LJ-trajectories/First-run/trajectory.xyz',':')
# This constructs the dictionary of properties for chemiscope
properties = {
    "coord": {
        "target": "atom",
        "values": coord,
        "description": "Coordination number of atom",
    },
    "cub": {
        "target": "atom",
        "values": cub,
        "description": "FCCCUBIC order parameter",
    },
    "fcub": {
        "target": "atom",
        "values": fcub,
        "description": "Transformed FCCUBIC order parameter",
    },
}
# This generates our chemiscope output
write_input("fccubic_chemiscope.json.gz", cutoff=1.5, frames=traj, properties=properties )
```

Put all the above together and see if you can produce a chemiscope representation like the one below

Chemiscope is invaluable for determining whether our order parameter is good at distinguishing the atoms within the solid from those within the liquid. If you visualize the FCCCUBIC or the transformed values of these parameters in the example above, you see that roughly half of the atoms are solid-like, and nearly half of the atoms are liquid-like. Notice also that we can use the dimensionality reduction that were discussed in Exercise 4: Path collective variables and clustering algorithms to develop these atomic order parameters. Generating the data to analyze using these algorithms is easier because we can get multiple sets of coordinates to analyze from each trajectory frame. The atoms are, after all, indistinguishable.

Once we have identified an order parameter we can analyse the distribution of values that it takes by using an input like the one shown below:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-6.txt
UNITS NATURAL
coord: COORDINATIONNUMBER __FILL__=1-5184 __FILL__={CUBIC D_0=1.2 D_MAX=1.5}
cub: FCCUBIC __FILL__=1-5184 __FILL__={CUBIC D_0=1.2 D_MAX=1.5} ALPHA=27
fcub: MTRANSFORM_MORE DATA=__FILL__ __FILL__={SMAP R_0=0.45 D_0=0.0 A=8 B=8}
coord_histo: HISTOGRAM __FILL__=coord STRIDE=10 __FILL__=2 GRID_MAX=15 __FILL__=100 __FILL__=DISCRETE
cub_histo: HISTOGRAM __FILL__=cub STRIDE=10 GRID_MIN=-1 __FILL__=1 __FILL__=100 __FILL__=DISCRETE
fcub_histo: HISTOGRAM __FILL__=fcub STRIDE=10 __FILL__=0 GRID_MAX=1 __FILL__=100 __FILL__=DISCRETE
__FILL__ __FILL__=coord_histo FILE=coord_histo.dat
__FILL__ __FILL__=cub_histo FILE=cub_histo.dat
__FILL__ __FILL__=fcub_histo FILE=fcub_histo.dat
```

The resulting distributions of order parameter values should look like this:

Try to use the information above to reproduce this plot. Notice that the distribution of order parameters for the FCC Cubic order parameters is bimodal. There is a peak corresponding to the value that this quantity takes in the box's liquid part. The second peak then corresponds to the value that the quantity takes in the box's solid part. The same cannot be said for the coordination number by contrast. Notice, last of all, that by transforming the values above we have an order parameter that is one for atoms that are within the solid and zero otherwise.

It is tempting to argue that the number of solid particles is equal to the number of atoms that have an order parameter that is greater than some threshold. A better way to calculate the number of solid particles, however, is to proceed as follows:

- Run simulations of the solid and liquid under the same thermodynamic conditions.
- Calculate the average values per atom value of the order parameter in these simulations of the solid ϕ_s and liquid ϕ_l.
- Calculate the number of solid n_l and liquid atoms n_l by solving the following pair of simultaneous equations $N = n_s + n_l$ and $\Phi = n_s\phi_s + n_l\phi_l$, where N is the total number of atoms. Φ, meanwhile, is given by:

$$\Phi = \sum_{i=1}^{N} \phi_i$$

The sum runs over all the atoms and where ϕ_i is the order parameter for the ith atom in the system.
This procedure works because Φ is an extensive quantity and is thus additive. If we have a system that contains a aolid liquid interface we can express the value of any extensive quantity, Ψ using:

$$\Psi = n_s\psi_s + n_l\psi_l + \psi_i$$

where ψ_s and ψ_l is the average value per atom value of the quantity ψ in the solid and liquid phases. ψ_i, meanwhile, is the so-called surface excess term which measures the contribution that the presence of the interface makes to the value of Ψ. When we find n_s by solving the simultaneous equations above, we assume that this excess term is zero.
To complete this exercise, you should run simulations of the solid, liquid and interface. You should use the ensemble average of the order parameter for your solid and liquid simulations to make a graph like the one below that shows the number of atoms of solid as a function of time for the system that contains the interface. The number of solid atoms has been determined by solving the simultaneous equations using the theory above. Notice that there are input configurations for the solid, liquid and interface systems in the data directory of the GitHub repository. The solid configuration is in solid.xyz. The liquid is in liquid.xyz, and the interface is in interface.xyz. The final result you get will look something like this:
Notice that the green line is the least noisy of these curves. The reason the line is less noisy is connected to the fact that you have the clearest delineation between solid and liquid atoms when you use this order parameter.
NB. We are running at constant volume because we are keeping things simple and using simplemd. If you are studying nucleation using these techniques, you should run at constant pressure.

11.20.6.9 Exercise 7: Applying these ideas in your research

The exercises in this tutorial have introduced you to the following ideas:

- Using chemiscope to visualize trajectories and collective variables.
- Using dimensionality reduction algorithms
- Using Path collective variables
- Clustering trajectories
- Dealing with indistinguishable atoms

What I would like to do in the follow-up session is to think about how you can use these ideas as you do your research. The first step in using any of these questions is asking a research question that is amenable to answering using one of these methods. In this final exercise, I would like you to think about using these methods in your research. The first step in doing so is asking yourself the question **what am I trying to find out?** The clearer you make this question, the easier you will find the subsequent research project. With this in mind, here are some examples of terrible research questions:

- Understanding protein-ligand interactions using machine learning algorithms.
- Understanding the mechanism via which limescale forms in pipes

- Understanding protein folding using machine learning.
- Studying transport through membrane proteins.
- Understanding the catalytic mechanism of iron in ammonia formation.

These research questions are terrible because the final result that you will obtain has not been specified. It is unclear when the project will be finished, or what success in this project looks like. With this in mind, consider the following alternative questions, which each have a clear goal. Each of the goals below maps on to one of the terrible research questions above:

- Calculate the free energy change when ligand A binds to protein B.
- Calculate the free energy change when single calcium and carbonate ions bind to copper surfaces.
- Calculate the free energy change when protein A transitions to its folded state.
- Calculate the free energy change as a sodium ion moves through a sodium channel protein.
- Calculate the free energy change when nitrogen, hydrogen and ammonia gas bind to iron surfaces.

For each of the questions above, it is clear what success will look like – you will have a converged estimate of the free energy difference. We have shown you how such estimates can be extracted in previous masterclasses and what it looks like when such calculations go wrong. Answering these research questions is thus simply a matter of applying what you have learned. Notice, also, how for each of the questions above, it is clear what you can do in the first step:

- Run a parallel tempering metadynamics simulation with the distance between the centres of mass of the protein and the ligand as a CV.
- Run a metadynamics simulation of a slab of copper in contact with an aqueous solution containing a single calcium ion. Use the z component of the vector connecting the calcium ion and the centre of mass of the copper slab as the CV. Run a metadynamics simulation using the distance from the protein's folded state as the collective variable.
- Run a metadynamics simulation of the sodium in the channel protein. Use the z position of the sodium relative to the centre of the channel protein as a CVs.
- Run a metadynamics simulation of a slab of iron in contact with a simulation box that is empty except for one nitrogen molecule. Use the z component of the vector connecting the centre of mass of the nitrogen and the centre of mass of the iron slab as the CV.

These simulations may or may not give you the desired free energy difference. You will at least get some trajectory data from them, however. You can then analyze the trajectories using a machine learning algorithm or similar to refine your approach. This refining stage is where the experience of your supervisor should be invaluable.
With all this in mind, think about your research project. Try to develop a research question to work on and an approach that you can use to tackle this question. Your question should not be some grand challenge.
It should be something simple that you know how to do. You do not need to run these simulations. I want you to think about the research question to discuss them at the follow-up session. I want you to think about how you can use these methods in your research as if you are not doing so, attending these masterclasses is pointless as you will never really apply these methods.

11.21 PLUMED Masterclass 21.7: Optimizing PLUMED performances

Authors

Giovanni Bussi and Max Bonomi

Date

April 21, 2021

11.21.1 Aims

In this Masterclass, we will discuss how to monitor and optimize the performances of a PLUMED-enhanced MD simulation.

11.21.2 Objectives

Once you have completed this Masterclass you will be able to:

- measure the performances of a PLUMED-enhanced simulation using the DEBUG action;
- optimize a metadynamics simulation;
- optimize GROMACS parallelization;
- define complex CVs in the PLUMED input file in a computationally efficient manner.

11.21.3 Setting up PLUMED

For this masterclass you will need versions of PLUMED and GROMACS that are compiled using the MPI library. Thus follow the instructions that are reported for PLUMED Masterclass 21.5: Simulations with multiple replicas. Natively-compiled GROMACS and PLUMED will be significantly faster than the conda versions that we are providing. Since we are focusing on performance here, this might be the right time to learn how to install them on your own.

11.21.4 Resources

The data needed to execute the exercises of this Masterclass can be found on GitHub. You can clone this repository locally on your machine using the following command:

```
git clone https://github.com/plumed/masterclass-21-7.git
```

Note

All the exercises were tested with PLUMED version 2.7.0 and GROMACS 2019.6

11.21.5 Exercises

Notice that the results of these exercises might depend on the details of the hardware and software you are using. It is thus instructive to test them on different architectures, or with different PLUMED or GROMACS versions.

11.21.5.1 Measuring performance

In these exercises you will have to maximise the performance of your simulation. When using GROMACS, the common way to report performances is to check at how many ns/day the simulation can produce. At the end of the log file you should find lines like these ones

```
               Core t (s)   Wall t (s)        (%)
       Time:      141.898       11.825     1200.0
                 (ns/day)     (hour/ns)
Performance:       14.628        1.641
```

Here, the higher the ns/day the faster your simulation will be. Another important information it the wallclock time, that indicates how many seconds elapsed since the start of your simulation. If you then run the same input file using PLUMED as well you will instead see something like this:

```
               Core t (s)   Wall t (s)        (%)
       Time:      170.519       14.210     1200.0
                 (ns/day)     (hour/ns)
Performance:       12.173        1.972
Finished mdrun on rank 0 Thu Apr 22 16:37:32 2021

PLUMED:                                              Cycles        Total      Average      Minimum      Maxim
PLUMED:                                                   1     2.009397     2.009397     2.009397     2.0093
PLUMED: 1 Prepare dependencies                         1001     0.004042     0.000004     0.000003     0.0000
PLUMED: 2 Sharing data                                 1001     0.151432     0.000151     0.000032     0.0199
```

```
PLUMED: 3 Waiting for data                          1001     0.021681     0.000022     0.000005     0.0130
PLUMED: 4 Calculating (forward loop)                1001     1.392920     0.001392     0.000349     0.0322
PLUMED: 5 Applying (backward loop)                  1001     0.303819     0.000304     0.000089     0.0255
PLUMED: 6 Update                                    1001     0.003255     0.000003     0.000002     0.0001
```

Notice that:

- The performance has decreased.
- The wallclock time has increased.
- PLUMED is reporting some information about the time spent using it.

Notice that the total time spent using PLUMED is approximately equal to the increment in the wallclock time. This might be different when using a GPU (and indeed the increment in the wallclock time should be smaller). This extra time measures the cost of using PLUMED. The goal of this Masterclass is to understand how to decrease this extra time without impacting the result of your simulation.
Notice that PLUMED gives a breakdown of the time spent. Some rows correspond to communication, and might become important if you run with a lot of MPI processes. Usually, most of the time is spent in the forward loop, where collective variables are calculated.
You can obtain a more detailed breakdown adding to your input this line:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aa-masterclass-21-7.txt
DEBUG DETAILED_TIMERS
```

The tail of the log will then look like this:

```
PLUMED:                                            Cycles        Total      Average      Minimum      Maxim
PLUMED:                                                 1     2.055374     2.055374     2.055374     2.0553
PLUMED: 1 Prepare dependencies                       1001     0.004168     0.000004     0.000003     0.0000
PLUMED: 2 Sharing data                               1001     0.118547     0.000118     0.000033     0.0160
PLUMED: 3 Waiting for data                           1001     0.008541     0.000009     0.000005     0.0000
PLUMED: 4 Calculating (forward loop)                 1001     1.449592     0.001448     0.000375     0.0342
PLUMED: 4A 0 @0                                      1001     0.004118     0.000004     0.000002     0.0001
PLUMED: 4A 4 d                                       1001     0.293253     0.000293     0.000011     0.0235
PLUMED: 4A 5 cn                                      1001     0.810816     0.000810     0.000308     0.0340
PLUMED: 4A 7 @7                                        11     0.000045     0.000004     0.000003     0.0000
PLUMED: 4A 8 @8                                      1001     0.225797     0.000226     0.000016     0.0210
PLUMED: 4A 9 @9                                      1001     0.066718     0.000067     0.000010     0.0160
PLUMED: 5 Applying (backward loop)                   1001     0.349111     0.000349     0.000120     0.0250
PLUMED: 5A 0 @9                                      1001     0.003697     0.000004     0.000003     0.0000
PLUMED: 5A 1 @8                                      1001     0.002814     0.000003     0.000002     0.0000
PLUMED: 5A 2 @7                                        11     0.000023     0.000002     0.000002     0.0000
PLUMED: 5A 4 cn                                      1001     0.102383     0.000102     0.000063     0.0134
PLUMED: 5A 5 d                                       1001     0.008055     0.000008     0.000006     0.0001
PLUMED: 5A 9 @0                                      1001     0.002417     0.000002     0.000002     0.0000
PLUMED: 5B Update forces                             1001     0.191250     0.000191     0.000020     0.0239
PLUMED: 6 Update                                     1001     0.003271     0.000003     0.000002     0.0001
```

Notice that for both the forward and the backward loop you are shown with a breakdown of the time required for each of the actions included in the input file. You should recognize the name of the actions that you defined, whereas unnamed actions are referred to with a generic `@`-number that corresponds to their position in the input file. From this detailed log you can also appreciate how often each action has been performed. PLUMED tries to optimize this, e.g., only computing variables when they are needed. This breakdown is very useful to know where you should direct your effort.

11.21.5.2 Exercise 1: Dissociation of NaCl in water

As a first test system we will consider a single Na Cl pair in a water box. The two ions are expected to attract each other, so that the bound conformation should be stable. The free energy as a function of the distance between the two ions should thus have a minimum followed by a barrier. At larger distances, it is not expected to become flat but rather to decrease as -2∗kBT log d, due to the entropic contribution, and then to grow again when reaching the boundaries of the simulation box. In this exercise we will compute the free-energy as a function of the distance between the two ions. As collective variables we will use the distance between the two ions as well as the number of water oxygens that are coordinated with the sodium ion.
In the `data/exercise1` folder of the `GitHub` repository of this Masterclass, you will find a `topol.tpr` file, which is needed to perform a MD simulation of this system with GROMACS. You can then create a `plumed.dat` file like this one as a starting point:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aa-masterclass-21-7.txt
# vim:ft=plumed
NA: GROUP ATOMS=1
CL: GROUP ATOMS=2
WAT: GROUP ATOMS=3-8544:3
d: DISTANCE ATOMS=NA,CL
cn: COORDINATION GROUPA=1 GROUPB=WAT R_0=0.3
PRINT ARG=d,cn STRIDE=100 FILE=COLVAR
METAD ARG=d,cn SIGMA=0.05,0.1 HEIGHT=0.1 PACE=10 BIASFACTOR=5
```

As usual, you can run a simulation with the following command

```
gmx_mpi mdrun -plumed plumed.dat -nsteps 100000
```

For the first three points, you can run relatively short simulations, so choose nsteps based on what you can quickly test on your machine. For the fourth point instead you will need to reach convergence. A time on the order of a ns should be more the sufficient.

11.21.5.3 Exercise 1a: Optimizing the calculation of a metadynamics bias

For this first point, we will focus on the calculation and update of the metadynamics bias. We will try to have a simulation that runs faster without changing significantly the result. Check the manual of METAD and find out how to speed up this calculation. A few hints:

- What matters is the deposition rate (that is: height/pace). Increasing the pace and height by the same factor should not change the result significantly.
- Grids will make calculation faster (especially for long runs) but update slower.

Notice that these changes are not expected to impact the convergence of the algorithm. Thus, you do not need a converged simulation to measure the impact on performances.

11.21.5.4 Exercise 1b: Optimizing the calculation of a coordination number

For this second point, we will focus on the calculation of one of the two biased collective variables, namely the coordination of the Na ion with water oxygens. We will try to have a simulation that runs faster without changing significantly the result. Check the manual of COORDINATION and find out how to speed up this calculation. A few hints:

- Neighbor lists might help, but be very careful with parameters.
- Starting with PLUMED v2.7 the construction of neighbor lists is parallelized. Performances might thus be very different if you test your input with PLUMED v2.6 or earlier.

Notice that these changes are not expected to impact the convergence of the algorithm. Thus, you do not need a converged simulation to measure the impact on performances.

11.21.5.5 Exercise 1c: Optimizing GROMACS parallelization

For this third point, we will try to make sure that GROMACS runs at its maximum speed. For this you will have to check GROMACS manual. A few hints:

- Based on the number of processors in your computer, play with the number of OpenMP threads and of MPI processes.
- Check if the `-pin on` option improves performances.

Notice that these changes are not expected to impact the convergence of the algorithm. Thus, you do not need a converged simulation to measure the impact on performances.

11.21.5.6 Exercise 1d: Optimizing metadynamics parameters

We will now make modifications to the algorithm so as to be able to arrive to the same result running a shorter simulation. Try to play with METAD parameters and see if you can improve them. A few hints:

- Changing hills width might affect that speed at which you fill free-energy basins.

- Limiting the domain of the collective variables that you explore might help, if you can predict what happens in the portion of the domain that the simulation does not explore.
- You can even try to reduce the number of CVs.

Notice that these changes are expected to impact the convergence of the algorithm. Thus, you do not need a converged simulation to measure the impact on performances, and you have to make sure that the statistical accuracy is comparable.

11.21.5.7 Exercise 1e: Computing the binding free energy

As a final step, analyze the simulations performed so far to compute the standard binding free energy between the two species. Notice that even though this is defined at 1M concentration, the calculations that you are running are actually at infinite dilution.

11.21.5.8 Exercise 2: Folding of the C-terminal domain (CTD) of the RfaH virulence factor

In this exercise, we will work with the C-terminal domain (CTD) of the RfaH virulence factor from *Escherichia coli* introduced in PLUMED Masterclass 21.4: Metadynamics (see Exercise 6: A 'real-life' application). This part of the system, which we refer to as RfaH-CTD, undergoes a dramatic conformational transformation from -barrel to -helical, which is stabilized by the N-terminal domain of the RfaH virulence factor (see Fig. masterclass-21-7-RfaH-CTD-fig). RfaH-CTD is simulated using a simplified, structure-based potential, called SMOG. The SMOG energy function has been designed to have two local minima corresponding to the -barrel and -helical states of RfaH-CTD. To achieve this goal, the SMOG energy function promotes native contacts, i.e. interactions that are present in the native structure(s). When using structure-based force fields, a function of the coordinates that is correlated with the energy of the system, such as the total number of native contacts, has been shown to be a good CV for enhanced-sampling simulations. Unfortunately, these types of CVs often involve a large number of atoms and are therefore computationally expensive to calculate at every step of the simulation. In this exercise, we will learn how to write and optimize these types of CVs.
In the `data/exercise2` folder of the GitHub repository of this Masterclass, you will find:

- two PDB files of RfaH-CTD in the -barrel (`stateA.pdb`) and -helical (`stateB.pdb`) states;
- a `topol.tpr` file, which is needed to perform a MD simulation of this system with GROMACS;
- a template PLUMED input file (`plumed.dat`) to perform a metadynamics simulation using the total number of native contacts as CV. These contacts are defined using only the -barrel conformation, which is the most populated state in the conditions we are simulating.

The provided PLUMED input file looks as follows:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aa-masterclass-21-7.txt
# reconstruct molecule
WHOLEMOLECULES ENTITY0=1-330

# CA-RMSDs from the two reference conformations
# useful for monitoring the distance from the two metastable states
rmsd_A: RMSD REFERENCE=stateA.pdb TYPE=OPTIMAL NOPBC
rmsd_B: RMSD REFERENCE=stateB.pdb TYPE=OPTIMAL NOPBC

# list of 379 distances between atoms that are closer
# than 0.6 nm in the reference PDB file (stateA.pdb, -barrel state)
d1: DISTANCE ATOMS=1,110 NOPBC
d2: DISTANCE ATOMS=1,115 NOPBC
# __FILL__
d379: DISTANCE ATOMS=239,265 NOPBC

# list of 379 switching functions to define a contact from the distance between two atoms
c1: CUSTOM FUNC=1-erf(x^4) ARG=d1 PERIODIC=NO
c2: CUSTOM FUNC=1-erf(x^4) ARG=d2 PERIODIC=NO
# __FILL__
c379: CUSTOM FUNC=1-erf(x^4) ARG=d379 PERIODIC=NO

# sum of switching functions = total number of contacts
cv: COMBINE ARG=c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19,c20,c21,c22,c23,c24,c25,c26
```

```
# metadynamics using "cv"
metad: METAD ARG=cv ...
 # Deposit a Gaussian every 500 time steps, with initial height
 # equal to 1.2 kJ/mol and bias factor equal to 60
 PACE=500 HEIGHT=1.2 BIASFACTOR=60
 # Gaussian width (sigma) based on the CV fluctuations in unbiased run
 SIGMA=10.0
 # Gaussians will be written to file
 FILE=HILLS
...

# print useful stuff
PRINT ARG=cv,rmsd_A,rmsd_B,metad.bias STRIDE=500 FILE=COLVAR
```

The objectives of this exercise are to:

1. optimize the distance-based CV provided in the template PLUMED input file. **The user should report the speedup obtained with the optimized CV with respect to the one defined in the provided input file**;
2. optimize the performances of well-tempered metadynamics, as done in Exercise 1a: Optimizing the calculation of a metadynam and Exercise 1d: Optimizing metadynamics parameters;
3. evaluate the stability of the -barrel state of RfaH-CTD (with error estimate). Have a look at PLUMED Masterclass 21.4: Metadynamics for more info.

Suggestions:

- the users should consult the PLUMED manual in order to optimize the proposed CV;
- the COORDINATION CV can be defined using a custom switching function, whose calculation can be made faster by using `AsmJit`. For more information, have a look `here`;
- try `Multiple time stepping` to apply the metadynamics bias every 2 or 4 steps using the `STRIDE` option on the METAD line.

Please keep in mind that:

- SMOG is significantly less computational demanding than all-atoms, explicit solvent force fields. However, the simulation of this system might take a few hours, so allocate enough time to complete this exercise;
- due to the special nature of the force field, please execute GROMACS using the following command: `gmx mdrun -plumed plumed.dat -ntomp 4 -noddcheck`. You can adjust the number of CPU cores you want to use (here 4, OpenMP parallelization), based on the available resources. The system is not particularly big, therefore using a large number of cores might be inefficient.

11.22 Advanced Methods in MD 2023: Metadynamics simulations with PLUMED

11.22.1 Aim

The aim of this tutorial is to train users to perform and analyze metadynamics simulations with PLUMED. This tutorial has been prepared by Max Bonomi (stealing a lot of material from other tutorials) for the `Advanced Methods in MD 2023` workshop, held in Copenhagen (Denmark) on December 11-12, 2023.

11.22.2 Objectives

Once this tutorial is completed users will be able to:

- Write the PLUMED input file to perform metadynamics simulations.
- Calculate the free energy as a function of the metadynamics collective variables.
- Unbias metadynamics simulations.
- Estimate the error in the reconstructed free energies using block analysis.
- Assess the convergence of metadynamics simulations.

11.22.3 Software installation

For this exercise we will need PLUMED 2.9.0 and a special version of GROMACS 2020.7 patched with PLUMED. The two software can be installed using the instructions reported here.

11.22.4 Resources

The TARBALL for this tutorial contains the following files:

- `diala.pdb`: a PDB file for alanine dipeptide in vacuo.
- `topol.tpr`: a GROMACS run (binary) file to perform MD simulations of alanine dipeptide.
- `do_block_fes.py`: a python script to perform error analysis of metadynamics simulations.

After dowloading the compressed archive to your local machine, you can unpack it using the following command:

```
tar xvzf advanced-methods.tar.gz
```

Once unpacked, all the files can be found in the `advanced-methods` directory. To keep things clean, it is recommended to run each exercise in a separate sub-directory that you can create inside `advanced-methods`.

11.22.5 Introduction

PLUMED can be used to compute collective variables (CVs) on a pre-calculated trajectory. However, PLUMED is most often use to add forces on the CVs during a MD simulation, for example, in order to accelerate sampling. To this aim, we have implemented a variety of possible biases acting on CVs. The complete documentation for all the biasing methods available in PLUMED can be found at the Bias page. In the following we will learn how to use PLUMED to perform and analyze a metadynamics simulation. Here you can find a brief recap of the metadynamics theory.
In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t \to \infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t \to \infty) = -\frac{\Delta T}{T + \Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent

of free-energy exploration: $\Delta T = 0$ corresponds to standard MD, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T + \Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135] [136].
We will play with a toy system, alanine dipeptide simulated in vacuo using the AMBER99SB-ILDN force field (see Fig. advanced-methods-ala-fig). This rather simple molecule is useful to understand data analysis and free-energy methods. This system is a nice example because it presents two metastable states separated by a high free-energy barrier. It is conventional use to characterize the two states in terms of Ramachandran dihedral angles, which are denoted with ϕ (phi) and ψ (psi) in Fig. advanced-methods-transition-fig.

11.22.6 Exercises

11.22.6.1 Exercise 1: My first metadynamics simulation

In this exercise we will setup and perform a well-tempered metadynamics run using the backbone dihedral ϕ as collective variable. During the calculation, we will also monitor the behavior of the other backbone dihedral ψ.
Here you can find a sample `plumed.dat` file that you can use as a template. Whenever you see an highlightedF↩ ILL string, this is a string that you must replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaa-advanced-methods.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=diala.pdb

# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
# you might want to use MOLINFO shortcuts
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
# here also you might want to use MOLINFO shortcuts
psi: TORSION ATOMS=__FILL__

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 500 time steps, with initial height equal to 1.2 kJ/mol
  PACE=500 HEIGHT=1.2
# The bias factor should be wisely chosen
  BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
  SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
...

# Print both collective variables on COLVAR file every 10 steps
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=__FILL__
```

The syntax for the command METAD is simple. The directive is followed by a keyword `ARG` followed by the labels of the CVs on which the metadynamics bias potential will act. The keyword `PACE` determines the stride of Gaussian deposition in number of time steps, while the keyword `HEIGHT` specifies the height of the Gaussian. For each CVs, one has to specify the width of the Gaussian by using the keyword `SIGMA`. Gaussian will be written to the file indicated by the keyword `FILE`.
In this example, the bias potential will be stored on a grid, whose boundaries are specified by the keywords `GRI↩ D_MIN` and `GRID_MAX`. Notice that you can provide either the number of bins for every CV (`GRID_BIN`) or the desired grid spacing (`GRID_SPACING`). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither `GRID_BIN` nor `GRID_SPACING`) and if Gaussian width is fixed, PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.
Once your `plumed.dat` file is complete, you can run a 10-ns long metadynamics simulations with the following command:

```
> gmx_mpi mdrun -s topol.tpr -nsteps 5000000 -plumed plumed.dat
```

During the metadynamics simulation, PLUMED will create two files, named COLVAR and HILLS. The COLVAR file contains all the information specified by the PRINT command, in this case the value of the backbone dihedrals ϕ and ψ every 10 steps of simulation. We can use gnuplot to visualize the behavior of the metadynamics CV ϕ during the simulation:

```
gnuplot> p "COLVAR" u 1:2
```

By inspecting Figure advanced-methods-phi-fig, we can see that the system is initialized in one of the two metastable states of alanine dipeptide. After a while (t=0.1 ns), the system is pushed by the metadynamics bias potential to visit the other local minimum. As the simulation continues, the bias potential fills the underlying free-energy landscape, and the system is able to diffuse in the entire phase space.
The HILLS file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content:

```
#! FIELDS time phi sigma_phi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
```

The line starting with FIELDS tells us what is displayed in the various columns of the HILLS file: the simulation time, the instantaneous value of ϕ, the Gaussian width and height, and the bias factor. We can use the HILLS file to visualize the decrease of the Gaussian height during the simulation, according to the well-tempered recipe:
If we look carefully at the scale of the y-axis, we will notice that in the beginning the value of the Gaussian height is higher than the initial height specified in the input file, which should be 1.2 kJ/mol. In fact, this column reports the height of the Gaussian scaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy.

Warning

> The fact that the Gaussian height is decreasing to zero should not be used as a measure of convergence of your metadynamics simulation!

11.22.6.2 Exercise 2: Estimating the free energy as a function of the metadynamics CVs

One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills can be used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.

To calculate the free energy as a function of ϕ, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called fes.dat in which the free-energy surface as function of ϕ is calculated on a regular grid. One can modify the default name for the free-energy file, as well as the boundaries and bin size of the grid, by using the following sum_hills options:

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

The result should look like this:
To give a preliminary assessment of the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The sum_hills option --stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option --mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --stride 100 --mintozero
```

one free energy is calculated every 100 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. The resulting plot should look like the following:
These two qualitative observations:

1. the system is diffusing rapidly in the entire CV space (Figure advanced-methods-phi-fig)
2. the estimated free energy does not significantly change as a function of time (Figure advanced-methods-metad-phifest-fig)

suggest that the simulation **might** be converged.

Warning

The two conditions listed above are necessary, but not sufficient to declare convergence. For a quantitative analysis of the convergence of metadynamics simulations, please have a look below at Exercise 4: Estimating the error in free energies using block-analysis.

11.22.6.3 Exercise 3: Reweighting (unbiasing) a metadynamics simulation

In the previous exercise we biased ϕ and computed the free energy as a function of the same variable directly from the metadynamics bias potential using the sum_hills utility. However, in many cases you might decide which variable should be analyzed *after* having performed a metadynamics simulation. For example, you might want to calculate the free energy as a function of CVs other than those biased during the metadynamics simulation, such as the dihedral ψ. At variance with standard MD simulations, you cannot simply calculate histograms of other variables directly from your metadynamics trajectory, because the presence of the metadynamics bias potential has altered the statistical weight of each frame. To remove the effect of this bias and thus be able to calculate properties of the system in the unbiased ensemble, you must reweight (unbias) your simulation.
There are multiple ways to calculate the correct statistical weight of each frame in your metadynamics trajectory and thus to reweight your simulation. For example:

1. weights can be calculated by considering the time-dependence of the metadynamics bias potential [3];
2. weights can be calculated using the metadynamics bias potential obtained at the end of the simulation and assuming a constant bias during the entire course of the simulation [50].

In this exercise we will use the second method, which resembles the umbrella-sampling reweighting approach. In order to compute the weights we will use the driver tool.
First of all, you need to prepare a `plumed_reweight.dat` file that is identical to the one you used for running your metadynamics simulation except for a few modifications. First, you need to add the keyword `RESTART=YES` to the METAD command. This will make this action behave as if PLUMED was restarting, i.e. PLUMED will read from the `HILLS` file the Gaussians that have previously been accumulated. Second, you need to set the Gaussian `HEIGHT` to zero and the `PACE` to a large number. This will actually avoid adding new Gaussians (and even if they are added they will have zero height). Finally, you need to modify the PRINT statement so that you write every frame and that, in addition to `phi` and `psi`, you also write `metad.bias`. You might also want to change the name of the output file to `COLVAR_REWEIGHT`.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaa-advanced-methods.txt
# Activate MOLINFO functionalities
MOLINFO STRUCTURE=diala.pdb

__FILL__ # here goes the definitions of the phi and psi CVs

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 10000000 time steps (never!), with initial height equal to 0.0 kJ/mol
  PACE=10000000 HEIGHT=0.0 # <- this is the new stuff!
# The bias factor should be wisely chosen
  BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
  SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
# Say that METAD should be restarting (= reading an existing HILLS file)
  RESTART=YES # <- this is the new stuff!
...

# Print out the values of phi, psi and the metadynamics bias potential
# Make sure you print out the 3 variables in the specified order at every step
PRINT ARG=__FILL__ FILE=COLVAR_REWEIGHT STRIDE=__FILL__  # <- also change this one!
```

Then run the driver tool using this command:

```
> plumed driver --mf_xtc traj_comp.xtc --plumed plumed_reweight.dat --kt 2.494339
```

Notice that you have to specify the value of k_BT in energy units. While running your simulation this information was communicated by the MD code.
As a result, PLUMED will produce a new `COLVAR_REWEIGHT` file with one additional column containing the metadynamics bias potential $V(s)$ calculated using all the Gaussians deposited along the entire trajectory. The beginning of the file should look like this:

```
#! FIELDS time phi psi metad.bias
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
 0.000000 -1.497988 0.273498 110.625670
 1.000000 -1.449714 0.576594 110.873141
 2.000000 -1.209587 0.831417 109.742353
 3.000000 -1.475975 1.279726 110.752327
```

You can easily obtain the weight w of each frame using the expression $w \propto \exp\left(\frac{V(s)}{k_BT}\right)$ (umbrella-sampling-like reweighting). At this point, you can read the `COLVAR_REWEIGHT` file using, for example, a python script and compute a weighted histogram. Alternatively, if you want PLUMED to do the weighted histograms for you, you can add the following lines at the end of the `plumed_reweight.dat` file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaa-advanced-methods.txt
# Use the metadynamics bias as argument
as: REWEIGHT_BIAS ARG=__FILL__

# Calculate histograms of phi and psi dihedrals every 50 steps
# using the weights obtained from the metadynamics bias potentials (umbrella-sampling-like reweighting)
# Look at the manual to understand the parameters of the HISTOGRAM action!
hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=50 BANDWIDTH=0.05 LOGWEIGHTS=as
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=50 BANDWIDTH=0.05 LOGWEIGHTS=as

# Convert histograms h(s) to free energies F(s) = -kBT * log(h(s))
ffphi: CONVERT_TO_FES GRID=hhphi
ffpsi: CONVERT_TO_FES GRID=hhpsi

# Print out the free energies F(s) to file once the entire trajectory is processed
DUMPGRID GRID=ffphi FILE=ffphi.dat
DUMPGRID GRID=ffpsi FILE=ffpsi.dat
```

and plot the result using `gnuplot`:

```
gnuplot> p "ffphi.dat" u 1:2 w lp
gnuplot> p "ffpsi.dat" u 1:2 w lp
```

You can now compare the free energies as a function of ϕ calculated:

1. directly from the metadynamics bias potential using sum_hills as done in Exercise 2: Estimating the free energy as a function of
2. using the reweighting procedure introduced in this exercise.

The results should be identical (see Fig. advanced-methods-fescomp-fig).
.

11.22.6.4 Exercise 4: Estimating the error in free energies using block-analysis

In the previous exercise, we calculated the *final* bias $V(s)$ on the entire metadynamics trajectory and we used this quantity to calculate the correct statistical weight of each frame that we need to reweight the biased simulation. In this exercise, we will see how this information can be used to calculate the error in the reconstructed free energies and assess whether our simulation is converged or not. Let's first calculate the un-biasing weights $w \propto \exp\left(\frac{V(s)}{k_BT}\right)$ from the `COLVAR_REWEIGHT` file obtained at the end of Exercise 3: Reweighting (unbiasing) a metadynamics simulation:

```
# Find maximum value of bias to avoid numerical errors when calculating the un-biasing weights
bmax=`awk 'BEGIN{max=0.}{if($1!="#!" && $4>max)max=$4}END{print max}' COLVAR_REWEIGHT`

# Print phi values and un-biasing (un-normalized) weights
awk '{if($1!="#!") print $2,exp(($4-bmax)/kbt)}' kbt=2.494339 bmax=$bmax COLVAR_REWEIGHT > phi.weight
```

If you inspect the `phi.weight` file, you will see that each line contains the value of the dihedral ϕ along with the corresponding (un-normalized) weight w for each frame of the metadynamics trajectory:

```
0.907347 0.0400579
0.814296 0.0169656
1.118951 0.0651276
1.040781 0.0714174
1.218571 0.0344903
1.090823 0.0700568
1.130800 0.0622998
```

At this point we can apply the block-analysis technique (for more info about the theory, have a look at Trieste tutorial: Averaging, histograms and block analysis) to calculate the average free energy across the blocks and the error as a function of block size. For your convenience, you can use the `do_block_fes.py` python script to read the `phi.weight` file and produce the desired output. We use a bash loop to test block sizes ranging from 1 to 1000:

```
# Arguments of do_block_fes.py
# - input file with CV value and weight for each frame of the trajectory: phi.weight
# - number of CVs: 1
# - CV range (min, max): (-3.141593, 3.141593)
# - # points in output free energy: 51
# - kBT (kJoule/mol): 2.494339
# - Block size: 1<=i<=1000 (every 10)
#
for i in `seq 1 10 1000`; do python3 do_block_fes.py phi.weight 1 -3.141593 3.141593 51 2.494339 $i; done
```

For each value of block size `N`, you will obtain a separate `fes.N.dat` file, containing the value of the ϕ variable on a grid, the average free energy across the blocks with its associated error (in kJ/mol) on each point of the grid:

```
   -3.141593       23.184653     0.080659
   -3.018393       17.264462     0.055181
   -2.895194       13.360259     0.047751
   -2.771994       10.772696     0.043548
   -2.648794        9.403544     0.042022
```

Finally, we can calculate the average error along each free-energy profile as a function of the block size:

```
for i in `seq 1 10 1000`; do a=`awk '{tot+=$3}END{print tot/NR}' fes.$i.dat`; echo $i $a; done > err.blocks
```

and visualize it using `gnuplot`:

```
gnuplot> p "err.blocks" u 1:2 w lp
```

As expected, the error increases with the block size until it reaches a plateau in correspondence of a dimension of the block that exceeds the correlation between data points (Fig. advanced-methods-block-phi).
What can we learn from this analysis about the convergence of the metadynamics simulation?

11.22.7 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Setup and run a metadynamics calculation.
- Compute free energies from the metadynamics bias potential using the sum_hills utility.
- Reweight a metadynamics simulation.
- Calculate errors and assess convergence.

11.23 Lugano tutorial: Brief guide to PLUMED syntax and analyzing trajectories

11.23.1 Aims

The aim of this tutorial is to introduce you to the PLUMED syntax. We will go through the writing of input files to calculate and print simple collective variables on a pre-calculated trajectory.

11.23.2 Learning Outcomes

Once this tutorial is completed, students will be able to:

- Write a simple PLUMED input file and use it with the PLUMED driver to analyze a trajectory.
- Print collective variables such as distances. (DISTANCE), torsional angles (TORSION), gyration radius (GYRATION), and RMSD (RMSD) using the PRINT action.
- Use MOLINFO shortcuts.
- Define and use virtual atoms, such as CENTER.
- Take care of periodic boundary conditions within PLUMED using WHOLEMOLECULES and be able to verify the result with DUMPATOMS.
- Write their own CVs directly in the input file using the CUSTOM action.

11.23.3 Install PLUMED

You can find detailed instructions about how to install PLUMED here:

```
https://github.com/plumed/conda
```

11.23.4 Resources

Before starting the tutorial, please create a separate directory, named `hands_on_1`, and enter it:

```
mkdir hands_on_1
cd hands_on_1
```

The reference trajectories and other files needed for the exercises proposed in this tutorial can be downloaded from `github` using the following command:

```
wget https://github.com/plumed/lugano2019/raw/master/handson_1/handson_1.tgz
```

The zip archive contains the following files:

- GB1_native.pdb : A PDB file with the native structure of the GB1 protein.
- traj-whole.xtc: A trajectory in xtc format. To make the exercise easier, GB1 has been made whole already.
- traj-broken.xtc: The same trajectory as it was originally produced by GROMACS. Here GB1 is broken by periodic boundary conditions and should be fixed.

The archive can be opened with the following command:

```
tar xvzf handson_1.tgz
```

This tutorial has been tested on PLUMED version 2.6.0

11.23.5 Instructions

PLUMED is a library that can be incorporated into many MD codes by adding a relatively simple and well documented interface. Once it is incorporated you can use PLUMED to perform a variety of different analyzes on the fly and to bias the sampling in the molecular dynamics engine. PLUMED can also, however, be used as a standalone code for analyzing trajectories. If you are using the code in this way you can, once PLUMED is compiled, run the plumed executable by issuing the command:

```
plumed <instructions>
```

Let's start by getting a feel for the range of calculations that we can use PLUMED to perform. Issue the following command now:

```
plumed --help
```

What is output when this command is run is a list of the tasks you can use PLUMED to perform. There are commands that allow you to patch an MD code, commands that allow you to run molecular dynamics and commands that allow you to build the manual. In this tutorial we will mostly be using PLUMED to analyze trajectories, however. As such most of the calculations we will perform will be performed using the driver tool. Let's look at the options we can issue to plumed driver by issuing the following command:

```
plumed driver --help
```

As you can see we can do a number of things with plumed driver. For all of these options, however, we are going to need to write a PLUMED input file. The syntax of the PLUMED input file is the same that we will use later to run enhanced sampling simulations, so all the things that you will learn now will be useful later when you will run PLUMED coupled to an MD code. In the following we are going to see how to write an input file for PLUMED.

11.23.5.1 The PLUMED internal units

By default the PLUMED inputs and outputs quantities in the following units:

- Energy - kJ/mol
- Length - nanometers
- Time - picoseconds

If you want to change these units you can do this using the UNITS keyword.

11.23.6 The syntax of the PLUMED input file

The main goal of PLUMED is to compute collective variables, which are complex descriptors than can be used to analyze a conformational change or a chemical reaction. This can be done either on-the-fly during molecular dynamics or a posteriori, using PLUMED as a post-processing tool. In both cases one, should create an input file with a specific PLUMED syntax. A sample input file is below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-1.txt
# Compute distance between atoms 1 and 10.
# Atoms are ordered as in the trajectory files and their numbering starts from 1.
# The distance is called "d" for future reference.
d: DISTANCE ATOMS=1,10

# Create a virtual atom in the center between atoms 20 and 30.
# The virtual atom only exists within PLUMED and is called "center" for future reference.
center: CENTER ATOMS=20,30

# Compute the torsional angle between atoms 1, 10, 20, and center.
# Notice that virtual atoms can be used as real atoms here.
# The angle is called "phi" for future reference.
phi1: TORSION ATOMS=1,10,20,center

# the same CV defined before can be split into multiple line
TORSION ...
LABEL=phi2
ATOMS=1,10,20,center
...

# Print d every 10 step on a file named "COLVAR1".
PRINT ARG=d STRIDE=10 FILE=COLVAR1

# Print phi1 and phi2 on another file names "COLVAR2" every 100 steps.
PRINT ARG=phi1,phi2 STRIDE=100 FILE=COLVAR2
```

In the input file above, each line defines a so-called action. An action could either compute a distance, or the center between two or more atoms, or print some value on a file. Each action supports a number of keywords, whose value is specified. Action names are highlighted in green and, clicking on them, you can go to the corresponding page in the manual that contains a detailed description for each keyword. Actions that support the keyword `STRIDE` are those that determine how frequently things are to be done. Notice that the default value for `STRIDE` is always 1. In the example above, omitting `STRIDE` keywords the corresponding COLVAR files would have been written for every

frame of the analyzed trajectory. All the other actions in the example above do not support the STRIDE keyword and are only calculated when requested. That is, d will be computed every 10 frames, and phi1 and phi2 every 100 frames. In short, you can think that for every snapshot in the trajectory that you are analyzing PLUMED is going to execute all the listed actions, though some of them are optimized out when STRIDE is different from 1.
Variables should be given a name (in the example above, d, phi1, and phi2), which is then used to refer to these variables. Lists of atoms should be provided as comma separated numbers, with no space. Virtual atoms can be created and assigned a name for later use.
You can find more information on the PLUMED syntax at Getting Started page of the manual. The complete documentation for all the supported collective variables can be found at the Collective Variables page.

11.23.7 Exercises

11.23.7.1 Exercise 1: Computing and printing simple collective variables

In this exercise, we will make practice with computing and printing collective variables. To analyze the trajectories provided here, you should:

- Create a PLUMED input file with a text editor (let us call it plumed.dat) similar to the one above.
- Run the command plumed driver command (see below)

Notice that you can also visualize trajectories with VMD directly. For example, the trajectory traj-whole.xtc can be visualized with the command vmd GB1_native.pdb traj-whole.xtc.
Let's prepare a PLUMED input file that calculates

- The gyration radius of all CA protein atoms (GYRATION). Look in the GB1_native.pdb file to retrieve the list of indexes of the CA atoms.
- The total number of contacts (COORDINATION) between all protein CA atoms. For COORDINATION, set reference distance R_0 to 8.0 A (be careful with units!!).

Here you can find a sample plumed.dat file that you can use as a template. Whenever you see an highlightedF↩ ILL string, this is a string that you should replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-1.txt
# Compute gyration radius on CA atoms:
r: GYRATION ATOMS=__FILL__

# Compute number of contacts between CA atoms
co: COORDINATION GROUPA=__FILL__ R_0=__FILL__

# Print the collective variables on COLVAR file every step
PRINT ARG=r,co FILE=COLVAR STRIDE=__FILL__
```

Once your plumed.dat file is complete, you can use it with the following command

```
> plumed driver --plumed plumed.dat --mf_xtc traj-broken.xtc
```

Scroll in your terminal to read the PLUMED log. As you can see, PLUMED gives a lot of feedback about the input that he is reading. There's the place where you can check if PLUMED understood correctly your input.
The command above will create a COLVAR file like this one:

```
#! FIELDS time r co
 0.000000 2.458704 165.184131
 1.000000 2.341932 164.546603
 2.000000 2.404708 162.606975
 3.000000 2.454297 143.850117
 4.000000 2.569342 147.110410
 5.000000 2.304027 163.608695
 6.000000 2.116676 177.549792
 7.000000 2.068599 183.177952
 8.000000 2.021605 181.929958
... more lines ...
```

Notice that the first line informs you about the content of each column.
In case you obtain different numbers, check your input, you might have made some mistake!
This file can then be visualized using the following python script:

```
import matplotlib.pyplot as plt
import plumed
colvar=plumed.read_as_pandas("COLVAR")
plt.plot(colvar.time,colvar.r)
plt.show()
plt.plot(colvar.time,colvar.co)
plt.show()
```

Now, look at what happens if you run the exercise twice. The second time, PLUMED will *back up* the previously produced file so as not to overwrite it. You can also *concatenate* your files by using the action RESTART at the beginning of your input file.
Finally, let's try to use the same input file with `traj-whole.xtc` and examine the results. Did you find the same results as with the previous trajectory? Why?

11.23.7.2 Exercise 2: MOLINFO shortcuts

PLUMED provides some shortcuts to select atoms with specific properties. To use this feature, you should specify the MOLINFO action along with a reference pdb file.
Let's try to use this functionality to calculate the backbone dihedral angle phi of residue 2 of GB1. This CV is defined by the action TORSION and a set of 4 atoms. For residue `i`, the dihedral phi is defined by these atoms: C(i-1),N(i),CA(i),C(i). After consulting the manual and inspecting GB1_native.pdb, let's define the dihedral angle in question in two different ways: using the MOLINFO shortcut to define psi of residue 2 and with an explicit list of 4 atoms.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-1.txt
# activate MOLINFO functionalities
MOLINFO STRUCTURE=__FILL__

# define phi dihedral of residue 2 as a list of 4 atoms
t1: TORSION ATOMS=__FILL__
# define the same dihedral using MOLINFO shortcuts
t2: TORSION ATOMS=__FILL__

# print the value of t1 and t2 every 10 steps
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=10
```

After completing the PLUMED input file above, let's use it to analyze the trajectory `traj-whole.xtc` that you downloaded at the start of this exercise. Now, let's plot the trajectory of the two CVs above. If you executed this exercise correctly, these two trajectories should be identical.

Note

> In the manual you can also learn how to use selections done with MDtraj or MDAnalysis. For instance, the CA atoms that you needed in Exercise 1: Computing and printing simple collective variables could have been picked with `{{@mda:name CA}}`.

11.23.7.3 Exercise 3: Virtual atoms

Sometimes, when calculating a CV, you may not want to use the positions of a number of atoms directly. Instead you may want to define a virtual atom whose position is generated based on the positions of a collection of other atoms. For example you might want to use the center of mass (COM) or the geometric center (CENTER) of a group of atoms.
In this exercise, you will learn how to specify virtual atoms and later use them to define a CV. Let's start by having a look at the PLUMED input file below.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-1.txt
# geometric center of first residue
first: CENTER ATOMS=1,2,3,4,5,6,7,8
# geometric center of last residue
last: CENTER ATOMS=427-436

# distance between centers of first and last residues, with PBCs
d1: DISTANCE ATOMS=first,last
# distance between centers of first and last residues, without PBCs
d2: DISTANCE ATOMS=first,last NOPBC

# print the two distances every step
PRINT ARG=d1,d2 STRIDE=1 FILE=COLVAR
```

Make a PLUMED input containing the above input and execute it on the trajectory `traj-broken.xtc` by making use of the following command:

```
plumed driver --mf_xtc traj-broken.xtc --plumed plumed.dat
```

Before we turn to analyzing what is output from this calculation there are a few things to note about this input file. Firstly, I should describe what this file instructs PLUMED to do. It tells PLUMED to:

1. calculate the position of the Virtual Atom 'first' as the CENTER of atoms from 1 to 8;
2. calculate the position of the Virtual Atom 'last' as the CENTER of atoms from 427 to 436;
3. calculate the distance between the two Virtual Atoms 'first' and 'last' and saves it in 'd1';
4. calculate the distance (ignoring periodic boundary conditions) between the two Virtual Atom 'first' and 'last' and saves it in 'd2';
5. print the content of 'd1' and 'd2' in the file COLVAR for every frame of the trajectory

Notice that in the input above we have used two different ways of writing the atoms used in the CENTER calculation:

1. ATOMS=1,2,3,4,5,6,7,8 is the explicit list of the atoms we need
2. ATOMS=427-436 is the range of atoms needed

Notice also that ranges of atoms can be defined with a stride which can also be negative as shown by the commands below, which are both equivalent:

1. ATOMS=from,to:by (i.e.: 427-436:2)
2. ATOMS=to,from:-by (i.e.: 436-427:-2)

Let's now analyze the output of the calculation by plotting the time series of the two CVs. Are they identical? What do you think is happening in those frames in which the two CVs are different?
You can repeat the same analysis on `traj-whole.xtc` and compare with the previous trajectory. Are the results identical?

11.23.7.4 Exercise 4: Taking care of periodic boundary conditions

As you should have noticed, in this tutorial we are working with two different trajectories of the GB1 protein:

- traj-broken.xtc: the original GROMACS trajectory in which GB1 is broken by periodic boundary conditions
- traj-whole.xtc: the trajectory processed by the `gmx trjconv` utility to fix discontinuities due to periodic boundary conditions

In many PLUMED CVs, periodic boundary conditions are automatically taken into account unless a special option (NOPBC) is used. In this way, the user can work directly with the raw trajectory `traj-broken.xtc`. Alternatively, PLUMED can reconstruct internally the coordinates of the system and thus fix discontinuities due to the periodic boundary conditions. In order to do so, the WHOLEMOLECULES action should be used.
In this exercise, we will learn how to use the WHOLEMOLECULES action. Let's ask PLUMED to internally make the structure of GB1 whole and print the new atoms positions on a new file, called `dump.gro`, every 10 steps.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-1.txt
# use WHOLEMOLECULES to make the entire protein whole
# let's use the range syntax to specify all GB1 atoms
WHOLEMOLECULES ENTITY0=__FILL__
# print the positions of all atoms every 10 steps
DUMPATOMS FILE=dump.gro ATOMS=__FILL__ STRIDE=__FILL__
```

Once your `plumed.dat` file is complete, you can use it with the following command

```
> plumed driver --plumed plumed.dat --mf_xtc traj-broken.xtc
```

and look at the output file with vmd. Have the problems with the periodic boundary conditions been fixed?
At this point, we can repeat Exercise 3: Virtual atoms after adding the WHOLEMOLECULES action at the beginning of the PLUMED input file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-1.txt
# make protein whole
WHOLEMOLECULES ENTITY0=__FILL__

# geometric center of first residue
first: CENTER ATOMS=1,2,3,4,5,6,7,8
# geometric center of last residue
last: CENTER ATOMS=427-436

# distance between centers of first and last residues, with PBCs
d1: DISTANCE ATOMS=first,last
# distance between centers of first and last residues, without PBCs
d2: DISTANCE ATOMS=first,last NOPBC

# print the two distances every step
PRINT ARG=d1,d2 STRIDE=1 FILE=COLVAR
```

We can now visualize the time line of the two CVs and compare it with the results of Exercise 3: Virtual atoms. Which CVs are identical and which are not? Why?

11.23.7.5 Exercise 5: Using CVs that measure the distance from a reference conformation

In many cases, it is useful to define a CV that quantifies the distance of the system from a reference conformation. In PLUMED, this can be achieved using a variety of different CVs. Please, have a look here for more info.
In this exercise, we will learn how to use the RMSD and DRMSD CVs. In order to do so, we need to edit a reference pdb file and identify:

- for RMSD, the atoms that you want to use for alignment and RMSD calculation
- for DRMSD, the atoms that you want to use for the DRMSD calculation

Please refer to the manual to understand how to specify the atoms needed to calculate the CA-RMSD and CA-D↩RMSD with respect to the native GB1 structure. After editing the reference pdb file, please complete the following input file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-1.txt
# RMSD on CA atoms, after optimal alignment
rmsd: RMSD REFERENCE=__FILL__ TYPE=OPTIMAL

# DRMSD on CA atoms
# Only pairs of atoms whose distance in the reference structure
# is within 0.1 and 0.8 nm are considered
drmsd: DRMSD REFERENCE=__FILL__ LOWER_CUTOFF=0.1 UPPER_CUTOFF=0.8

# print both CVs to file
PRINT ARG=__FILL__ STRIDE=1 FILE=COLVAR
```

Once your `plumed.dat` file is complete, you can use it with the following command

```
> plumed driver --plumed plumed.dat --mf_xtc traj-broken.xtc
```

and also on the trajectory file in which problems due to the periodic boundaries have been fixed:

```
> plumed driver --plumed plumed.dat --mf_xtc traj-whole.xtc
```

Can you comment the results obtained and the differences - if any - between the two trajectories? What is happening to the protein during the course of the simulation? Are the two metrics, RMSD and DRMSD, equivalent?

11.23.7.6 Exercise 6: Creating your own CV directly in the PLUMED input file

In PLUMED, you can define your own CV directly in the input file by writing it as a function of existing CVs using the CUSTOM action. PLUMED will then automatically calculate the derivatives with the respect to the atoms positions. Let's look at the following example.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-1.txt
# distance between atoms 10 and 12
dAB: DISTANCE ATOMS=10,12
# distance between atoms 10 and 15
dAC: DISTANCE ATOMS=10,15

# custom CV defined as the difference between dAC->y and dAB->x
diff: CUSTOM ARG=dAB,dAC FUNC=y-x PERIODIC=NO

# print the 3 CVs to file every 10 steps
PRINT ARG=dAB,dAC,diff FILE=COLVAR STRIDE=10
```

In this example, a custom CV is defined as the difference of two distances between pairs of atoms.
Please complete the following input file to calculate two new CVs from those defined in Exercise 5: Using CVs that measure the distanc
:

- the average of RMSD and DRMSD, calculated on all the CA atoms of GB1 (easy)
- the minimum distance between RMSD and DRMSD (a bit more difficult). To define this function, you can creatively take inspiration from the path CV `s` (see PATH).

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-1.txt
# RMSD on CA atoms, after optimal alignment
rmsd: RMSD REFERENCE=__FILL__ TYPE=OPTIMAL

# DRMSD on CA atoms
# Only pairs of atoms whose distance in the reference structure
# is within 0.1 and 0.8 nm are considered
drmsd: DRMSD REFERENCE=__FILL__ LOWER_CUTOFF=0.1 UPPER_CUTOFF=0.8

# average of RMSD and DRMSD
ave: CUSTOM ARG=rmsd,drmsd FUNC=__FILL__ PERIODIC=NO

# minimum distance between RMSD and DRMSD
min: CUSTOM ARG=rmsd,drmsd FUNC=__FILL__ PERIODIC=NO

# print all 4 CVs to file every step
PRINT ARG=__FILL__ STRIDE=1 FILE=COLVAR
```

Once your `plumed.dat` file is complete, you can use it with the following command

```
> plumed driver --plumed plumed.dat --mf_xtc traj-broken.xtc
```

and then check that the custom CVs are indeed what they are expected by plotting the `COLVAR` file with `matplotlib`.

11.24 Lugano tutorial: Using restraints

11.24.1 Aims

The aim of this tutorial is to introduce the users to the use of constant biases in PLUMED and their application to perform a simple Umbrella Sampling WHAM simulation.

11.24.2 Objectives

- Apply a restraint on a simulations over one or more collective variables
- Understand the effect of a restraint on the acquired statistics
- Perform a simple un-biasing of a restrained simulation
- Add an external potential in the form of an analytical or numerical function
- Perform an Umbrella Sampling WHAM simulation

11.24.3 Resources

The TARBALL for this tutorial contains the following files:

- diala.pdb: a PDB file for alanine dipeptide in vacuo
- topol.tpr: a GROMACS run file to perform MD of alanine dipeptide
- run_us.sh: a script to perform a serial US wham simulation
- wham.py: a script to perform the WHAM analysis
- do_fes.py: a script to generate fes from CV and weights

Also notice that the `.solutions` direction of the tarball contains correct input files for the exercises. This tutorial has been tested with version 2.5.

11.24.4 Introduction

PLUMED can calculate conformational properties of a system a posteriori as well as on-the-fly. This information can be use to manipulate a simulation on-the-fly. This means adding energy terms in addition to those of the original Hamiltonian. This additional energy terms are usually referred as Bias. In the following we will see how to apply a constant bias potential with PLUMED. It is preferable to run each exercise in a separate folder.

11.24.4.1 Biased sampling

A system at temperature T samples conformations from the canonical ensemble:

$$P(q) \propto e^{-\frac{U(q)}{k_B T}}$$

. Here q are the microscopic coordinates and k_B is the Boltzmann constant. Since q is a highly dimensional vector, it is often convenient to analyze it in terms of a few collective variables (see Lugano tutorial: Brief guide to PLUMED syntax and analyzing). The probability distribution for a CV s is

$$P(s) \propto \int dq e^{-\frac{U(q)}{k_B T}} \delta(s - s(q))$$

This probability can be expressed in energy units as a free energy landscape $F(s)$:

$$F(s) = -k_B T \log P(s)$$

.
Now we would like to modify the potential by adding a term that depends on the CV only. That is, instead of using $U(q)$, we use $U(q) + V(s(q))$. There are several reasons why one would like to introduce this potential. One is to avoid that the system samples some un-desired portion of the conformational space. As an example, imagine that you want to study dissociation of a complex of two molecules. If you perform a very long simulation you will be able to see association and dissociation. However, the typical time required for association will depend on the size of the simulation box. It could be thus convenient to limit the exploration to conformations where the distance between the two molecules is lower than a given threshold. This could be done by adding a bias potential on the distance between the two molecules. Another example is the simulation of a portion of a large molecule taken out from its initial context. The fragment alone could be unstable, and one might want to add additional potentials to keep the fragment in place. This could be done by adding a bias potential on some measure of the distance from the experimental structure (e.g. on root-mean-square deviation).
Whatever CV we decide to bias, it is very important to recognize which is the effect of this bias and, if necessary, remove it a posteriori. The biased distribution of the CV will be

$$P'(s) \propto \int dq e^{-\frac{U(q)+V(s(q))}{k_B T}} \delta(s - s(q)) \propto e^{-\frac{V(s(q))}{k_B T}} P(s)$$

and the biased free energy landscape

$$F'(s) = -k_B T \log P'(s) = F(s) + V(s) + C$$

Thus, the effect of a bias potential on the free energy is additive. Also notice the presence of an undetermined constant C. This constant is irrelevant for what concerns free-energy differences and barriers, but will be important later when we will learn the weighted-histogram method. Obviously the last equation can be inverted so as to obtain the original, unbiased free-energy landscape from the biased one just subtracting the bias potential

$$F(s) = F'(s) - V(s) + C$$

Additionally, one might be interested in recovering the distribution of an arbitrary observable. E.g., one could add a bias on the distance between two molecules and be willing to compute the unbiased distribution of some torsional angle. In this case there is no straightforward relationship that can be used, and one has to go back to the relationship between the microscopic probabilities:

$$P(q) \propto P'(q) e^{\frac{V(s(q))}{k_B T}}$$

The consequence of this expression is that one can obtained any kind of unbiased information from a biased simulation just by weighting every sampled conformation with a weight

$$w \propto e^{\frac{V(s(q))}{k_B T}}$$

That is, frames that have been explored in spite of a high (disfavoring) bias potential V will be counted more than frames that has been explored with a less disfavoring bias potential.
We will make use as a toy-model of alanine dipeptide: we will see how we can use an iterative approach to build a constant bias to speed up the sampling.

Note

Create a folder for each exercise and use sub-folders if you want to run the same simulation with multiple choices for the parameters

11.24.5 Exercise 1: Preliminary run with alanine dipeptide

Alanine dipeptide is characterized by multiple minima separated by relatively high free energy barriers. Here we will explore the conformational space of alanine dipeptide using a standard MD simulation, then instead of using the free energy as an external potential we will try to fit the potential using gnuplot and add a bias using an analytical function of a collective variable with CUSTOM and BIASVALUE .
As a first test lets run an MD and generate on-the-fly the free energy as a function of the phi and psi collective variables separately.
This is an example input file to calculate the phi and psi angles on the fly and accumulate two 1D histograms from which calculating the free energy.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-2.txt
#SETTINGS MOLFILE=user-doc/tutorials/lugano-2/diala.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=diala.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2
hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1
ffphi: CONVERT_TO_FES GRID=hhphi
ffpsi: CONVERT_TO_FES GRID=hhpsi
DUMPGRID GRID=ffphi FILE=fes_phi STRIDE=100000
DUMPGRID GRID=ffpsi FILE=fes_psi STRIDE=100000
PRINT ARG=phi,psi FILE=colvar.dat STRIDE=50
```

run it with gromacs as

```
gmx mdrun -s topol -plumed plumed.dat -nb cpu -v
```

from the colvar file it is clear that we can quickly explore two minima but that the region for positive phi is not accessible. Ideally we would like to speed up the sampling of regions that are not visited spontaneously by MD. We have multiple possibilities. One option could be to use as a bias the opposite of the accumulated free-energy using EXTERNAL . Another option can be to fit the FES and use the fit. This is what we will do, but first of all take a look at the fes accumulated in time.

```
>gnuplot
plot for [i=0:9] 'analysis.'.i.'.fes_phi' u 1:2 w l t''.i
rep 'fes_phi' u 1:2 w l t'final'
plot for [i=0:9] 'analysis.'.i.'.fes_psi' u 1:2 w l t''.i
rep 'fes_psi' u 1:2 w l t'final'
```

So first we need to fit the opposite of the free energy as a function of phi in the region explored with a periodic function, because of the gaussian like look of the minima we can fit it using the von Mises distribution. In gnuplot

```
>gnuplot
gnuplot>plot 'fes_phi' u 1:(-$2) w l
```

Now find a value such as the fes is always positive, e.g. ∼38

```
gnuplot>plot 'fes_phi' u 1:(-$2+38) w l
gnuplot>f(x)=exp(k1*cos(x-a1))+exp(k2*cos(x-a2))
gnuplot>k1=2
gnuplot>k2=2
gnuplot>fit [-2.9:-0.7] f(x) 'fes_phi' u 1:(-$2+38) via k1,a1,k2,a2
gnuplot>rep f(x)
```

The function and the resulting parameters can be used to run a new biased simulation:

11.24.6 Exercise 2: First biased run with alanine dipeptide

To the above file we add a few lines to define using CUSTOM a function of the angle phi.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-2.txt
#SETTINGS MOLFILE=user-doc/tutorials/lugano-2/diala.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=diala.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2

CUSTOM ...
ARG=phi
LABEL=doubleg
FUNC=exp(__FILL__*cos(x-__FILL__))+exp(__FILL__*cos(x-__FILL__))
PERIODIC=NO
... CUSTOM

b: BIASVALUE ARG=doubleg

hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1
ffphi: CONVERT_TO_FES GRID=hhphi
ffpsi: CONVERT_TO_FES GRID=hhpsi
DUMPGRID GRID=ffphi FILE=fes_phi STRIDE=100000
DUMPGRID GRID=ffpsi FILE=fes_psi STRIDE=100000
PRINT ARG=phi,psi,b.bias FILE=colvar.dat STRIDE=50
```

It is now possible to run a second simulation and observe the new behavior. The system quickly explores a new minimum. While a quantitative estimate of the free energy difference of the old and new regions is out of the scope of the current exercise what we can do is to add a new von Mises function centered in the new minimum with a comparable height, in this way we can hope to facilitate a back and forth transition along the phi collective variable. Look at the old and new free energy and add a third exponential function to CUSTOM centered in the new minimum.

```
gnuplot> plot 'fes_phi' u 1:(-$2+38) w l
gnuplot> f(x)=exp(k3*cos(x-a3))
gnuplot>k3=2
gnuplot> fit [0.3:1.8] f(x) 'fes_phi' u 1:(-$2+38) via k3,a3
```

We can now run a third simulation where both regions are biased.

11.24.7 Exercise 3: Second biased run with alanine dipeptide

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-2.txt
#SETTINGS MOLFILE=user-doc/tutorials/lugano-2/diala.pdb
```

```
MOLINFO STRUCTURE=diala.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2

CUSTOM ...
ARG=phi
LABEL=tripleg
FUNC=exp(__FILL__*cos(x-__FILL__))+exp(__FILL__*cos(x-__FILL__))+exp(__FILL__*cos(x-__FILL__))
PERIODIC=NO
... CUSTOM

b: BIASVALUE ARG=tripleg

hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1
ffphi: CONVERT_TO_FES GRID=hhphi
ffpsi: CONVERT_TO_FES GRID=hhpsi
DUMPGRID GRID=ffphi FILE=fes_phi STRIDE=100000
DUMPGRID GRID=ffpsi FILE=fes_psi STRIDE=100000
PRINT ARG=phi,psi,b.bias FILE=colvar.dat STRIDE=50
```

With this third simulation it should be possible to visit both regions as a function on the phi torsion. The resulting free energy is now reporting about the biased simulation is flatter than the former even if not flat everywhere. Now it is possible to reweight the sampling and obtain a better free energy estimate along phi.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-2.txt
#SETTINGS MOLFILE=user-doc/tutorials/lugano-2/diala.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=diala.pdb

phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2

MATHEVAL ...
ARG=phi
LABEL=tripleg
FUNC=__FILL__
PERIODIC=NO
... MATHEVAL

b: BIASVALUE ARG=tripleg
as: REWEIGHT_BIAS ARG=b.bias

hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1 LOGWEIGHTS=as
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1 LOGWEIGHTS=as
ffphi: CONVERT_TO_FES GRID=hhphi STRIDE=100000
ffpsi: CONVERT_TO_FES GRID=hhpsi STRIDE=100000

DUMPGRID GRID=ffphi FILE=ffphi.dat
DUMPGRID GRID=ffpsi FILE=ffpsi.dat

PRINT ARG=phi,psi,b.bias FILE=colvar.dat STRIDE=50
```

Of notice that the reweighting is also applied to the psi collective variable. If you have performed your simulations in different folder you can compare the effect of the bias on phi on the free energy of psi. For a single simulation with a constant bias the reweighting is simple, the weight of each frame is exp(+bias(cv(t))/kt). So it is possible to perform the reweighting by hand at any time.
Now you have performed an original Umbrella Sampling calculation. This is not particularly easy to setup nor robust, even if from a modern perspective it is a very rough implementation of METAD
In the next exercise we will perform a WHAM Umbrella Sampling simulation.

11.24.8 Exercise 4: WHAM Umbrella Sampling

In this case we will run many simulations with a strong harmonic restraint centered around specific values of phi in such a way to cover all possible values, keep each simulation close to its specific value, allow for overlap between neighbor simulations, i.e. simulations centered around consecutive phi values. The simulation can be either performed in parallel by preparing starting configurations close to each value or sequentially, extracting a good starting conformation from the former simulations. In the specific case of alanine dipeptide we can even just start always from the same configuration and let the bias quickly move it close to the target values.

To run the simulation in scalar you can make use of the provided bash script that is:

```
for AT in -3.00  -2.75 -2.50 -2.25 -2.00 -1.75 -1.50 -1.25 -1.00 -0.75 -0.50 -0.25 0.00 0.25 0.50 0.75 1.00 1.
do

cat >plumed.dat << EOF
# vim:ft=plumed
MOLINFO STRUCTURE=diala.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2
#
# Impose an umbrella potential on CV 1
# with a spring constant of 250 kjoule/mol
# at fixed points along phi
#
restraint-phi: RESTRAINT ARG=phi KAPPA=250.0 AT=$AT
# monitor the two variables and the bias potential from the two restraints
PRINT STRIDE=20 ARG=phi,psi,restraint-phi.bias FILE=COLVAR$AT
EOF

gmx mdrun -plumed plumed.dat -nsteps 100000 -x traj$AT.xtc -c cout$AT.gro -nb cpu

done
```

you can run it using

```
./run_us.sh
```

Plotting the phi collective variable for all replica you will see that each simulation has explored a well defined region of the conformation space as defined by phi. To perform the WHAM merging of the windows we need to 1) collect all the frames

```
gmx trjcat -f traj*.xtc -cat -o concatenated.xtc
```

2) calculate the values for all employed biases applied on each frame for this we can write a plumed-wham.dat file including all the biases used in the former simulations:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-2.txt
#SETTINGS MOLFILE=user-doc/tutorials/lugano-2/diala.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=diala.pdb
phi: TORSION ATOMS=@phi-2

RESTRAINT ARG=phi KAPPA=250.0 AT=-3.00
RESTRAINT ARG=phi KAPPA=250.0 AT=-2.75
RESTRAINT ARG=phi KAPPA=250.0 AT=-2.50
RESTRAINT ARG=phi KAPPA=250.0 AT=-2.25
RESTRAINT ARG=phi KAPPA=250.0 AT=-2.00
RESTRAINT ARG=phi KAPPA=250.0 AT=-1.75
RESTRAINT ARG=phi KAPPA=250.0 AT=-1.50
RESTRAINT ARG=phi KAPPA=250.0 AT=-1.25
RESTRAINT ARG=phi KAPPA=250.0 AT=-1.00
RESTRAINT ARG=phi KAPPA=250.0 AT=-0.75
RESTRAINT ARG=phi KAPPA=250.0 AT=-0.50
RESTRAINT ARG=phi KAPPA=250.0 AT=-0.25
RESTRAINT ARG=phi KAPPA=250.0 AT=0.00
RESTRAINT ARG=phi KAPPA=250.0 AT=0.25
RESTRAINT ARG=phi KAPPA=250.0 AT=0.50
RESTRAINT ARG=phi KAPPA=250.0 AT=0.75
RESTRAINT ARG=phi KAPPA=250.0 AT=1.00
RESTRAINT ARG=phi KAPPA=250.0 AT=1.25
RESTRAINT ARG=phi KAPPA=250.0 AT=1.50
RESTRAINT ARG=phi KAPPA=250.0 AT=1.75
RESTRAINT ARG=phi KAPPA=250.0 AT=2.00
RESTRAINT ARG=phi KAPPA=250.0 AT=2.25
RESTRAINT ARG=phi KAPPA=250.0 AT=2.50
RESTRAINT ARG=phi KAPPA=250.0 AT=2.75
RESTRAINT ARG=phi KAPPA=250.0 AT=3.00

PRINT ARG=*.bias FILE=biases.dat STRIDE=10
PRINT ARG=phi FILE=allphi.dat STRIDE=10

plumed driver --mf_xtc concatenated.xtc --plumed plumed-wham.dat
```

3) run the iterative WHAM optimization and get a weight per frame

```
python wham.py biases.dat 25 2.49
```

where 25 is the number of windows and 2.49 is the temperature in energy unit. After some time the result is a file weight.dat with one weight per frame that can be used to calculate any possible property of the system. For example the free energy profile along phi.
To do so edit the weight.dat file to add 3 blank lines and then

```
paste allphi.dat weights.dat | grep -v \# > allphi-w.dat
python do_fes.py allphi-w.dat 1 -3.1415 3.1415 50 2.49 fes.dat
```

the resulting profile will be disappointing, error estimate and convergence will be discussed in the following tutorials, but clearly simulations are too short. A more advanced approach would be to use the configurations obtained from the former simulations to generate multiple replicas and then perform the US again for longer time and possible in parallel. The syntax is presented in the following but the exercise is possible only if plumed is compiled with mpi

11.24.9 Exercise 5: WHAM Umbrella Sampling in parallel (optional)

Here we use the "replica" syntax of plumed to write a single plumed input file for all the windows:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-2.txt
#SETTINGS FILENAME=plumed.dat MOLFILE=user-doc/tutorials/lugano-2/diala.pdb
# this is plumed.dat
# vim:ft=plumed
MOLINFO STRUCTURE=diala.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2
#
# Impose an umbrella potential on CV 1
# with a spring constant of 250 kjoule/mol
# at fixed points along phi
#
restraint-phi: RESTRAINT ...
    ARG=phi KAPPA=250.0
    AT=@replicas:{
    -3.00 -2.75 -2.50 -2.25
    -2.00 -1.75 -1.50 -1.25
    -1.00 -0.75 -0.50 -0.25
     0.00 0.25 0.50 0.75
     1.00 1.25 1.50 1.75
     2.00 2.25 2.50 2.75 3.00
    }
...
# monitor the two variables and the bias potential from the restraint
PRINT STRIDE=20 ARG=phi,psi,restraint-phi.bias FILE=COLVAR
```

The @replicas syntax allow to define different values for a variable for the different replicas.

```
mpiexec -np 25 gmx_mpi -s topol -plumed plumed.dat -multi 25 -replex 100 -nb cpu -nsteps 100000
```

In this case we run 25 parallel simulations and we also try to perform replica-exchange between neighbor replicas. Once the simulation is finished the trajectories can be concatenated and analyzed with WHAM making use of the plumed native implementation:

```
gmx_mpi trjcat -f traj*.xtc -o concatenated.xtc -cat
```

Write a new plumed-wham.dat

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-2.txt
INCLUDE FILE=plumed.dat
  h1: WHAM_HISTOGRAM ...
     ARG=phi BIAS=restraint-phi.bias TEMP=300
     GRID_MIN=-pi GRID_MAX=pi GRID_BIN=100
     BANDWIDTH=0.1
...
  fes1: CONVERT_TO_FES TEMP=300 GRID=h1
  DUMPGRID GRID=fes1 FILE=fes1.dat
```

And again use the driver in parallel:

```
mpiexec -np 25 plumed driver --mf_xtc concatenated.xtc --plumed plumed-wham.dat --multi 25
```

11.25 Lugano tutorial: Metadynamics simulations with PLUMED

11.25.1 Aims

The aim of this tutorial is to train users to perform metadynamics simulations with PLUMED. Error analysis, which is very very important and requires extensive discussions, will be done in a separate tutorial.

11.25.2 Objectives

Once this tutorial is completed students will be able to:

- Write the PLUMED input file to perform metadynamics simulations
- Calculate the free energy from a metadynamics run
- Monitor the behavior of variables in a metadynamics run

11.25.3 Resources

The TARBALL for this project contains the following files:

- diala.pdb: a PDB file for alanine dipeptide in vacuo
- topol.tpr: a GROMACS run file to perform MD of alanine dipeptide

This tutorial has been tested on version 2.5.

11.25.4 Introduction

We have seen that PLUMED can be used to compute collective variables. However, PLUMED is most often use to add forces on the collective variables. To this aim, we have implemented a variety of possible biases acting on collective variables. The complete documentation for all the biasing methods available in PLUMED can be found at the Bias page. In the following we will see how to build an adaptive bias potential with metadynamics. Here you can find a brief recap of the metadynamics theory.
In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t\to\infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t\to\infty) = -\frac{\Delta T}{T+\Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard MD, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T + \Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135] [136].
We will play with a toy system, alanine dipeptide simulated in vacuo using the AMBER99SB-ILDN force field (see Fig. lugano-3-ala-fig). This rather simple molecule is useful to understand data analysis and free-energy methods. This system is a nice example because it presents two metastable states separated by a high free-energy barrier. It is conventional use to characterize the two states in terms of Ramachandran dihedral angles, which are denoted with Φ and Ψ in Fig. lugano-3-transition-fig .

11.25.4.1 Exercise 1: my first metadynamics calculation

In this exercise we will setup and perform a well-tempered metadynamics run using the backbone dihedral ϕ as collective variable. During the calculation, we will also monitor the behavior of the other backbone dihedral ψ.
Here you can find a sample `plumed.dat` file that you can use as a template. Whenever you see an highlightedF↩ ILL string, this is a string that you should replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-3.txt
# vim:ft=plumed
MOLINFO STRUCTURE=diala.pdb
# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
# you might want to use MOLINFO shortcuts
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
psi: TORSION ATOMS=__FILL__

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 500 time steps, with initial height equal to 1.2 kJ/mol
  PACE=500 HEIGHT=1.2
# the bias factor should be wisely chosen
  BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
  SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
...

# Print both collective variables and the value of the bias potential on COLVAR file
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=10
```

The syntax for the command METAD is simple. The directive is followed by a keyword ARG followed by the labels of the CVs on which the metadynamics potential will act. The keyword PACE determines the stride of Gaussian deposition in number of time steps, while the keyword HEIGHT specifies the height of the Gaussian in kJ/mol. For each CVs, one has to specify the width of the Gaussian by using the keyword SIGMA. Gaussian will be written to the file indicated by the keyword FILE.
In this example, the bias potential will be stored on a grid, whose boundaries are specified by the keywords GRID_↩ MIN and GRID_MAX. Notice that you can provide either the number of bins for every collective variable (GRID_BIN) or the desired grid spacing (GRID_SPACING). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither GRID_BIN nor GRID_SPACING) and if Gaussian width is fixed, PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.
Once your `plumed.dat` file is complete, you can run a 10-ns long metadynamics simulations with the following command

```
> gmx mdrun -s topol.tpr -nsteps 5000000 -plumed plumed.dat
```

During the metadynamics simulation, PLUMED will create two files, named COLVAR and HILLS. The COLVAR file contains all the information specified by the PRINT command, in this case the value of the CVs every 10 steps of

simulation, along with the current value of the metadynamics bias potential. We can use `gnuplot` to visualize the behavior of the CV during the simulation, as reported in the COLVAR file:

```
gnuplot> p "COLVAR" u 1:2
```

By inspecting Figure lugano-3-phi-fig, we can see that the system is initialized in one of the two metastable states of alanine dipeptide. After a while (t=0.1 ns), the system is pushed by the metadynamics bias potential to visit the other local minimum. As the simulation continues, the bias potential fills the underlying free-energy landscape, and the system is able to diffuse in the entire phase space.
The HILLS file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content:

```
#! FIELDS time phi sigma_phi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
```

The line starting with FIELDS tells us what is displayed in the various columns of the HILLS file: the simulation time, the instantaneous value of ϕ, the Gaussian width and height, and the bias factor. We can use the HILLS file to visualize the decrease of the Gaussian height during the simulation, according to the well-tempered recipe:
If we look carefully at the scale of the y-axis, we will notice that in the beginning the value of the Gaussian height is higher than the initial height specified in the input file, which should be 1.2 kJ/mol. In fact, this column reports the height of the Gaussian scaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy.

11.25.4.2 Exercise 2: estimating the free energy

One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills should be used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.
To calculate the free energy as a function of ϕ, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called `fes.dat` in which the free-energy surface as function of ϕ is calculated on a regular grid. One can modify the default name for the free energy file, as well as the boundaries and bin size of the grid, by using the following options of sum_hills :

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

The result should look like this:
To assess the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The option --stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option --mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --stride 100 --mintozero
```

one free energy is calculated every 100 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. The resulting plot should look like the following:
These two qualitative observations:

- the system is diffusing efficiently in the collective variable space (Figure lugano-3-phi-fig)
- the estimated free energy does not change significantly as a function of time (Figure lugano-3-metad-phifest-fig)

suggest that the simulation most likely converged.

Warning

The fact that the Gaussian height is decreasing to zero should not be used as a measure of convergence of your metadynamics simulation!

Note

The two observations above are necessary, but qualitative conditions for convergence. A quantitative assessment of convergence can be obtained by performing an error analysis in a later tutorial.

11.25.4.3 Exercise 3: the role of the bias factor.

The bias factor allows you to choose how extensive your sampling of the CV space will be. If you choose it too large, you will explore a large reason of the CV space and your simulation will take more time to converge. If you choose it too small, you will not be able to cross the barriers you are interested in.
Try to run your simulation with different values of the bias factor. Going low to anything that is greater than 1 should give meaningful results, although if the chosen value is too low the system will only explore a limited portion of the space.
Which is the minimum value of the bias factor that allows you to see, in the simulated time, transitions between the two relevant minima?

11.25.5 Exercise 4: reweighting

In the previous exercise we biased ϕ and compute the free energy as a function of the same variable. Many times you want to decide which variable you want to analyze *after* having performed a simulation. In order to do so you must reweight your simulation.
There are multiple ways to reweight a metadynamics simulations. In order to calculate these weights, we can use either of these two approaches:
1) Weights are calculated by considering the time-dependence of the metadynamics bias potential [3];
2) Weights are calculated using the metadynamics bias potential obtained at the end of the simulation and assuming a constant bias during the entire course of the simulation [50].
In this exercise we will use the umbrella-sampling-like reweighting approach (Method 2). In order to compute the weights we will use the driver tool.
First of all, prepare a `plumed_reweight.dat` file that is identical to the one you used for running your simulation but add the keyword `RESTART=YES` to the METAD command. This will make this action behave as if PLUMED was restarting. It will thus read the already accumulated hills and continue adding more. In addition, set hills height to zero and the pace to a large number. This will actually avoid adding new hills (and even if they are added they will have zero height). Finally, change the PRINT statement so that you write every frame (`STRIDE=1`) and that, in addition to `phi` and `psi`, you also write `metad.bias.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-3.txt
__FILL__ # here goes the definitions of the CVs

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 500 time steps, with initial height equal to 1.2 kJ/mol
  PACE=10000000 HEIGHT=0.0 # <- this is the new stuff!
# the bias factor should be wisely chosen
  BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
  SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
# Say that METAD should be restarting
  RESTART=YES # <- this is the new stuff!
...

PRINT ARG=phi,psi,metad.bias FILE=COLVAR STRIDE=1  # <- also change this one!
```

Then run the driver using this command

```
> plumed driver --ixtc traj_comp.xtc --plumed plumed.dat --kt 2.5
```

Notice that you have to specify the value of k_BT in energy units. While running your simulation this information was taken from the MD code.
As a result, PLUMED will produce a new COLVAR file with an additional column. The beginning of the file should look like this:

```
#! FIELDS time phi psi metad.bias
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
 0.000000 -1.497988 0.273498 110.625670
 1.000000 -1.449714 0.576594 110.873141
 2.000000 -1.209587 0.831417 109.742353
 3.000000 -1.475975 1.279726 110.752327
```

The last column will give as, in energy units, the logarithm of the weight of each frame. You can easily obtain the weight of each frame using the expression $w \propto \exp\left(\frac{V(s)}{k_B T}\right)$. You might want to read the COLVAR file in python and compute a weighted histogram.
If you want PLUMED to do the histograms for you, you can just add the following lines that you learned in Lugano tutorial: Using restraints to the plumed input file:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-3.txt
as: REWEIGHT_BIAS ARG=__FILL__

hhphi: HISTOGRAM ARG=phi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1 LOGWEIGHTS=as
hhpsi: HISTOGRAM ARG=psi STRIDE=50 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.1 LOGWEIGHTS=as
ffphi: CONVERT_TO_FES GRID=hhphi
ffpsi: CONVERT_TO_FES GRID=hhpsi

DUMPGRID GRID=ffphi FILE=ffphi.dat
DUMPGRID GRID=ffpsi FILE=ffpsi.dat
```

and plot the result using gnuplot.

```
gnuplot> p "ffphi.dat"
gnuplot> p "ffpsi.dat"
```

11.25.6 Exercise 5: larger orthogonal barriers

Alanine dipeptide is often considered as a too-simple system to understand the typical problems that one will then see in real systems. This is very much not true! We will here see how to make the exercise arbitrarily difficult, to the point that metadynamics will not work anymore.
The difficult case of metadynamics is when there are variables that (a) are orthogonal to the biased ones and (b) exhibit large energy barriers. The first point implies that when you flatten the distribution of the biased collective variables you are not accelerating the sampling of the orthogonal variables. The second point implies that those variable will take a lot of time to explore different metastable states.
Often these variables are not known and they are thus called "hidden variables". In the case of alanine dipeptide, we can easily add a barrier on Ψ with some additional PLUMED command. We will add a Gaussian barrier centered at $\Psi = 0.5$. To do so, add to the PLUMED input file that you prepared for the first exercise (that is: biasing Φ alone) the following lines

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-3.txt
# first shift the psi variable.
# setting the new periodicity to -pi,pi will make sure that the barrier
# is a continuous function of the coordinates
shift1:   CUSTOM ARG=psi   FUNC=x-0.5 PERIODIC=-pi,pi
shift2:   CUSTOM ARG=psi   FUNC=x+2.5 PERIODIC=-pi,pi
# then compute the barrier energy.
# this would be a Gaussian with wifth 0.3. You can pick the height as you like
barrier:  CUSTOM ARG=shift1,shift2 FUNC=__FILL__*exp(-0.5*x^2/0.2^2)+__FILL__*exp(-0.5*y^2/0.2^2) PERIODIC=NO
# then add the barrier to the total energy of the system.
BIASVALUE ARG=barrier
```

Something high like 25 kj/mol should create a significant difficulty. Notice that this means a barrier 10 $k_B T$ high in a direction that is not being biased.
You should see something like this
If you look at this series you will clearly see that there are clear changes in the Phi dynamics. If you look at Psi dynamics you should be able to see that these changes correspond to transitions in Psi.
This is an indication that an important slow variable is missing.
Find the minimum value of the barrier required for the simulation not to converge in the simulated timescale.

11.25.7 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Setup and run a metadynamics calculation.
- Compute free energies from the metadynamics bias potential using the sum_hills utility.
- Identify problems when a hidden variable exhibit a large barrier

11.26 Lugano tutorial: Calculating error bars

11.26.1 Aims

This tutorial will teach you how to use block averaging techniques to compute the error bars on the estimates for the ensemble average and the free energy that you obtain from a biased simulation. Please note that the ensemble averages that you obtain from simulations are always estimates and that you should thus **always** endeavor to provide an estimate of the error bar.

11.26.2 Objectives

Once this tutorial is completed students will

- Be able to explain why it is important to compute error bars when calculating averages and free energy surfaces from enhanced sampling calculations.
- Be able to use PLUMED to calculate ensemble averages and histograms using the keywords AVERAGE and HISTOGRAM.
- Be able to use PLUMED to perform block analysis of trajectory data using the keywords AVERAGE and HISTOGRAM.
- Be able to explain how block analysis can be used to detect problems with error bar underestimation in correlated data.

11.26.3 Resources

The `TARBALL` for this project contains the following files:

- in : The input file for simplemd that contains the parameters for the MD simulation.
- input.xyz : An initial configuration for the cluster that we are studying in this tutorial.
- plumed.dat : An empty input file for PLUMED

This tutorial has been tested on v2.5 but it should also work with other versions of PLUMED.
Also notice that the `.solutions` direction of the tarball contains correct input files for the exercises. Please only look at these files once you have tried to solve the problems yourself. Similarly the tutorial below contains questions for you to answer that are shown in bold. You can reveal the answers to these questions by clicking on links within the tutorial but you should obviously try to answer things yourself before reading these answers.

11.26.4 Introduction

In this tutorial we are going to study a very simple physical system; namely, seven Lennard Jones atoms in a two dimensional space. This simple system has been extensively studied as has often been used to benchmark new simulation techniques. In addition, the potential energy landscape has been fully characterized and it is known that only the four structurally-distinct minima shown below exist:
In the exercises that follow we are going to learn how to use PLUMED to determine the relative free energies of these four structures by running molecular dynamics simulations as well as how to find suitable error bars on the energy of these minima. First of all, however, we are going to learn how to estimate the average energy of this system and how to compute the error on our estimate for the average. We will thus start with a very brief recap of the theory behind taking an ensemble average.

11.26.5 Background

When performing unbiased and biased simulations the aim is **always** to estimate the ensemble average for some quantity $\langle A \rangle$. We know from statistical mechanics that, if we are in the canonical (NVT) ensemble, the value of this ensemble average is given by:

$$\langle A \rangle = \frac{\int \mathrm{d}x\mathrm{d}p A(x) e^{-\frac{H(x,p)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}}$$

where $H(x,p)$ is the Hamiltonian for our system, T is the temperature and k_B is the Boltzmann constant. We also know, however, that for all but the simplest possible systems, it is impossible to solve the integrals in this expression analytically. Furthermore, because this expression involves integrals over all the $3N$ position and $3N$ momentum coordinates, using a numerical integration method that employs a set of regularly spaced grid points in the $6N$ dimensional phase space would be prohibitively expensive. We are thus forced to instead generate a time series of random variables and to approximate the ensemble average using:

$$\langle A \rangle \approx \frac{1}{T} \sum_{t=1}^{T} A_t \qquad \text{Equation 1}$$

where each A_t in the expression above is a sample from the distribution:

$$P(A_t = a) = \frac{\int \mathrm{d}x\mathrm{d}p \delta(A(x) - a) e^{-\frac{H(x,p)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}}$$

This distribution (thankfully) is exactly the distribution we are sampling from if we compute the values the observable A takes during the course of in an equilibrated molecular dynamics trajectory. We can thus calculate an approximate value for $\langle A \rangle$ by computing the value of A for each of the frames in our trajectory and by computing the average value that A takes over the trajectory using equation 1. It is critical to remember, however, that the value we obtain for $\langle A \rangle$ when we compute it this way is itself a random variable. When reporting ensemble averages calculated in this way we should thus endeavor to quantify the error in our estimate of this quantity by computing multiple estimates for $\langle A \rangle$ and by using these multiple estimates to compute a variance for the underlying random variable. This tutorial will explain how this such error bars are computed in practice. At some stage you may find it useful to watch the following videos in order to understand the theory that is behind these calculations a little better.

11.26.6 Getting started

11.26.6.1 Using PLUMED as an MD code

Before getting into the business of computing an ensemble average we first need to setup the system we are going to study. In this tutorial we are going to use the MD code **simplemd** that is part of PLUMED. You can run this code by issuing the command:

```
plumed simplemd < in
```

where in here is the input file from the tar ball for this tutorial, which is shown below:

```
nputfile input.xyz
outputfile output.xyz
temperature 0.5
tstep 0.005
friction 0.1
forcecutoff 2.5
listcutoff  3.0
ndim 2
nstep 200000
nconfig 1000 trajectory.xyz
nstat   1000 energies.dat
```

This input instructs PLUMED to perform 200000 steps of MD at a temperature of $k_B T = 0.5\epsilon$ starting from the configuration in input.xyz. The timestep in this simulation is 0.005 $\sqrt{\epsilon m \sigma^2}$ and the temperature is kept fixed using a Langevin thermostat with a relaxation time of $0.1\sqrt{\epsilon m \sigma^2}$. Trajectory frames are output every 1000 MD steps to

a file called trajectory.xyz. Notice also that in order to run the calculation above you need to provide an empty file called plumed.dat. This file is the input file to the PLUMED plugin, which, because this file is empty, is doing nothing when we run the calculation above.

Run a calculation using simplemd and the input above and visualize the trajectory that is output. Describe what happens during this calculation and explain why this is happening.

You can visualize what occurs during the trajectory by using a visualization package such as VMD (https://www.ks.uiuc.edu/Research/vmd/). If you are using VMD you can see the MD trajectory by using the command:

```
vmd trajectory.xyz
```

You should observe that all the atoms fly apart early on in the simulation and that the cluster evaporates. The cluster evaporates because at a temperature of $k_BT = 0.5\epsilon$ the gas state has a lower free energy than than the cluster state.

Change the parameters in the input file for simplemd so as to prevent the cluster from evaporating.

To prevent the cluster from evaporating you need to lower the temperature in the file in. The cluster will not evaporate if the temperature is set equal to $k_BT = 0.2\epsilon$.

Now try to think how we can use a bias potential to stop the cluster from evaporating. Why might using a bias potential be preferable to the method that you have just employed? N.B. The next exercise is in the hidden section below so you need to expand it. Please try to come up with your own answer to the question of what bias potential we should be using before expanding this section by thinking about the material that was covered in Lugano tutorial: Using restraints.

If we lower the temperature of the simulation very little will happen. Yes the cluster will no longer evaporate but at the same time we will not see any transitions between the various basins in this energy landscape. We thus can use a bias potential to prevent the cluster from exploring gaseous configurations that do not interest us instead of lowering the temperature. In other words, we are going to add restraints that will prevent the cluster from evaporating. The particular restraint we are going to use will prevent all the atoms from moving more than 2σ from the center of mass of the cluster. As the masses of all the atoms in the cluster are the same we can compute the position of the center of mass using:

$$\mathbf{x}_{\rm com} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$$

where $\mathbf{x}_i$ is the position of the atom with the index i. The distance between the atom with index i and the position of this center of mass, d_i, can be computed using Pythagoras' theorem. These distances are then restrained by using the following potential:

$$V(d_i) = \{ 1\,00 * (d_i - 2.0)^2 \text{if} \quad d_i > 20 \text{otherwise}$$

as you can see this potential has no effect on the dynamics when these distances are less than 2 ϵ. If any atom is more than 2 ϵ from the center of mass, however, this potential will drive it back towards the center of mass. The following cell contains a skeleton input file for PLUMED that gets it to calculate and apply this bias potential.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-4.txt
# this optional command tells VIM that this is a PLUMED file and to color the text accordingly
# vim: ft=plumed

# This tells PLUMED we are using Lennard Jones units
UNITS NATURAL

# Calculate the position of the center of mass.  We can then refer to this position later in the input using t
com: COM __FILL__

# Add the restraint on the distance between com and the first atom
d1: DISTANCE __FILL__
UPPER_WALLS ARG=d1 __FILL__

# Add the restraint on the distance between com and the second atom
d2: DISTANCE __FILL__
UPPER_WALLS  __FILL__

# Add the restraint on the distance between com and the third atom
d3: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fourth atom
d4: DISTANCE __FILL__
```

```
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fifth atom
d5: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the sixth atom
d6: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the seventh atom
d7: DISTANCE __FILL__
UPPER_WALLS __FILL__
```

Copy and paste the content above into the file plumed.dat and then fill in the blanks by looking up the documentation for these actions online and by reading the description of the calculation that you are to run above. Once you have got a working plumed.dat file run a calculation using simplemd again at a temperature of $k_BT = 0.5\epsilon$ and check to see if the bias potential is indeed preventing the cluster from evaporating.

11.26.7 Block averaging

The previous sections showed you how to set up the simulations of the Lennard Jones cluster and reviewed some of the material on adding static bias potentials that was covered in the earlier hands-on sessions in the meeting. Now that we have completed all this we can move to the material on calculating appropriate error bars that we will cover in this tutorial. In this section you are going to work through the process of block averaging the trajectory yourself for a simple case in order to better understand the theory. In the final section we will then apply this technique to a more complex case. Without further ado then lets run a trajectory and collect some data to analyze.

Run a simulation of the Lennard Jones cluster at $k_BT = 0.2\epsilon$ using for 12000 steps using the input file below (but with the blanks filled in obviously). This calculation outputs the potential energy of the system for every tenth step in the trajectory to a file called energy.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-4.txt
# this optional command tells VIM that this is a PLUMED file and to color the text accordingly
# vim: ft=plumed

# This tells PLUMED we are using Lennard Jones units
UNITS NATURAL

# Calculate the position of the center of mass.  We can then refer to this position later in the input using t
com: COM __FILL__

# Add the restraint on the distance between com and the first atom
d1: DISTANCE __FILL__
UPPER_WALLS ARG=d1 __FILL__

# Add the restraint on the distance between com and the second atom
d2: DISTANCE __FILL__
UPPER_WALLS  __FILL__

# Add the restraint on the distance between com and the third atom
d3: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fourth atom
d4: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fifth atom
d5: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the sixth atom
d6: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the seventh atom
d7: DISTANCE __FILL__
```

```
UPPER_WALLS __FILL__

# Get the potential energy
e: ENERGY
# Print the potential energy to a file
PRINT ARG=__FILL__ FILE=energy STRIDE=10
```

The exercise below will take you through the process of calculating block averages and hence error bars on the data you generated.
Notice that we can calculate the block averages that were required for the block averaging technique that was explained in the programming exercise using PLUMED directly. The input below (once you fill in the gaps) calculates and prints block averages over windows of 100 trajectory frames. See if you can fill in the blanks and compare the result you obtain with the result that you obtain by running a python script to convince yourself that PLUMED calculates these block averages correctly.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-4.txt
# this optional command tells VIM that this is a PLUMED file and to color the text accordingly
# vim: ft=plumed

# This tells PLUMED we are using Lennard Jones units
UNITS NATURAL

# Calculate the position of the center of mass.  We can then refer to this position later in the input using t
com: COM __FILL__

# Add the restraint on the distance between com and the first atom
d1: DISTANCE __FILL__
UPPER_WALLS ARG=d1 __FILL__

# Add the restraint on the distance between com and the second atom
d2: DISTANCE __FILL__
UPPER_WALLS  __FILL__

# Add the restraint on the distance between com and the third atom
d3: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fourth atom
d4: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fifth atom
d5: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the sixth atom
d6: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the seventh atom
d7: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Get the potential energy
e: ENERGY
# Calculate block averages of the potential energy
av_e: AVERAGE ARG=__FILL__ CLEAR=__FILL__ STRIDE=__FILL__
# Print the block averages of the potential energy to a file
PRINT ARG=__FILL__ STRIDE=__FILL__ FILE=energy
```

At some point (probably not during the tutorial as you will not have time) you can use the following video and quiz to understand the theory behind this process of block averaging.

11.26.8 Calculating the free energy surface

In this final exercise we are going to run a metadynamics simulation in order to see the Lennard Jones cluster explore all of the basins in the energy landscape that were shown in figure lugano-4-lj7-minima. We will extract a free energy surface from this simulation trajectory and will use the block averaging technique that we learnt about in the previous section to quote error bars on this free energy surface. There are three important differences between

the way we apply the block averaging techinique in this section and the way that we applied the block averaging technique in the previous section; namely:

- The block averaging technique is applied on on the histogram that is estimated from the simulation. As the free energy surface is a function of the histogram we have do some propegation of errors to get the final error bar.
- The free energy surface we are extracting **is not** a single number as it was in the previous section. It is a function evaluated on the grid. We thus have to apply the block averaging technique for the value of the free energy at each grid point separately.
- The simulation in this case is biased so we have to reweight in order to get the unbiased free energy surface.

We will not dwell too much on these issues in what follows. For the interested reader they are discussed at length in https://arxiv.org/abs/1812.08213. Furthermore, the Trieste tutorial: Averaging, histograms and block analysis tutorial deals with each of these issues in turn. If you have sufficient time at the end you may therefore like to work through the exercises in that tutorial in order to better understand how the block averaging technique that was discussed in the previous section has been extended so as to resolve these issues.

11.26.8.1 Running the metadynamics simulation

We can drive transitions between the four possible minima in the Lennard-Jones-seven potential energy landscape by biasing the second and third central moments of the distribution of coordination numbers. The nth central moment of a set of numbers, $\{X_i\}$ can be calculated using:

$$\mu^n = \frac{1}{N}\sum_{i=1}^{N}(X_i - \langle X\rangle)^n \qquad \text{where} \qquad \langle X\rangle = \frac{1}{N}\sum_{i=1}^{N} X_i$$

Furthermore, we can compute the coordination number of our Lennard Jones atoms using:

$$c_i = \sum_{i\neq j} \frac{1-\left(\frac{r_{ij}}{1.5}\right)^8}{1-\left(\frac{r_{ij}}{1.5}\right)^{16}}$$

where r_{ij}__FILL__ is the distance between atom i and atom j. The following cell contains a skeleton input file for PLUMED that gets it to perform metadynamics using the second and third central moments of the distribution of coordination numbers as a CV.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-4.txt
# this optional command tells VIM that this is a PLUMED file and to color the text accordingly
# vim: ft=plumed

# This tells PLUMED we are using Lennard Jones units
UNITS NATURAL

# Calculate the position of the center of mass.  We can then refer to this position later in the input using t
com: COM __FILL__

# Add the restraint on the distance between com and the first atom
d1: DISTANCE __FILL__
UPPER_WALLS ARG=d1 __FILL__

# Add the restraint on the distance between com and the second atom
d2: DISTANCE __FILL__
UPPER_WALLS  __FILL__

# Add the restraint on the distance between com and the third atom
d3: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fourth atom
d4: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fifth atom
d5: DISTANCE __FILL__
UPPER_WALLS __FILL__
```

```
# Add the restraint on the distance between com and the sixth atom
d6: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the seventh atom
d7: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Calculate the collective variables
c1: COORDINATIONNUMBER SPECIES=__FILL__ MOMENTS=__FILL__ SWITCH={RATIONAL __FILL__ }

# Do metadynamics
METAD ARG=__FILL__ HEIGHT=__FILL__ PACE=__FILL__ SIGMA=__FILL__ GRID_MIN=-1.5,-1.5 GRID_MAX=2.5,2.5 GRID_BIN=5
```

This input should be modified to instruct PLUMED to add Gaussian kernels with a bandwidth of 0.1 in both the second and third moment of the distribution of coordination numbers and a height of 0.05 ϵ every 500 MD steps. The metadynamics calculation should then be run using simplemd at a temperature of $k_BT = 0.1\epsilon$. You can then run a simplemd calculation using the following input:

```
inputfile input.xyz
outputfile output.xyz
temperature 0.1
tstep 0.005
friction 1
forcecutoff 2.5
listcutoff  3.0
ndim 2
nstep 1000000
nconfig 100 trajectory.xyz
nstat   1000 energies.dat
```

and the command

```
plumed simplemd < in
```

11.26.8.2 Extracting block averages for the histogram

Having now run the metadynamics we will need to post process our trajectory with **driver** in order to extract the free energy by reweighting. Furthermore, notice that, in order to do our block averaging, we are going to want to extract multiple estimates for the histogram so that we can do our block averaging. We are thus going to use the following input file to extract our estimates of the histogram:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-4.txt
# this optional command tells VIM that this is a PLUMED file and to color the text accordingly
# vim: ft=plumed

UNITS NATURAL

# We can delete the parts of the input that specified the walls and disregrad these in our analysis
# It is OK to do this as we are only interested in the value of the free energy in parts of phase space
# where the bias due to these walls is not acting.

c1: COORDINATIONNUMBER SPECIES=__FILL__ MOMENTS=__FILL__ SWITCH={RATIONAL __FILL__}

# The metadynamics bias is restarted here so we consider the final bias as a static bias in our calculations
METAD ARG=__FILL__ HEIGHT=0.05 PACE=50000000 SIGMA=0.1,0.1 GRID_MIN=-1.5,-1.5 GRID_MAX=2.5,2.5 GRID_BIN=500,50

# This adjusts the weights of the sampled configurations and thereby accounts for the effect of the bias poten
rw: REWEIGHT_BIAS TEMP=0.1

# Calculate the histogram and output it to a file
hh: HISTOGRAM ARG=c1.* GRID_MIN=-1.5,-1.5 GRID_MAX=2.5,2.5 GRID_BIN=200,200 BANDWIDTH=0.02,0.02 LOGWEIGHTS=__F
DUMPGRID GRID=hh FILE=my_histogram.dat STRIDE=2500
```

Once you have filled in the blanks in this input you can then run the calculation by using the command:

```
> plumed driver --ixyz trajectory.xyz  --initial-step 1
```

You must make sure that the HILLS file that was output by your metadynamics simulation is available in the directory where you run the above command. If that condition is satisfied though you should generate a number of files

containing histograms that will be called: analysis.0.my_histogram.dat, analysis.1.myhistogram.dat etc. These files contain the histograms constructed from each of the blocks of data in your trajectory. You can merge them all to get the final free energy surface, which can be calculated using the well known relation between the histogram, $P(s)$, and the free energy surface, $F(s)$:

$$F(s) = -k_B T \ln P(s)$$

that is employed in the following python script:

```
import math
import glob
import numpy as np
# Here are some numbers you will need to change if you run this script on grids generated in different
      contexts
temp = 0.1               # Boltzmann's constant multiplied by the temperature at which the simulation was
      performed
grid_dimension = 2       # Number of collective variables that you provided using the ARG keyword
filename = "my_histogram.dat"  # The name you specified the data to output to in the DUMPGRID command
# Function to read in histogram data and normalization
def readhistogram( fname ) :
        # Read in the histogram data
        data = np.loadtxt( fname )
        with open( filename, "r" ) as myfile :
                for line in myfile :
                        if line.startswith("#! SET normalisation") : norm = line.split()[3]
        return float(norm), data
# Read in the grid file header to work out what fields we have
with open( filename, "r" ) as myfile :
        for line in myfile :
                if line.startswith("#! FIELDS") : fieldnames = line.split()
# Check if derivatives have been output in the grid by investigating the header
nextg = 1
if len(fieldnames)>(2+grid_dimension+1) :
        nextg = 1 + grid_dimension
        assert len(fieldnames)==(2+grid_dimension + nextg)
# Read in a grid
norm, griddata = readhistogram( filename )
norm2 = norm*norm
# Create two np array that will be used to accumulate the average grid and the average grid squared
average = np.zeros( len(griddata[:,0]) )
average_sq = np.zeros( len(griddata[:,0]) )
average[:] = norm*griddata[:, grid_dimension]
average_sq[:] = norm*griddata[:, grid_dimension]*griddata[:, grid_dimension]
# Now sum the grids from all all the analysis files you have
for filen in glob.glob( "analysis.*." + filename ) :
        tnorm, newgrid = readhistogram( filen )
        norm = norm + tnorm
        norm2 = norm2 + tnorm*tnorm
        average[:] = average[:] + tnorm*newgrid[:, grid_dimension]
        average_sq[:] = average_sq[:] + tnorm*newgrid[:, grid_dimension]*newgrid[:, grid_dimension]
# Compte the final average grid
average = average / norm
# Compute the sample variance for all grid points
variance = (average_sq / norm) - average*average
# Now multiply by bessel correction to unbias the sample variance and get the population variance
variance = ( norm /(norm-(norm2/norm)) ) * variance
# And lastly divide by number of grids and square root to get an error bar for each grid point
ngrid = 1 + len( glob.glob( "analysis.*." + filename ) )
errors = np.sqrt( variance / ngrid )
mean_error, denom = 0, 0
for i in range(len(errors)) :
        if np.abs(average[i])>0 :
                errors[i] = errors[i] / average[i]
                mean_error = mean_error + errors[i]
                denom = denom + 1
        else : errors[i] = 0
# Calculate average error over grid and output in header
mean_error = mean_error / denom
print("# Average error for free energy on grid equals ", mean_error )
# Output the final free energy
for i in range(0,len(griddata[:,0])) :
        for j in range(0,grid_dimension) : print( griddata[i,j], end=" " )
        print( -temp*np.log(average[i]), temp*errors[i] )
        # We added spaces every time the y coordinate changes value to make the output readable by gnuplot
        if i%201==0 and i>0 : print()
```

Copy this script to a file called merge-histograms.py and then run it on your data by executing the command:

```
> python merge-histograms.py > final-histogram.dat
```

This will output the final average histogram together with some error bars. You can plot the free energy surface you obtain by using gnuplot and the following command:

```
gnuplot> sp 'final-histogram.dat' u 1:2:3 w pm3d
```

Similarly you can get a sense of how the error in the estimate of the free energy depends on the value of the CV by using the command:

```
gnuplot> sp 'final-histogram.dat' u 1:2:4 w pm3d
```

More usefully, however, if you open the final-histogram.dat file you find that the first line reads:

```
# Average error for historgram is <average-histogram-error> and thus average energy in free energy is <average
```

You can thus read off the average error in the estimate of the free energy from this top line directly.
Repeat the analysis of the trajectory that was discussed in this section with different block sizes. Use the results you obtain to draw a graph showing how the average error on the estimate of the free energy depends on the block size
You should be able to extract a graph that looks something like the one shown below. The error is small when the block size is small because the correlations between the trajectory frames cause this quantity to be underestimated. As the block size increases, however, the error increases until it eventually flattens out.

11.26.9 Conclusions and extensions

This exercise has explained the block averaging technique and has shown you how this technique can be used to extract the errors in estimates of the free energy. You can learn more about the background to this technique and the business of reweighting biased trajectories in particular by working through Trieste tutorial: Averaging, histograms and block analysis or by reading https://arxiv.org/abs/1812.08213.

11.27 Lugano tutorial: Dimensionality reduction

11.27.1 Aims

This tutorial will show you how to you can use PLUMED to perform dimensionality reduction. The tutorial will try not to focus on the application of one particular algorithm but will instead try to show you the principles behind the implementation of these algorithms that has been adopted within PLUMED. By the end of the tutorial you will thus be able to design your own dimensionality reduction algorithm.

11.27.2 Objectives

Once this tutorial is completed students will

- Be able to use COLLECT_FRAMES to store a trajectory for later analysis
- Be able to use PCA to perform principal component analysis
- Be able to construct a dissimilarity matrix using EUCLIDEAN_DISSIMILARITIES
- Be able to select a subset of landmark points to analyze with particular dimensionality reduction algorithm.
- Be able to construct low dimensional representations using CLASSICAL_MDS and SKETCH_MAP.
- Be able to generate projections of non-landmark points by using PROJECT_ALL_ANALYSIS_DATA

11.27.3 Resources

The TARBALL for this project contains the following files:

- beta-hairpin.pdb : A pdb file containing the protein that we are going to study in this tutorial in a beta hairpin configuration. This input will be used as a template so that we can use the names of special groups in many of the inputs that follow.

In addition, you will also need to get a copy of the trajectory that we will be analyzing in this tutorial by executing the following command:

```
wget https://github.com/plumed/lugano2019/raw/master/handson_5/traj.dcd
```

The trajectory we are analyzing is a smaller version of the trajectory that was analyzed in the following paper:

- https://www.frontiersin.org/articles/10.3389/fmolb.2019.00046/full

In this paper the trajectory was analyzed with a variety of different dimensionality reduction algorithms and the results were compared. The paper may, therefore, be of interest.
This tutorial has been tested on v2.5 but it should also work with other versions of PLUMED.
Also notice that the `.solutions` direction of the tarball contains correct input files for the exercises. Please only look at these files once you have tried to solve the problems yourself. Similarly the tutorial below contains questions for you to answer that are shown in bold. You can reveal the answers to these questions by clicking on links within the tutorial but you should obviously try to answer things yourself before reading these answers.

11.27.4 Introduction

In all of the previous tutorials we have used functions that take the position of all the atoms in the system - a $3N$ dimensional vector, where N is the number of atoms as input. This function then outputs a single number - the value of the collective variable - that tells us where in a low dimensional space we should project that configuration. Problems can arise because this collective-variable function is many-to-one and it may thus be difficult to distinguish between every different pair of structurally distinct conformers of our system.
In this tutorial we are going to introduce an alternative approach to this business of finding collective variables. In this alternative approach we are going to stop trying to seek out a function that can take any configuration of the atoms (any $3N$-dimensional vector) and find its low dimensional projection on the collective variable axis. Instead we are going to take a set of configurations of the atoms (a set of $3N$-dimensional vectors of atom positions) and try to find a sensible set of projections for these configurations. We are going to find this low dimensional representation by seeking out an isometry between the space containing the $3N$-dimensional vectors of atom positions and some lower-dimensional space. This idea is explained in more detail in the following video and details on the various algorithms that we are using in the tutorial can be found in:

- https://arxiv.org/abs/1907.04170

As you will find out if you read the chapter that is linked above there are multiple ways to construct an isometric embedding of a trajectory. This tutorial will thus try to teach you a set of basic ideas and will then encourage you to experiment and to develop your own strategy for representing the data set.

11.27.5 Exercises

11.27.5.1 Collecting the trajectory

The first thing that we need to learn to do in order to run these dimensionality reduction algorithms is to store the trajectory so that we can analyze in later. The following input (once the blanks are filled in) will take the positions of the non-hydrogen atoms in our protein and store them every 1 step in an object that we can refer to later in the input using the label data. All the configurations stored in data will then be output to a pdb file once the whole trajectory is read in. Fill in the blanks in the input below now:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-5.txt
# This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=__FILL__ MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES __FILL__=@nonhydrogens

# This should output the atomic positions for the frames that were collected to a pdb file called traj.pdb
OUTPUT_ANALYSIS_DATA_TO_PDB USE_OUTPUT_DATA_FROM=__FILL__ FILE=traj.pdb
```

Then, once all the blanks are filled in, run the command using:

```
plumed driver --mf_dcd traj.dcd
```

Notice that the above input stored the atomic positions of the atoms. We can use the atomic positions in many of the dimensionality reductions that will be discussed later in this tutorial or we can use a high-dimensional vector

of collective variables. The following input thus gives an example of which shows you can compute and store the values the Ramachandran angles of the protein took in all the trajectory frames so that they can be analyzed using a dimensionality reduction algorithm. Try to fill in the blanks on this input and then run this form of analysis on the trajectory using the command above once more:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-5.txt
# This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=__FILL__ MOLTYPE=protein

# The following commands compute all the Ramachandran angles of the protein for you
r2-phi: TORSION ATOMS=@phi-2
r2-psi: TORSION ATOMS=@psi-2
r3-phi: TORSION ATOMS=@phi-3
r3-psi: TORSION ATOMS=@psi-3
r4-phi: TORSION __FILL__
r4-psi: TORSION __FILL__
r5-phi: TORSION __FILL__
r5-psi: TORSION __FILL__
r6-phi: TORSION __FILL__
r6-psi: TORSION __FILL__
r7-phi: TORSION __FILL__
r7-psi: TORSION __FILL__
r8-phi: TORSION __FILL__
r8-psi: TORSION __FILL__
r9-phi: TORSION __FILL__
r9-psi: TORSION __FILL__
r10-phi: TORSION __FILL__
r10-psi: TORSION __FILL__
r11-phi: TORSION __FILL__
r11-psi: TORSION __FILL__
r12-phi: TORSION __FILL__
r12-psi: TORSION __FILL__
r13-phi: TORSION __FILL__
r13-psi: TORSION __FILL__
r14-phi: TORSION __FILL__
r14-psi: TORSION __FILL__
r15-phi: TORSION __FILL__
r15-psi: TORSION __FILL__
r16-phi: TORSION __FILL__
r16-psi: TORSION __FILL__

# This command stores all the Ramachandran angles that were computed
cc: COLLECT_FRAMES __FILL__=r2-phi,r2-psi,r3-phi,r3-psi,r4-phi,r4-psi,r5-phi,r5-psi,r6-phi,r6-psi,r7-phi,r7-ps

# This command outputs all the Ramachandran angles that were stored to a file called angles_data
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=__FILL__ ARG=cc.* FILE=angles_data
```

11.27.5.2 Performing PCA

Having learned how to store data for later analysis with a dimensionality reduction algorithm lets now apply principal component analysis (PCA) upon our stored data. In principal component analysis a low dimensional projections for our trajectory are constructed by:

- Computing a covariance matrix from the trajectory data
- Diagonalizing the covariance matrix.
- Calculating the projection of each trajectory frame on a subset of the eigenvectors of the covariance matrix.

To perform PCA using PLUMED we are going to use the following input with the blanks filled in:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-5.txt
# This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=__FILL__ MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES __FILL__=@nonhydrogens
# This diagonalizes the covariance matrix
pca: PCA USE_OUTPUT_DATA_FROM=__FILL__ METRIC=OPTIMAL NLOW_DIM=2
```

```
# This projects each of the trajectory frames onto the low dimensional space that was
# identified by the PCA command
dat: PROJECT_ALL_ANALYSIS_DATA USE_OUTPUT_DATA_FROM=__FILL__ PROJECTION=__FILL__

# This should output the atomic positions for the frames that were collected and analyzed using PCA
OUTPUT_ANALYSIS_DATA_TO_PDB USE_OUTPUT_DATA_FROM=__FILL__ FILE=traj.pdb
# This should output the PCA projections of all the coordinates
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=__FILL__ ARG=dat.* FILE=pca_data

# These next three commands calculate the secondary structure variables.  These
# variables measure how much of the structure resembles an alpha helix, an antiparallel beta sheet
# and a parallel beta sheet.  Configurations that have different secondary structures should be projected
# in different parts of the low dimensional space.
alpha: ALPHARMSD RESIDUES=all
abeta: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1.0
pbeta: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1.0

# These commands collect and output the secondary structure variables so that we can use this information to
# determine how good our projection of the trajectory data is.
cc2: COLLECT_FRAMES ARG=alpha,abeta,pbeta
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=cc2 ARG=cc2.* FILE=secondary_structure_data
```

To generate the projection you run the command:

```
plumed driver --mf_dcd traj.dcd
```

I would recommend visualizing this data using the GISMO plugin to VMD. You can find instructions on how to compile this code on the page below:
http://epfl-cosmo.github.io/sketchmap/index.html?page=code
(you don't need to compile the sketch-map code) Once GISMO is installed you should have an option to open it when you open vmd. The option to open GISMO can be found under Extensions>Analysis>GISMO. To visualize the results from what we have just done you should need to follow the following instructions:

- Open vmd and load the pdb file that was output: traj.pdb
- Open GISMO and load the pca projections file: pca_data
- Open GISMO and load the secondary structure variables: secondary_structure_data
- You can safely ignore the error message that GISMO will give at this stage.
- Now choose to plot the quantities dat.coord-1 and dat.coord-2 on the x and y axis respectively. Color the points using cc2.alpha.

If you follow the instructions above you should get an image like the one shown below:
You can click on the various points in the plot and VMD will show you the structure in the corresponding trajectory frame. Furthermore, you can get a particularly useful representation of the structures by adding the following text to your ∼/.vmdrc file:

```
user add key m {
  puts "Automatic update of secondary structure, and alignment to first frame"
  trace variable vmd_frame w structure_trace
  rmsdtt
  rmsdtt::doAlign
  destroy $::rmsdtt::w
  clear_reps top
  mol color Structure
  mol selection backbone
  mol representation NewCartoon
  mol addrep top
}
```

With this text in your ∼/.vmdrc file VMD will align all the structures with the first frame and then show the cartoon representation of each structure when you press the m button on your keyboard

11.27.5.3 Performing MDS

In the previous section we performed PCA on the atomic positions directly. In the section before last, however, we also saw how we can store high-dimensional vectors of collective variables and then use these vectors as input to a dimensionality reduction algorithm. We might legitimately ask, therefore, if we can do PCA using these high-dimensional vectors as input rather than atomic positions. The answer to this question is yes as long as the CV is not periodic. If any of our CVs are not periodic we cannot analyze them using the PCA action. We can, however, formulate the PCA algorithm in a different way. In this alternative formulation, which is known as classical multidimensional scaling (MDS) we do the following:

- We calculate the matrix of distances between configurations
- We perform an operation known as centering the matrix.
- We diagonalize the centered matrix
- The eigenvectors multiplied by the square root of the corresponding eigenvalue can then be used as a set of projections for our input points.

This method is used less often the PCA as the matrix that we have to diagonalize here in the third step can be considerably larger than the matrix that we have to diagonalize when we perform PCA. In fact in order to avoid this expensive diagonalization step we often select a subset of so called landmark points on which to run the algorithm directly. Projections for the remaining points are then found by using a so-called out-of-sample procedure. This is what has been done in the following input:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-5.txt
 This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=beta-hairpin.pdb MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES ATOMS=@nonhydrogens
# This should output the atomic positions for the frames that were collected and analyzed using MDS
OUTPUT_ANALYSIS_DATA_TO_PDB USE_OUTPUT_DATA_FROM=cc FILE=traj.pdb

# The following commands compute all the Ramachandran angles of the protein for you
r2-phi: TORSION ATOMS=@phi-2
r2-psi: TORSION ATOMS=@psi-2
r3-phi: TORSION ATOMS=@phi-3
r3-psi: TORSION ATOMS=@psi-3
r4-phi: TORSION __FILL__
r4-psi: TORSION __FILL__
r5-phi: TORSION __FILL__
r5-psi: TORSION __FILL__
r6-phi: TORSION __FILL__
r6-psi: TORSION __FILL__
r7-phi: TORSION __FILL__
r7-psi: TORSION __FILL__
r8-phi: TORSION __FILL__
r8-psi: TORSION __FILL__
r9-phi: TORSION __FILL__
r9-psi: TORSION __FILL__
r10-phi: TORSION __FILL__
r10-psi: TORSION __FILL__
r11-phi: TORSION __FILL__
r11-psi: TORSION __FILL__
r12-phi: TORSION __FILL__
r12-psi: TORSION __FILL__
r13-phi: TORSION __FILL__
r13-psi: TORSION __FILL__
r14-phi: TORSION __FILL__
r14-psi: TORSION __FILL__
r15-phi: TORSION __FILL__
r15-psi: TORSION __FILL__
r16-phi: TORSION __FILL__
r16-psi: TORSION __FILL__

# This command stores all the Ramachandran angles that were computed
angles: COLLECT_FRAMES __FILL__=r2-phi,r2-psi,r3-phi,r3-psi,r4-phi,r4-psi,r5-phi,r5-psi,r6-phi,r6-psi,r7-phi,r
# Lets now compute the matrix of distances between the frames in the space of the Ramachandran angles
```

```
distmat: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=__FILL__ METRIC=EUCLIDEAN
# Now select 500 landmark points to analyze
fps: LANDMARK_SELECT_FPS USE_OUTPUT_DATA_FROM=__FILL__ NLANDMARKS=500
# Run MDS on the landmarks
mds: CLASSICAL_MDS __FILL__=fps NLOW_DIM=2
# Project the remaining trajectory data
osample: PROJECT_ALL_ANALYSIS_DATA USE_OUTPUT_DATA_FROM=__FILL__ PROJECTION=__FILL__

# This command outputs all the projections of all the points in the low dimensional space
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=__FILL__ ARG=osample.* FILE=mds_data

# These next three commands calculate the secondary structure variables.  These
# variables measure how much of the structure resembles an alpha helix, an antiparallel beta sheet
# and a parallel beta sheet.  Configurations that have different secondary structures should be projected
# in different parts of the low dimensional space.
alpha: ALPHARMSD RESIDUES=all
abeta: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1.0
pbeta: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1.0

# These commands collect and output the secondary structure variables so that we can use this information to
# determine how good our projection of the trajectory data is.
cc2: COLLECT_FRAMES ARG=alpha,abeta,pbeta
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=cc2 ARG=cc2.* FILE=secondary_structure_data
```

This input collects all the torsional angles for the configurations in the trajectory. Then, at the end of the calculation, the matrix of distances between these points is computed and a set of landmark points is selected using a method known as farthest point sampling. A matrix that contains only those distances between the landmarks is then constructed and diagonalized by the CLASSICAL_MDS action so that projections of the landmarks can be constructed. The final step is then to project the remainder of the trajectory using the PROJECT_ALL_ANALYSIS_DATA action. Try to fill in the blanks in the input above and run this calculation now using the command:

```
plumed driver --mf_dcd traj.dcd
```

Once the calculation has completed you can, once again, visualize the data generated using the GISMO plugin.

11.27.5.4 Performing sketch-map

The two algorithms (PCA and MDS) that we have looked at thus far are both linear dimensionality reduction algorithms. In addition to these there are a whole class of non-linear dimensionality reduction reduction algorithms which work by transforming the matrix of dissimilarities between configurations, calculating geodesic rather than Euclidean distances between configurations or by changing the form of the loss function that is optimized. In this final exercise we are going to use an algorithm that uses the last of the these three strategies to construct a non-linear projection. The algorithm is known as sketch-map and an input for sketch-map is provided below:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-5.txt
# This reads in the template pdb file and thus allows us to use the @nonhydrogens
# special group later in the input
MOLINFO STRUCTURE=__FILL__ MOLTYPE=protein

# This stores the positions of all the nonhydrogen atoms for later analysis
cc: COLLECT_FRAMES __FILL__=@nonhydrogens
# This should output the atomic positions for the frames that were collected and analyzed using MDS
OUTPUT_ANALYSIS_DATA_TO_PDB USE_OUTPUT_DATA_FROM=__FILL__ FILE=traj.pdb

# The following commands compute all the Ramachandran angles of the protein for you
r2-phi: TORSION ATOMS=@phi-2
r2-psi: TORSION ATOMS=@psi-2
r3-phi: TORSION ATOMS=@phi-3
r3-psi: TORSION ATOMS=@psi-3
r4-phi: TORSION __FILL__
r4-psi: TORSION __FILL__
r5-phi: TORSION __FILL__
r5-psi: TORSION __FILL__
r6-phi: TORSION __FILL__
r6-psi: TORSION __FILL__
r7-phi: TORSION __FILL__
r7-psi: TORSION __FILL__
r8-phi: TORSION __FILL__
r8-psi: TORSION __FILL__
r9-phi: TORSION __FILL__
```

```
r9-psi: TORSION __FILL__
r10-phi: TORSION __FILL__
r10-psi: TORSION __FILL__
r11-phi: TORSION __FILL__
r11-psi: TORSION __FILL__
r12-phi: TORSION __FILL__
r12-psi: TORSION __FILL__
r13-phi: TORSION __FILL__
r13-psi: TORSION __FILL__
r14-phi: TORSION __FILL__
r14-psi: TORSION __FILL__
r15-phi: TORSION __FILL__
r15-psi: TORSION __FILL__
r16-phi: TORSION __FILL__
r16-psi: TORSION __FILL__

# This command stores all the Ramachandran angles that were computed
angles: COLLECT_FRAMES __FILL__=r2-phi,r2-psi,r3-phi,r3-psi,r4-phi,r4-psi,r5-phi,r5-psi,r6-phi,r6-psi,r7-phi,r
# Lets now compute the matrix of distances between the frames in the space of the Ramachandran angles
distmat: EUCLIDEAN_DISSIMILARITIES USE_OUTPUT_DATA_FROM=__FILL__ METRIC=EUCLIDEAN
# Now select 500 landmark points to analyze
fps: LANDMARK_SELECT_FPS USE_OUTPUT_DATA_FROM=__FILL__ NLANDMARKS=500
# Run sketch-map on the landmarks
smap: SKETCH_MAP __FILL__=fps NLOW_DIM=2 HIGH_DIM_FUNCTION={SMAP R_0=6 A=8 B=2} LOW_DIM_FUNCTION={SMAP R_0=6 A
# Project the remaining trajectory data
osample: PROJECT_ALL_ANALYSIS_DATA USE_OUTPUT_DATA_FROM=__FILL__ PROJECTION=__FILL__

# This command outputs all the projections of all the points in the low dimensional space
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=__FILL__ ARG=osample.* FILE=smap_data

# These next three commands calculate the secondary structure variables.  These
# variables measure how much of the structure resembles an alpha helix, an antiparallel beta sheet
# and a parallel beta sheet.  Configurations that have different secondary structures should be projected
# in different parts of the low dimensional space.
alpha: ALPHARMSD RESIDUES=all
abeta: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1.0
pbeta: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1.0

# These commands collect and output the secondary structure variables so that we can use this information to
# determine how good our projection of the trajectory data is.
cc2: COLLECT_FRAMES ARG=alpha,abeta,pbeta
OUTPUT_ANALYSIS_DATA_TO_COLVAR USE_OUTPUT_DATA_FROM=cc2 ARG=cc2.* FILE=secondary_structure_data
```

This input collects all the torsional angles for the configurations in the trajectory. Then, at the end of the calculation, the matrix of distances between these points is computed and a set of landmark points is selected using a method known as farthest point sampling. A matrix that contains only those distances between the landmarks is then constructed and diagonalized by the CLASSICAL_MDS action and this set of projections is used as the initial configuration for the various minimization algorithms that are then used to optimize the sketch-map stress function. As in the previous exercise once the projections of the landmarks are found the projections for the remainder of the points in the trajectory are found by using the PROJECT_ALL_ANALYSIS_DATA action. Try to fill in the blanks in the input above and run this calculation now using the command:

```
plumed driver --mf_dcd traj.dcd
```

Once the calculation has completed you can, once again, visualize the data generated using the GISMO plugin.

11.27.6 Conclusions and extensions

This tutorial shown you that running dimensionality reduction algorithms using PLUMED involves the following stages:

- Data is collected from the trajectory using COLLECT_FRAMES.
- Landmark points are selected using a Landmark Selection algorithm
- The distances between the trajectory frames are computed using EUCLIDEAN_DISSIMILARITIES
- A loss function is optimized in order to generate projections of the landmarks.

- Projections of the non-landmark points are generated using PROJECT_ALL_ANALYSIS_DATA.

There are multiple choices to be made in each of the various stages described above. For example, you can change the particular sort of data this is collected from the trajectory, there are multiple different ways to select landmarks, you can use the distances directly or you can transform them, you can use various different loss function and you can optimize the loss function using a variety of different algorithms. In this final exercise of the tutorial I thus want you to experiment with these various different choices that can be made. Use the data set that we have been working with throughout this tutorial and try to construct an interesting representation of it using some combination of Actions that we have not explored in the tutorial. Some things you can perhaps try:

- Try sketch-map with RMSD distances as input rather than angles
- Try using different Landmark Selection algorithms
- Try using different numbers of landmarks
- Try to use PCA followed by sketch-map
- See if you can work out how to draw contour plot showing the free energy as a function of the low-dimensional coordinates.

11.28 Lugano tutorial: Using PLUMED with LAMMPS

11.28.1 Aims

In this tutorial I will show you how you can use PLUMED to perform a simulation in which a mixture of Lennard Jones particles are forced to separate and mix. You then will be encouraged to investigate how the work done during this process depends on the simulation parameters or the parameters that are used for the enhanced sampled algorithm.

11.28.2 Objectives

Once this tutorial is completed students will have used PLUMED coupled with LAMMPS to perform an investigation of their own design.

11.28.3 Resources

The `TARBALL` for this project contains the following files:

- lammps-equilibration.in : The input for LAMMPS that we will use for our setup and equilibration steps of our investigation
- lammps-production.in : The input for LAMMPS that will be used for the production steps of our investigation
- xdistances.dat : A PLUMED file that defines the collective variables that we will use in the investigation. We will include this file in the PLUMED input files we write for this tutorial.

This tutorial has been tested on v2.5 but it should also work with other versions of PLUMED.
Also notice that the `.solutions` direction of the tarball contains correct input files for the exercises. Please only look at these files once you have tried to solve the problems yourself. Similarly the tutorial below contains questions for you to answer that are shown in bold. You can reveal the answers to these questions by clicking on links within the tutorial but you should obviously try to answer things yourself before reading these answers.

11.28.4 Introduction

This tutorial is somewhat different to the others that you have done. The tutorial will show you how to run a simulation on a binary Lennard Jones system using PLUMED and LAMMPS. In this simulation the two types of atoms will be forced to separate and mix. You will then be encouraged to design your investigation to further investigate this phenomenon by means of simulation. There are a variety of different things you can investigate. You could

investigate how the work done to separate the two components depends on the parameters of the Lennard Jones model or temperature, you could investigate whether you can reduce the amount of hysteresis that is observed with the current simulation parameters. You may even decide that the method proposed is not a particularly good way to investigate the phenomenon of phase separation and propose a different method for simulating the process.

11.28.5 Exercises

11.28.5.1 Installing LAMMPS with PLUMED

You can find the instructions about how to install LAMMPS patched with PLUMED here:

```
https://github.com/plumed/conda#install-lammps
```

11.28.5.2 Setting up the simulation and equilibrating

We can now run a simulation with LAMMPS. If we take the input lammps-equilibration.in that we downloaded from the tarball we can run this simulation using the following commands:

```
touch plumed.dat
/Users/gareth/Plumed-tutorial/LAMMPS/lammps-5Jun19/build/lmp < lammps-equil.in
```

(In the second command here you obviously need to specify the location where you compiled LAMMPS instead of the location where I compiled LAMMPS).
As you can see if you visualize the trajectory output.xyz that is generated by this simulation this input creates an initial configuration containing 1000 atoms. There are 500 Ar atoms and 500 Ne atoms and in the initial configuration all the Ne atoms are placed in one half of the simulation box while the Ar atoms are placed in the other half of the box. During the simulation you have just run you should observe that these two types of atom mix together as the system equilibrates. We are now going to run this simulation again but we will now use PLUMED to monitor the extent how the system comes to equilibrium. In particular, we are going to use a filled in version of the following input file to monitor the energy and the cell parameters over the course of the simulation:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-6a.txt
# Specify that we are in natural units
UNITS NATURAL

# Retrieve the value of the potential energy
ee: ENERGY
# Retrieve the 3 cell vectors
cell: CELL
# Compute the square moduli of three cell vectors
aaa2: COMBINE ARG=cell.ax,cell.ay,cell.az POWERS=2,2,2 PERIODIC=NO
bbb2: COMBINE ARG=cell.bx,cell.by,cell.bz POWERS=2,2,2 PERIODIC=NO
ccc2: COMBINE ARG=cell.cx,cell.cy,cell.cz POWERS=2,2,2 PERIODIC=NO
# Compute the moduli of the three cell vectors
aaa: __FILL__
bbb: __FILL__
ccc: __FILL__

# Print the energy and the moduli of the three cell vectors
PRINT ARG=ee,aaa,bbb,ccc FILE=colvar STRIDE=10
```

Rerun the equilibration calculation that you just performed but with a plumed.dat file that now contains a filled-in version of the above input. You should observe that the energy and cell lengths that are output to the colvar file initially change substantially, which is normal as we are running in the NPT ensemble. After a while, however, these quantities equilibrate and then simply fluctuate around some fixed value.

11.28.5.3 Simulating phase separation

Having equilibrated our system lets now simulate phase separation. What we are going to do is force the Ne atoms to be in a different part of the simulation cell from the Ar atoms. We will thus force the system to be a un-mixed, two phase system from being a mixed, single-phase system. In other words, we are going to force the system to go back to a configuration much like the one that we started our simulation from. Before we get on to running the simulation, however, we are going to need to copy a few pieces to the directory. In particular, we need the following files:

- We need to start our simulation from the equilibrated configuration that we arrived in at the end of the simulation run that we have just performed. We can start in this why by copying the lammps restart file that was output from that simulation: lj.equilibration.restart.100000
- We need the lammps input file called lammps-production.in that we obtained from the tarball that we downloaded at the beginning of the exercise
- We need the PLUMED file called xdistances.dat that we downloaded from the tar ball. This file defines one virtual atom and three collective variables and looks something like the following:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-6a.txt
origin: FIXEDATOM AT=0,0,0

dx: XDISTANCES ...
  ATOMS1=1,origin
  ATOMS2=2,origin
  BETWEEN={GAUSSIAN LOWER=0 UPPER=5.4373815 SMEAR=0.1}
...

da: XDISTANCES ...
  ATOMS1=1,origin
  ATOMS2=2,origin
  ATOMS3=3,origin
  ATOMS4=4,origin
  BETWEEN1={GAUSSIAN LOWER=0 UPPER=5.4373815 SMEAR=0.1}
  BETWEEN2={GAUSSIAN LOWER=-5.4373815 UPPER=0 SMEAR=0.1}
...
```

(In the actual file there are 500 ATOMS keywords in the first XDISTANCES command and 1000 ATOMS keywords in the second XDISTANCES command, however, which is why I have not shown the whole file.)
These XDISTANCES commands compute the x-component of the vector connecting the pair of specified atom. The BETWEEN keywords then specify that we want to calculate how many of these x-components are between a particular lower and upper bound. In this particular case, the FIXEDATOM is placed at the origin (center) of the simulation cell and the CVs monitor how many of the x-components of the vector connecting the origin and each atom are positive and how many are negative. We thus divide the box into equally sized regions and count how many atoms are in each of these regions. To this correctly, however, you have to ensure that the parameters for the BETWEEN keywords are set correctly. Wherever you see 5.4373815 in the input file you need to replace this value by half the simulation box length that you will use for these production calculations. You can get the simulation box for the initial configuration (the final configuration from the equilibration simulation) from the colvar file that was output from your equilibration simulation. You simply need to look at the value that this quantity took in the last line of the colvar file. This quantity will stay fixed throughout your simulation as in this production run we are going to use the NVT ensemble.
With these modifications in place lets now look at the PLUMED input file we are going to use for this calculation:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-6a.txt
UNITS NATURAL
# Include the file that defines our collective variable
INCLUDE FILE=__FILL__

# Add a restraint that ensures that the number of atoms in the left part of the box stays the same as the numb
# in the right part of the box
ccc: CUSTOM ARG=__FILL__ FUNC=x-y PERIODIC=NO
RESTRAINT ARG=ccc AT=0.0 KAPPA=10

# Add a moving restraint that forces the Ar atoms (1-500) to separate from the Ne atoms and to then mix again
res: MOVINGRESTRAINT ARG=__FILL__ AT0=230 AT1=450 AT2=230 KAPPA0=100 STEP0=0 STEP1=50000 STEP2=100000
PRINT ARG=da.*,dx.between,res.* FILE=colvar STRIDE=10
```

Fill in the blanks in the input above and then try to run the simulation. You should see as the simulation progresses the Ne and Ar atoms un-mix into two separate phases. They then remix and by the end of the simulation there is only one phase once more.

11.28.6 Conclusions

This tutorial has shown you how you can run a simulation in which an interesting physical phenomenon is forced to occur. The phenomenon in question is phase separation - the simulation was started from a mixed phase of Ar and

Ne. These two chemical components were then forced to separate during the simulation and a two phase system was formed with all the Ne atoms in one part of the simulation box and all the Ar atoms in the other part of the box. Your task is now to investigate this phenomenon further. Some questions you might want to ask yourself as you design your investigation are the following:

- How do you get a quantitative estimate of the energetic cost for performing the phase separation reaction?
- How does this energetic cost depend on the simulation parameters e.g. ratio of Ar/Ne, temperature, pressure?
- How does this energetic cost depend on the ratio of force field parameters?
- Is the method that you were taught in this tutorial the best method for studying this process? What other methods might we use to investigate phase separation?

11.29 Lugano tutorial: Binding of a ion and a dinucleotide

11.29.1 Aims

In this tutorial I will show you how you can use PLUMED and metadynamics to study the binding between a ion and a dinucleotide.

11.29.2 Objectives

Once this tutorial is completed students will

- Know how to enhance binding between molecules using metadynamics.
- Know how to analyze metadynamics simulations.
- Know how to compute standard affinities.

11.29.3 Resources

The reference trajectory and other files can be obtained at this path

```
wget https://github.com/plumed/lugano2019/raw/master/lugano-6b.tgz
```

This tutorial has been tested on v2.5 but it should also work with other versions of PLUMED.

11.29.4 Introduction

For this tutorial we will consider a practical application. The system is actually taken from [137]. It is one of the simplest systems studied in that paper, namely a guanine dinucleotide monophosphate binding a Mg ion. This is a very simple binding event, but a very similar procedure might be used to study binding of a ligand on a protein. The reason why this exercise is particularly simple is that instead of the ligand we have a single ion, thus with no internal degree of freedom, and instead of a protein with a complex binding pocket we have a dinucleotide. We are also assuming to know which is the proper binding site, since we can easily guess that the most stable binding will happen on the phosphate.
Since running these simulations on your laptop would take too long, you will start with the output files obtained with a decently long simulation and analyze them.

Warning

> The trajectory is too short (approx 20ns) to obtain converged results! To get statistically significant numbers, please run it longer.

Before continuing, please read carefully the `plumed.dat` file that was used to produce the simulation, since there you will find all the explanations about which variables were biased and how.
In case you want to do analysis with python, you can use the included `plumed_pandas.py` module, which is a preview of a feature that will be available in plumed 2.6. It requires pandas to be installed (use `conda install pandas`) and allows to extract columns from a COLVAR file by name. It works in this way:

```
> import plumed_pandas
> import matplotlib.pyplot as plt
> df=plumed_pandas.read_as_pandas("COLVAR")
# shows the head of the file:
> df.head()
# plot distance between Mg and phosphate
> plt.plot(df["dp"][:],".")
# plot coordination number of Mg with water
> plt.plot(df["cn"][:],".")
```

11.29.5 Exercises

11.29.5.1 Exercise 1: Computing the free energy as a function of the biased variables.

As the title says, just compute the free-energy landscape as a function of the biased collective variable (namely, distance between the Mg ion and the phosphate and coordination number of the Mg ion with water oxygen atoms). You should just use sum_hills with the usual options. In order to visualize the result with gnuplot you might use something like this:

```
gnuplot> set pm3d map
gnuplot> sp "fes.dat" u 1:2:3
```

You should obtain a plot similar to this one:

11.29.5.2 Exercise 2: Visualizing the trajectory

This exercise is optional and is not needed to continue with the next points. However, it is a very good idea to do it in order to have a better understanding of what the system is doing!
Beware that the periodic boundary conditions were broken. You can adjust them using PLUMED with an input like this one (please fill the gaps)

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/aaaa-lugano-6b.txt
MOLINFO STRUCTURE=conf.pdb
WHOLEMOLECULES ENTITY0=@nucleic
c: CENTER ATOMS=@nucleic
mg:  GROUP ATOMS=__FILL__ # find the serial number of the Mg ion
WRAPAROUND AROUND=c ATOMS=mg

# check documentation of WRAPAROUND!
# you should also know how many atoms make a water molecule
WRAPAROUND AROUND=c ATOMS=@water GROUPBY=__FILL__

# dump your trajectory
DUMPATOMS ATOMS=@mdatoms FILE=whole.xtc
# writing all atoms you will be able to reuse the same pdb for opening.
# e.g. vmd conf.pdb whole.xtc
```

11.29.5.3 Exercise 3: Reweighting your free energy

Now reweight your free energy and compute it as a function of:

- distance between Mg and phosphate
- distance between Mg and geometric center of RNA
- coordination number between Mg and water

The free energy as a function of the distance between Mg and geometric center of RNA can be used to identify the bulk region. In order to do so, normalize it adding the correct entropic term $k_B T \log d^2$, and find a region where the free energy is approximately constant to represent the bulk region. You can for instance use the following commands in gnuplot

```
gnuplot> p "fes_dc" u 1:2 , "" u 1:($2+2.5*log($1**2))
```

Below you can find reference results
Also try to compute conditional free energies:

- coordination number between Mg and water *assuming Mg is bound to phosphate*.

- coordination number between Mg and water *assuming Mg is in the bulk*.

A possible way to do so you can use UPDATE_IF to extract portions of trajectory such that the Mg is bound or unbound.
Below you can find reference results

11.29.5.4 Exercise 4: Standard affinity

Now use the weights that you computed in the previous exercise to compute the absolute binding affinity of the Mg to the phosphate. In order to do so you should compute the relative probability of seeing the Mg bound to the phosphate and in the bulk region and normalize to 1 mol/M concentration.
For instance, if you define bulk the region between 1.5 and 2.5 nm, you should divide the weight of the unbound state by a factor $\frac{4\pi}{3}(2.5^3 - 1.5^3)/V_{mol}$ where $V_{mol} = 1.66nm^3$ is the volume corresponding to the inverse of 1 mol/L concentration. The absolute binding affinity is then defined as $-k_B T \log \frac{w_B}{w_U}$. You should obtain a value of approximately 50.4 kj/mol

Warning

The trajectory is too short (approx 20ns) to obtain converged results! To get statistically significant numbers, please run it longer. Also notice that the result might depend a lot on the used force field.

11.30 Lugano tutorial: Computing proton transfer in water

11.30.1 Aims

In this tutorial I will show you how you can use PLUMED and metadynamics in combinations with CP2K.

11.30.2 Objectives

Once this tutorial is completed students will

- Know how to enhance the sampling in an ab initio simulation.

11.30.3 Resources

The TARBALL for this project contains the following files:

- H-transfer.inp: a CP2K input file to perform BO-MD and Free Energy calculations with PLUMED
- H-transfer.pdb: a PDB file with the starting configuration for a few water molecules

This tutorial has been tested on v2.5 but it should also work with other versions of PLUMED.

11.30.4 Introduction

For this tutorial we will consider a practical application. The aim is that of studying proton transfer in water. The system is simplified and the accuracy of the ab initio simulation is not production like so do not reuse the CP2K input for real-life applications.

11.30.5 Exercises

In this example the system is initially in a configuration where there are H3O+ and an OH- molecule separated by a few other water molecules. In a standard MD the system will quickly equilibrate. Here the aim is to use metadynamics to estimate the free energy of this process and to understand the role of the solvent.
When you download CP2K it already includes the interface for PLUMED but it must be compiled using the proper flags, check CP2K installation instructions.

11.30.6 Exercise 1: Preliminary run

The starting configuration is represented in the H-transfer.pdb. While a simple CP2K input file to perform BO-MD is written in H-transfer.inp. In particular here the section to enable PLUMED is commented out initially.

```
# &FREE_ENERGY
#   &METADYN
#     USE_PLUMED .TRUE.
#     PLUMED_INPUT_FILE ./plumed.dat
#   &END METADYN
# &END FREE_ENERGY
```

To run a preliminary simulation it is enough to:

```
cp2k.sopt H-transfer.inp >& log &
```

use the plumed driver and VMD to choose to water molecule relatively far apart to study the proton transfer.

11.30.7 Exercise 2: Proton transfer

In this exercise you are challenged to

- Select two water molecule to be kept far apart using LOWER_WALLS and the distance between their two oxygen atoms.
- Use the distances of an hydrogen from both oxygen atoms to setup a first METAD calculation to study the proton transfer between the selected water molecules
- Test more complex CVs maybe taking into account the role of the other molecules.
- Think how to study proton transfer in a general way instead than between two specific water molecules and using a specific hydrogen.

The number of steps in the simulation is initially set to 100, this should be increased so to allow a more extensive exploration of the conformational space. This simulation can be slow and is better run on a workstation with multiple processors.

11.31 Lugano tutorial: Folding free energy for a protein described by a go-model

11.31.1 Aims

The aim of this tutorial is to train users to learn the syntax of complex collective variables and use them to analyze MD trajectories of realistic biological systems and bias them with metadynamics.

11.31.2 Objectives

Once this tutorial is completed students will be able to:

- Write the PLUMED input file to use complex CVs for analysis
- Analyze trajectories and calculate the free energy of complex biological systems
- Perform error analysis and evaluate convergence in realistic situations
- Setup, run, and analyze metadynamics simulations of a complex system

11.31.3 Resources

The TARBALL for this tutorial contains the following files:

- GB1.tpr: a GROMACS run file to perform MD simulation of GB1
- GB1_native.pdb: a PDB file for the folded structure of GB1
- GB1_smog.top: the GROMACS topology file
- GB1.mdp: the gromacs simulation parameters file
- GB1_start.gro: the starting configuration of the simulation

11.31.4 Introduction

In this tutorial we propose exercises on the protein G B1 domain described using a structure-based potential obtained using SMOG (smog-server.org)

11.31.5 Exercise 1: Protein G folding simulations

GB1 is a small protein domain with a simple beta-alpha-beta fold. It is a well studied protein that folds on the millisecond time scale. Here we use a structure based potential and well-tempered metadynamics to study the free energy of folding and unfolding. In the TARBALL of this exercise we provide the files needed to run the simulation, the user should write the plumed input file needed to bias the sampling.
The users are expected to:

- setup and perform a well-tempered metadynamics simulation
- evaluate convergence and error calculation of the metadynamics simulation
- calculate the free energy difference between the folded and unfolded state of this protein
- evaluate the robustness of the former by reweighting the resulting free energy as function of different CVs

The users are free to choose his/her favorite CVs and they are encouraged to use the on-line manual to create their own PLUMED input file. However, we encourage all the users to experiment at least with the following CVs to characterize the free-energy landscape of GB1:

- RMSD with respect to the folded state
- DRMSD with respect to the folded state
- GYRATION
- ALPHABETA
- DIHCOR

Unfortunately secondary structure collective variables cannot be used in this case because the model does not include hydrogen atoms.
The users should first perform a preliminary simulations and use this to select one or more CV to be later employed for a METAD or RESTRAINT (umbrella sampling) simulation. Once you are satisfied by the convergence of your simulation, you can use one of the reweighting algorithms proposed to evaluate the free-energy difference between folded and unfolded state as a function of multiple collective variables.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/aaaa-lugano-6d.txt
#SETTINGS MOLFILE=user-doc/tutorials/lugano-6d/GB1_native.pdb
#this allows you to use short-cut for dihedral angles
MOLINFO STRUCTURE=GB1_native.pdb

#add here the collective variables you want to bias

#add here the RESTRAINT or METAD bias, remember that for METAD you need to set: one SIGMA per CV, one single H
#Using GRIDS can increase the performances, so set a as many GRID_MIN and GRID_MAX as the number of CVs with r
#(i.e an RMSD will range between 0 and 3, while ALPHABETA and DIHCOR will range between 0 and N of dihedrals).

#add here the printing and/or other analyssi
```

11.31.6 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Analyze trajectories of realistic biological systems using complex CVs
- Extract conformations that correspond to local free-energy minima
- Apply block analysis to estimate error in the reconstructed free-energy profiles
- Calculate ensemble averages of experimental CVs
- Reweight well-tempered metadynamics simulations

11.32 Using Hamiltonian replica exchange with GROMACS

When patching GROMACS with PLUMED, it is also possible to perform Hamiltonian replica exchange with different topologies. Although this feature is provided together with PLUMED, it is actually a new feature for GROMACS itself that can be enabled using the `-hrex` flag of `mdrun`. This implementation is very close to the one used to produce the data in this paper [138]. In case you find it useful, please cite this paper.

Warning

> This feature is currently used by several groups and should be robust enough. However, be sure that you understand its limitations and to perform all the tests discussed in this page before using it in production.

In this short tutorial you will learn how to do two things:

- Generate scaled topologies with `plumed partial_tempering`. This is actually not directly related to the `-hrex` flag.
- Run GROMACS with replica exchange and multiple topologies.

11.32.1 Generate scaled topologies

Plumed comes with a `partial_tempering` command line tool that can be used to generate scaled topologies. Notice that you might want to generate these topologies by yourself. This step is totally independent from the use of the `-hrex` flag.
The `partial_tempering` tool can be invoked as follows:

```
plumed partial_tempering scale < processed.top
```

where scale is the Hamiltonian scaling factor and processed.top is a post-processed topology file (i.e. produced with grompp -pp) where each "hot" atom has a "_" appended to the atom type, e.g. change this

```
     1          HC      1    ACE    HH31      1      0.1123      1.008   ; qtot 0.1123
```

to this:

```
     1          HC_     1    ACE    HH31      1      0.1123      1.008   ; qtot 0.1123
```

Notice that the section that should be edited is the [atoms] section for all the molecules that you wish to affect (typically only for the solute, but you may also want to change solvent parameters). If you select all the atoms of the solute, you will be able to prepare topologies such as those needed in a REST2 simulation [139].
Also remember to first produce the processed.top file with grompp -pp. Editing a normal topol.top file will not work, because it does not contain all the parameters. The processed.top file should not have any "#include" statement.

```
# produce a processed topology
grompp -pp
# choose the "hot" atoms
vi processed.top
# generate the actual topology
plumed partial_tempering $scale < processed.top > topol$i.top
```

Warnings:

- It's not very robust and there might be force-field dependent issues!

- It certainly does not work with CHARMM cmap

Suggested tests:

1. Compare partial_tempering with scale=1.0 to non-scaled force field. E.g.

```
grompp -o topol-unscaled.tpr
grompp -pp
vi processed.top # choose the "hot" atoms appending "_". You can choose whatever.
plumed partial_tempering 1.0 < processed.top > topol-scaled.top # scale with factor 1
grompp -p topol-scaled.top -o topol-scaled.tpr
# Then do a rerun on a trajectory
mdrun -s topol-unscaled.tpr -rerun rerun.trr
mdrun -s topol-scaled.tpr -rerun rerun.trr
# and compare the resuling energy files. they should be identical
```

2. Compare partial_tempering with scale=0.5 to non-scaled force field. Repeat the same procedure but using "plumed partial_tempering 0.5". Choose all the atoms in all the relevant [atoms] sections (e.g. solute, solvent and ions). In the two resulting energy files you should see: long range electrostatics, LJ, and dihedral energy is *half* in the scaled case all other terms (bonds/bends) are identical.

11.32.2 Run GROMACS

If GROMACS has been patched with PLUMED it should accept the -hrex option in mdrun. Please double check this (mdrun -h should list this possibility). Notice that not all the versions of GROMACS allow this feature. First of all prepare separate topologies for each replicas using `plumed partial_tempering` tool as shown above (or using some other tool). Then run a normal replica exchange with gromacs adding the flag "-hrex" on the command line.
A complete run script could be adapted from the following

```
# five replicas
nrep=5
# "effective" temperature range
tmin=300
tmax=1000

# build geometric progression
list=$(
awk -v n=$nrep \
    -v tmin=$tmin \
    -v tmax=$tmax \
  'BEGIN{for(i=0;i<n;i++){
    t=tmin*exp(i*log(tmax/tmin)/(n-1));
    printf(t); if(i<n-1)printf(",");
  }
}'
)

# clean directory
rm -fr \#*
rm -fr topol*

for((i=0;i<nrep;i++))
do

# choose lambda as T[0]/T[i]
# remember that high temperature is equivalent to low lambda
  lambda=$(echo $list | awk 'BEGIN{FS=",";}{print $1/$'$((i+1))';}')
# process topology
# (if you are curious, try "diff topol0.top topol1.top" to see the changes)
  plumed partial_tempering $lambda < processed.top > topol$i.top
# prepare tpr file
# -maxwarn is often needed because box could be charged
  grompp_mpi_d  -maxwarn 1 -o topol$i.tpr -f grompp$i.mdp -p topol$i.top
done
mpirun -np $nrep gmx_mpi mdrun_d -v -plumed plumed.dat -multi $nrep -replex 100 -nsteps 15000000 -hrex -dlb no
```

Notice that total cell could be charged. This happens whenever the scaled portion of the system is not neutral. There should be no problem in this. When used with pbc, GROMACS will add a compensating background.
Suggested check:

- Try with several identical force fields (hard code the same lambda for all replicas in the script above) and different seed/starting point. Acceptance should be 1.0

Notice that when you run with GPUs acceptance could be different from 1.0. The reason is that to compute the acceptance GROMACS is sending the coordinates to the neighboring replicas which then recompute energy. If all the tpr files are identical, one would expect energy to be identically to the originally computed one. However, calculations made with GPUs are typically not reproducible to machine precision. For a large system, even an error in the total energy for the last significant digit could be on the order of a kJ. This results in practice in a lower acceptance. The error induced on the final ensemble is expected to be very small.
Warnings:

- Topologies should have the same number of atoms, same masses and same constraint topology. Some of these differences (e.g. masses) are not explicitly checked and might lead to unnoticed errors in the final results.
- Choose neighbor list update (nstlist) that divides replex. Notice that running with GPUs GROMACS is going to change nstlist automatically, be sure that it still divides replex. Only if you are sure that you patched G↩ ROMACS using PLUMED version 2.7 or later, you can assume that GROMACS will automatically adjust the nstlist so as to be compatible with replex. See `#579`.
- It seems that when using multiple processes per replica it is necessary to switch off dynamic load balancing (`-dlb no`) otherwise the simulation could crash randomly, see `#410`.
- Option -hrex requires also option -plumed. If you do not care about plumed, just provide an empty plumed.dat file.
- It should work correctly if replicas have different force-field, temperature, lambda, pressure, in any combination. However, not all these combinations have been tried in practice, so please first test the code on a system for which you know the result.

11.33 Julich tutorial: Developing CVs in plumed

11.33.1 Aims

The aim of this tutorial is to introduce users to the tutorials in the developer manual of PLUMED. We will learn a little bit about the structure of the code and how this structure can be visualized using the tree diagrams in the developer manual. You will then learn how to implement a new collective variable in a way that uses as much of the functionality that is already within the code and that thus avoids the duplication of code. Finally, you will learn how to write regression tests and how these can be used for continuous integration.

11.33.2 Learning Outcomes

By the end of this session you will know how to:

- access the developer manual for PLUMED.
- find the tutorial information for implementing new collective variables, multicolvars, functions, biases and analysis routines.
- find the tree diagrams showing the class structure in the PLUMED developer manual.
- exploit the functionality within `PLMD::multicolvar` in order to write a reasonably complex collective variable quickly.
- write regression tests for PLUMED and understand how these are used for continuous integration.

To do this module you must understand the basics of object-oriented programming. Information on object oriented programming and how it is used within plumed can be found `here` .

11.33.3 Resources

The tarball for this project contains the following directories:

- rt-coord : a directory containing input files for doing a regtest on an already existing collective variable.
- rt-second-shell : a directory containing input files for doing a regtest on the collective variable you will implement within this tutorial

At the start of the exercise you should move these two directories to the plumed2/regtest/multicolvar directory of your PLUMED source. You should see many other directories called rt... within the same directory. If you change into any of these directories and issue the following set of commands:

```
source ../../../sourceme.sh
make
```

Then you will have run one of the regression tests of PLUMED. If the test runs without a hitch you should get an output something like this:

```
../../scripts/run
Thu Aug 20 14:33:00 IST 2015
Running regtest in /Users/gareth/Projects/CVception/plumed2/regtest/multicolvar/rt22
cp: ../tmp is a directory (not copied).
++ Test type: driver
++ Arguments: --plumed plumed.dat --trajectory-stride 10 --timestep 0.005 --ixyz trajectory.xyz --dump-forces
++ Processors: 0
/Users/gareth/Projects/CVception/plumed2/regtest/multicolvar/rt22/tmp
Run driver
Done. Here is the error file:
```

11.33.4 Introduction

PLUMED has two manuals: a manual that is for users of the code and a manual that is for developers of the code. If you are interested in implementing new features in the code your first point of call should thus be the developer manual, which can be found here . Alternatively, you can get to the front page of the developer manual by clicking the USER/DEVELOPER logo in the top right hand corner of any page of the user manual.

One of nicest features of PLUMED for the developer is that all the code and documentation for any new PLUMED command all appears together in a single file. When you come to implement a new feature it is thus relatively unlikely that you will have to modify any of the files you downloaded. Adding a new feature is simply a matter of adding one further cpp file containing the new method and the documentation for this method. We are able to achieve this by exploiting abstract base classes and polymorphism. All classes that calculate collective variables, print these variables or calculate biases thus inherit from a single base class called Action. You can read about the Action class here . Notice also that this page also shows how all the various classes within the code inherit from this single base class. It is perhaps worth spending a little while browsing through the various branches of this tree and understanding how the classes at each level become increasingly specialized and thus fit for particular purposes.

Lets say you want to implement a new collective variable in PLUMED. One way to start this task would be to write a new class that inherits from the PLMD::Colvar base class.

If you click on the box in the tree for PLMD::Colvar and follow various links on the various subject pages you will eventually get to the following page , which will give you a step-by-step set of instructions for implementing a new collective variable. If you look through the manual you can find similar pages that provide you with instructions for implementing new analysis methods, functions and so on. Our suggestion when you implement something new would thus be to find some similar functionality in the code, look at how it is implemented and to look in the developer manual at the descriptions of the classes that have been inherited. This process is what we will try and take you through in this short tutorial.

11.33.5 Instructions

11.33.5.1 Calculating a reasonably complex collective variable

In this first exercise I would like you to look through the manual and work out how to use the functionality that is already available in PLUMED to calculate the following collective variable:

$$s = \sum_{i=1}^{108} 1 - \frac{1 - \left(\frac{c_i}{4}\right)^6}{1 - \left(\frac{c_i}{4}\right)^{12}}$$

where c_i is equal to the coordination number of atom i, i.e.:

$$c_i = \sum_{j \neq i} \frac{1 - \left(\frac{r_{ij}}{1}\right)^6}{1 - \left(\frac{r_{ij}}{1}\right)^{12}}$$

So r_{ij} is the distance between atom i and atom j. This collective variable measures the number of coordination numbers that are more than 4 so the summations in the above expressions run over all 108 atoms in this particular system.
Write an input file that computes s and outputs its value to a file called colvar and that computes both the analytical and numerical derivatives of s and outputs this information to a file called deriv. You are going to run this calculation in the rt-coord directory that you downloaded with the tarball for this exercise so you should create this input file within that directory. You will need to output the quantities in the colvar and deriv files with only four decimal places. When you are content with your input run the calculation by typing make. If you have done everything correctly you should see the output that was discussed above.
In setting up your input files you may find it useful to watch our introductory video and this video on some of the features that are available in the code. Obviously, the user manual will be indispensable as well.

11.33.5.2 Implementing a new collective variable

In this second exercise I would like you to implement the following collective variable:

$$s = \sum_{i=1}^{108} \int_{57}^{59} \exp\left(-\frac{(x_i - x)^2}{2}\right) \mathrm{d}x$$

where x_i is number of atoms in the second coordination sphere of the ith atom, i.e.:

$$s = \sum_{i \neq j} \int_{2}^{4} \exp\left(-\frac{(r_{ij} - r)^2}{2}\right) \mathrm{d}r$$

So r_{ij} is the distance between atom i and atom j. This collective variable measure the number of atoms that have between 57 and 59 atoms in their second coordination sphere. There are a number of observations we can make here that will help you enormously:

- Notice that CV is similar to the coordination number CV you calculated in the first example. For both CVs there is a sum over a quantity that is calculated for different sets of atoms. You should thus look at what was done by the routines that calculate the first CV and see what you can reuse.
- Notice that with the first CV we can calculate the number of coordination numbers that are within a certain range as well as the number of coordination numbers that are less than a certain threshold.
- You can set the link cell cutoff equal to

```
std::numeric_limits<double>::max()
```

- Lastly, this tool will prove very useful.

Write an input file that computes s and outputs its value to a file called colvar and that computes both the analytical and numerical derivatives of s and outputs this information to a file called deriv. When outputting numbers to colvar and deriv you should output numbers to four and three decimal places respectively. You are going to run this calculation in the rt-second-shell directory that you downloaded with the tarball for this exercise so you should create this input file within that directory. When you are content with your input run the calculation by typing make. If you have done everything correctly you should see the output that was discussed above.
If you have sufficient time try use doxygen within PLUMED to write the documentation for your new collective variable.

11.33.6 Final thoughts

The aim of this tutorial is not so much to implement the CV. I do not think it is a particularly interesting or useful CV. Instead the hope is that by doing it you will get some idea of how to implement things within PLUMED. Based on my experience there are three key things I wish I could go back and tell my younger self about programming, which I would urge you to learn now:

- **Write less code** The more code you write the more bugs you generate. To be totally clear this is not me trying to say I am better at coding than you are - the same statement holds true if I replace each you in the above with an I - it holds true for everyone. In addition, there are lots of smart people out in the world writing code (and again I am not talking about the developers of plumed). Their code is properly tested and faster than anything that you will write so spend your time learning how it works so that you can re-use it.

- **Test your code** Notice that I set up directories that you could use to test your code for this tutorial. Testing should always be part of your development work flow and coming up with ways to test your code is hard, which is why we often don't do it enough. Also don't develop stuff in a vacuum get in touch with people who are using the code. You need to test your code on systems that people actually want to simulate and not just on dummy problems. In addition, you should learn to use profiling tools such as Valgrind and instruments on the mac so that you can find memory leaks and bottlenecks in your code. When you start doing this properly you will realize that you cannot possibly properly maintain all the code that you have written and you will see clearly that you that you need to write less code.

- **Write documentation for your code** Undocumented software is useless. You need to explain how to use your code and to fill your manual with examples of how the code can be used to solve specific and interesting problems. That is to say that you need to provide examples of calculations that users actually want to perform. To be clear here writing a manual that tells users how something like the shake algorithm works in general terms is actually pretty useless. After all a user can go off and find out this information from a textbook, you could thus replace your description with a link to a textbook. What users want from you is an example that is similar to the calculation they actually want to set up. If I were providing documentation for an implementation of shake in an MD code I would thus explain in detail how to set up shake to constrain all the angles and bonds in all the water molecules in my system as it is a fair bet that this is how it will be being used by many users.

I hope you have seen from this tutorial that PLUMED tries to help you with all of these aims. We have a developer manual where we try to explain how to use the code that is already there. In addition, you can quite quickly generate nicely formatted user documentation for new CVs and it is easy to add new regression tests that allow you to test new features. Notice also that whenever commits are made on the origin git repository we use a website called `travis-ci` to test that the code works across a range of platforms.

11.34 ESDW Workshop Lyon 2019: A very simple introduction to PLUMED

11.34.1 Aims

The aim of this tutorial is to give you a quick introduction to the PLUMED syntax and to some of the types of calculations that can be done with PLUMED. It was prepared for a two and a half hour session on PLUMED in the following CECAM extended software development workshop on topics in classical MD:
`https://www.cecam.org/workshop-1802.html`
This tutorial is very straightforward and not at all exhaustive. In fact it skims over many of the subtleties associated with performing these types of calculations using PLUMED. We have thus suggested additional topics that you might like to look into throughout the tutorial in order to take this further.
If you are interested in using the techniques described in the tutorial in your own work a good starting point might be to read the chapter we wrote about using metadynamics from within PLUMED:
`https://arxiv.org/abs/1812.08213`

11.34.2 Objectives

Once this tutorial is completed students will be able to:

- Run simulations using the internal PLUMED MD engine simplemd.
- Use PLUMED to add additional forces in order to prevent clusters from dissociating.
- Use PLUMED to calculate and PRINT the value of a collective variable.
- Use the PLUMED HISTOGRAM and CONVERT_TO_FES actions to calculate an estimate of the free energy as a function of some collective variables.
- Run metadynamics using the action METAD in order to enhance sampling along collective variables of interest.

11.34.3 Resources

The TARBALL for this project contains the following files:

- in : The input file for simplemd that contains the parameters for the MD simulation.
- input.xyz : An initial configuration for the cluster that we are studying in this tutorial.
- plumed.dat : An empty input file for PLUMED

This tutorial has been tested on v2.5 but it should also work with other versions of PLUMED.
Also notice that the `.solutions` direction of the tarball contains correct input files for the exercises. Please only look at these files once you have tried to solve the problems yourself. Similarly the tutorial below contains questions for you to answer that are shown in bold. You can reveal the answers to these questions by clicking on links within the tutorial but you should obviously try to answer things yourself before reading these answers.

11.34.4 Introduction

In this tutorial we are going to study a very simple physical system; namely, seven Lennard Jones atoms in a two dimensional space. This simple system has been extensively studied as has often been used to benchmark new simulation techniques. In addition, the potential energy landscape has been fully characterized and it is known that only the four structurally-distinct minima shown below exist:
In the exercises that follow we are going to learn how to use PLUMED to determine the relative free energies of these four structures by running molecular dynamics simulations.

11.34.5 Exercises

Before completing the following exercises you will need to download and install PLUMED. In addition, you will need to ensure that the PLUMED executable can be found within your path.
You can download PLUMED from:
`http://www.plumed.org/get-it` or `https://github.com/plumed/plumed2`
If you are running PLUMED on a Mac you can install the executable directly using macports and the command:

```
sudo port install plumed
```

If you download the source code you can find instructions on compilation in the PLUMED page on Installation. If you run make install PLUMED or install with macports PLUMED will be within your path.
If, however, you do not wish to do a make install you can simple make PLUMED. You can then add the location of the PLUMED executable to your path by issuing the command:

```
source sourceme.sh
```

from within the plumed2 directory.

11.34.5.1 Using PLUMED as an MD code

PLUMED is primarily used as a plugin to other larger molecular dynamics codes. Mainly for software testing and educational purposes, however, we do have an MD code with a rather limited set of options within PLUMED. You can run this code by issuing the command:

```
plumed simplemd < in
```

where in here is the input file from the tar ball for this tutorial, which is shown below:

```
nputfile input.xyz
outputfile output.xyz
temperature 0.5
tstep 0.005
friction 0.1
forcecutoff 2.5
listcutoff  3.0
ndim 2
nstep 200000
nconfig 1000 trajectory.xyz
nstat   1000 energies.dat
```

This input instructs PLUMED to perform 200000 steps of MD at a temperature of $k_BT = 0.5\epsilon$ starting from the configuration in input.xyz. The timestep in this simulation is 0.005 $\sqrt{\epsilon m\sigma^2}$ and the temperature is kept fixed using a Langevin thermostat with a relaxation time of $0.1\sqrt{\epsilon m\sigma^2}$. Trajectory frames are output every 1000 MD steps to a file called trajectory.xyz. Notice also that in order to run the calculation above you need to provide an empty file called plumed.dat. This file is the input file to the PLUMED plugin itself, which, because this file is empty, is doing nothing when we run the calculation above.

Run a calculation using simplemd and the input above and visualize the trajectory that is output. Describe what happens during this calculation and explain why this is happening.

You can visualize what occurs during the trajectory by using a visualization package such as VMD (https://www.ks.uiuc.edu/Research/vmd/). If you are using VMD you can see the MD trajectory by using the command:

```
vmd trajectory.xyz
```

You should observe that all the atoms fly apart early on in the simulation and that the cluster evaporates. The cluster evaporates because at a temperature of $k_BT = 0.5\epsilon$ the gas state has a lower free energy than than the cluster state.

Change the parameters in the input file for simplemd so as to prevent the cluster from evaporating.

To prevent the cluster from evaporating you need to lower the temperature in the file in. The cluster will not evaporate if the temperature is set equal to $k_BT = 0.2\epsilon$.

11.34.5.2 Using PLUMED to specify a force field

If we lower the temperature of the simulation very little will happen. Yes the cluster will no longer evaporate but at the same time we will not see any transitions between the various basins in this energy landscape. In this section we are thus going to see how we can use PLUMED to add a bias potential that can prevent the cluster from exploring gaseous configurations that do not interest us. In other words, we are going to see how we can use PLUMED to add restraints that will prevent the cluster from evaporating. Before getting on to that, however, we first need to understand a little about how the PLUMED input file works. As explained in The structure of a PLUMED input file and Getting Started when you write a PLUMED input file you are writing a script that calculates functions of the atomic positions.

Read the description of the PLUMED input syntax in the linked pages in the paragraph above and then attempt the exercise that follows

We are going to prevent the cluster from evaporating by adding a restraint on all the atoms so as to prevent them from moving more than 2σ from the center of mass of the cluster. As the masses of all the atoms in the cluster are the same we can compute the position of the center of mass using:

$$\mathbf{x}_{\rm com} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

where $\mathbf{x}_i$ is the position of the ith atom. The distance between the ith atom and the position of this center of mass, d_i, can be computed using Pythagoras' theorem. These distances are then restrained by using the following potential:

$$V(d_i) = \{ 1\ 00 * (d_i - 2.0)^2 \text{if} \quad d_i > 20 \text{otherwise}$$

as you can see this potential has no effect on the dynamics when these distances are less than 2 ϵ. If any atom is more than 2 ϵ from the center of mass, however, this potential will drive it back towards the center of mass. The following cell contains a skeleton input file for PLUMED that gets it to calculate and apply this bias potential.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/lyon.txt
# this optional command tells VIM that this is a PLUMED file and to color the text accordingly
# vim: ft=plumed

# This tells PLUMED we are using Lennard Jones units
UNITS NATURAL

# Calculate the position of the center of mass.  We can then refer to this position later in the input using t
com: COM __FILL__

# Add the restraint on the distance between com and the first atom
d1: DISTANCE __FILL__
UPPER_WALLS ARG=d1 __FILL__

# Add the restraint on the distance between com and the second atom
d2: DISTANCE __FILL__
UPPER_WALLS  __FILL__

# Add the restraint on the distance between com and the third atom
d3: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fourth atom
d4: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the fifth atom
d5: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the sixth atom
d6: DISTANCE __FILL__
UPPER_WALLS __FILL__

# Add the restraint on the distance between com and the seventh atom
d7: DISTANCE __FILL__
UPPER_WALLS __FILL__
```

Copy and paste the content above into the file plumed.dat and then fill in the blanks by looking up the documentation for these actions online and by reading the description of the calculation that you are to run above. Once you have got a working plumed.dat file run a calculation using simplemd again at a temperature of $k_B T = 0.2\epsilon$ and check to see if the bias potential is indeed preventing the cluster from evaporating.

11.34.5.2.1 Extensions

Try the following exercises if you have time and if you wish to take this exercise further:

- See if you can rewrite the input above in a more compact way by making use of DISTANCES and UWALLS.
- Checkout the hack-the-tree branch of PLUMED. This version of PLUMED allows you to pass vector quantities between actions as well as scalars. See if you can rewrite the above input using this version of PLUMED by making use of the MATHEVAL and BIASVALUE actions.

11.34.5.3 Calculating collective variables

We can easily determine if there are transitions between the four possible minima in the Lennard-Jones-seven potential energy landscape by monitoring the second and third central moments of the distribution of coordination

numbers. The nth central moment of a set of numbers, $\{X_i\}$ can be calculated using:

$$\mu^n = \frac{1}{N}\sum_{i=1}^{N}(X_i - \langle X \rangle)^n \qquad \text{where} \qquad \langle X \rangle = \frac{1}{N}\sum_{i=1}^{N} X_i$$

Furthermore, we can compute the coordination number of our Lennard Jones atoms using:

$$c_i = \sum_{i \neq j} \frac{1 - \left(\frac{r_{ij}}{1.5}\right)^8}{1 - \left(\frac{r_{ij}}{1.5}\right)^{16}}$$

where r_{ij}__FILL__ is the distance between atom i and atom j. The following cell contains a skeleton input file for PLUMED that gets it to calculate and print the values these two quantities take every 100 time steps to a file called colvar.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/lyon.txt
c1: COORDINATIONNUMBER SPECIES=__FILL__ MOMENTS=__FILL__ SWITCH={RATIONAL __FILL__ }
PRINT ARG=__FILL__ FILE=colvar STRIDE=100
```

Copy and paste the content above into your plumed.dat file after the lines that you added in the previous exercise that added the restraints to stop the cluster from evaporating. Fill in the blanks in the input above and, once you have a working plumed.dat file, run a calculation using simplemd at a temperature of $k_BT = 0.2\epsilon$. Once you have generated the trajectory plot the time series of second and third moment values using gnuplot or some other plotting package.

11.34.5.3.1 Extensions

Try the following exercises if you have time and if you wish to take this exercise further:

- Investigate how SPRINT coordinates are defined and calculated using PLUMED and try to compute them for this particular system.
- Read about other collective variables that can be calculated using PLUMED. A topic that is particularly important to understand is how PLUMED deals with periodic boundary conditions.

11.34.5.4 Estimating the free energy surface

When you plotted the time series for the collective variables in the previous exercise you should have seen that the values of these collective variables fluctuate around a number of different constant values for long periods of time. There are then infrequent jumps and after these jumps the value of the collective variable settles down and begins to fluctuate around a different constant value. There are jumps in the value of the CV whenever the system moves from one minimums illustrated above to another. What we would ideally like to extract from our simulation are the relative free energies of these various minimums. From elementary statistical mechanics, however, we know that the free energy is given by:

$$F(s) = -k_B T \ln P(s)$$

Consequently, if we can extract an estimate of the probability that the CVs take particular values, s, then we can get an estimate of the free energy. Extracting an estimate that for the probability that the collective variables take a particular value is easy, however. We just need to construct a histogram from the CV values that the system took during the trajectory. The following cell contains a skeleton input file for PLUMED that gets it to estimate this histogram and that converts and outputs this estimate of the histogram to a file called fes.dat.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/lyon.txt
hh: HISTOGRAM __FILL__ __FILL__=-1.0,-1.0 __FILL__=1.5,1.5 GRID_BIN=300,300 __FILL__=DISCRETE STRIDE=100
fes: CONVERT_TO_FES GRID=_FILL__ __FILL__=0.2
DUMPGRID GRID=__FILL__ FILE=fes.dat
```

Copy and paste the content above into your plumed.dat file after the lines that you added in the previous two exercises. Fill in the blanks in the input above and, once you have a working plumed.dat file, run a calculation using simplemd at a temperature of $k_BT = 0.2\epsilon$. Once you have generated the trajectory create a contour plot of the free energy surface that is contained in fes.dat using gnuplot or some other plotting package.

The free energy surface that you get from this calculation should look something like the figure shown on the left below. Please note that in order to fully sample configuration space and in order to get a reasonably converged estimate of the free energy you are going to have to use many more time steps than you did in previous exercises.

The figure shown on the right is the free energy surface obtained if you run at a temperature of $k_B T = 0.1\epsilon$. As you can see a much smaller volume of configuration space is explored at this lower temperature

Rerun the calculation of the free energy surface at a lower temperature. What differences do you observe at the lower temperature? Why might these differences be problematic?

When you run the simulation at a lower temperature you should see that the system no longer explores all the basins in the free energy landscape. If the temperature is sufficiently low the system will remain trapped in the original configuration throughout the whole simulation run. At very low temperatures this may be the correct behavior and the free energy estimate you obtain may prove to be a reasonable estimate. At the same time, however, lowering the temperature makes the barrier crossing a rarer event. It thus becomes less likely that you will observe a crossing between basins in a lower temperature simulation. The fact that the histogram shows that the probability of observing these configurations is zero may thus simply be a consequence of the fact that the simulation is not converged. The free energy will thus look like the one shown below instead of looking like the one in the previous hidden section:

11.34.5.4.1 Extensions

Try the following exercises if you have time and if you wish to take this exercise further:

- Try to recalculate the free energy surface but use kernel density estimation when calculating the HISTOGRAM.
- Work through Trieste tutorial: Averaging, histograms and block analysis and try to apply the ideas this tutorial covers to calculate appropriate error bars the estimate of the free energy surface that you extracted in the previous exercises.

11.34.5.5 Performing metadynamics

We can use an enhanced sampling algorithm to overcome the problems that we encountered at the end of the last exercise and to more fully explore configuration space even at low temperature. In this final exercise we are thus going to perform a metadynamics simulation on our Lennard Jones cluster. In metadynamics the system is forced to explore more of configuration space by a history dependent bias that has the following functional form:

$$V(s,t) = \sum_{\tau=0}^{t} w(\tau) \exp\left(-\frac{(s(t) - s(\tau))^2}{2\sigma^2} \right)$$

This bias potential consists of a sum of Gaussian kernels with bandwidth σ that are centered at the values of the CV that the system took at previous times. The weights of the kernels in this sum are time dependent because we are going to use the well-tempered variant of metadynamics which introduces one further parameter $\gamma = \frac{T+\Delta T}{T}$.

To run metadynamics on the Lennard Jones cluster that we have been studying in this exercise using PLUMED you would delete the lines that you used to compute the histogram from the plumed input file that we have been using thus far and you would add a line like the one shown below.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/lyon.txt
METAD ARG=__FILL__ HEIGHT=__FILL__ PACE=__FILL__ SIGMA=__FILL__ GRID_MIN=-1.5,-1.5 GRID_MAX=2.5,2.5 GRID_BIN=5
```

This input should be modified to instruct PLUMED to add Gaussian kernels with a bandwidth of 0.1 in both the second and third moment of the distribution of coordination numbers and a height of 0.05 ϵ every 50 MD steps. The metadynamics calculation should then be run using simplemd at a temperature of $k_B T = 0.1\epsilon$.

A nice feature of the metadynamics method is that, if the simulation is converged, the final bias potential is a related to the underlying free energy surface by:

$$F(s) = -\frac{T + \Delta T}{\Delta T} V(s,t)$$

This formula is implemented in the tool sum_hills that is part of the PLUMED package. To run sum_hills on the output of your metadynamics simulation you can run the command:

```
plumed sum_hills --hills HILLS
```

This command should output a file called fes.dat that contains the final estimate of the free energy surface that was calculated using metadynamics.

Try to run the sum_hills command on the HILLS file output by your metadynamics simulation. Visualize the fes.dat file that is output by using gnuplot or some similar plotting package.

11.34.5.5.1 Extensions Try the following exercises if you have time and if you wish to take this exercise further:

- Work through Trieste tutorial: Averaging, histograms and block analysis and try to apply the ideas this tutorial covers to calculate an estimate of the free energy surface using metadynamics and reweighting.
- Try to estimate the errors on your estimate of the free energy surface.

11.34.6 Conclusions

This exercise has hopefully given you a sense of the sorts of calculations that can be performed using PLUM↩ ED. Hopefully you now have an understanding of how PLUMED input files are structured so that variables can be passed between different actions. Furthermore, you should have seen how PLUMED can be used both to analyze trajectories and to add in additional forces that force the system to undergo transitions that it would not otherwise undergo.
Obviously, given the short time available to us in this tutorial, we have only scratched the surface in terms of what PLUMED can be used to do and in terms of how metadynamics calculations are run in practice. If you want to understand more about how to run PLUMED we would recommend that you start by reading the chapter that we provided a link to in the introduction to this tutorial. For now though well done on completing the exercise and good luck with your future PLUMED-related endeavors.

11.35 MARVEL tutorial: Analyzing CVs

11.35.1 Aims

The aim of this tutorial is to introduce you to the PLUMED syntax. We will go through the writing of input files to calculate and print simple collective variables. We will then discuss how we can use PLUMED to analyze a trajectory by calculating ensemble averages, histograms and free energy surfaces.

11.35.2 Learning Outcomes

Once this tutorial is completed students will:

- Know how to write PLUMED input files that can be used to calculate and print collective variables.
- Be able to calculate a collective variable that take the position of center of mass as input.
- Know how to write a PLUMED input file that can be used to calculate an ensemble average.
- Know how to write a PLUMED input file that can be used to calculate a histogram. Students will also learn how this histogram can be converted into a free energy surface.

11.35.3 Resources

The `tarball` for this project contains the following files:

- trajectory-short.xyz : a (short) trajectory for a 16 residue protein in xyz format. All the calculations with plumed driver that will be performed during this tutorial will use this trajectory.
- template.pdb : a single frame from the trajectory that can be used in conjunction with the MOLINFO command
- in : An input file for the simplemd code that forms part of PLUMED
- input.xyz : A configuration file for Lennard-Jones solid with an FCC solid structure

11.35.4 Instructions

PLUMED is a library that can be incorporated into many MD codes by adding a relatively simple and well documented interface. Once it is incorporated you can use PLUMED to perform a variety of different analyzes on the fly and to bias the sampling in the molecular dynamics engine. PLUMED can also, however, be used as a standalone code for analyzing trajectories. If you are using the code in this way you can, once PLUMED is compiled, run the plumed executable by issuing the command:

```
plumed <instructions>
```

Let's start by getting a feel for the range of calculations that we can use PLUMED to perform. Issue the following command now:

```
plumed --help
```

What is output when this command is run is a list of the tasks you can use PLUMED to perform. There are commands that allow you to patch an MD code, commands that allow you to run molecular dynamics and commands that allow you to build the manual. In this tutorial we will mostly be using PLUMED to analyze trajectories, however. As such most of the calculations we will perform will be performed using the driver tool. Let's look at the options we can issue to plumed driver by issuing the following command:

```
plumed driver --help
```

As you can see we can do a number of things with plumed driver. For all of these options, however, we are going to need to write a PLUMED input file. Before we get on to writing input files for PLUMED there is information Codes interfaced with PLUMED here which provides details on what the other PLUMED tools do and instructions for how to interface PLUMED with an MD code. You may like to look at this information now or you might prefer to return after you have finished the exercises here.

11.35.4.1 The PLUMED internal units

By default the PLUMED inputs and outputs quantities in the following units:

- Energy - kJ/mol
- Length - nanometers
- Time - picoseconds

If you want to change these units you can do this using the UNITS keyword.

11.35.4.2 Introduction to the PLUMED input file

Many input files for PLUMED provides specifications for one or more CVs. These specifications are then followed by an instruction to PLUMED to PRINT these CVs and a termination line. Comments are denoted with a # and the termination of the input for PLUMED is marked with the keyword ENDPLUMED. Any words that follow the E↩ NDPLUMED line are ignored by PLUMED. You can also introduce blank lines as these will not be interpreted by PLUMED.
The following input can be used analyze the DISTANCE between the two terminal carbons of a 16 residues peptide. The PRINT command after the DISTANCE command ensures that the results (i.e. the distances calculated) are printed into the file named COLVAR.

```
#my first PLUMED input:
e2edist: DISTANCE ATOMS=2,253

#printout frequency
PRINT ARG=e2edist STRIDE=1 FILE=COLVAR

#endofinput
ENDPLUMED
here I can write what I want it won't be read.
```

Let's use this simple input file to analyze the trajectory included in the RESOURCES. To do so copy the input above into a file called plumed.dat and then issue the command below:

```
plumed driver --plumed plumed.dat --ixyz trajectory-short.xyz --length-units 0.1
```

Notice the –length-units 0.1 flag here. This tells PLUMED to convert the positions in the xyz file here, which are in Angstroms, into nm, which remember are The PLUMED internal units

When this command finishes running you should have a file called COLVAR. If you look at it's contents (using the command more COLVAR for instance) you will find that the first two lines read:

```
#! FIELDS time e2edist
 0.000000 2.5613161
```

The first line of the COLVAR files tells you what values are in each of the columns. The remaining lines then tell you the values these quantities took in the various trajectory frames. We can plot this data using gnuplot (or our favorite graphing program) by issuing the following commands:

```
gnuplot
p 'COLVAR' u 1:2 w l
```

What this graph shows is the value of the distance that we calculated using PLUMED as a function of time. As you can see the distance fluctuates about as the atoms in our system move about in accordance with the various forces that act upon them.
Right so hopefully that wasn't too hard. What we are going to next is we are going to try to understand the input file that we have written a bit better.

11.35.4.3 The PLUMED input syntax

The input file that we issued in the last section looked something like this:

```
e2edist: DISTANCE ATOMS=2,253
PRINT ARG=e2edist STRIDE=1 FILE=COLVAR
```

What happens if we reverse the order of the two commands in the input file. In other words, what would have happened if we had run with an input file that looked like this:

```
PRINT ARG=e2edist STRIDE=1 FILE=COLVAR
e2edist: DISTANCE ATOMS=2,253
```

Run the input file above using the commands described in the previous section to find out.
If everything is working correctly the input above should crash with the following error message:

```
+++ Internal PLUMED error
+++ file Action.cpp, line 234
+++ message: ERROR in input to action PRINT with label @0 : cannot find action named e2edist (hint! the action
```

Take a moment to read that error message and to try to think what it might mean before reading on.
To understand the error lets look at the correct input again:

```
e2edist: DISTANCE ATOMS=2,253
PRINT ARG=e2edist STRIDE=1 FILE=COLVAR
```

You should think of the PLUMED input syntax as a kind of scripting language with commands and variables. The first line in the above file thus tells PLUMED to calculate the distance between atoms 2 and 253 and **to store the value of this distance in a variable called e2edist** . The fact that this quantity is stored in a variable is important as it ensures that we can access this quantity when we come to issue commands later in the input file. So in the second line above we print a quantity. What quantity should be printed? e2edist - the distance between atom 2 and atom 253. Easy!
So why does the second input, that has the order of these two commands reversed, not work? Well the variable e2edist has to be defined before it can be used because in PLUMED the commands are executed in the same order as they are defined in the input file. We thus cannot do anything with e2edist (i.e. PRINT it) without first explaining how it is calculated.
This input demonstrates the key idea of the PLUMED syntax. Quantities calculated by commands such as DISTANCE are given labels (e2edist) so that they can be reused when performing other commands. This idea is discussed in more depth in the following video https://www.youtube.com/watch?v=PxJP16qNC↩Ys.
If you understand this idea though you are 90% of the way to understanding how to used PLUMED. Well done.

11.35.4.4 Center of mass positions

When calculating many collective variables it is useful to not think in terms of calculating them directly based on the positions of a number of atoms. It is useful to instead think of them as being calculated from the position of one or more virtual atoms whose positions are generated based on the position of a collection of other atoms. For example you might want to calculate the distance between the center of masses of two molecules. In this case it is useful to calculate the two positions of the centers of mass first and to then calculate the distance between the centers of mass. The PLUMED input that you would use for such a calculation reflects this way of thinking. An example of a PLUMED input that can be used to perform this sort of calculation is shown below:

```
first: CENTER ATOMS=1,2,3,4,5,6
last: CENTER ATOMS=251-256

e2edist: DISTANCE ATOMS=2,253
comdist: DISTANCE ATOMS=first,last

PRINT ARG=e2edist,comdist STRIDE=1 FILE=COLVAR

ENDPLUMED
```

Make a PLUMED input containing the above input and execute it on the trajectory that you downloaded at the start of the exercise by making use of the commands in section Introduction to the PLUMED input file
Before we turn to analyzing what is output from this calculation there are a few things to note about this input file. Firstly, I should describe what this file instructs PLUMED to do. It tells PLUMED to:

1. calculate the position of the Virtual Atom 'first' as the CENTER of atoms from 1 to 6;
2. calculate the position of the Virtual Atom 'last' as the CENTER of atoms from 251 to 256;
3. calculate the distance between atoms 2 and 253 and saves it in 'e2edist';
4. calculate the distance between the two atoms 'first' and 'last' and saves it in 'comdist';
5. print the content of 'e2edist' and 'comdist' in the file COLVAR

Notice that in the input above we have used two different ways of writing the atoms used in the CENTER calculation:

1. ATOMS=1,2,3,4,5,6 is the explicit list of the atoms we need
2. ATOMS=251-256 is the range of atoms needed

Notice also that ranges of atoms can be defined with a stride which can also be negative as shown by the commands below, which are both equivalent:

1. ATOMS=from,to:by (i.e.: 251-256:2)
2. ATOMS=to,from:-by (i.e.: 256-251:-2)

Lets now return to the business of analyzing what was output by the calculation. Lets begin by looking at the contents of the COLVAR file that was output. When you do so you should find that the first few lines of this file read:

```
#! FIELDS time e2edist comdist
 0.000000 2.516315 2.464043
```

Notice that at variance with the file that was output in the previous section we now have three columns of numbers in the output file. Given the PRINT command that we used in the input to this calculation though this new behavior makes a lot of sense.
Lets now plot contents of the COLVAR file so we can compare the behavior of the distance between the two terminal carbons and the distance between the centers of the mass of the two terminal residues in this trajectory (these two distances are what the above input is calculating). To plot this data issue the following commands:

```
gnuplot
p 'COLVAR' u 1:2 w l, '' u 1:3 w l
```

Given what you observe for the behavior of these two distance what do you now expect to see in the trajectory? Let's look at the trajectory to see if we are right. To look at the trajectory issue the following commands:

```
vmd template.pdb trajectory-short.xyz
```

Lets summarize what we have learned from these sections thus far. We have seen that:

- PLUMED provides a simple syntax that can be used to calculate the DISTANCE between any pair of atoms in our system.
- PLUMED also allows us to calculate the positions of virtual atom (e.g. CENTER) and that we can calculate the DISTANCE between these quantities.
- Calculating these quantities is useful because it allows us to simplify the high-dimensional information contained in a trajectory.

Now, obviously, PLUMED can do much more than calculate the distances between pairs of atoms as we will start to see that in the following sections.

11.35.4.5 Calculating torsion angles

In the previous sections we have seen how we can use PLUMED to calculate distances and how by plotting these distances we can begin to simplify the high dimensional data contained in a trajectory. Obviously, calculating a DISTANCE is not always the best way to simplify the information contained in a trajectory and we often find we have to work with other more-complex quantities. PLUMED thus started as a library that was used to gather all the various implementations people had written for different collective variables (CVs) that people had used to "analyze" trajectories over the years (analyze is in inverted commas here because, as you will see if you continue to use PLUMED, we use CVs to do much more than simply analyze trajectories).
Now we will not have time to go over all the quantities that can be calculated in this tutorial. Once you understand the basic principles, however, you should be able to use the manual to work out how to calculate other quantities of interest. With this in mind then lets learn how to calculate a TORSION. As with DISTANCE the atoms that we specify in our TORSION command can be real or virtual. In the example below two real atoms and a virtual atom are used:

```
first: CENTER ATOMS=1-6
last: CENTER ATOMS=251-256
cvtor: TORSION ATOMS=first,102,138,last

PRINT ARG=cvtor STRIDE=1 FILE=COLVAR

ENDPLUMED
```

Copy this input to a PLUMED input file and use it to analyze the trajectory you downloaded at the start of this exercise by using the commands described in section Introduction to the PLUMED input file then plot the CV output using gnuplot.
As you can hopefully see calculating TORSION values and other CVs is no more difficult than calculating DISTANCE values. In fact it is easier as generally when you calculate the torsion angles of a protein you often wish to calculate particular, named torsion angles (i.e. the ϕ and ψ angles). The MOLINFO command makes it particularly easy to do this. For instance suppose that you want to calculate and print the ϕ angle in the sixth residue of the protein and the ψ angle in the eighth residue of the protein. You can do so using the following input:

```
#SETTINGS MOLFILE=user-doc/tutorials/marvel-1/template.pdb
MOLINFO STRUCTURE=template.pdb
phi6: TORSION ATOMS=@phi-6
psi8: TORSION ATOMS=@psi-8
PRINT ARG=phi6,psi8 FILE=colvar
```

Copy this input to a PLUMED input file and use it to analyze the trajectory you downloaded at the start of this exercise by using the commands described in section Introduction to the PLUMED input file then plot the CV output using gnuplot. Notice that you will need the template.pdb file you downloaded at the start of this exercise in order for this to run.

11.35.4.6 An exercise with the radius of gyration

Lets now see if you can use everything you have learned to setup a PLUMED input file of your own. What I would like you to do is to write a PLUMED input file that measures the Radius of Gyration GYRATION for the configurations in each of the frames in the trajectory that you downloaded at the start of this exercise. This radius of gyration will be calculated using the positions of all the atoms in that trajectory.
NOTE: if what you need for one or more variables is a long list of atoms and not a virtual atom you can use the keyword GROUP. A GROUP can be defined using ATOMS in the same way we saw before, in addition it is also possible to define a GROUP by reading a GROMACS index file.

```
ca: GROUP ATOMS=9,16,31,55,69,90,102,114,124,138,160,174,194,208,224,238
```

Now 'ca' is not a virtual atom but a simple list of atoms.

11.35.4.7 Coordination numbers

In the previous sections we have learned how PLUMED can be used to calculate simple functions of atomic positions such as the DISTANCE between pairs of atoms. As discussed here (https://www.youtube.↩com/watch?v=iDvZmbWE5ps) many of the more complicated collective variables that we use to analyze simulations can be calculated by computing these simple quantities multiple times for different sets of atoms. That is to say we can calculate many more complicated collective variables using:

$$s = \sum_i g[f(\{\mathbf{X}_i\})]$$

Here g is a function of a scalar and f is a function that takes in the positions of a set of atoms $\{\mathbf{X}\}$ and outputs a scalar. The sum then runs over the different sets of atoms from which the quantity f can be calculated. This is all rather abstract so lets make it more concrete by considering an example. Suppose that we want to calculate the coordination number of atom k. What we need to do is:

1. We need to calculate the distance between atom k and every atom in the system that is not atom k. This will be the set of sets of atoms that we have to perform the sum above on. Furthermore, the function f in the above will be Pythagoras theorem, which is the function we use to calculate the distance between a pair of atoms.

2. We need to transform the distances calculated by a switching function (f in the above) that is equal to one if the distance is less than or equal to the typical length of a chemical bond and zero otherwise.

Lets thus use PLUMED to calculate the coordination number of atom 9. We can do this as follows:

```
d1: DISTANCES GROUPA=9 GROUPB=1-8,10-256 LESS_THAN={RATIONAL D_0=0.16 R_0=0.01 D_MAX=0.2}
PRINT ARG=d1.* FILE=colvar
```

Copy this input file to a PLUMED input file. Before using it to analyze the trajectory that you downloaded at the start of the exercise using the commands described in section Introduction to the PLUMED input file try to guess what value this coordination number will take. Hint: what element is atom 9?
Now see if you can adjust the above input to calculate the coordination number of atom 5. What is the coordination number of this atom and why does it take this value?

11.35.4.8 Multicolvar

In the previous section we exploited a feature of PLUMED known as multicolvar when calculating the coordination number. When using this feature we are not confined to simply calculating coordination numbers. For instance the input below allows us to calculate a number of distances and to then calculate the mean of the distribution of distances, the minimum distance in the distribution, the maximum distance in the distribution and the second moment of the distribution (the variance).

```
ca: GROUP ATOMS=9,16,31,55,69,90,102,114,124,138,160,174,194,208,224,238
dd: DISTANCES GROUP=ca MEAN MIN={BETA=50} MAX={BETA=0.02} MOMENTS=2

PRINT ARG=dd.mean,dd.min,dd.max,dd.moment-2 STRIDE=1 FILE=COLVAR

ENDPLUMED
```

Try to copy this input now and to use it to analyze the trajectory you downloaded at the start of the exercise using the commands described in section Introduction to the PLUMED input file.

Multicolvar is not just for DISTANCES though. The infrastructure of multicolvar has been used to develop many PLUMED collective variables. One interesting example is the set of Secondary Structure CVs (ANTIBETARMSD, PARABETARMSD and ALPHARMSD). You can use the input below to calculate the degree of anti-beta secondary structure in each of the trajectory frames by copying this input to a PLUMED input file and by exploiting the commands to run driver that were described in section Introduction to the PLUMED input file.

```
#SETTINGS MOLFILE=user-doc/tutorials/marvel-1/template.pdb
MOLINFO STRUCTURE=template.pdb
abeta: ANTIBETARMSD RESIDUES=all TYPE=DRMSD LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} STRANDS_CUTOFF=1

PRINT ARG=abeta.lessthan STRIDE=1 FILE=COLVAR

ENDPLUMED
```

We can do a large number of other things with multicolvar. If you are interested this topic is described in more detail in the tutorial: Belfast tutorial: Steinhardt Parameters.

11.35.4.9 Understanding the need for ensemble averages

In the previous sections we have learned how we can use PLUMED to calculate collective variables from simulation trajectories and have seen how, by plotting how these collective variables change as a function of time, we can get a handle on what happens during the trajectory. Generally this level of analysis is not sufficient for us to publish a paper about the simulations we have run, however. In this section we are going to run some molecular dynamics simulations in order to understand why.
You are going to need to do the following set of things in order to do this exercise:

1. Take the two files you downloaded at the start of this exercise that are called called in and input.xyz and place them in a directory.
2. In the same directory write an input file for PLUMED called plumed.dat that calculates and prints the distance between atoms 2 and 3 to a file called colvar.
3. Run simplemd by issuing the command:

```
plumed simplemd < in
```

1. Open the file called input.xyz and modify the z-coordinate of the 1st atom. It should currently be equal to zero. Set it equal to 0.1.
2. Run simplemd again using the command above.
3. Plot the colvar files output during the two calculations using:

```
gnuplot
p 'colvar' w l, 'bck.0.colvar' w l
```

If you have done everything correctly you should see that the values of the distance in the early parts of the simulation are similar but that the values of the distance in the two simulations are very different by the end of the simulations.
Allow me to explain what we have just done. simplemd is a tool for running molecular dynamics using the Lennard-Jones potential. We have thus just run two very similar molecular dynamics calculations. The only difference between these two calculations was in the z-coordinate of the first atom in the input structure. The initial velocities of all the atoms were the same and the initial positions of all other atoms were identical. In spite of these similarities, however, the trajectories that we obtain in the two calculations are very different. This is important as it tells us that the time series of CV values that we get from a single MD simulation is (in and of itself) not particularly valuable as we would have got a completely different time series of values if we had run the calculation with only a slightly different input structure. We should, therefore, **think of the trajectory output from a molecular dynamics simulations as a time series of random variables** and analyze it accordingly.
The justification for thinking of a trajectory as a time series of random variables is based on statistical mechanics which tells us that a system with a constant number of atoms and a constant volume evolving at a fixed temperature has a probability:

$$P(q) \propto e^{-\frac{U(q)}{k_B T}}$$

of being in a microstate q with internal energy $U(q)$ as discussed and explained in the videos that can be found on the following pages:

- http://gtribello.github.io/mathNET/PRINCIPLE_OF_EQUAL_APRIORI_PROBABILITIES.html
- http://gtribello.github.io/mathNET/GENERALIZED_PARTITION_FUNCTION.html
- http://gtribello.github.io/mathNET/CANONICAL_ENSEMBLE.html

Now, and as already explained above, given that each microstate is sampled with this particular probability it makes sense to think of the microstates, and by extension the collective variables values that we calculate from these microstates, as random variables. We can thus define the ensemble average for a particular collective variable, A, using:

$$\langle A \rangle = \int A(q) P(q) \mathrm{d}q$$

where the integral here runs over all possible microstates, $A(q)$ is the function that allows us to calculate the value of the CV from the positions of the atoms in microstate q and where $P(q)$ is the "probability" (as it appears in an integral it is, strictly speaking, a probability density) of being in microstate q which was introduced above. For more information on this business of random variables and ensemble averages (or expectations) check out the videos that can be found on the following pages:

- http://gtribello.github.io/mathNET/RANDOM_VARIABLES.html
- http://gtribello.github.io/mathNET/EXPECTATION.html

Instead of using the formula for the ensemble average given above we can estimate an ensemble averages by taking a time series of random variables (like our trajectory) adding all the values of the random variable together (in our case the CV values in each of the trajectory frames) and dividing by the number of random variables that were added together. This works because of results known as the law of large numbers and the central limit theorem, which you can find videos on here:

- http://gtribello.github.io/mathNET/LAW_OF_LARGE_NUMBERS.html
- http://gtribello.github.io/mathNET/CENTRAL_LIMIT_THEOREM.html

Alternatively, we can justify this way of analyzing our trajectory by thinking of the set of sampled frames as random variables generated from Markov chain.
These mathematical objects are discussed here:

- http://gtribello.github.io/mathNET/MARKOV_PROPERTY.html
- http://gtribello.github.io/mathNET/CHAPMAN_KOLMOGOROV_RELATION.html
- http://gtribello.github.io/mathNET/TRANSIENT_RECURRENT_STATES.html
- http://gtribello.github.io/mathNET/STATIONARY_DIST_MARKOV.html

Regardless, however, and as we will see in the following section we can estimate ensemble averages of collective variables using:

$$\langle A \rangle \approx \frac{1}{T} \sum_t A[q(t)]$$

where the sum runs over the set of T microstates, $q(t)$, in our trajectory and where $A[q(t)]$ is the function that is used to calculate the value of the collective variable from the positions of the atoms in the microstate. When we do so we find that the values of the ensemble averages from different trajectories are reasonably consistent and certainly much more consistent than the set of instantaneous CV values.

11.35.4.10 Calculating ensemble averages using PLUMED

Repeat the steps from the previous section that were used to run the two MD calculations with slightly different input configurations. This time, however, your PLUMED input should look like this:

```
d1: DISTANCE ATOMS=2,3
d1a: AVERAGE ARG=d1 STRIDE=10
PRINT ARG=d1,d1a FILE=colvar STRIDE=10
```

If you now plot the values of the CV output from these two calculations together with the estimates of the ensemble averages using something like the commands below:

```
gnuplot
p 'colvar' u 1:2 w l, '' u 1:3 w l, 'bck.0.colvar' u 1:2 w l, '' u 1:3 w l
```

You should see that, although the instantaneous values of the CVs differ greatly in the two simulations, the average values of the CVs are relatively similar for both simulations.

Lets discuss the AVERAGE command that we have added here and what this does. In essence it accumulates the sum of all the distances calculated by the DISTANCE command labelled d1. When it comes time to output this accumulated average (every 10 trajectory steps) this accumulated sum is divided by the number of trajectory

frames that have been summed in calculated the current sum. In other words, the average allows us to compute the following quantity:

$$a = \frac{1}{N}\sum_{i=1}^{N} d(t_i)$$

where $d(t_i)$ is the value of the distance at time t_i. Now you may reasonably ask: what the keyword STRIDE=10 is doing in this command?
Well this is telling PLUMED to only add distances into the accumulated average every 10 steps. In other words, when calculating the average we are disregarding the value of the distance in frames 1,2,3,4,5,6,7,8 and 9. Why are we disregarding this information though? Well because there are correlations between the values the CV takes in two adjacent trajectory frames. Ideally, we want the values of the distance we are averaging to be independent and identical random variables.

Before we move on to calculating histograms and free energy surfaces I have a little challenge for you. If you are able to answer the following question then you really understand what the AVERAGE command is doing. Try running the calculation above using the following input:

```
d1: DISTANCE ATOMS=2,3
d1a: AVERAGE ARG=d1 STRIDE=10 CLEAR=10
PRINT ARG=d1,d1a FILE=colvar STRIDE=10
```

Based on what you see when you plot the colvar file and the information on the page about the AVERAGE command what is the CLEAR=10 keyword telling PLUMED to do?

11.35.4.11 Calculating histograms

Most of the time, we are not really interested in calculating ensemble averages for particular collective variables. What we would really like is the probability that the collective variable takes a particular value or set of values. In other words, and as discussed in the following video, we would like the probability as a function of collective variable value:

- https://www.youtube.com/watch?v=-1NLaqOJKS0

We can calculate approximate histograms like these using PLUMED. Furthermore, and as discussed in the video below, when we do this what we are really doing is we are calculating multiple ensemble averages at once:

- http://gtribello.github.io/mathNET/histogram-video.html

To calculate these ensemble averages we must make use of the HISTOGRAM command. An input file for a calculation of this sort is provided below. Use this input now and the input files for simple MD from the previous two sections to calculate the histogram of DISTANCE values that are explored in these trajectories.

```
d1: DISTANCE ATOMS=2,3
histogram: HISTOGRAM ARG=d1 BANDWIDTH=0.05 GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=200 STRIDE=10
DUMPGRID GRID=histogram FILE=histo STRIDE=25000
```

You can plot the histogram output from this simulation using:

```
gnuplot
p 'histo' w l
```

You should see that there is a large peak in the histogram at around 2.0 indicating that this is the value of the distance that the atoms were most likely to have during the course of the simulation. N.B. The histogram is unlikely to be converged with a trajectory this short.
Lets now look at the syntax of the HISTOGRAM command. The first thing to note is the similarities between what is done with this command and what is done with the AVERAGE command. Once again we have a STRIDE keyword for HISTOGRAM (and a CLEAR keyword for that matter) that tells us how often data should be added to to the accumulated averages. Furthermore, in both these commands we have an additional instruction with its own STRIDE keyword (PRINT for AVERAGE and DUMPGRID for HISTOGRAM) that tells us how frequently the accumulated averages should be output to human readable files.

A substantial difference between these two input files is the object the label histogram refers to. In all previous examples the label of an action has referred to a scalar quantity that was calculated by that action. In the example above, however, the label histogram refers to the function accumulated on a grid that is evaluated by the HISTOGRAM command. This is why we cannot print this quantity using PRINT - it is not a simple scalar valued function anymore. As you become more familiar with PLUMED you will find that these labels can refer to a range of different types of mathematical object.

Lets now see if we can bring together everything we have learned in this tutorial in order to analyze the protein trajectory that was downloaded at the start of the exercise.

11.35.4.12 A histogram for the protein trajectory

We are going to calculate the HISTOGRAM from our protein trajectory as a function of two different collective variables: ANTIBETARMSD and the average distance between the ca atoms of our protein backbone. The input that allows us to calculate perform this analysis is shown below:

```
#SETTINGS MOLFILE=user-doc/tutorials/marvel-1/template.pdb
# Read in protein structure template
MOLINFO STRUCTURE=template.pdb
# Calculate collective variables
abeta: ANTIBETARMSD RESIDUES=all TYPE=DRMSD LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} STRANDS_CUTOFF=1
ca: GROUP ATOMS=9,16,31,55,69,90,102,114,124,138,160,174,194,208,224,238
DISTANCES GROUP=ca MEAN MIN={BETA=50} MAX={BETA=0.02} MOMENTS=2 LABEL=dd
# Print instaneous values of collective variables
PRINT ARG=abeta.lessthan,dd.mean,dd.min,dd.max,dd.moment-2 STRIDE=1 FILE=COLVAR
# Accumulate histogram of collective variables
hh: HISTOGRAM ARG=abeta.lessthan,dd.mean KERNEL=DISCRETE GRID_MIN=0,0.8 GRID_MAX=4,1.2 GRID_BIN=40,40
# Output histogram - N.B. when we are running with driver we can not provide a STRIDE.
# The histogram will then only be output once all the trajectory frames have been read in and analyzed
DUMPGRID GRID=hh FILE=histo
```

Try running the input above on the trajectory that you downloaded at the start of this exercise by using the commands detailed in section Introduction to the PLUMED input file. You can plot the two dimensional histogram output using the following commands:

```
gnuplot
set pm3d map
sp 'histo' w pm3d
```

If you do so though, you will probably find that the structure in the histogram, $H(s)$, is a bit difficult to visualize because the probability changes from point to point by multiple orders of magnitude. This is why we often convert the histogram, $H(s)$, to a free energy surface, $F(s)$, using:

$$F(s) = -k_B T \ln H(s)$$

If you want to use PLUMED to output the free energy rather than the histogram you need to use the CONVERT_TO_FES command as shown below:

```
#SETTINGS MOLFILE=user-doc/tutorials/marvel-1/template.pdb
# Read in protein structure template
MOLINFO STRUCTURE=template.pdb
# Calculate collective variables
abeta: ANTIBETARMSD RESIDUES=all TYPE=DRMSD LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} STRANDS_CUTOFF=1
ca: GROUP ATOMS=9,16,31,55,69,90,102,114,124,138,160,174,194,208,224,238
DISTANCES GROUP=ca MEAN MIN={BETA=50} MAX={BETA=0.02} MOMENTS=2 LABEL=dd
# Print instaneous values of collective variables
PRINT ARG=abeta.lessthan,dd.mean,dd.min,dd.max,dd.moment-2 STRIDE=1 FILE=COLVAR
# Accumulate histogram of collective variables
hh: HISTOGRAM ARG=abeta.lessthan,dd.mean KERNEL=DISCRETE GRID_MIN=0,0.8 GRID_MAX=4,1.2 GRID_BIN=40,40
fes: CONVERT_TO_FES GRID=hh TEMP=300
# Output free energy - N.B. when we are running with driver we can not provide a STRIDE.
# The histogram will then only be output once all the trajectory frames have been read in and analyzed
DUMPGRID GRID=fes FILE=fes.dat
```

Notice though that even when we do this complicated looking calculation we are still, underneath it all, calculating functions of a large number of ensemble averages.

11.35.5 Conclusions and further work

If you have worked through all of this tutorial make sure that you have understood it by ensuring that you understand what the list of learning outcomes in section Learning Outcomes means and that you can use PLUMED to perform all these tasks. In terms of further work you should investigate issues related to the convergence of estimates of ensemble averages such as block averaging. You might like to investigate how long your simulations have to be in order to obtain reliable estimates of the ensemble average for a collective variable and reliable estimates for the free energy as a function of a collective variable. Alternatively, you might like to explore other collective variables that could be used to analyze the protein trajectory that you have worked on in this tutorial.

11.36 MARVEL tutorial: Path CVs

11.36.1 Aims

Consider the two overlain protein structures that are shown in the figure below.
Can you see the difference between these two structures? Can you think of a collective variable that could be used to study the substantial change in structure?
Your answers to the questions posed above are hopefully: yes I can see the difference between the two structures - the upper loop is radically different in the two cases - and no I have absolutely no idea as to how to create a collective variable that might be used to study the change in structure.
These answers are interesting as they cut to the very heart of what is interesting about biomolecular systems. In fact this difficulty is one of the reasons why such systems are so widely studied. If you think for a moment about solid state systems any transition usually involves a substantial change in symmetry.
Low energy configurations are usually high symmetry while higher energy configurations have a low symmetry. This makes it easy to design collective variables to study solid state transitions - you simply measure the degree of symmetry in the system (see Belfast tutorial: Steinhardt Parameters). In biomolecular systems by contrast the symmetry does not change substantially during a folding transition. The unfolded state has a low symmetry but the folded state also has a low symmetry, which is part of the reason that it is so difficult to find the folded state from the amino acid sequence alone.
With all this in mind the purpose of this tutorial is to learn about how we can design collective variables that can be used to study transitions between different states of these low-symmetry systems. In particular, we are going to learn how we can design collective variables that describe how far we have progressed along some pathway between two configurations with relatively low symmetry. We will in most of this tutorial study how these coordinates work in a two-dimensional space as this will allow us to visualize what we are doing. Hopefully, however, you will be able to use what you learn from this tutorial to generalize these ideas so that you can use PATH and PCAVARS in higher-dimensional spaces.

11.36.2 Learning Outcomes

Once this tutorial is completed students will:

- be able to explain what is computed by a PCAVARS coordinate and write down expressions for these quantities.
- be able to write a PLUMED input file that calculates and prints a PCAVARS coordinate.
- be able to write down an expression for the quantity contained in the s and z components of a PATH collective variable.
- be able to write PLUMED input files that calculate PATH collective variables for a range of different metrics.
- be able to measure the quality of a transition state by calculating the isocommittor.

11.36.3 Resources

The `tarball` for this project contains the following files:

- transformation.pdb : a trajectory that shows the transition between the C_7ax and C_7eq conformers of alanine dipeptide.

- pca-reference.pdb : a file that gives the start and end points of the vector that connects the C_7ax and C_7eq conformers. This file contains the positions of the atoms in these two structures.
- PCA-isocommittor : a directory containing the files required to run isocommittor simulations that monitor the values of ϕ, ψ and the PCA coordinate.
- PATH-isocommittor : a directory containing the files required to run isocommittor simulations that monitor the values of the PATH collective variable $S(X)$.
- 2CV-isocommittor : a directory containing the files required to run isocommittor simulations that monitor the values of the PATH collective variables $S(X)$ and $Z(X)$

11.36.4 Instructions

In this tutorial we are going to be considering a conformational transition of alanine dipeptide. In particular we are going to be considering the transition between the two conformers of this molecule shown below:
Alanine dipeptide is a rather well-studied biomolecule (in fact it is an over studied molecule!). It is well known that you can understand the inter-conversion of the two conformers shown above by looking at the free energy surface as a function of the ϕ and ψ Ramachandran angles as shown below:
In this tutorial we are not going to use these coordinates to study alanine dipeptide. Instead we are going to see if we can find a single collective variable that can distinguish between these two states.

11.36.4.1 PCA coordinates

Consider the free energy surface shown in figure marvel-2-rama-fig. It is clear that either the ϕ (x-axis) or the ψ (y-axis) angle of the molecule can be used to distinguish between the two configurations shown in figure marvel-2-transition-fig. Having said that, however, given the shape of landscape and the associated thermal fluctuations we would expect to see in the values of these angles during a typical simulation, it seems likely that ϕ will do a better job at distinguishing between the two configurations. ψ would most likely be a bad coordinate as when the molecule is in the C_7eq configuration the ψ angle can fluctuate to any value with only a very small energetic cost. If we only had information on how the ψ angle changed during a simulation we would thus struggle to distinguish a transition to the C_7ax configuration from a thermal fluctuations. It has been shown that metadynamics simulations that use just the ϕ angle as a collective variable can effectively drive the system from the C_7ax configuration to the C_7eq configuration. We will not repeat these calculations here but will instead use driver to determine how the values of ϕ and ψ change as we move between along the transition path that connects the C_7ax configuration to the C_7eq state.

```
t1: TORSION ATOMS=2,4,6,9
t2: TORSION ATOMS=4,6,9,11
PRINT ARG=t1,t2 FILE=colvar
```

Lets run this now on trajectory that describes the transition from the C_7ax configuration to the C_7eq configuration. To run this calculation copy the input file above into a file called plumed.dat and run the command below:

```
plumed driver --mf_pdb transformation.pdb
```

Try plotting each of the two torsional angles in this file against time in order to get an idea of how good a job each one of these coordinates at distinguishing between the various configurations along the pathway connecting the C_7ax and C_7eq configurations. What you will see is that in both cases the CV does not increase/decrease monotonically as the transition progresses.
We can perhaps come up with a better coordinate that incorporates changes in both ϕ and ψ by using the coordinate illustrated in the figure below.
We can even use PLUMED to calculate this coordinate by using the input shown below:

```
t1: TORSION ATOMS=2,4,6,9
t2: TORSION ATOMS=4,6,9,11
tc: COMBINE ARG=t1,t2 COEFFICIENTS=2.621915,-2.408714 PERIODIC=NO
PRINT ARG=t1,t2,tc FILE=colvar
```

Try calculating the values of the above collective variables for each of the configurations in the transformation.pdb file by using the command that was given earlier.
Notice that what we are using here are some well known results on the dot product of two vectors here. Essentially if the values of the Ramachandran angles in the C_7eq configuration are (ϕ_1, ψ_1) and the Ramachandran angles

in the C_7ax configuration are (ϕ_2, ψ_2). If our instantaneous configuration is (ϕ_3, ψ_3) we can thus calculate the following projection on the vector connecting the C_7eq state to the C_7ax state:

$$s = (\phi_2 - \phi_1).(\phi_3 - \phi_1) + (\psi_2 - \psi_1).(\psi_3 - \psi_1)$$

which is just the dot product between the vector connecting the point (ϕ_1, ψ_1) to (ϕ_2, ψ_2) and the vector connecting the point (ϕ_1, ψ_1) to (ϕ_3, ψ_3). If we call these two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ we can write this dot product as:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = |\mathbf{v}_1||\mathbf{v}_2|cos(\alpha)$$

where $|\mathbf{v}_1|$ and $|\mathbf{v}_2|$ are the magnitudes of our two vectors and where α is the angle between the two of them. Elementary trigonometry thus tells us that if $\mathbf{v}_1$ is a unit vector (i.e. if it has magnitude 1) the dot product is thus equal to the projection of the vector $\mathbf{v}_2$ on $\mathbf{v}_2$ as shown in figure pca-figure.
This is an useful idea. In fact it is the basis of the PCAVARS collective variable that is implemented in PLUMED so we can (almost) calculate the projection on this vector by using the input shown below:

```
t1: TORSION ATOMS=2,4,6,9
t2: TORSION ATOMS=4,6,9,11
p: PCAVARS REFERENCE=angle-pca-reference.pdb TYPE=EUCLIDEAN
PRINT ARG=t1,t2,p.* FILE=colvar
```

We cannot, however, do this in practice (we also shouldn't really use the previous input either) as the TORSION angles that we use to define our vectors here are periodic. In this next section we will thus look at how we can avoid this problem of periodicity by working in a higher dimensional space.

11.36.4.2 PCA with the RMSD metric

In the previous section I showed how we can use the projection of a displacement on a vector as a collective variable. I demonstrated this in a two dimensional space as this makes it easy to visualize the vectors involved. We are not forced to work with two dimensional vectors, however. We can instead find the vector that connects the C_7eq and C_7ax states in some higher dimensional space and project our current coordinate on that particular vector. In fact we can even define this vector in the space of the coordinates of the atoms. In other words, if the 3 N coordinate of atomic positions is $\mathbf{x}^{(1)}$ for the C_7eq configuration and $\mathbf{x}^{(2)}$ for the C_7ax configuration and if the instantaneous configuration of the atoms is $\mathbf{x}^{(3)}$ we can use the following as a CV:

$$s = \sum_{i=1}^{3N} (x_i^{(2)} - x_i^{(1)})(x_i^{(3)} - x_i^{(1)})$$

where the sum here runs over the $3N$-dimensional vector that defines the positions of the N atoms in the system. This is what (in a manner of speaking - I will return to this point momentarily) is calculated by this PLUMED input:

```
t1: TORSION ATOMS=2,4,6,9
t2: TORSION ATOMS=4,6,9,11
p: PCAVARS REFERENCE=pca-reference.pdb TYPE=OPTIMAL
PRINT ARG=t1,t2,p.* FILE=colvar
```

Use this input to analyze the set of configurations that are in the transformation.pdb file.
Let's now look further at the caveat that I alluded to before we ran the calculations. I stated that we are only calculating:

$$s = \sum_{i=1}^{3N} (x_i^{(2)} - x_i^{(1)})(x_i^{(3)} - x_i^{(1)})$$

in a manner of speaking. The point is that we would not want to calculate exactly this quantity because the vectors of displacements that are calculated in this way includes both rotational and translational motion. This is a problem as the majority of the change in moving from the C_7ax configuration shown in figure marvel-2-transition-fig to the C_7eq configuration shown in figure marvel-2-transition-fig comes from the translation of all the atoms. To put this another way if I had, in figure marvel-2-transition-fig, shown two images of the C_7ax configuration side by side the displacement in the positions of the atoms in those two structures would be similar to the displacement of the atoms in marvel-2-transition-fig as as the majority of the displacement in the vector of atomic positions comes about because I have translated all the atoms in the molecule rightwards by a fixed amount. I can, however, remove these translational displacements from consideration when calculating these vectors. In addition, I can also remove any displacements due rotation in the frame of reference of the molecule. If you are interested in how this is done

in practice you can read about it on the manual page about the RMSD collective variable. For what concerns us here, however, all we need to know is that when we use the OPTIMAL metric we are calculating a vector which tells us how far the atoms have been displaced in moving from structure A to structure B in a way that excludes any displacements due to translation of the center of mass of the molecule or any displacements that occur due to rotation of the Cartesian frame.

11.36.4.3 The isocommittor surface

In the previous sections I have been rather loose when talking about better and worse collective variables in that I have not been clear in my distinction between what makes a collective variable good and what makes a collective variable bad. In this section I thus want to discuss one method that we can use to judge the quality of a collective variable. This method involves calculating the so called isocommittor. The essential notion behind this technique is that there will be a saddle point between the two states of interest (the C_7ax and C_7eq configurations in our alanine dipeptide example). If the free energy is plotted as a function of a good collective variable the location of this dividing surface - the saddle point - will appear as a maximum.
Lets suppose that we now start a simulation with the system balanced precariously on this maximum. The system will, very-rapidly, fall off the maximum and move towards either the left or the right basin. Furthermore, if this maximum provides a good representation of the location of the transition state ensemble - in other words if the position of maximum in the low-dimensional free energy surface tells us something about the structure in the transition state ensemble - then 50% of trajectories started from this point will fall to the left and 50% will fall to the right. If by contrast the maximum in the free energy surface does not represent the location of the transition state well then there will be an imbalance between the number of trajectories that move rightwards and the number that move leftwards. We can think of this business of the isocommittor one further way, however. If in the vicinity of the transition state between the two basins the collective variable is perpendicular to the surface separating these two states half of the trajectories that start from this configuration will move to the left in CV space while the other half will move to the right. If the dividing surface and the CV are not perpendicular in the vicinity of the transition state, however, there will be an imbalance between the number of trajectories that move rightwards and the number of trajectories that move leftwards. We can thus determine the goodness of a collective variable by shooting trajectories from what we believe is the transition state and examining the number of trajectories that move into the left and right basins.
Lets make all this a bit more concrete by looking at how we might calculate the isocommittor by using some of the collective variables we have introduced in this exercise. You will need to go into the directory in the tar ball that you downloaded that is called PCA-isocommittor. In this directory you will find a number of files that will serve as input to gromacs 5 and PLUMED. You thus need to ensure that you have an installed version of gromacs 5 patched with PLUMED on your computer in order to perform this exercise. In addition to these input files you will find a bash script called script.sh, which we are going to use in order to set of a large number of molecular dynamics simulations. If you open script.sh you will find the following lines near the top:

```
GROMACS_BIN=/Users/gareth/MD_code/gromacs-5.1.1/build/bin
GROMACS=$GROMACS_BIN/gmx
source $GROMACS_BIN/GMXRC.bash
```

These will need to be adjusted so that the GROMACS_BIN variable points at the bin directory of the gromacs build on your computer. Once you have made this modification though you can run the calculation by issuing the following command in the PCA-isocommittor directory:

```
./script.sh
```

This command submits 50 molecular dynamics simulations that all start from a configuration that lies between the C_7eq and C_7ax configurations of alanine dipeptide. In addition, this command also generates some scripts that allow us to visualize how the ϕ, ψ and PCA coordinates that we introduced in the previous sections change during each of these 50 simulations. If you load gnuplot and issue the command:

```
load "script_psi.gplt"
```

you see how ψ changes during the course of the 50 simulations. Similarly the gnuplot command:

```
load "script_phi.gplt"
```

will show how how ϕ changes during the course of the 50 simulations and:

```
load "script_pca.gplt"
```

gives you the information on the PCA coordinates. If you look at the ψ and PCA data first you can see clearly that it is very difficult to distinguish the configurations that moved to the C_7eq basin from the trajectories that moved to the C_7ax basin. By contrast if you look at the data on the ϕ angles you can indeed use these plots to distinguish the trajectories that moved to C_7eq from the trajectories that moved to C_7ax. It is abundantly clear, however, that the number of trajectories that moved to C_7eq is not equal to the number of trajectories that moved to C_7ax. This CV, therefore, is not capturing the transition state ensemble.
One thing you will have seen from these examples is that the PCAVARS coordinate that were introduced in the previous sections provides an extremely poor model for the transition state ensemble. The value of the isocommittor at the maximum for both of these variables is not at all close to 50%. In fact the CV is not even particularly good at capturing the difference between these two states. If you look at the free energy surface shown below it is perhaps clear why.
You can see the location of the saddle point between these two states in this surface and it is very clear that the vector connecting the C_7eq state to the C_7ax state does not pass through this point. In fact it would be extremely fortuitous if a vector connecting an initial state and a final state also passed through the intermediate transition state between them. We can, after all, define the equation of straight line (a vector) if we are given only two points on it. In the next section we are thus going to see how we can resolve this problem by introducing a non-linear (or curvilinear) coordinate.

11.36.4.4 Path collective variables

Consider the black path that connects the C_7ax and C_7eq states in the free energy shown below:
This black pathways appears to be the "perfect" coordinate for modelling this conformational transition as it passes along the lowest energy pathway that connects the two states and because it thus passes over the lowest saddle point that lies between them. We can calculate such a coordinate with PLUMED by using the input file below (I will return to the mathematical details of how this works momentarily)

```
path: PATH REFERENCE=path-reference.pdb TYPE=OPTIMAL LAMBDA=15100.
PRINT ARG=* STRIDE=2 FILE=colvar FMT=%12.8f
```

Lets thus use this input and run an isocommittor analysis using the location of the maximum in this coordinate as the start point for all our trajectories. Everything you need to do this analysis is in the PATH-isocommittor directory. Once again you will find that there is a script.sh bash script inside this directory, which, as in the previous section, you can use to run a large number of molecular dynamics simulation. Furthermore, similarly to the last section you will need to begin this exercise by modifying the location of path to gromacs within this script. Once you have made this modification submit your molecular dynamics jobs by issuing the command:

```
./script.sh
```

You can then plot the data output using gnuplot and the command:

```
load "script_path.gplt"
```

Unlike what we saw for the PCAVARS variables in the previous section we find that it is easy to use these PATH variables to distinguish those configurations that moved to C_7ax from those that moved to C_7eq. Having said that, however, we still have a large imbalance between the number of trajectories that move rightwards and the number that move leftwards. We are thus still a long way from unambiguously identifying the location of the transition state ensemble for this system.

11.36.4.5 The mathematics of path collective variables

Let's now take a moment to discuss the mathematics of these coordinates, which is not so complicated if we think about what they do through an analogy. Suppose that you were giving your friend instructions as to how to get to your house and lets suppose these instructions read something like this:

1. Take the **M1 motorway** and get off **at junction 5**
2. At **the roundabout** you need to take **the third exit** towards **Crumlin**
3. Follow the road as far as the **farmers arms** then take the **next left**

4. The house is **on the corner by the garage.**

If you think about these instructions in the abstract what you have is a set of way markers (the item I have put in bold) in a particular order. The list of way markers is important as is the order they appear in in the instructions so we incorporate both these items in the PATH coordinates that we have just used to study the transition between the transition between C_7eq and C_7ax. In these coordinates the way markers are a set of interesting points in a high dimensional space. In other words, these are like the configurations of C_7eq and C_7ax that we used in the previous sections when talking about PCAVARS. Each of these configurations lies along the path connecting C_7eq and C_7ax and, as in the directions example above, one must pass them in a particular order in order to pass between these two conformations. The final CV that we have used above is thus:

$$S(X) = \frac{\sum_{i=1}^{N} i \ \exp^{-\lambda|X-X_i|}}{\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|}}$$

In this expression $-X_i|$ is the distance between the instantaneous coordinate of the system, X, in the high-dimensional space and X_i is the coordinate of the ith way mark in the path. The largest exponential in the sum that appears in the numerator and the denominator will thus be the one that corresponds to the point that is closest to where the system currently lies. In other words, $S(X)$, measures the position on a (curvilinear) path that connects two states of interest as shown in red in the figure below:

11.36.4.6 The Z(X) collective variable

You may reasonably ask what the purpose these PATH collective variables variables serve given that in this case they seem to do no better than ϕ when it comes to the tests we have performed on calculating the isocommittor. To answer this question we are going to run one final set of isocommittor simulations. The input for these calculations are in the directory called 2CV-isocommittor. Once again you will find that there is a script.sh bash script inside this directory, which, as in the previous section, you can use to run a large number of molecular dynamics simulation. Furthermore, similarly to the last section you will need to begin this exercise by modifying the location of path to gromacs within this script. Once you have made this modification submit your molecular dynamics jobs by issuing the command:

```
./script.sh
```

You can then plot the data output using gnuplot and the commands:

```
load "script_pca.gplt"
```

and

```
load "script_path.gplt"
```

What is plotted by these commands is slightly different from what was plotted in the previous exercises where we calculated the isocommittor.
Instead of plotting the value of the CV against simulation time the above commands plot the values that 2CVs take during the simulation. The script called script_path.gplt plots the value of the $S(X)$ collective variable on the x-axis and the value of the following quantity on the y-axis:

$$Z(X) = -\frac{1}{\lambda} \log(\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|})$$

What this quantity measures is shown in green in the figure marvel-2-ab-sz-fig. Essentially it measures the distance between the instantaneous configuration the system finds itself in and the path that is marked out using the way markers. If you plot the data using script_path.gplt what you thus see is that the system never moves very far from the path that is defined using the PATH command. In short the system follows this path from the transition state back to either the C_7eq or C_7ax configuration.
We can calculate a quantity similar to $Z(X)$ for the PCAVARS collective variables. Furthermore, when we plot the data we have generated using this exercise using script_pca.gplt the value this quantity takes is shown plotted against the instantaneous value of the PCAVARS collective variable. If you compare this graph with what was obtained when you plotted the output from PATH above you see that the system has moved very far from the PCAVARS coordinate.

This exercise illustrates the strength of these PATH collective variables. We can use a PATH to monitor how a large number of coordinates change during a chemical transition. Furthermore, we can use $Z(X)$ to measure how much real trajectories deviate from our PATH and thus have a quantitative measure of how well our PATH represents the true transition mechanism. Compare this with using a single CV such as ϕ. When use use a single CV we map the high-dimensional data from the trajectory into a lower dimensional space. We thus lose some information on what occurs during the transition.
To be clear we also loose information on the what occurs during the transition when we use PATH as any mapping into a lower dimensional space deletes information. In the PATH case, however, we can use the value of $Z(X)$ to measure how much data has been lost in mapping the trajectory onto $S(X)$.
These two coordinates, $S(X)$ and $Z(X)$, are very flexible. They are thus been used widely in the literature on modelling conformational changes of biomolecules. A part of this flexibility comes because one can use any set of way markers to define the PATH. Another flexibility comes, however, when you recognize that you can also change the way in which the distance, $-X_i|$, is calculated in the two formulas above. For example this distance can be calculated using the RMSD distance or it can be calculated by measuring the sum of the squares of a set of displacements in collective variable values (see TARGET). Changing the manner in which the distance between path way points is calculated thus provides a way to control the level of detail that is incorporated in the description of the reaction PATH.

11.36.4.7 Optimizing path collective variables

Hopefully the previous sections have allowed you to understand how PATH collective variables work and the sorts of problems they might be used to solve. If you have one of these problems to solve the next reasonable question to ask is: how to collect the set of reference frames that serve as the way markers on your PATH. Unfortunately, there is no single answer to this question. Different researchers have used different methods including using packages that morph one protein structure into another, using information from prior molecular dynamics or enhanced sampling calculations and even using PATH collective variables that change adaptively as the simulation progresses. Ultimately, you will need to find the best method for solving your particular problem. Having said that, however, there is some general guidance on setting up PATH collective variable and it is this that we will focus on in this section.
The first thing that you will need to double check is the spacing between the frames in your PATH. Lets suppose that your PATH has N of these way markers upon it you will need to calculate is the $N \times N$ matrix of distances between way markers. That is to say you will have to calculate the distance $_j - X_i|$ between each pair of frames. The values of the distance in this matrix for a good PATH are shown in the figure below:
For contrast the values of the distances in this matrix for a bad PATH are shown in the figure below:
If the distance matrix looks like the second of the two figures shown above this is indicates that the frames in the PATH that have been chosen are not particularly effective. Lets suppose that we have a PATH with four way markers upon it. In order for the $S(x)$ CV that was defined earlier to work well frame number 3 must be further from frame number 1 than frame number 2. Similarly frame number 4 must be still further from frame number 1 than frame number 3. This is what the gull wing shape in marvel-2-good-matrix-fig is telling us. The order of the frames in the rows and columns of the matrix is the same as the order that they are run through in the sums in the equation for $S(X)$. The shape of the surface in this figure shows that the distance between frames i and j increases monotonically as the magnitude of the difference between i and j is increased, which is what is required.
A second important requirement of a good PATH is shown in the figure below:

A good PATH has an approximately equal spacing between the neighboring frames along it. In other words, the distance between frame 1 and frame 2 is approximately equal to the distance between frame 2 and frame 3 as shown above. When this condition is satisfied a good criterion for selecting a suitable λ parameter to use is:

$$\lambda = \frac{2.3(N-1)}{\sum_{i=1}^{N-1} |X_i - X_{i+1}|}$$

11.36.5 Conclusions and further work

If you have worked through all of this tutorial make sure that you have understood it by ensuring that you understand what the list of learning outcomes in section Learning Outcomes means and that you can use PLUMED to perform all these tasks. You might then want to read the original paper on the PATH collective variable method as well as a few other articles in which these coordinates have been used to analyze simulations and to accelerate sampling.

- Davide Branduardi and Francesco Luigi Gervasio and Michele Parrinello From A to B in free energy space J. Chem. Phys., 126, 054103 (2007)

If you are interested in learning more about isocommittor surfaces and the transition state ensemble you should read up on the transition path sampling method.

11.37 Optimizing PLUMED performance

Authors

Giovanni Bussi

Date

February 19, 2019

This document describes the PLUMED tutorial held at CINECA, February 2019. The aim of this tutorial is to learn how to optimize simulations performed using PLUMED. Although the presented input files are correct, the users are invited to **refer to the literature to understand how the parameters of enhanced sampling methods should be chosen in a real application.**
Users are also encouraged to follow the links to the full PLUMED reference documentation and to wander around in the manual to discover the many available features and to do the other, more complete, tutorials.

11.37.1 How to optimize a simulation performed with PLUMED

The input files for this exercise can be found in this TARBALL.
We will now learn how to optimize a simulation where PLUMED is used on-the-fly to compute or bias collective variables. You should use the provided `topol.tpr` file.

11.37.1.1 Run a simulation with GROMACS alone

In order to run a simulation with gromacs you can use the following command

```
gmx_mpi mdrun -nsteps 500 -v -nb cpu -ntomp 12 -pin on
```

This will run a simulation for 500 steps, without using any GPU-acceleration, and with 12 OpenMP threads. Later at the end of the tutorial we will see how the speed of GROMACS+PLUMED changes when using the GPU. Adjust the number of threads based on the number of processors that you have on each node. 500 steps should be sufficient to get an estimate of the simulation speed if you use OpenMP only. Notice that if you use MPI parallelism with GROMACS more steps are needed due to dynamic load balancing.
As a first step, make a table showing the performance (ns per day) with different number of OpenMP threads. On my workstation the result is

Number of threads	Performance (ns/day)	Wallclock time (s)
12	27.225	3.180
6	12.677	6.829
3	8.827	9.807
1	3.685	23.491

Scaling is approximately linear.

11.37.1.2 Run a simulation with GROMACS+PLUMED

Now you can run GROMACS with PLUMED using the following input file

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=conf.pdb
# @water and @hydrogens are special groups introduce in PLUMED 2.5!
wat: GROUP ATOMS=@water
ow:  GROUP ATOMS=@water REMOVE=@hydrogens
mg:  GROUP ATOMS=10484
p:   GROUP ATOMS=@P-2
dp:    DISTANCE ATOMS=mg,p
cn:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261
```

```
PRINT ARG=dp,cn
```

The command to run the simulation will be

```
gmx_mpi mdrun -nsteps 500 -v -nb cpu -ntomp 12 -pin on -plumed plumed.dat
```

The total time required by this simulation should **increase**. This is because PLUMED occupies some of the CPU cycles in order to:

- copy the coordinates of the requested atoms
- compute the requested collective variables
- print them on a file

Take note of the increment in the wall-clock time. On my workstation it is approx 1.5 seconds more for these 500 steps with 12 OpenMP threads. Check how much is the impact when you change the number of threads.

11.37.1.3 Timing individual variables

In order to know which of your collective variables is taking more time to be computed, you should use the DEBUG DETAILED_TIMERS flag (place it after MOLINFO). The end of the `md.log` file should now look similar to this

```
PLUMED:                                      1     1.138360     1.138360     1.138360     1.1383
PLUMED: 1 Prepare dependencies             501     0.001654     0.000003     0.000003     0.0000
PLUMED: 2 Sharing data                     501     0.052691     0.000105     0.000092     0.0048
PLUMED: 3 Waiting for data                 501     0.004488     0.000009     0.000008     0.0000
PLUMED: 4 Calculating (forward loop)       501     0.444763     0.000888     0.000849     0.0018
PLUMED: 4A 1 @1                            501     0.001294     0.000003     0.000002     0.0000
PLUMED: 4A 6 dp                            501     0.007342     0.000015     0.000013     0.0000
PLUMED: 4A 7 cn                            501     0.422879     0.000844     0.000807     0.0017
PLUMED: 4A 8 @8                            501     0.001087     0.000002     0.000002     0.0000
PLUMED: 5 Applying (backward loop)         501     0.112352     0.000224     0.000210     0.0021
PLUMED: 5A 0 @8                            501     0.000713     0.000001     0.000001     0.0000
PLUMED: 5A 1 cn                            501     0.069576     0.000139     0.000132     0.0003
PLUMED: 5A 2 dp                            501     0.002932     0.000006     0.000005     0.0000
PLUMED: 5A 7 @1                            501     0.000806     0.000002     0.000001     0.0000
PLUMED: 5B Update forces                   501     0.029823     0.000060     0.000050     0.0019
PLUMED: 6 Update                           501     0.009887     0.000020     0.000017     0.0001
```

Notice that running with DETAILED_TIMERS might slow down a bit more your simulation.
Each line tells you how expensive was the calculation of each collective variable. Find which is the most expensive! We will focus on it in the next points

11.37.1.4 Optimizing coordination numbers using neighbor lists

The COORDINATION of multiple atoms can be very expensive for a number of reasons:

- It might require the calculation of a large number of distances
- It might require many atoms to be copied from GROMACS to PLUMED

In this specific example, the coordination number will require a number of calculations that is proportional to the number of water molecules in the system.
Repeat the timing above using neighbor lists. Here's how you should modify the line computing the COORDINA↩ TION in order to enable neighbor lists

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
cn:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261 NLIST NL_STRIDE=20 NL_CUTOFF=1.0
```

There are two critical parameters:

- the stride for the neighbor lists (here 20 steps).
- the cutoff distance (here 1 nm).

You should be **very careful** since using neighbor lists introduces approximations that might invalidate your calculation! The recommended procedure is to first perform a simulation where you compute your variable with different settings and compare the result. For instance:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=conf.pdb
wat: GROUP ATOMS=@water
ow:  GROUP ATOMS=@water REMOVE=@hydrogens
mg:  GROUP ATOMS=10484
p:   GROUP ATOMS=@P-2
dp:    DISTANCE ATOMS=mg,p
cn:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261
# use increasing values of the stride
cn2:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261 NLIST NL_STRIDE=2  NL_CUTOFF=1.0
cn10:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261 NLIST NL_STRIDE=10 NL_CUTOFF=1.0
cn20:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261 NLIST NL_STRIDE=20 NL_CUTOFF=1.0
cn50:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261 NLIST NL_STRIDE=50 NL_CUTOFF=1.0
# notice that here we are using a regular expression to select all coordination numbers
PRINT ARG=dp,(cn.*) FILE=COLVAR STRIDE=1
```

In the COLVAR files you will see multiple columns corresponding to values computed using different strides. You should pick a column such that the reported value is not too different from the one in the third column (that is: without neighbor lists). "Not too different" of course depends on how much you want to trade in accuracy vs speed.

Warning

> As of PLUMED 2.5, the construction of the neighbor list is not parallelized! As a consequence, the advantage of using it might be not compensated by the slow time required to construct it. This might change in a future PLUMED version. In addition, notice that the time for constructing the list is not present in the breakdown of the timers seen above, but is part of the "Prepare dependencies" stage.

Once you picked a setting that gives you results that are accurate enough, you can measure the speed using that setting alone using an input where only that specific setting is used. You will probably see that using neighbor lists is usually inconvenient unless you run with a single or very few cores.

11.37.1.5 Biasing your CVs: multiple time stepping

So far we have used PLUMED to analyze a CV on the fly. In principle, when you analyze CVs you typically only print them with a given stride (say, every 100 steps). This of course moderates significantly their cost. However, when you bias a CV you typically need to compute it at every step. Let's say that you want to use a restraint to make sure that the Mg ion is always six coordinated with water. This can be obtained using a RESTRAINT located AT=6. Remove the PRINT action from your input file and replace it with the following actions:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
# apply a restraint. this will be computed at every step by default
RESTRAINT ARG=cn AT=6 KAPPA=5.0
# only print every 100 steps
PRINT ARG=dp,cn FILE=COLVAR STRIDE=100
```

Now measure the speed of your simulation. You should see some overhead due to PLUMED. You can now try to use multiple-time-stepping [140]. To do so add a STRIDE option to the RESTRAINT line. You can also use the EFFECTIVE_ENERGY_DRIFT action to print a kind of "total energy drift" due to the application of the PLUMED bias. In molecular dynamics simulation you usually monitor the drift in the total energy in order to choose the time step. Here you can use the value of the effective energy drift to choose the stride for multiple-time-stepping.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
RESTRAINT ARG=cn AT=6 KAPPA=5.0 STRIDE=2
EFFECTIVE_ENERGY_DRIFT PRINT_STRIDE=100 FILE=eff
```

Using a STRIDE=5 you should get something like this

```
#! FIELDS time effective-energy
 0.000000 0.000000
 0.200000 0.012010
 0.400000 0.024574
 0.600000 0.105866
 0.800000 0.178739
 1.000000 0.254018
```

The effective energy will drift quickly! Using a STRIDE=2 instead the effective energy will be very stable

```
#! FIELDS time effective-energy
 0.000000 0.000000
 0.200000 0.000980
 0.400000 0.000041
 0.600000 -0.000162
 0.800000 0.000078
 1.000000 -0.000395
```

Note

The STRIDE option is not included in the manual, but can be used for all the bias actions to activate multiple-time-stepping.

11.37.1.6 Using grids in metadynamics

Let's try to perform a metadynamics simulation biasing the coordination number and the distance between the magnesium ion and the phosphate. Parameters are similar to those used in [137], although we use here a shorter deposition pace in order to artificially increase the computational cost.

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=conf.pdb
wat: GROUP ATOMS=@water
ow:  GROUP ATOMS=@water REMOVE=@hydrogens
mg:  GROUP ATOMS=10484
p:   GROUP ATOMS=@P-2
dp:    DISTANCE ATOMS=mg,p
cn:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261
metadP: METAD ARG=dp,cn SIGMA=0.05,0.1 HEIGHT=0.3 PACE=100 TEMP=300 BIASFACTOR=15
PRINT ARG=dp,cn FILE=COLVAR STRIDE=100
```

If you use this input for a very long simulation you will realize that most of the time is spent within the METAD action. The reason is simple: in its standard implementation, metadynamics is implemented as a history dependent potential where, every time we need to compute the force, a sum over the past history is performed. The history is saved in the HILLS file (have a look at it). When this file becomes too large the simulation will slow down. Without the need to run a very long simulation, we can see the problem by artificially creating a very long HILLS file. Every line of the HILLS file contains:

- The value of time.
- The coordinates at which the Gaussian function was deposited.
- The width of the deposited Gaussian function
- The height
- A number called bias-factor (check METAD manual to know what it means).

We can make an artificially long HILLS file and then use it to restart our simulation.

```
var="$(<HILLS)
for((i=0;i<=10000;i++)) ; do echo "$var" ; done | awk '{if($1!="#!") print $1,$2,$3,$4,$5,$6/100000,$7; else p
```

Now modify your plumed.dat file so that it will read the HILLS_long file:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
MOLINFO STRUCTURE=conf.pdb
wat: GROUP ATOMS=@water
ow:  GROUP ATOMS=@water REMOVE=@hydrogens
mg:  GROUP ATOMS=10484
p:   GROUP ATOMS=@P-2
dp:    DISTANCE ATOMS=mg,p
cn:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261
metadP: METAD ARG=dp,cn SIGMA=0.05,0.1 HEIGHT=0.3 PACE=100 TEMP=300 BIASFACTOR=15 FILE=HILLS_long RESTART=YES
PRINT ARG=dp,cn FILE=COLVAR STRIDE=100
```

Run your simulation and check the timing. Which is the most expensive action now?
The standard solution to the problem above is to use a grid to store the Gaussian functions. In this manner, at the first step PLUMED will take some time to put all the Gaussian functions on the grid, but subsequent calculations of the force will be much faster, since they will just require that the action look up the value on the grid (rather than a sum over the history).
Time the simulation using the following lines to perform METAD:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
metadP: METAD ARG=dp,cn SIGMA=0.05,0.1 HEIGHT=0.3 PACE=100 TEMP=300 BIASFACTOR=15 FILE=HILLS_long RESTART=YES
```

Which is the most expensive action now?

Warning

If a simulation reaches a point outside of the grid PLUMED will stop! You should be generous in the boundaries, but not too much or your simulation might require too much memory and crash

11.37.1.7 Running GROMACS on the GPU

All the exercises so far were done running GROMACS on the CPU. You should have noticed that the wall-clock time of a simulation run using PLUMED is approximately equal to the sum of:

- the wall-clock time of an equivalent simulation **not** using PLUMED, plus
- the total time required by PLUMED, shown at the end of the `md.log` file.

Let's see what happens using the GPU. In this case the first couple of thousands steps are kind of different since GROMACS tries to optimize the GPU load, so we should run a longer simulation to estimate the simulation speed. For simplicity, use the following `plumed.dat` file:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/performance-optimization.txt
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=conf.pdb
DEBUG DETAILED_TIMERS
wat: GROUP ATOMS=@water
ow:  GROUP ATOMS=@water REMOVE=@hydrogens
mg:  GROUP ATOMS=27275
p:   GROUP ATOMS=@P-2
dp:    DISTANCE ATOMS=mg,p
cn:    COORDINATION GROUPA=mg GROUPB=ow R_0=0.261
PRINT ARG=dp,(cn.*) FILE=COLVAR STRIDE=100
RESTRAINT ARG=cn AT=5 KAPPA=5.0
```

Let's compare the timings with/without GPU and with/without PLUMED using the following commands

```
gmx_mpi mdrun -nsteps 10000 -v -ntomp 12 -pin on -nb cpu
gmx_mpi mdrun -nsteps 10000 -v -ntomp 12 -pin on -plumed plumed.dat -nb cpu
gmx_mpi mdrun -nsteps 10000 -v -ntomp 12 -pin on
gmx_mpi mdrun -nsteps 10000 -v -ntomp 12 -pin on -plumed plumed.dat
```

When we run with PLUMED, let's also take note of the total time spent in PLUMED, written in the `md.log` file. Here's the result on my workstation:

GPU	PLUMED	Wall t (s)	PLUMED time (s)
no	no	49.931	/
no	yes	65.469	13.44
yes	no	19.648	/
yes	yes	21.745	12.36

PLUMED is not running on the GPU, so the total time spent in PLUMED is roughly the same both with and without GPU (12-13 seconds). However, when GROMACS runs using the GPU the load balancing transfers part of the load to the GPU. As a consequence, the total wall time is incremented by approximately 1 sec.

11.38 Old Tutorials

Trieste tutorial: Analyzing trajectories using PLUMED	This tutorial explains how to use PLUMED to analyze trajectories
Trieste tutorial: Averaging, histograms and block analysis	Averaging, histograms and block averaging
Trieste tutorial: Using restraints	This tutorial explains how to use PLUMED to run simple restrained simulations and account for the bias in the analysis
Trieste tutorial: Metadynamics simulations with PLUMED	This tutorial explains how to use PLUMED to run metadynamics simulations
Trieste tutorial: Running and analyzing multi-replica simulations	This tutorial explains how to use PLUMED to run and analyze multi-replica simulations
Trieste tutorial: Real-life applications with complex CVs	This tutorial explains how to use PLUMED to run metadynamics simulations

Belfast tutorial: Analyzing CVs	This tutorial explains how to use plumed to analyze CVs
Belfast tutorial: Adaptive variables I	How to use path CVs
Belfast tutorial: Adaptive variables II	Dimensionality reduction and sketch maps
Belfast tutorial: Umbrella sampling	Umbrella sampling, reweighting, and weighted histogram
Belfast tutorial: Out of equilibrium dynamics	How to run a steered MD simulations and how to estimate the free energy
Belfast tutorial: Metadynamics	How to run a metadynamics simulation
Belfast tutorial: Replica exchange I	Parallel tempering and Metadynamics, Well-Tempered Ensemble
Belfast tutorial: Replica exchange II and Multiple walkers	Bias exchange and multiple walkers
Belfast tutorial: NMR restraints	NMR restraints
Belfast tutorial: Steinhardt Parameters	Steinhardt Parameters
Cambridge tutorial	A short 2 hours tutorial that introduces Well-Tempered Metadynamics, Bias-Exchange Metadynamics and Replica-Average Metadynamics
CINECA tutorial	A short 2 hours tutorial that introduces analysis, well-tempered metadynamics, and multiple-restraints umbrella sampling.
Moving from PLUMED 1 to PLUMED 2	This tutorial explains how plumed 1 input files can be translated into the new plumed 2 syntax.
Munster tutorial	A short 3 hours tutorial that introduces analysis, well-tempered metadynamics, and multiple-restraints umbrella sampling.

11.38.1 Trieste tutorial: Analyzing trajectories using PLUMED

11.38.1.1 Aims

The aim of this tutorial is to introduce the users to the PLUMED syntax. We will go through the writing of simple collective variable and we will use them to analyze existing trajectories.

11.38.1.2 Objectives

Once this tutorial is completed students will be able to:

- Write a simple PLUMED input file and use it with the PLUMED driver to analyze a trajectory.
- Use the GROUP keyword to make the input file compact and easy to read and to quickly build complex atom groups.
- Print collective variables such as distances (DISTANCE), torsional angles (TORSION), gyration radius (GYRATION), and coordination numbers (COORDINATION) using the PRINT action.
- Computing the geometric center of a group of atoms using CENTER.
- Know how to take care of periodic boundary conditions within PLUMED using WHOLEMOLECULES and WRAPAROUND, and be able to verify the result with DUMPATOMS.
- Extract from a trajectory snapshots satisfying specific conditions using UPDATE_IF.

11.38.1.3 Resources

The TARBALL for this project contains the following files:

- ref.pdb : A PDB file with a RNA duplex solvated in a water box and a Mg ion.
- traj-whole.xtc: A trajectory for the same system in GROMACS xtc format. To make the exercise easier, RNA duplex has been made whole already.

- traj-broken.xtc: The same trajectory as it was originally produced by GROMACS. Here the RNA duplex is broken and should be fixed.

This tutorial has been tested on a pre-release version of version 2.4. However, it should not take advantage of 2.4-only features, thus should also work with version 2.3.
Also notice that in the `.solutions` directory of the tarball you will find correct input files. Please only look at these files after you have tried to solve the problems yourself.

11.38.1.4 Introduction

This tutorial asks you to compute a variety of different collective variables using PLUMED for a particular trajectory and to compare the files and graphs that you obtain with the correct ones that are shown online. Compared to some of the other tutorials that are available here this tutorial contains considerably less guidance so in doing this tutorial you will have to learn how to consult the manual. If you would like a more guided introduction to PLUMED it might be better to start with the tutorials Belfast tutorial: Analyzing CVs or MARVEL tutorial: Analyzing CVs. Also notice that, whereas this tutorial was tested using a pre-release version of PLUMED 2.4, it should be completely feasible using PLUMED 2.3.

11.38.1.5 Using PLUMED from the command line

As we will see later, PLUMED provides a library that can be combined with multiple MD codes. However, in this tutorial we will only use PLUMED to analyze trajectories that have been produced already. Once PLUMED is installed you can run a `plumed` executable that can be used for multiple purposes:

```
> plumed --help
```

Here we will use the driver tool, that allows you to process an already existing trajectory.

```
> plumed driver --help
```

What we will need is:

- A trajectory to be analyzed (provided).
- A PLUMED input file (you do it!).

The syntax of the PLUMED input file is the same that we will use later to run enhanced sampling simulations, so all the things that you will learn now will be useful later when you will run PLUMED coupled to an MD code. In the following we are going to see how to write an input file for PLUMED.

11.38.1.6 The structure of a PLUMED input file

The main goal of PLUMED is to compute collective variables, which are complex descriptors than can be used to analyze a conformational change or a chemical reaction. This can be done either on the fly, that is during molecular dynamics, or a posteriori, using PLUMED as a post-processing tool. In both cases one should create an input file with a specific PLUMED syntax. A sample input file is below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-1.txt
# this is optional and tell to VIM that this is a PLUMED file
# vim: ft=plumed
# see comments just below this input file

# Compute distance between atoms 1 and 10.
# Atoms are ordered as in the trajectory files and their numbering starts from 1.
# The distance is called "d" for future reference.
d: DISTANCE ATOMS=1,10

# Create a virtual atom in the center between atoms 20 and 30.
# The virtual atom only exists within PLUMED and is called "center" for future reference.
center: CENTER ATOMS=20,30

# Compute the torsional angle between atoms 1, 10, 20, and center.
# Notice that virtual atoms can be used as real atoms here.
# The angle is called "phi" for future reference.
phi: TORSION ATOMS=1,10,20,center
```

```
# Compute some function of previously computed variables.
# In this case we compute the cosine of angle phi and we call it "d2"
d2: MATHEVAL ...
  ARG=phi FUNC=cos(x)
  PERIODIC=NO
...
# The previous command has been split in multiple lines.
# It could have been equivalently written in a single line:
#   d2: MATHEVAL ARG=phi FUNC=cos(x) PERIODIC=NO

# Print d and d2 every 10 step on a file named "COLVAR1".
PRINT ARG=d,d2 STRIDE=10 FILE=COLVAR1

# Print phi on another file names "COLVAR2" every 100 steps.
PRINT ARG=phi STRIDE=100 FILE=COLVAR2
```

Note

If you are a VIM user, you might find convenient configuring PLUMED syntax files, see Using VIM syntax file. Syntax highlighting is particularly useful for beginners since it allows you to identify simple mistakes without the need to run PLUMED. In addition, VIM has a full dictionary of available keywords and can help you by autocomplete your commands.

In the input file above, each line defines a so-called action. An action could either compute a distance, or the center between two or more atoms, or print some value on a file. Each action supports a number of keywords, whose value is specified. Action names are highlighted in green and, clicking on them, you can go to the corresponding page in the manual that contains a detailed description for each keyword. Actions that support the keyword `STRIDE` are those that determine how frequently things are to be done. Notice that the default value for `STRIDE` is always 1. In the example above, omitting `STRIDE` keywords the corresponding COLVAR files would have been written for every frame of the analyzed trajectory. All the other actions in the example above do not support the `STRIDE` keyword and are only calculated when requested. That is, `d` and `d2` will be computed every 10 frames, and `phi` every 100 frames. In short, you can think that for every snapshot in the trajectory that you are analyzing PLUMED is going to execute all the listed actions, though some of them are optimized out when `STRIDE` is different from 1.
Also notice that PLUMED works using kJ/nm/ps as energy/length/time units. This can be personalized using UNITS, but we will here stay with default values.
Variables should be given a name (in the example above, `d`, `phi`, and `d2`), which is then used to refer to these variables. Instead of `a: DISTANCE ATOMS=1,2` you might equivalently use `DISTANCE ATOMS=1,2 L↩ ABEL=a`. Lists of atoms should be provided as comma separated numbers, with no space. Virtual atoms can be created and assigned a name for later use.
You can find more information on the PLUMED syntax at Getting Started page of the manual. The complete documentation for all the supported collective variables can be found at the Collective Variables page.
To analyze the trajectory provided here, you should:

- Create a PLUMED input file with a text editor (let us call it `plumed.dat`) similar to the one above.
- Run the command `plumed driver --mf_xtc traj.xtc --plumed plumed.dat`.

Here `traj.xtc` is the trajectory that you want to analyze. Notice that driver can read multiple file formats using embedded molfile plugins from VMD (that's where the `mf` letters come from).
Notice that you can also visualize trajectories with VMD directly. Trajectory `traj.xtc` can be visualized with the command `vmd ref.pdb traj-whole.xtc`.
In the following we will make practice with computing and printing collective variables.

11.38.1.6.1 Exercise 1: Computing and printing collective variables

Analyze the `traj-whole.xtc` trajectory and produce a colvar file with the following collective variables.

- The gyration radius of the solute RNA molecule (GYRATION). Look in the `ref.pdb` file which are the atoms that are part of RNA (search for the first occurrence of a water molecule, residue name `SOL`). Remember that you don't need to list all the atoms: instead of `ATOMS=1,2,3,4,5` you can write `ATOMS=1-5`.
- The torsional angle (TORSION) corresponding to the glycosidic chi angle χ of the first nucleotide. Since this nucleotide is a purine (guanine), the proper atoms to compute the torsion are O4' C1 N9 C4. Find their serial number in the `ref.pdb` file or learn how to select a special angle reading the MOLINFO documentation.

- The total number of contacts (COORDINATION) between all RNA atoms and all water oxygen atoms. For COORDINATION, set reference distance R_0 to 2.5 A (be careful with units!!). Try to be smart in selecting the water oxygen atoms without listing all of them explicitly.
- Same as before but against water hydrogen. Also in this case you should be smart to select water hydrogen atoms. Documentation of GROUP might help.
- Distance between the Mg ion and the geometric center of the RNA duplex (use CENTER and DISTANCE).

Notice that some of the atom selections can be made in a easier manner by using the MOLINFO keyword with a proper reference PDB file. Also read carefully the Groups and Virtual Atoms page before starting. Here you can find a sample plumed.dat file that you can use as a template. Whenever you see an highlightedFILL string, this is a string that you should replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-1.txt
# First load information about the molecule.
MOLINFO __FILL__
# Notice that this is special kind of "action" ("setup action")
# that is only used during setup. It will not be re-executed at each step.

# Define some group that will make the rest of the input more readable
# Here are the atoms belonging to RNA.
rna: GROUP ATOMS=1-258
# This is the Mg ion. A group with atom is also useful!
mg:  GROUP ATOMS=6580
# This group should contain all the atoms belonging to water molecules.
wat: GROUP ATOMS=__FILL__
# Select water oxygens only:
owat: GROUP __FILL__
# Select water hydrogens only:
hwat: GROUP __FILL__

# Compute gyration radius:
r: GYRATION ATOMS=__FILL__
# Compute the Chi torsional angle:
c: TORSION ATOMS=__FILL__
# Compute coordination of RNA with water oxygens
co: COORDINATION GROUPA=rna GROUPB=owat R_0=__FILL__
# Compute coordination of RNA with water hydrogens
ch: COORDINATION GROUPA=rna GROUPB=hwat __FILL__

# Compute the geometric center of the RNA molecule:
ce: CENTER ATOMS=__FILL__
# Compute the distance between the Mg ion and the RNA center:
d: DISTANCE ATOMS=__FILL__

# Print the collective variables on COLVAR file
# No STRIDE means "print for every step"
PRINT ARG=r,c,co,ch,d FILE=COLVAR
```

Once your plumed.dat file is complete, you can use it with the following command

```
> plumed driver --plumed plumed.dat --mf_xtc whole.xtc
```

Scroll in your terminal to read the PLUMED log. As you can see, PLUMED gives a lot of feedback about the input that he is reading. There's the place where you can check if PLUMED understood correctly your input.
The command above will create a file COLVAR like this one:

```
#! FIELDS time r c co ch d
#! SET min_c -pi
#! SET max_c pi
 0.000000 0.788694 -2.963150 207.795793 502.027244 0.595611
 1.000000 0.804101 -2.717302 208.021688 499.792595 0.951945
 2.000000 0.788769 -2.939333 208.347867 500.552127 1.014850
 3.000000 0.790232 -2.940726 211.274315 514.749124 1.249502
 4.000000 0.796395 3.050949 212.352810 507.892198 2.270682
```

Notice that the first line informs you about the content of each column and the second and third lines tell you that variable c (the χ torsion) is defined between $-\pi$ and $+\pi$.
In case you obtain different numbers, check your input, you might have made some mistake!
This file can then be shown with gnuplot

```
gnuplot> p "COLVAR" u 1:2, "" u 1:3
```

As a final note, look at what happens if you run the exercise twice. The second time, PLUMED will *back up* the previously produced file so as not to overwrite it. You can also *concatenate* your files by using the action RESTART at the beginning of your input file.
In this first exercise we only computed simple functions of the atomic coordinates. PLUMED is very flexible and allows you to also combine these functions to create more complicated variables. These variables can be useful when you want to describe a complex conformational change. PLUMED implements a number of functions that can be used to this aim that are described in the page Functions. Look at the following example:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-1.txt
# Distance between atoms 1 and 2:
d1: DISTANCE ATOMS=1,2
# Distance between atoms 1 and 3:
d2: DISTANCE ATOMS=1,3
# Distance between atoms 1 and 4:
d3: DISTANCE ATOMS=1,4

# Compute the sum of the squares of those three distances:
c: COMBINE ARG=d1,d2,d3 POWERS=2,2,2 PERIODIC=NO

# Sort the three distances:
s: SORT ARG=d1,d2,d3
# Notice that SORT creates a compund object with three components:
# s.1: the smallest distance
# s.2: the middle distance
# s.3: the largest distance

p: MATHEVAL ARG=d1,d2,d3 FUNC=x*y*z PERIODIC=NO

# Print the sum of the squares and the largest among the three distances:
PRINT FILE=COLVAR ARG=c,s.3
```

In case you have many distances to combine you can also use regular expressions to select them using `ARG=(d.)`, see Regular Expressions.
Notice for many functions you should say to PLUMED if the function is periodic. See Functions for a detailed explanation of how to choose this keyword.
You might think that it is easier to combine the variables after you have written them already, using, e.g., an awk or python script. That's fine if you are analyzing a trajectory. However, as we will learn later, computing variables within PLUMED you will be able to add bias potentials on those combinations, influencing their dynamics. Actually, you could implement any arbitrarily complex collective variable using just DISTANCE and MATHEVAL! Anyway, if the CV combinations that you are willing to use can be computed easily with some external program, do it and compare the results with the output of the PLUMED driver.

11.38.1.6.2 Exercise 1b: Combining collective variables

As an optional exercise, create a file with the following quantities:

- The sum of the distances between Mg and each of the phosphorous atoms.
- The distance between Mg and the closest phosphorous atom.

Notice that the serial numbers of the phosphorous atoms can be easily extracted using the following command

```
> grep ATOM ref.pdb | grep " P " | awk '{print $2}'
```

Here's a template input file to be completed by you.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-1.txt
# First load information about the molecule.
MOLINFO __FILL__

# Define some group that will make the rest of the input more readable
mg:  GROUP ATOMS=6580 # a group with one atom is also useful!

# Distances between Mg and phosphorous atoms:
d1: DISTANCE ATOMS=mg,33
d2: DISTANCE __FILL__
__FILL__
d6: DISTANCE __FILL__
# You can use serial numbers, but you might also use MOLINFO strings
```

```
# Compute the sum of these distances
c: COMBINE __FILL__

# Compute the distance between Mg and the closest phosphorous atom
s: SORT __FILL__

# Print the requested variables
PRINT FILE=COLVAR __FILL__
```

Notice that using the collective variable DISTANCES you might be able to do the same with a significantly simpler input file! If you have time, also try that and compare the result.
The resulting `COLVAR` file should look like this one:

```
#! FIELDS time c s.1
 0.000000 6.655622 0.768704
 1.000000 7.264049 0.379416
 2.000000 7.876489 0.817820
 3.000000 8.230621 0.380191
 4.000000 13.708759 2.046935
```

11.38.1.7 Solving periodic-boundary conditions issues

While running PLUMED can also dump the coordinate of the internally stored atoms using DUMPATOMS. This might seem useless (coordinates are already contained in the original trajectory) but can be used in the following cases:

- To dump coordinates of virtual atoms that only exist within PLUMED (e.g. a CENTER).
- To dump snapshots of our molecule conditioned to some value of some collective variable (see UPDATE_IF).
- To dump coordinates of atoms that have been moved by PLUMED.

The last point is perhaps the most surprising one. Some of the PLUMED actions can indeed move the stored atoms to positions better suitable for the calculation of collective variables.
The previous exercise was done on a trajectory where the RNA was already whole. For the next exercise you will use the `traj-broken.xtc` file instead, which is a real trajectory produced by GROMACS. Open it with VMD to understand what we mean with broken

```
> vmd ref.pdb traj-broken.xtc
```

Select `Graphics`, then `Representations`, then type `nucleic` in the box `Selected Atoms`. You will see that your RNA duplex is not whole. This is not a problem during MD because of periodic boundary conditions. However, it is difficult to analyze this trajectory. In addition, some collective variables that you might want to compute could require the molecules to be whole (an example of such variables is RMSD).
You might think that there are alternative programs that can be used to reconstruct the molecules the molecules that are broken by the periodic boundary correctly in your trajectory *before* analyzing it. However, you should keep in mind that if you need to compute CVs on the fly to add a bias potential on those (as we will to in the next tutorials) you will have to learn how to reconstruct the molecules that are broken by the periodic boundary within PLUMED. If you know alternative tools that can reconstruct the molecules that are broken by the periodic boundary, it is a good idea to also use them and compare the result with PLUMED.

11.38.1.7.1 Exercise 2: Solving PBC issues and dump atomic coordinates

Analyze the provided trajectory `traj-broken.xtc` and use the DUMPATOMS action to produce new trajectories in `gro` format that contain:

- The RNA duplex made whole (not broken by periodic boundary conditions). You should read carefully the documentation of WHOLEMOLECULES.
- The whole RNA duplex aligned to a provided template (structure `reference.pdb`). See FIT_TO_TEMPLATE, using `TYPE=OPTIMAL`. Notice that you should provide to FIT_TO_TEMPLATE a pdb file with only the atoms that you wish to align. Use the `ref.pdb` file as a starting point and remove the lines non containing RNA atoms. More details on PDB files in PLUMED can be found here.
- The whole RNA duplex and Mg ion, but only including the snapshots where Mg is at a distance equal to at most 4 A from phosphorous atom of residue 8. Search for the serial number of the proper phosphorous atom in the PDB file and use the UPDATE_IF action to select the frames.

- The whole RNA duplex plus water molecules and mg ion wrapped around the center of the duplex. Compute first the center of the duplex with CENTER then wrap the molecules with WRAPAROUND. Make sure that individual water molecules are not broken after the move!

Here you can find a template input file to be completed by you.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-1.txt
# First load information about the molecule.
MOLINFO __FILL__

# Define here the groups that you need.
# Same as in the previous exercise.
rna: GROUP ATOMS=__FILL__
mg:  GROUP ATOMS=__FILL__
wat: GROUP ATOMS=__FILL__

# Make RNA duplex whole.
WHOLEMOLECULES __FILL__

# Dump first trajectory in gro format.
# Notice that PLUMED understands the format based on the file extension
DUMPATOMS ATOMS=rna FILE=rna-whole.gro

# Align RNA duplex to a reference structure
# This should not be the ref.pdb file but a new file with only RNA atoms.
FIT_TO_TEMPLATE REFERENCE=__FILL__ TYPE=OPTIMAL
# Notice that before using FIT_TO_TEMPLATE we used WHOLEMOLECULES to make RNA whole
# This is necessary otherwise you would align a broken molecule!

# Dump the aligned RNA on a separate file
DUMPATOMS ATOMS=rna FILE=rna-aligned.gro

# Compute the distance between the Mg and the Phosphorous from residue 8
d: DISTANCE ATOMS=mg,__FILL__ ## put the serial number of the correct phosphorous here

# here we only dump frames conditioned to the value of d
UPDATE_IF ARG=d __FILL__
DUMPATOMS ATOMS=rna,mg FILE=rna-select.gro
UPDATE_IF ARG=d __FILL__ # this command is required to close the UPDATE_IF above

# compute the center of the RNA molecule
center: CENTER ATOMS=rna

# Wrap atoms correctly
WRAPAROUND ATOMS=mg AROUND=__FILL__
WRAPAROUND ATOMS=wat AROUND=center __FILL__ # anything missing here?

# Dump the last trajectory
DUMPATOMS ATOMS=rna,wat,mg FILE=rna-wrap.gro
```

After you have prepared a proper `plumed.dat` file, you can use it with the following command

```
> plumed driver --plumed plumed.dat --mf_xtc broken.xtc
```

Visualize the resulting trajectories using VMD. Since the `gro` files already contain atom names, you do not need to load the `pdb` file first. For instance, the first trajectory can be shown with

```
> vmd rna-whole.gro
```

TODO: I should perhaps add reference plots
If you just simulate a single solute molecule in water it is easy to understand how to pick the right options for WHOLEMOLECULES. However, if you have multiple molecules it can be rather tricky. In the example above, we used WHOLEMOLECULES on the RNA molecule which is actually a duplex, that is two separated chains. This was correct for the following reasons:

- the two chains are kept together by hydrogen bonds, and
- the last atom of the first chain is always close to the first atom of the second chain.

In case the two molecules can separate from each other this would be rather problematic.
We will now see what happens when using WHOLEMOLECULES on multiple molecules *incorrectly*.

11.38.1.7.2 Exercise 2b: Mistakes with WHOLEMOLECULES

Prepare a PLUMED input file that makes all the water molecules whole. Use the following template

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-1.txt
# First load information about the molecule.
MOLINFO __FILL__

# Define here the groups that you need
rna: GROUP ATOMS=__FILL__
mg:  GROUP ATOMS=__FILL__
wat: GROUP ATOMS=__FILL__

# Make RNA whole
WHOLEMOLECULES ENTITY0=rna

# Now make water whole as if it was a single molecule
WHOLEMOLECULES ENTITY0=wat

# And dump the resulting trajectory
DUMPATOMS ATOMS=rna,wat,mg FILE=wrong.gro
```

Now look at the resulting file with `vmd wrong.gro`. Can you understand which is the problem?
The important take-home message here is that when you want to reconstruct periodic boundary conditions correctly in systems with multiple molecules you should be careful and always verify with DUMPATOMS that the system is doing what you expect.
In an exercise above we used FIT_TO_TEMPLATE. This action uses as a reference a PDB file which typically contains a subset of atoms (those that are fitted). However, when you apply FIT_TO_TEMPLATE with `TYPE=OPTIMAL`, the whole system is translated and rotated. The whole system here means all atoms plus the vectors defining the periodic box.

11.38.1.7.3 Exercise 2c: Mastering FIT_TO_TEMPLATE

Check how the periodic box rotates when using FIT_TO_TEMPLATE. Use the following template

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-1.txt
# First load information about the molecule.
MOLINFO __FILL__

# Define here the groups that you need
rna: GROUP ATOMS=__FILL__
mg:  GROUP ATOMS=__FILL__
wat: GROUP ATOMS=__FILL__

# Make RNA whole
WHOLEMOLECULES ENTITY0=rna

# Here's a compund variable with the box vectors
# computed before aligning RNA
cell_before: CELL

# Now we align RNA
FIT_TO_TEMPLATE __FILL__ TYPE=OPTIMAL

# Here's a compund variable with the box vectors
# computed after aligning RNA
cell_after: CELL
PRINT ARG=cell_before.* FILE=CELL_BEFORE
PRINT ARG=cell_after.* FILE=CELL_AFTER
```

You should obtains files like the ones reported below.
`CELL_BEFORE` should be

```
#! FIELDS time cell_before.ax cell_before.ay cell_before.az cell_before.bx cell_before.by cell_before.bz cell_
 0.000000 4.533710 0.000000 0.000000 0.000000 4.533710 0.000000 2.266860 2.266860 3.205821
 1.000000 4.533710 0.000000 0.000000 0.000000 4.533710 0.000000 2.266860 2.266860 3.205821
 2.000000 4.533710 0.000000 0.000000 0.000000 4.533710 0.000000 2.266860 2.266860 3.205821
 3.000000 4.533710 0.000000 0.000000 0.000000 4.533710 0.000000 2.266860 2.266860 3.205821
 4.000000 4.533710 0.000000 0.000000 0.000000 4.533710 0.000000 2.266860 2.266860 3.205821
```

`CELL_AFTER` should be

```
#! FIELDS time cell_after.ax cell_after.ay cell_after.az cell_after.bx cell_after.by cell_after.bz cell_after.
 0.000000 4.533710 -0.000059 -0.000008 0.000059 4.533710 -0.000172 2.266895 2.266952 3.205730
```

```
1.000000 -0.396226 4.289476 -1.413481 -1.244340 1.260309 4.173460 2.249665 3.307132 2.134590
2.000000 -3.016552 1.123968 -3.192434 -1.055123 -4.375593 -0.543533 -4.309790 -1.356178 0.375612
3.000000 -4.083873 1.923282 -0.421306 0.339577 -0.267554 -4.513051 -3.243502 -2.069026 -2.398628
4.000000 -4.020722 2.094622 -0.029688 -1.060483 -1.979827 3.938298 -1.263169 2.532008 3.542306
```

As you can see, the generating vectors of the periodic lattice *before* fitting are constant. On the other hand, *after* fitting these vectors change so as to keep RNA correctly aligned to its reference structure.
Later on you will learn how to add a bias potential on a give collective variable. In principle, you could also add a RESTRAINT to the `cell_after.*` variables of the last example. This would allow you to force your molecule to a specific orientation.

11.38.1.7.4 Conclusions In summary, in this tutorial you should have learned how to use PLUMED to:

- Manipulate atomic coordinates.
- Compute collective variables.

All of this was done by just reading an already available trajectory. Notice that there are many alternative tools that could have been used to do the same exercise. Indeed, if you are familiar with other tools, it might be a good idea to also try them and compare the results. The special things of working with PLUMED are the following:

- PLUMED implements a vast library of useful collective variables. Browse the manual and search for ideas that are suitable for your system.
- PLUMED has a simple and intuitive syntax to combine collective variables ending up in descriptors capable to characterize complex conformational changes.
- And finally, the most special thing: any collective variable that can be computed within PLUMED can also be biased while you are running your MD simulation! You will learn more later about this topic.

The last point is probably the main reason why PLUMED exists and what distinguishes it from other available software.

11.38.2 Trieste tutorial: Averaging, histograms and block analysis

11.38.2.1 Introduction

The aim of this tutorial is for you to understand how we analyze trajectory data by calculating ensemble averages. One key point we try to make in this tutorial is that you must **always** calculate averaged quantities from our simulation trajectories. Consequently, in order to understand the analysis that is done when we perform molecular dynamics and Monte Carlo simulations, we will need to understand some things about probability and statistics. In fact, the things that we need to understand about probability and statistics are going to be the main subject of this tutorial. Therefore, in order to make our lives easier, we are not going to work with simulation trajectories in this tutorial. We are going to use model data instead.

11.38.2.2 Objectives

Once this tutorial is completed students will:

- Be able to explain the role played by the central limit theorem in many of the analysis methods that we perform on simulation trajectories.
- Be able to use PLUMED to calculate ensemble averages and histograms using the keywords AVERAGE and HISTOGRAM.
- Be able to use PLUMED to perform block analysis of trajectory data using the keywords AVERAGE and HISTOGRAM.
- Be able to use PLUMED to calculate unbiased ensemble averages and histograms from biased trajectories by using the keywords AVERAGE, HISTOGRAM and REWEIGHT_BIAS.
- Be able to explain how block analysis can be used to detect problems with error bar underestimation in correlated data.

11.38.2.3 Background

Let's begin by thinking about what is actually output by a simulation trajectory. You probably know by now that molecular dynamics gives us a trajectory that provides us with information on how the positions and velocities of the all the atoms in the system change with time. Furthermore, you should by now understand that we can use PLUMED to reduce the amount of information contained in each of the trajectory frames to a single number or a low-dimensional vector by calculating a collective variable or a set of collective variables. As we saw in Trieste tutorial: Analyzing trajectories using PLUMED this process of lowering the dimensionality of the data contained in a simulation trajectory was vital as we understand very little by watching the motions of the hundreds of atoms that the system we are simulating contains. If we monitor the appropriate key variables, however, we can determine whether a chemical reaction has taken place, whether the system has undergone a phase transition to a more ordered form or if a protein has folded. Even so if all we do is monitor the values the the collective variables take during the simulation our result can only ever be qualitative and we can only say that we observed this process to take place under these particular conditions on one particular occasion. Obviously, we would like to be able to do more - we would like to be able to make quantitative predictions based on the outcomes of our simulations.
The first step in moving towards making these quantitative predictions involves rethinking what precisely our simulation trajectory has provided us with. We know from rudimentary statistical mechanics that when we perform a molecular dynamics simulation in the NVT ensemble the values of the collective variables that we obtain for each of our trajectory frames, X, are samples from the following probability distribution:

$$P(s') = \frac{\int \mathrm{d}x\mathrm{d}p\delta(s(x) - s')e^{-\frac{H(x,p)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}}$$

In this expression the integral signs are used to represent $6N$-dimensional integrals that run over all the possible positions and momenta that the N atoms in our system can take. $H(x,p)$ is the Hamiltonian and k_B and T are the Boltzmann constant and the temperature respectively. The quantity calculated by this quotient is the probability that our CV will take a value s'. The quantity in the denominator of the above expression is the canonical partition function while the δ function in the integral in the numerator ensures that only those configurations that have a CV value, $s(x)$, equal to s' contribute to the integral in the numerator.
The fact that we know what distribution the X-values obtained from our trajectory frames are taken from is not particularly helpful as it is impossible to calculate the integrals in the expression above analytically. The fact that we know that our trajectory frames represent samples from a distribution is helpful, however, because of a result know as the Central Limit Theorem. This theorem states the following:

$$\lim_{n\to\infty} P\left(\frac{\frac{S_n}{n} - \langle Y \rangle}{\sqrt{\frac{\langle (\delta Y)^2 \rangle}{n}}} \leq z \right) = \Phi(z)$$

In this expression $\Phi(z)$ is the cumulative probability distribution function for a standard normal distribution and S_n is a sum of n independent samples from a probability distribution - in our case the probability distribution that we introduced in the previous equation. $\rangle$ is the ensemble average for the quantity $Y(x)$, which, in our case, we can calculate as follows:

$$\langle Y \rangle = \frac{\int \mathrm{d}x\mathrm{d}p Y(x) e^{-\frac{H(x,p)}{k_B T}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_B T}}}$$

Lastly, the quantity $\langle (\delta Y)^2 \rangle$ is a measure of the extent of the fluctuations we will observe in the Y values that we samples. This quantity is calculated using:

$$\langle (\delta Y)^2 \rangle = \langle Y^2 \rangle - \langle Y \rangle^2$$

The statement of the theorem provided above is compact way of stating the formal theorem. This is undoubtedly pleasing to mathematicians but to understand why this idea is so important in the context of molecular simulation it is useful to think about this in a slightly less formal way. The central limit theorem essentially tells us that if we take the set of random variables we extract from a simulation trajectory, which we know represent samples from the complicated distribution in the first equation above, and we add all these random variables together and divide by n we can think of the final number we obtain as a random variable that is taken from a Gaussian distribution. In other words, the probability density function for the sample mean, $\frac{S_n}{n}$, that we get from our trajectory frames is given by:

$$P(S_n) = \frac{1}{\sqrt{\frac{2\pi\langle (\delta Y)^2 \rangle}{n}}} \exp\left(-\frac{\frac{S_n}{n} - \langle Y \rangle}{2\frac{\langle (\delta Y)^2 \rangle}{n}} \right)$$

This function is shown plotted for various values of n in the movie at https://www.youtube.com/watch?v=-7hlP-2dG_o&feature=youtu.be.
You can see clearly that this distribution becomes more strongly peaked around $\langle Y \rangle$ (which I set equal to 0 in the movie) as n increases.
This observation is important as it ensures that the probability that $\frac{S_n}{n}$ lies close to the true value of the expectation value of our distribution, $\langle Y \rangle$, increases as we increase the value of n. The central limit theorem therefore allows us to get an estimate for the ensemble average for a particular quantity Y by taking repeated samples of Y from our distribution. These samples can be taken by, for example, performing a molecular dynamics simulation. Furthermore, and as we will see in the exercises that follow, we can also get an estimate of how much we might expect the system to fluctuate about this average. Incidentally, if you are confused at this stage you might want to work through these two videos and exercises in order to get a better understanding of the central limit theorem, confidence limits and error bars:

- Error bars exercise: http://gtribello.github.io/mathNET/error_bar_video.html
- Confidence limits exercise: http://gtribello.github.io/mathNET/central-limit-theorem-video.html

11.38.2.4 Instructions

11.38.2.4.1 Calculating an ensemble average As discussed in the introduction we are going to be using model data in this exercise so we must begin by generating some model data to analyze using PLUMED. The following short python script will generate 10000 (pseudo) random variables between 0 and 1 from a uniform distribution in a format that PLUMED can understand:

```
import random
print("#! FIELDS time rand")
for i in range(0,10001):
        print(i, random.uniform(0,1) )
```

Copy the contents of the box above to a plain text file called generate_data.py, save the file and then execute the script within it by running:

```
> python generate_data.py > my_data
```

This will generate a file called my_data that contains 10001 uniform random variables. The first few lines of this file should look something like this:

```
#! FIELDS time rand
0 0.7880696770414992
1 0.6384371499082688
2 0.01373858851099563
3 0.30879241147755776
```

PLUMED will ignore the first number in the colvar file as it assumes this is the initial configuration you provided for the trajectory. The sample mean will thus be calculated from 10000 random variables. Plot this data now using gnuplot and the command:

```
gnuplot> p 'my_data' u 1:2 w p
```

The probability distribution that we generated these random variables is considerably simpler than the probability distribution that we would typically sample from during an MD simulation. Consequently, we can calculate the exact values for the ensemble average and the fluctuations for this distribution. These are:

$$\langle X \rangle = \int_0^1 x \mathrm{d}x = \left[\frac{x^2}{2} \right]_0^1 = \frac{1}{2} \langle X^2 \rangle = \int_0^1 x^2 \mathrm{d}x = \left[\frac{x^3}{3} \right]_0^1 = \frac{1}{3} \langle (\delta X)^2 \rangle = \langle X^2 \rangle - \langle X \rangle^2 = \frac{1}{12}$$

Lets now try estimating these quantities by calculating $\frac{S_n}{n}$ from the points we generated and exploiting the central limit theorem. We can do this calculation by writing a PLUMED input file that reads:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
data: READ FILE=my_data VALUES=rand
d2: MATHEVAL ARG=data VAR=a FUNC=a*a PERIODIC=NO
av: AVERAGE ARG=data STRIDE=1
av2: AVERAGE ARG=d2 STRIDE=1
PRINT ARG=av,av2 STRIDE=10000 FILE=colvar
```

If you copy this input to a file called plumed.dat you can then run the calculation by executing:

```
> plumed driver --noatoms
```

When the calculation is finished you should have a file called colvar that contains the estimate of the ensemble averages $\langle X \rangle$ and $\langle X^2 \rangle$. To be clear, the quantities output in this file are:

$$\langle X \rangle = \frac{1}{N} \sum_{i=1}^{N} X_i \qquad \text{and} \qquad \langle X^2 \rangle \frac{1}{N} \sum_{i=1}^{N} X_i^2$$

We can calculate the fluctuations, $(\delta X)^2$, from them using:

$$(\delta X)^2 = \langle X^2 \rangle - \langle X \rangle^2$$

We can then compare the values that we got for these estimated values with those that we got for the true values. You should find that the agreement is reasonable but not perfect.

11.38.2.4.2 Calculating a histogram We can use what we have learned about calculating an ensemble average to calculate an estimate for the probability density function or probability mass function for our random variable. The theory behind what we do here is explained in this video http://gtribello.github.io/mathNE↩T/histogram-video.html
To do such a calculation with PLUMED on the random variables we generated from the uniform distribution in the previous section we would use an input like the one below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
data: READ FILE=my_data VALUES=rand
hh: HISTOGRAM ARG=data STRIDE=1 GRID_MIN=0 GRID_MAX=1.0 GRID_BIN=20 KERNEL=DISCRETE
DUMPGRID GRID=hh FILE=my_histogram.dat
```

Once again you can run this calculation by using the command:

```
> plumed driver --noatoms
```

Once this calculation is completed you can then plot the output using gnuplot and the command:

```
gnuplot> p 'my_histogram.dat' u 1:2 w l
```

In the previous section we compared the estimates we got for the ensemble average with the exact analytical values, which we could determine because we know that the data was sampled from a uniform distribution between 0 and 1. We can do a similar thing with the histogram. **Can you determine what the true (analytic) values for the probabilities in the histogram above should be?**
Also notice that the probability estimate for the first and last points in our histogram are lower than the probability estimates for the other points.
Can you determine why these two points have a lower probability?

11.38.2.4.3 Problem I: Making the best use of the data As discussed in previous sections the sample mean of the CV values from our trajectory frames is a random variable and the central limit theorem tells us something about the the **distribution** from which this random variable is drawn. The fact that we know what distribution the sample mean is drawn from is what allows us to **estimate** the ensemble average and the fluctuations. The fact that this recipe is only estimating the ensemble average is critical and this realization should always be at the forefront of our minds whenever we analyze our simulation data. The point, once again, is that the sample mean for CV values from the simulation is random. As is explained in this video http://gtribello.github.io/mathNE↩T/central-limit-theorem-video.html, however, we can use the central limit theorem to calculate a range, $\{\langle X \rangle - \epsilon, \langle X \rangle + \epsilon\}$, that the sample mean for the CV values, $\frac{S_n}{n}$, will fall into with a probability p_c using:

$$\epsilon = \sqrt{\frac{\langle (\delta X)^2 \rangle}{n}} \Phi^{-1} \left(\frac{p_c + 1}{2} \right)$$

Here Φ^{-1} is the inverse of the cumulative probability distribution function for a normal distribution with mean 0 and variance 1. As you can see this range gets smaller as the number of samples from which you calculate the mean,

n, increases. As is shown in the figure below, however, the rate at which this range narrows in size is relatively small.

This graph hopefully illustrates to you that an estimate of the ensemble average that is taken over 500 trajectory frames is not guaranteed to lie significantly closer to the true ensemble average than an estimate taken from 100 trajectory frames. For this reason, we might choose to split the data into blocks that all have equal length. Will will then estimate the average for each of these blocks separately. As we will see in the remainder of this exercise this process of block averaging has a number of other advantages. For now though we are just going to use it to test that the results from the various parts of ``the trajectory" are all consistent.
We can perform a block averaging on the data we generated at the start of the first exercise above by using PLUMED and the input file below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
data: READ FILE=my_data VALUES=rand
av: AVERAGE ARG=data STRIDE=1 CLEAR=1000
PRINT ARG=av STRIDE=1000 FILE=colvar
```

Once again this calculation can be executed by running the command:

```
> plumed driver --noatoms
```

The key differences between this input and the ones that we have seen previously is the CLEAR keyword on the line AVERAGE. The instruction CLEAR=1000 tells PLUMED that the data that has been accumulated for averaging should be reset to zero (cleared) every 1000 steps. If you look at the subsequent PRINT command in the above input you see the instruction STRIDE=1000. The above input thus accumulates an average over 1000 trajectory frames. This average is then output to the file colvar on step 1000 and the accumulated data is immediately cleared after printing so that a new average over the next 1000 steps can be accumulated. We can plot the averages that were output from the calculation above by using gnuplot and the following command:

```
gnuplot> p 'colvar' u 1:2 w l
```

If you try this now you should see that all the average values that were calculated are relatively consistent but that there are differences between them. **Try to calculate the size of ϵ for a 90 % confidence limit around this random variable using the formula that was provided above. How many of the averages that you extracted using PLUMED lie within this range? Is this behavior inline with your expectations based on your understanding of what a confidence limit is?**
We can also perform block averaging when we estimate histograms using PLUMED. The following input will calculate these block averaged histograms for the data we generated at the start of this exercise using PLUMED.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
data: READ FILE=my_data VALUES=rand
hh: HISTOGRAM ARG=data STRIDE=1 GRID_MIN=0 GRID_MAX=1.0 GRID_BIN=20 KERNEL=DISCRETE CLEAR=1000
DUMPGRID GRID=hh FILE=my_histogram.dat STRIDE=1000
```

Notice that the input here has the same structure as the input for the AVERAGE. Once again we have a CLE↩
AR=1000 keyword that tells PLUMED that the data that has been accumulated for calculating the histogram should be deleted every 1000 steps. In addition, we can set a STRIDE for DUMPGRID and thus output the histogram from each of these blocks of trajectory data separately. The difference between the output from this input and the output from the input above is that in this case we have multiple output files. In particular, the input above should give you 10 output files which will be called:

- analysis.0.my_histogram.dat = analysis of data in first 1000 frames
- analysis.1.my_histogram.dat = analysis of data in second 1000 frames
- analysis.2.my_histogram.dat = analysis of data in third 1000 frames
- analysis.3.my_histogram.dat = analysis of data in fourth 1000 frames
- analysis.4.my_histogram.dat = analysis of data in fifth 1000 frames
- analysis.5.my_histogram.dat = analysis of data in sixth 1000 frames
- analysis.6.my_histogram.dat = analysis of data in seventh 1000 frames
- analysis.7.my_histogram.dat = analysis of data in eighth 1000 frames

- analysis.8.my_histogram.dat = analysis of data in ninth 1000 frames
- my_histogram.dat = analysis of data in tenth 1000 frames

We can plot all of these histograms using gnuplot and the command:

```
gnuplot> p 'analysis.0.my_histogram.dat' u 1:2 w l, 'analysis.1.my_histogram.dat' u 1:2 w l, 'analysis.2.my_hi
```

Performing a comparison between the results from each of these blocks of data is more involved than the analysis we performed when comparing the ensemble averages as we have more data. The essential idea is the same, however, and, if you have time at the end of the session, you might like to see if you can write a script that determines what fraction of the many averages we have calculated here lie within the confidence limits.

11.38.2.4.4 Problem II: Dealing with rare events and simulation biases

As is discussed in many of the tutorials that are provided here one of the primary purposes of PLUMED is to use simulation biases to resolve the rare events problem. When we use these technique we modify the Hamiltonian, $H(x,p)$, and add to it some bias, $V(x)$. The modified Hamiltonian thus becomes:

$$H'(x,p) = H(x,p) + V(x)$$

and as such the values of the collective variables that we obtain from each of our trajectory frames are samples from the following distribution:

$$P(s') = \frac{\int \mathrm{d}x\mathrm{d}p\delta(s(x)-s')e^{-\frac{H(x,p)}{k_BT}}e^{-\frac{V(x)}{k_BT}}}{\int \mathrm{d}x\mathrm{d}p e^{-\frac{H(x,p)}{k_BT}}e^{-\frac{V(x)}{k_BT}}}$$

Using a bias in this way is enormously helpful as we can ensure that we sample from a particular part of configuration space. We appear to have sacrificed, however, the ability to extract estimates of ensemble averages for the unbiased distribution using the central limit theorem. If we calculate the mean from a set of trajectory frames that are sampled from the distribution above we will get an estimate for the ensemble average in this biased distribution. As we will see in this exercise, however, this is not really a problem as we can use reweighting techniques to extract ensemble averages for the unbiased distribution.

Lets begin by generating some new trajectory data. The following python script generates a set of random variables from a (truncated) normal distribution with $\sigma = 0.5$ and $\mu = 0.6$. In other words, points are generated from the following probability distribution but if they don't fall in a range between 0 and 1 they are discarded:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

```
import random
n = 0
print("#! FIELDS time rand")
while True :
        x = random.gauss( 0.6, 0.1732 )
        if (x>=0) & (x<=1) :
                print(n, x )
                n = n + 1
        if n==10001 : break
```

Copy the script above to a file called gen-normal.py and then execute the python script within it using the command:

```
> python gen-normal.py > normal_data
```

This should produce a file called normal_data. The first few lines of this file will look something like the following:

```
#! FIELDS time rand
0 0.7216627600374712
1 0.7460765782434674
2 0.7753011891527442
3 0.5643419178297695
```

Use what you have learned from the exercises above to estimate the ensemble average from these generated data points using PLUMED. If you want you can use block averaging but it does not matter if you do it by just considering all the data in the input file. What would you expect the ensemble average to be and how does the estimate you get from PLUMED compare with the true value?

You should have found that the ensemble average that you get when you perform the experiment described in the previous paragraph is different from the ensemble average that you get when you considered the uniform distribution. This makes sense as the distribution we sampled from here is approximately:

$$P(x) = \frac{1}{0.03 * \sqrt{2\pi}} \exp\left(-\frac{x - 0.6}{2(0.03)^2} \right)$$

This is different from the distribution we sampled from in the previous exercises. A question we might therefore ask is: can we extract the ensemble average for the uniform distribution that we sampled in previous exercises by sampling from this different distribution? In the context of the experiment we are performing here with the random variables this question perhaps feels somewhat absurd and pointless. In the context of molecular simulation, however, answering this question is essential as our ability to extract the true, unbiased distribution from a simulation run with a bias relies on being able to perform this sort of reweighting.
The true value of the ensemble average, which we estimated when we analyzed the data generated from the Gaussian using the python script above is given by:

$$\langle X \rangle = \frac{\int_0^1 x e^{-V(x)} \mathrm{d}x}{\int_0^1 e^{-V(x)} \mathrm{d}x} \qquad \text{where} \qquad V(x) = \frac{(x-x)^2}{2\sigma^2}$$

By contrast the ensemble average in the exercises involving the uniform distribution is given by:

$$\langle Y \rangle = \frac{\int_0^1 y \mathrm{d}y}{\int_0^1 \mathrm{d}y} = \frac{\int_0^1 y e^{-V(y)} e^{+V(y)} \mathrm{d}y}{\int_0^1 e^{-V(y)} e^{+V(y)} \mathrm{d}y}$$

We can use the final expression here to reweight the data that we sampled from the Gaussian. By doing so can thus extract ensemble averages for the uniform distribution. The trick here is calculate the following weighted mean rather than the unweighted mean that we calculated previously:

$$\langle Y \rangle \approx \frac{\sum_i Y_i e^{+V(Y_i)}}{\sum_i e^{+V(Y_i)}}$$

where here the sums run over all the random variables, the Y_is, that we sampled from the Gaussian distribution. If you used the script above the $V(Y_i)$ values were output for you so you can calculate this estimate of the ensemble average for the unbiased distribution in this case using the following PLUMED input.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
UNITS NATURAL  # This ensures that Boltzmann's constant is one
data: READ FILE=normal_data VALUES=rand IGNORE_FORCES
mm: RESTRAINT ARG=data AT=0.6 KAPPA=33.333
rw: REWEIGHT_BIAS TEMP=1
av: AVERAGE ARG=data STRIDE=1 LOGWEIGHTS=rw
PRINT ARG=av STRIDE=10000 FILE=colvar
```

Try to run this calculation now using:

```
> plumed driver --noatoms
```

and see how close you get to the ensemble average for the uniform distribution. Once you have done this try the following input, which allows you to compute the reweighted histogram.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
UNITS NATURAL # This ensures that Boltzmann's constant is one
data: READ FILE=normal_data VALUES=rand IGNORE_FORCES
mm: RESTRAINT ARG=data AT=0.6 KAPPA=33.333
rw: REWEIGHT_BIAS TEMP=1
hh: HISTOGRAM ARG=data STRIDE=1 GRID_MIN=0 GRID_MAX=1.0 GRID_BIN=20 KERNEL=DISCRETE LOGWEIGHTS=rw
DUMPGRID GRID=hh FILE=my_histogram.dat
```

Plot the histogram that you obtain from this calculation using the command:

```
gnuplot> p 'my_histogram.dat' w l
```

Now suppose that the data we generated for this exercise had come from an MD simulation. What would the unbiased Hamiltonian in this MD simulation have looked like and what functional form would our simulation bias have taken?

11.38.2.4.5 Problem III: Dealing with correlated variables As a good scientist you have probably noticed that we have failed to provide error bars around our estimates for the ensemble average in previous sections. Furthermore, you may have found that rather odd given that I showed you how to estimate the variance in the data in the very first exercise. This ignoring of the error bars has been deliberate, however, as there is a further problem with the data that comes from molecular dynamics trajectories that we have to learn to deal with and that will affect our ability to estimate the error bars. This problem is connected to the fact that **the central limit theorem only holds when we take a sum of uncorrelated and identical random variables.** Random variables that are generated from simulation trajectories will contain correlations simply because the system will most likely not diffuse from one edge of CV space to the other during a single timestep. In other words, the random CV value that we calculate from the $(i+1)$th frame of the trajectory will be similar to the value obtained from the ith trajectory frame. This problem can be resolved, however, and to show how we will thus use the following python script to generate some correlated model data:

```
import random
prev = 0.;
print("#! FIELDS time rand")
for i in range(0,10001):
        new = 0.95*prev + 2*random.uniform(0,1) - 1
        print( i, new/2. + 0.5 )
        prev = new
```

Copy the python script above to a filled called correlated-data.py and then execute the script using the command:

```
> python correlated-data.py > correlated_data
```

This should output a file called correlated_data. The first few lines of this file should look something like the following:

```
#! FIELDS time rand
0 0.17818391042061332
1 0.33368077871483476
2 0.0834749323925259
```

The autocorrelation function, $R(\tau)$ provides a simple method for determining whether or not there are correlations between the random variables, X, in our time series. This function is defined as follows:

$$R(\tau) = \frac{\langle (X_t - \langle X \rangle)(X_{t+\tau} - \langle X \rangle) \rangle}{\langle (\delta X)^2 \rangle}$$

At present it is not possible to calculate this function using PLUMED. If we were to do so for the samples taken from the uniform distribution and the correlated samples that were taken from the distribution the python script above we would find that the auto correlation functions for these random variables look something like the figures shown below:

To understand what these functions are telling us lets deal with the samples from the uniform distribution first. The autocorrelation function in this case has a value of 1 when τ is equal to 0 and a value of 0 in all other cases. This function thus tells us that each random variable is perfectly correlated with itself but that there are no correlations between the distributions we sampled adjacent variables from. In other words, each of the random variables we generate are independent. If we now look at the autocorrelation function for the distribution that is sampled by the python script above we see that the autocorrelation function slowly decays to zero. There are, therefore, correlations between the random variables that we generate that we must account for when we perform our analysis.

We account for the correlations in the data when we do our analysis by performing a block analysis. To understand what precisely this involves we are going to perform the analysis with PLUMED and explain the results that we get at each stage of the process. We will begin by analyzing the data we generated by sampling from the uniform distribution using the following PLUMED input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
data: READ FILE=correlated_data VALUES=rand
av5: AVERAGE ARG=data STRIDE=1 CLEAR=5
PRINT ARG=av5 FILE=colvar5 STRIDE=5
av10: AVERAGE ARG=data STRIDE=1 CLEAR=10
PRINT ARG=av10 FILE=colvar10 STRIDE=10
av15: AVERAGE ARG=data STRIDE=1 CLEAR=15
PRINT ARG=av15 FILE=colvar15 STRIDE=15
av20: AVERAGE ARG=data STRIDE=1 CLEAR=20
PRINT ARG=av20 FILE=colvar20 STRIDE=20
av25: AVERAGE ARG=data STRIDE=1 CLEAR=25
PRINT ARG=av25 FILE=colvar25 STRIDE=25
av30: AVERAGE ARG=data STRIDE=1 CLEAR=30
```

```
PRINT ARG=av30 FILE=colvar30 STRIDE=30
av35: AVERAGE ARG=data STRIDE=1 CLEAR=35
PRINT ARG=av35 FILE=colvar35 STRIDE=35
av40: AVERAGE ARG=data STRIDE=1 CLEAR=40
PRINT ARG=av40 FILE=colvar40 STRIDE=40
av45: AVERAGE ARG=data STRIDE=1 CLEAR=45
PRINT ARG=av45 FILE=colvar45 STRIDE=45
av50: AVERAGE ARG=data STRIDE=1 CLEAR=50
PRINT ARG=av50 FILE=colvar50 STRIDE=50
av55: AVERAGE ARG=data STRIDE=1 CLEAR=55
PRINT ARG=av55 FILE=colvar55 STRIDE=55
av60: AVERAGE ARG=data STRIDE=1 CLEAR=60
PRINT ARG=av60 FILE=colvar60 STRIDE=60
av65: AVERAGE ARG=data STRIDE=1 CLEAR=65
PRINT ARG=av65 FILE=colvar65 STRIDE=65
av70: AVERAGE ARG=data STRIDE=1 CLEAR=70
PRINT ARG=av70 FILE=colvar70 STRIDE=70
```

Copy the input above to a file called plumed.dat and run the calculation using the command:

```
> plumed driver --noatoms
```

This calculation should output 14 colvar files, which should then be further analyzed using the following python script:

```
import numpy as np
import math
for i in range(1,15):
        # Read in each colvar file
        fmult = 5*i
        dat = np.loadtxt( 'colvar' + str(fmult) )
        # Compute the square of all the average values in the colvar
        sq = dat[1:,1]**2
        # Now compute the average over all the averages
        mean = np.sum( dat[1:,1] ) / len( dat[1:,1] )
        # Compute the average of the squares of the individual averages
        mean2 = np.sum( sq ) / len( sq )
        # Compute the population variance amongst the block averages
        population_variance = mean2 - mean*mean
        # Convert the population variance into a sample variance by multiplying by the bessel factor
        sample_variance = ( len( sq ) / ( len(sq) - 1 ) )*population_variance
        # Print out the length of the blocks, the final average taken over all blocks and the square
        # root of the sample variance divided by the number of data points that this estimate was
        # calcualted from.  This last term is a measure of the error bar
        print( fmult, mean, math.sqrt( sample_variance / len(sq) ) )
```

Copy this script to a file called block-average-script.py and then execute the contents using the command:

```
> python block-average-script.py > my_averages.dat
```

This will output a single file called my_averages.dat that can be plotted using gnuplot and the command:

```
gnuplot> p 'my_averages.dat' u 1:2:3 w e, 'my_averages.dat' w l
```

The final result should be a graph like that shown in the left panel of the figure below. The figure on the right shows what you should obtain when you repeat the same analysis on the correlated data that we generated using the python script in this section.

The output that you obtain from these two calculations is explained in the video at: http://gtribello.github.io/mathNET/block_averaging_video.html Before watching this explanation though take some time to note down the differences between the two graphs above. Try to look through the scripts above and to understand what is being done in the PLUMED inputs and the python scripts above so as to work out what the data in these two figures are telling you.

11.38.2.4.6 Putting it all together In this final exercise we are going to try to combine everything we have seen in the previous sections. We are going to sample from the distribution that was introduced in Problem II: Dealing with rare events and simulation biases. This time though we are not going to generate random variables from the Gaussian directly. We are instead going to use Monte Carlo sampling. This sampling method is going to give us correlated data so we will need to use ideas from Problem III: Dealing with correlated variables in order to get a proper estimate of the error bars. Furthermore, we are not going to try to extract ensemble averages that tell us about the distribution we sampled from. Instead we are going to reweight using the ideas from Problem II: Dealing with rare events and simulation biases and extract the unbiased distribution.

The first step in doing all this is, as always, to generate some data. The python script below will generate this data:

```
import math
import random
# Energy given by a harmonic potential centered at 0.6
# This ensures that our data represent samples from a Gaussian with
# mean 0.6 and variance 0.1732
def calc_eng( x ) :
        return 0.5*33.333*pow((x-0.6),2)
x = 0.5
eng = calc_eng( x )
print("#! FIELDS time rand")
for i in range(0,100010):
        # Generate new random position from old position
        newx = x + 0.1*random.uniform(0,1) - 0.05
        # Apply periodic boundary conditions
        if( newx > 1.0 ) : newx = newx - 1.0
        if( newx < 0.0 ) : newx = newx + 1.0
        # Monte Carlo criterion
        new_eng = calc_eng( newx )
        if( new_eng<eng ) :
                x, eng = newx, new_eng
        elif( random.uniform(0,1)<math.exp(-new_eng)/math.exp(-eng) ) :
                x, eng = newx, new_eng
        if( i%10==0 ) : print( i/10, x )
```

Copy this script to a filled called do-monte-carlo.py and execute the contents of the script using the command:

```
> python do-monte-carlo.py > monte_carlo_data
```

This will run a short Monte Carlo simulation that generates (time-correlated) random data from a (roughly) Gaussian distribution by attempting random translational moves of up to 0.1 units. The command above will generate a file called monte_carlo_data the first few lines of which should look something like the following:

```
#! FIELDS time rand
0 0.17818391042061332
1 0.33368077871483476
2 0.0834749323925259
```

An autocorrelation function that was calculated using data generated using this script is shown below. You can clearly see from this figure that there are correlations between the adjacent data points in the time series and that we have to do block averaging as a result.

We can use the following PLUMED input to calculate block averages for the unbiased histogram from the data we generated:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-2.txt
UNITS NATURAL # This ensures that Boltzmann's constant is one
data: READ FILE=monte_carlo_data VALUES=rand IGNORE_FORCES
mm: RESTRAINT ARG=data AT=0.6 KAPPA=33.333
rw: REWEIGHT_BIAS TEMP=1
hh: HISTOGRAM ARG=data STRIDE=1 GRID_MIN=0 GRID_MAX=1.0 GRID_BIN=20 KERNEL=DISCRETE LOGWEIGHTS=rw CLEAR=500
DUMPGRID GRID=hh FILE=my_histogram.dat STRIDE=500
```

Notice that this input instructs PLUMED to calculate block averages for the histogram from each set of 500 consecutive frames in the trajectory.
I have worked out that this is an appropriate length of time to average over by performing the analysis described in Problem III: Dealing with correlated variables.
We will come back to how precisely I did this momentarily, however. For the time being though you can execute this input using:

```
> plumed driver --noatoms
```

Executing this command will generate a number of files containing histograms. The following python script will merge all this data and calculate the final histogram together with the appropriate error bars.

```
import math
import glob
import numpy as np
# Here are some numbers you will need to change if you run this script on grids generated in different
      contexts
nquantities = 1          # Number of quanities that have been evaluated on the grid
grid_dimension = 1       # Number of collective variables that you provided using the ARG keyword
filename = "my_histogram.dat"  # The name you specified the data to output to in the DUMPGRID command
# Function to read in histogram data and normalization
def readhistogram( fname ) :
```

```
        # Read in the histogram data
        data = np.loadtxt( fname )
        with open( filename, "r" ) as myfile :
                for line in myfile :
                        if line.startswith("#! SET normalisation") : norm = line.split()[3]
        return float(norm), data
# Read in the grid file header to work out what fields we have
with open( filename, "r" ) as myfile :
        for line in myfile :
                if line.startswith("#! FIELDS") : fieldnames = line.split()
# Check if derivatives have been output in the grid by investigating the header
nextg = 1
if len(fieldnames)>(2+grid_dimension+nquantities) :
        nextg = 1 + grid_dimension
        assert len(fieldnames)==(2+grid_dimension + nquantities*nextg)
# Read in a grid
norm, griddata = readhistogram( filename )
norm2 = norm*norm
# Create two np array that will be used to accumulate the average grid and the average grid squared
average = np.zeros((nquantities, len(griddata[:,0])))
average_sq = np.zeros((nquantities, len(griddata[:,0])))
for i in range(0,nquantities) :
        average[i,:] = norm*griddata[:,nquantities+i*nextg]
        average_sq[i,:] = norm*griddata[:,nquantities+i*nextg]*griddata[:,nquantities+i*nextg]
# Now sum the grids from all all the analysis files you have
for filen in glob.glob( "analysis.*." + filename ) :
        tnorm, newgrid = readhistogram( filen )
        norm = norm + tnorm
        norm2 = norm2 + tnorm*tnorm
        for i in range(0,nquantities) :
                average[i,:] = average[i,:] + tnorm*newgrid[:,nquantities+i*nextg]
                average_sq[i,:] = average_sq[i,:] +
      tnorm*newgrid[:,nquantities+i*nextg]*newgrid[:,nquantities+i*nextg]
# Compte the final average grid
average = average / norm
# Compute the sample variance for all grid points
variance = (average_sq / norm) - average*average
# Now multiply by bessel correction to unbias the sample variance and get the population variance
variance = ( norm /(norm-(norm2/norm)) ) * variance
# And lastly divide by number of grids and square root to get an error bar for each grid point
ngrid = 1 + len( glob.glob( "analysis.*." + filename ) )
errors = np.sqrt( variance / ngrid )
# Calculate average error over grid and output in header
for i in range(0,nquantities) :
        mean_error = sum(errors[i,:]) / len(errors[i,:])
        print("# Average error for " + str(i+1) + "th averaged function on grid equals ", mean_error )
# Output the final average grid
for i in range(0,len(griddata[:,0])) :
        for j in range(0,grid_dimension) : print( griddata[i,j], end=" " )
        for j in range(0,nquantities) : print( average[j,i], errors[j,i], end=" " )
        print()
```

Copy this script to a file called merge-histograms.py and then run its contents by executing the command:

```
> python merge-histograms.py > final-histogram.dat
```

This will output the final average histogram together with some error bars. You can plot this function using gnuplot by executing the command:

```
gnuplot> p 'final-histogram.dat' u 1:2:3 w e, '' u 1:2 w l
```

Where are the error bars in our estimate of the histogram the largest? Why are the errors large in these regions?

Notice that the file output by the script above also contains information on the average error per grid point in the header. The quantity that is calculated here is:

$$\text{average error} = \frac{1}{N}\sum_{i=1}^{N}\sigma_i$$

In this expression N is the total number of grid points at which the function was evaluated and σ_i is the error bar for the estimate of the function that was calculated for the ith grid point. The average error is a useful quantity as we can plot it and thus check that our blocks are large enough to correct for the correlation between our data points. In other words, we can use this quantity in the same way that we used the error around the average in Problem III: Dealing with correlated variables. A plot of the average error versus the size of the blocks that were used in for the block averaging is shown below. This figure demonstrates that a block average length of 500 is certainly long enough to correct for the problems due to the correlations in the values produced at different times:

If you have sufficient time try to see if you can reproduce this plot using the data you generated

11.38.2.5 Extensions

The exercises in the previous sections have shown you how we can calculate ensemble averages from trajectory data. You should have seen that performing block averaging is vital as this technique allows us to deal with the artifacts that we get because of the correlations in the data that we obtain from simulation trajectories.
What we have seen is that when this technique is used correctly we get reasonable error bars. If block averaging is not performed, however, we can underestimate the errors in our data and thus express a false confidence in the reliability of our simulation results.

The next step for you will be to use this technique when analyzing the data from your own simulations.

11.38.3 Trieste tutorial: Using restraints

11.38.3.1 Aims

The aim of this tutorial is to introduce the users to the use of constant biases in PLUMED.

11.38.3.2 Objectives

- Apply a restraint on a simulations over one or more collective variables
- Understand the effect of a restraint on the acquired statistics
- Perform a simple un-biasing of a restrained simulation
- Add an external potential in the form of an analytical or numerical function

11.38.3.3 Resources

The TARBALL for this tutorial contains the following files:

- wdimer.pdb: a PDB file for two molecules of water in vacuo
- wdimer.tpr: a GROMACS run file to perform MD of two water molecules
- diala.pdb: a PDB file for alanine dipeptide in vacuo
- diala.tpr: a GROMACS run file to perform MD of alanine dipeptide
- do_block_histo.py: the python script from Trieste tutorial: Averaging, histograms and block analysis to perform block averaging over histograms

This tutorial has been tested on a pre-release version of version 2.4. However, it should not take advantage of 2.4-only features, thus should also work with version 2.3.

11.38.3.4 Introduction

PLUMED can calculate conformational properties of a system a posteriori as well as on-the-fly. This information can be use to manipulate a simulation on-the-fly. This means adding energy terms in addition to those of the original Hamiltonian. This additional energy terms are usually referred as Bias. In the following we will see how to apply a constant bias potential with PLUMED. It is preferable to run each exercise in a separate folder.

11.38.3.4.1 Biased sampling A system at temperature T samples conformations from the canonical ensemble:

$$P(q) \propto e^{-\frac{U(q)}{k_B T}}$$

. Here q are the microscopic coordinates and k_B is the Boltzmann constant. Since q is a highly dimensional vector, it is often convenient to analyze it in terms of a few collective variables (see Trieste tutorial: Analyzing trajectories using PLUMED). The probability distribution for a CV s is

$$P(s) \propto \int dq e^{-\frac{U(q)}{k_B T}} \delta(s - s(q))$$

This probability can be expressed in energy units as a free energy landscape $F(s)$:

$$F(s) = -k_B T \log P(s)$$

.
Now we would like to modify the potential by adding a term that depends on the CV only. That is, instead of using $U(q)$, we use $U(q) + V(s(q))$. There are several reasons why one would like to introduce this potential. One is to avoid that the system samples some un-desired portion of the conformational space. As an example, imagine that you want to study dissociation of a complex of two molecules. If you perform a very long simulation you will be able to see association and dissociation. However, the typical time required for association will depend on the size of the simulation box. It could be thus convenient to limit the exploration to conformations where the distance between the two molecules is lower than a given threshold. This could be done by adding a bias potential on the distance between the two molecules. Another example is the simulation of a portion of a large molecule taken out from its initial context. The fragment alone could be unstable, and one might want to add additional potentials to keep the fragment in place. This could be done by adding a bias potential on some measure of the distance from the experimental structure (e.g. on root-mean-square deviation).
Whatever CV we decide to bias, it is very important to recognize which is the effect of this bias and, if necessary, remove it a posteriori. The biased distribution of the CV will be

$$P'(s) \propto \int dq e^{-\frac{U(q)+V(s(q))}{k_B T}} \delta(s - s(q)) \propto e^{-\frac{V(s(q))}{k_B T}} P(s)$$

and the biased free energy landscape

$$F'(s) = -k_B T \log P'(s) = F(s) + V(s) + C$$

Thus, the effect of a bias potential on the free energy is additive. Also notice the presence of an undetermined constant C. This constant is irrelevant for what concerns free-energy differences and barriers, but will be important later when we will learn the weighted-histogram method. Obviously the last equation can be inverted so as to obtain the original, unbiased free-energy landscape from the biased one just subtracting the bias potential

$$F(s) = F'(s) - V(s) + C$$

Additionally, one might be interested in recovering the distribution of an arbitrary observable. E.g., one could add a bias on the distance between two molecules and be willing to compute the unbiased distribution of some torsional angle. In this case there is no straightforward relationship that can be used, and one has to go back to the relationship between the microscopic probabilities:

$$P(q) \propto P'(q) e^{\frac{V(s(q))}{k_B T}}$$

The consequence of this expression is that one can obtained any kind of unbiased information from a biased simulation just by weighting every sampled conformation with a weight

$$w \propto e^{\frac{V(s(q))}{k_B T}}$$

That is, frames that have been explored in spite of a high (disfavoring) bias potential V will be counted more than frames that has been explored with a less disfavoring bias potential.
We will make use of two toy models: the first is a water dimer, i.e. two molecules of water in vacuo, that we will use to compare the effect of a constant bias on the equilibrium properties of the system that in this case can be readily computed. The second toy model is alanine dipeptide in vacuo. This system is more challenging to characterize with a standard MD simulation and we will see how we can use an interactive approach to to build a constant bias that will help in flattening the underlying free energy surface and thus sped up the sampling.

Note

Create a folder for each exercise and use sub-folders if you want to run the same simulation with multiple choices for the parameters

11.38.3.5 Exercise 1: converged histogram of the water dimer relative distance

First of all let's start to learn something about the water dimer system by running a first simulations. You can start by creating a folder with the dimer.tpr file and run a simulation.

```
> gmx mdrun -s dimer.tpr
```

In this way we have a 25ns long trajectory that we can use to have a first look at the behavior of the system. Is the sampling of the relative distance between the two water molecules converged?
Use plumed driver to analyze the trajectory and evaluate the quality of the sampling.
Here you can find a sample `plumed.dat` file that you can use as a template. Whenever you see an highlightedF↩ ILL string, this is a string that you should replace.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-3.txt
# vim:ft=plumed
#compute the distance between the two oxygens
d: DISTANCE ATOMS=1,4
#accumulate block histograms
hh: HISTOGRAM ARG=d STRIDE=10 GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=200 KERNEL=DISCRETE CLEAR=10000
#and dump them
DUMPGRID GRID=hh FILE=my_histogram.dat STRIDE=10000
# Print the collective variable.
PRINT ARG=d STRIDE=10 FILE=distance.dat

> plumed driver --mf_xtc traj_comp.xtc --plumed plumed.dat
> python3 do_block_histo.py > final-histo-10000.dat
```

If there is something you don't remember about this procedure go back and check in Trieste tutorial: Averaging, histograms and block a . There you can also find a python script to perform block averaging of the histograms and assess the error. The result should be comparable with the following: Notice the peak at 0.9 nm, this is the effect of using cut-off for the calculation of the interactions in the simulation (check the run-dimer.mdp file for the properties of the run)

11.38.3.6 Exercise 2: Apply a linear restraint on the same collective variable

Now we will try to apply a linear restraint on the relative distance and compare the resulting distribution. The new sampling will reflect the effect of the bias. Be careful about the statistics: in the simulation of exercise 1 you were post processing a trajectory of 125000 frames accumulating one frame every ten in an histogram and clearing the histogram after 10000 steps. As a result you had 12 blocks in the form of 11 analysis.$*$ files and a final block named my_histogram.dat. In the following try to accumulate on the fly the same amount of statistics. Look into the .mdp file to see how often frames are written in a trajectory. If you write too many analysis.$*$ files (i.e. 100 files plumed will fail with an error).

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
# vim:ft=plumed
#compute the distance between the two oxygens
d: DISTANCE __FILL__
#accumulate block histograms
hh: HISTOGRAM ARG=d KERNEL=DISCRETE STRIDE=500 CLEAR=500000 GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=200
#and dump them
DUMPGRID __FILL__

#apply a linear restraint
lr: RESTRAINT ARG=d KAPPA=0 AT=0 SLOPE=2.5

# Print the collective variable and the bias.
PRINT __FILL__
```

In a new folder we can run this new simulation this time biasing and analyzing the simulation on-the-fly.

```
> gmx mdrun -s dimer.tpr -plumed plumed.dat
```

The histogram should look different.
The effect of a constant bias is that of systematically changing the probability of each conformation by a factor $\exp(+V_{bias}/k_BT)$. This means that it is easily possible to recover the un-biased distribution at least in the regions of the conformational space that have been thoroughly sampled. In practice the statistical weight of each frame is not 1 anymore but is given by the exponential of the bias.
In order to recover the unbiased distribution we can post process the simulation using plumed driver to recalculate the bias felt by each frame and store this information to analyze any property. Furthermore plumed can also automatically use the bias to reweight the accumulated histogram.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
# vim:ft=plumed
d: DISTANCE __FILL__

lr: RESTRAINT __FILL__
as: REWEIGHT_BIAS TEMP=298

HISTOGRAM ...
  LOGWEIGHTS=as
  ARG=d
  STRIDE=10
  GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=200
  KERNEL=DISCRETE
  CLEAR=10000
... HISTOGRAM

DUMPGRID __FILL__
PRINT ARG=*.* FILE=COLVAR STRIDE=1
```

Be careful again about the difference in the way statistics is accumulated on-the-fly or for post processing. This is not critical for the result but is important in order to have comparable histograms, that is histograms with comparable noise. Remember to give different names to the new histogram otherwise the one obtained before will be overwritten.

```
> plumed driver --mf_xtc traj_comp.xtc --plumed plumed.dat
```

Note

To run block analysis of both sets of histograms you need to edit the python script because the file name is hard coded.

```
> python3 do_block_histo.py > histo-biased.dat
> python3 do_block_histo.py > histo-reweighted.dat
```

Now the resulting histogram should be comparable to the reference one.

11.38.3.7 Exercise 3: Apply a quadratic restraint on the same collective variable

Do you expect a different behavior? This time we can write the plumed input file in such a way to compare directly the biased and unbiased histograms.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-3.txt
# vim:ft=plumed
#calculate the distance
d: DISTANCE ATOMS=1,4
#apply the quadratic restraint centered at a distance of 0.5 nm
lr: RESTRAINT ARG=d KAPPA=10 AT=0.5
#accumulate the biased histogram
hh: HISTOGRAM ARG=d STRIDE=500 GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=200 KERNEL=DISCRETE CLEAR=500000
#dumpit
DUMPGRID GRID=hh FILE=my_histogram.dat STRIDE=500000
#calculate the weights from the constant bias
as: REWEIGHT_BIAS TEMP=298
#accumulate the unbiased histogram
hhu: HISTOGRAM ARG=d STRIDE=500 GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=200 KERNEL=DISCRETE CLEAR=500000 LOGWEIGHTS=a
#dumpit
DUMPGRID GRID=hhu FILE=myhistu.dat STRIDE=500000
#print distance and bias
PRINT ARG=d,lr.bias FILE=distance.dat STRIDE=50
```

The comparison of the two histograms with the former will show the effect of the weak quadratic bias on the simulation.

Note

To run block analysis of both sets of histograms you need to edit the python script because the file name is hard coded.

```
> python3 do_block_histo.py > histo-biased.dat
> python3 do_block_histo.py > histo-reweighted.dat
```

11.38.3.8 Exercise 4: Apply an upper wall on the distance.

In the above cases we have always applied weak biases. Sometimes biases are useful to prevent the system in reaching some region of the conformational space. In this case instead of using RESTRAINT , we can make use of lower or upper restraints, e.g. LOWER_WALLS and UPPER_WALLS.
What happen to the histogram when we use walls?

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
# vim:ft=plumed
d: DISTANCE ATOMS=1,4
uw: UPPER_WALLS ARG=d KAPPA=1000 AT=2.5
# accumulate the biased histogram
__FILL__
#dumpit
__FILL__
# calculate the weights from the constant bias
__FILL__
#accumulate the unbiased histogram
__FILL__
#dumpit
__FILL__
#print distance and bias
__FILL__
```

Run it.

```
> gmx mdrun -s dimer.tpr -plumed plumed.dat
```

If we have not sampled a region thoroughly enough it is not possible to estimate the histogram in that region even using reweighting (reweighting is not magic!).

11.38.3.9 Exercise 5: Evaluate the free energy and use it as an external restraint

The main issue in sampling rare events is that importance sampling algorithms spend more time in low energy regions and if two low energy regions are separated by a high energy one is unlikely for the sampling algorithm to cross the high energy region and reach the other low energy one. From this point of view an algorithm based on random sampling will work better in crossing the barrier. A particularly efficient sampling can be obtained if one would know the underlying free energy and thus use that to bias the sampling and make the sampling probability uniform in the regions of relevant interest. In this exercise we will make use of the free-energy estimate along the distance collective variable to bias the sampling of the same collective variable in the dimer simulation. To do so we will make use of a table potential applied using the Bias EXTERNAL. We first need to get a smooth estimate of the free-energy from our fist reference simulations, we will do this by accumulating a histogram with kernel functions, that is continuous function centered at the value of the accumulated point and added accumulated on the discrete representation of the histogram, see Kernel density estimation .

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
# vim:ft=plumed
#calculate the distance
d: DISTANCE ATOMS=1,4
#accumulate the histogram using a gaussian kernel with 0.05 nm width
hh2: HISTOGRAM ARG=d STRIDE=10 GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=400 BANDWIDTH=0.05
#convert to a free energy
ff: CONVERT_TO_FES GRID=__FILL__ TEMP=__FILL__
#dump the free energy
DUMPGRID GRID=__FILL__ FILE=__FILL__
```

by running plumed driver on the reference trajectory we obtain a free energy estimate.

```
> plumed driver --mf_xtc traj_comp.xtc --plumed plumed.dat
```

The resulting file for the free energy should be edited in order to:

- Invert the sign of the free-energy and of its derivative
- Remove some unused flag and regions with infinite potential at the boundaries

The file looks like:

```
#! FIELDS d ff dff_d
#! SET min_d 0
#! SET max_d 4.0
#! SET nbins_d  400
#! SET periodic_d false
0.060000 -34.9754 185.606
0.070000 -26.0117 184.362
0.080000 -20.8195 181.39
0.090000 -17.5773 176.718
```

where the first column is the grid spacing, the second the free energy and the third the derivative of the free energy. You can edit the file as you want, for example using the following bash lines:

```
grep \# ff.dat | grep -v normalisation > external.dat
grep -v \# ff.dat | awk '{print $1, -$2, -$3}' | grep -v inf >> external.dat
```

Furthermore edit the first line of external.dat from

```
#! FIELDS d ff dff_d
```

to

```
#! FIELDS d ff.bias der_d
```

Now we have an external potential that is the opposite of the free energy and we can use it in a new folder to bias a simulation:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
# vim:ft=plumed
d: DISTANCE ATOMS=1,4
EXTERNAL ARG=d FILE=__FILL__ LABEL=ff
# accumulate the biased histogram
__FILL__
#dumpit
__FILL__
# calculate the weights from the constant bias
__FILL__
#accumulate the unbiased histogram
__FILL__
#dumpit
__FILL__
#print distance and bias
__FILL__
```

Run it.

```
> gmx mdrun -s dimer.tpr -plumed plumed.dat
```

How do the biased and unbiased histograms look like? In the following we will apply this concept to sample the conformational space of a more complex system.

11.38.3.10 Exercise 6: Preliminary run with Alanine dipeptide

Alanine dipeptide is characterized by multiple minima separated by relatively high free energy barriers. Here we will explore the conformational space of alanine dipeptide using a standard MD simulation, then instead of using the free energy as an external potential we will try to fit the potential using gnuplot and add a bias using an analytical function of a collective variable with MATHEVAL and BIASVALUE .
As a first test lets run an MD and generate on-the-fly the free energy as a function of the phi and psi collective variables separately.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-3.txt
#SETTINGS MOLFILE=user-doc/tutorials/old_tutorials/trieste-3/aladip/aladip.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=aladip.pdb
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2
hhphi: HISTOGRAM ARG=phi STRIDE=10 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05
hhpsi: HISTOGRAM ARG=psi STRIDE=10 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05
ffphi: CONVERT_TO_FES GRID=hhphi TEMP=298
ffpsi: CONVERT_TO_FES GRID=hhpsi TEMP=298
DUMPGRID GRID=ffphi FILE=ffphi.dat
DUMPGRID GRID=ffpsi FILE=ffpsi.dat
PRINT ARG=phi,psi FILE=colvar.dat STRIDE=10
```

from the colvar file it is clear that we can quickly explore two minima but that the region for positive phi is not accessible. Instead of using the opposite of the free energy as a table potential here we introduce the function MATHEVAL that allows defining complex analytical functions of collective variables, and the bias BIASVALUE that allows using any continuous function of the positions as a bias.
So first we need to fit the opposite of the free energy as a function of phi in the region explored with a periodic function, because of the gaussian like look of the minima we can fit it using the von Mises distribution. In gnuplot

```
>gnuplot
gnuplot>plot 'ffphi.dat' u 1:(-$2+31) w l
gnuplot>f(x)=exp(k1*cos(x-a1))+exp(k2*cos(x-a2))
gnuplot>fit [-3.:-0.6] f(x) 'ffphi.dat' u 1:(-$2+31) via k1,a1,k2,a2
gnuplot>rep f(x)
```

The function and the resulting parameters can be used to run a new biased simulation:

11.38.3.11 Exercise 7: First biased run with Alanine dipeptide

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
#SETTINGS MOLFILE=user-doc/tutorials/trieste-3/aladip/aladip.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=aladip.pdb

phi: TORSION ATOMS=@phi-2
__FILL__

MATHEVAL ...
ARG=phi
LABEL=doubleg
FUNC=exp(__FILL)+__FILL__
PERIODIC=NO
... MATHEVAL

b: BIASVALUE ARG=__FILL__
PRINT __FILL__
```

It is now possible to run a second simulation and observe the new behavior. The system quickly explores a new minimum. While a quantitative estimate of the free energy difference of the old and new regions is out of the scope of the current exercise what we can do is to add a new von Mises function centered in the new minimum with a comparable height, in this way we can hope to facilitate a back and forth transition along the phi collective variable.

```
>gnuplot
gnuplot> ...
```

We can now run a third simulation where both regions are biased.

11.38.3.12 Exercise 8: Second biased run with Alanine dipeptide

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
#SETTINGS MOLFILE=user-doc/tutorials/trieste-3/aladip/aladip.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=aladip.pdb

phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2


MATHEVAL ...
ARG=phi
LABEL=tripleg
FUNC=exp(k1*cos(x-a1))+exp(k2*cos(x-a2))+exp(k3*cos(x+a3))
PERIODIC=NO
... MATHEVAL

b: BIASVALUE ARG=tripleg

__FILL__

ENDPLUMED
```

With this third simulation it should be possible to visit both regions as a function on the phi torsion. Now it is possible to reweight the sampling and obtain a better free energy estimate along phi.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-3.txt
#SETTINGS MOLFILE=user-doc/tutorials/trieste-3/aladip/aladip.pdb
# vim:ft=plumed
MOLINFO STRUCTURE=aladip.pdb

phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2

MATHEVAL ...
ARG=phi
LABEL=tripleg
FUNC=__FILL__
PERIODIC=NO
... MATHEVAL

b: BIASVALUE ARG=tripleg
as: REWEIGHT_BIAS TEMP=298
hhphi: HISTOGRAM ARG=phi STRIDE=10 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05 LOGWEIGHTS=as
hhpsi: HISTOGRAM ARG=psi STRIDE=10 GRID_MIN=-pi GRID_MAX=pi GRID_BIN=600 BANDWIDTH=0.05 LOGWEIGHTS=as
ffphi: CONVERT_TO_FES GRID=hhphi TEMP=298
ffpsi: CONVERT_TO_FES GRID=hhpsi TEMP=298

DUMPGRID GRID=ffphi FILE=ffphi.dat
DUMPGRID GRID=ffpsi FILE=ffpsi.dat

PRINT ARG=phi,psi FILE=colvar.dat STRIDE=10
```

If you have time you can extend this in two-dimensions using at the same time the phi and psi collective variables.

11.38.4 Trieste tutorial: Metadynamics simulations with PLUMED

11.38.4.1 Aims

The aim of this tutorial is to train users to perform metadynamics simulations with PLUMED, analyze the results, calculating free-energies as a function of the collective variables used, and estimating the associated error.

11.38.4.2 Objectives

Once this tutorial is completed students will be able to:

- Write the PLUMED input file to perform metadynamics simulations
- Calculate the free energy from a metadynamics run
- Compute the error associated to the reconstructed free energy
- Evaluate the convergence of a metadynamics simulation
- Assess the choice of the collective variables

11.38.4.3 Resources

The TARBALL for this project contains the following files:

- diala.pdb: a PDB file for alanine dipeptide in vacuo
- topol.tpr: a GROMACS run file to perform MD of alanine dipeptide
- do_block_fes.py: a python script to perform error analysis

This tutorial has been tested on a pre-release version of version 2.4. However, it should not take advantage of 2.4-only features, thus should also work with version 2.3.

Note

We suggest to run the three exercises in three separate directories. For Exercise 3, you will need the output of the first two exercises, so don't delete it!

11.38.4.4 Introduction

We have seen that PLUMED can be used to compute collective variables. However, PLUMED is most often use to add forces on the collective variables. To this aim, we have implemented a variety of possible biases acting on collective variables. The complete documentation for all the biasing methods available in PLUMED can be found at the Bias page. In the following we will see how to build an adaptive bias potential with metadynamics. Here you can find a brief recap of the metadynamics theory.
In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t\to\infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t\to\infty) = -\frac{\Delta T}{T+\Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard MD, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T+\Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135].
We will play with a toy system, alanine dipeptide simulated in vacuo using the AMBER99SB-ILDN force field (see Fig. trieste-4-ala-fig). This rather simple molecule is useful to benchmark data analysis and free-energy methods. This system is a nice example because it presents two metastable states separated by a high free-energy barrier. It is conventional use to characterize the two states in terms of Ramachandran dihedral angles, which are denoted with Φ and Ψ in Fig. trieste-4-transition-fig .

11.38.4.5 Exercise 1: my first metadynamics calculation

11.38.4.5.1 Exercise 1a: setup and run In this exercise we will setup and perform a well-tempered metadynamics run using the backbone dihedral ϕ as collective variable. During the calculation, we will also monitor the behavior of the other backbone dihedral ψ.
Here you can find a sample `plumed.dat` file that you can use as a template. Whenever you see an highlightedF↩ ILL string, this is a string that you should replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-4.txt
# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
psi: TORSION ATOMS=__FILL__

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Deposit a Gaussian every 500 time steps, with initial height equal to 1.2 kJ/mol
PACE=500 HEIGHT=1.2
# the bias factor should be wisely chosen
BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
SIGMA=__FILL__
# Gaussians will be written to file and also stored on grid
FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
...

# Print both collective variables and the value of the bias potential on COLVAR file
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=10
```

The syntax for the command METAD is simple. The directive is followed by a keyword ARG followed by the labels of the CVs on which the metadynamics potential will act. The keyword PACE determines the stride of Gaussian deposition in number of time steps, while the keyword HEIGHT specifies the height of the Gaussian in kJ/mol. For each CVs, one has to specify the width of the Gaussian by using the keyword SIGMA. Gaussian will be written to the file indicated by the keyword FILE.
In this example, the bias potential will be stored on a grid, whose boundaries are specified by the keywords GRID_↩ MIN and GRID_MAX. Notice that you can provide either the number of bins for every collective variable (GRID_BIN) or the desired grid spacing (GRID_SPACING). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither GRID_BIN nor GRID_SPACING) and if Gaussian width is fixed, PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.
Once your `plumed.dat` file is complete, you can run a 10-ns long metadynamics simulations with the following command

```
> gmx mdrun -s topol.tpr -nsteps 5000000 -plumed plumed.dat
```

During the metadynamics simulation, PLUMED will create two files, named COLVAR and HILLS. The COLVAR file contains all the information specified by the PRINT command, in this case the value of the CVs every 10 steps of simulation, along with the current value of the metadynamics bias potential. We can use `gnuplot` to visualize the behavior of the CV during the simulation, as reported in the COLVAR file:

```
gnuplot> p "COLVAR" u 1:2
```

By inspecting Figure trieste-4-phi-fig, we can see that the system is initialized in one of the two metastable states of alanine dipeptide. After a while (t=0.1 ns), the system is pushed by the metadynamics bias potential to visit the other local minimum. As the simulation continues, the bias potential fills the underlying free-energy landscape, and the system is able to diffuse in the entire phase space.
The HILLS file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content:

```
#! FIELDS time phi sigma_phi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
```

The line starting with FIELDS tells us what is displayed in the various columns of the HILLS file: the simulation time, the instantaneous value of ϕ, the Gaussian width and height, and the bias factor. We can use the HILLS file to visualize the decrease of the Gaussian height during the simulation, according to the well-tempered recipe:
If we look carefully at the scale of the y-axis, we will notice that in the beginning the value of the Gaussian height is higher than the initial height specified in the input file, which should be 1.2 kJ/mol. In fact, this column reports the height of the Gaussian scaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy.

11.38.4.5.2 Exercise 1b: estimating the free energy One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills should be

used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.
To calculate the free energy as a function of ϕ, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called `fes.dat` in which the free-energy surface as function of ϕ is calculated on a regular grid. One can modify the default name for the free energy file, as well as the boundaries and bin size of the grid, by using the following options of sum_hills :

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

The result should look like this:
To assess the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The option --stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option --mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --stride 100 --mintozero
```

one free energy is calculated every 100 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. The resulting plot should look like the following:
These two qualitative observations:

- the system is diffusing efficiently in the collective variable space (Figure trieste-4-phi-fig)
- the estimated free energy does not change significantly as a function of time (Figure trieste-4-metad-phifest-fig)

suggest that the simulation most likely converged.

Warning

> The fact that the Gaussian height is decreasing to zero should not be used as a measure of convergence of your metadynamics simulation!

Note

> The two observations above are necessary, but qualitative conditions for convergence. A quantitative assessment of convergence can be obtained by performing an error analysis of the reconstructed free-energy profile, as explained in the last exercise

11.38.4.6 Exercise 2: playing with collective variables

In this exercise, we will run a well-tempered metadynamics simulation on alanine dipeptide in vacuum, this time using as CV the backbone dihedral ψ. Please complete the template `plumed.dat` file used in the previous exercise to run this calculation.
Once your `plumed.dat` file is complete, you can run a 10-ns long metadynamics simulations with the following command

```
> gmx mdrun -s topol.tpr -nsteps 5000000 -plumed plumed.dat
```

As we did in the previous exercise, we can use COLVAR to visualize the behavior of the CV during the simulation. Here we will plot at the same time the evolution of the metadynamics CV ψ and of the other dihedral ϕ.

```
gnuplot> p "COLVAR" u 1:2, "" u 1:3
```

By inspecting Figure trieste-4-metad-psi-phi-fig, we notice that something different happened compared to the previous exercise. At first the behavior of ψ looks diffusive in the entire CV space. However, around t=1 ns, ψ seems trapped in a region of the CV space in which it was previously diffusing without problems. The reason is that the non-biased CV ϕ after a while has jumped into a different local minima. Since ϕ is not directly biased, one has to wait for this (slow) degree of freedom to equilibrate before the free energy along ψ can converge.
Try to repeat the analysis done in the previous exercise, i.e. calculate the estimate of the free energy as a function of time, first step to assess the convergence of this metadynamics simulation.

11.38.4.7 Exercise 3: estimating the error in free-energies using block-analysis

In this exercise, we will calculate the error associated to the free-energy reconstructed by a well-tempered metadynamics simulation. The free energy and the errors will be calculated using the block-analysis technique explained in a previous lesson (Trieste tutorial: Averaging, histograms and block analysis). The procedure can be used to estimate the error in the free-energy as a function of the collective variable(s) used in the metadynamics simulation, or for any other function of the coordinates of the system.
First, we will calculate the "un-biasing" weights associated to each conformation sampled during the metadynamics run. In order to calculate these weights, we can use either of these two approaches:
1) Weights are calculated by considering the time-dependence of the metadynamics bias potential [3];
2) Weights are calculated using the metadynamics bias potential obtained at the end of the simulation and assuming a constant bias during the entire course of the simulation [50].
In this exercise we will use the umbrella-sampling-like reweighting approach (Method 2).
To calculate the weights, we need to use the PLUMED driver utility and read the HILLS file along with the GR↩ OMACS trajectory file produced during the metadynamics simulation. Let's consider the metadynamics simulation carried out in Exercise 1. We need to prepare the `plumed.dat` input file to use in combination with driver. Here you can find a sample `plumed.dat` file that you can use as a template. Whenever you see an highlightedFILL string, this is a string that you should replace.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-4.txt
# Read old Gaussians deposited on HILLS file
RESTART
# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
psi: TORSION ATOMS=__FILL__

# Activate well-tempered metadynamics in phi
metad: __FILL__ ARG=__FILL__ ...
# Set the deposition stride to a large number
PACE=10000000 HEIGHT=1.2 BIASFACTOR=__FILL__
# Gaussian width (sigma) should be chosen based on CV fluctuation in unbiased run
SIGMA=__FILL__
# Gaussians will be read from file and stored on grid
FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
...

# Print both collective variables and the value of the bias potential on COLVAR file
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=1
```

Once your `plumed.dat` file is complete, you can use the driver utility to back-calculated the quantities needed for the error calculation

```
plumed driver --plumed plumed.dat --mf_xtc traj_comp.xtc
```

The COLVAR file produced by driver should look like this:

```
#! FIELDS time phi psi metad.bias
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
 0.000000 0.907347 -0.144312 103.117323
 1.000000 0.814296 -0.445819 100.974351
 2.000000 1.118951 -0.909782 104.329630
 3.000000 1.040781 -0.991788 104.559590
 4.000000 1.218571 -1.020024 102.744053
```

Please check your `plumed.dat` file if your output looks different! Once the final bias has been evaluated on the entire metadynamics simulations, we can easily calculate the "un-biasing weights" using the umbrella-sampling-like approach:

```
# find maximum value of bias
bmax=`awk 'BEGIN{max=0.}{if($1!="#!" && $4>max)max=$4}END{print max}' COLVAR`

# print phi values and weights
awk '{if($1!="#!") print $2,exp(($4-bmax)/kbt)}' kbt=2.494339 bmax=$bmax COLVAR > phi.weight
```

If you inspect the `phi.weight` file, you will see that each line contains the value of the dihedral ϕ along with the corresponding weight:

```
0.907347 0.0400579
0.814296 0.0169656
1.118951 0.0651276
1.040781 0.0714174
1.218571 0.0344903
1.090823 0.0700568
1.130800 0.0622998
```

At this point we can apply the block-analysis technique we have learned in the Trieste tutorial: Averaging, histograms and block analysis tutorial to calculate for different block sizes the average free-energy and the error. For your convenience, you can use the `do_block_fes.py` python script to read the `phi.weight` file and produce the desired output. We use a bash loop to use block sizes ranging from 1 to 1000:

```
for i in `seq 1 10 1000`; do python3 do_block_fes.py phi.weight 1 -3.141593 3.018393 51 2.494339 $i; done
```

For each value of block length `N`, you will obtain a separate `fes.N.dat` file, containing the value of the ϕ variable on a grid, the average free-energy, and the associated error (in kJ/mol):

```
  -3.141593       23.184653     0.080659
  -3.018393       17.264462     0.055181
  -2.895194       13.360259     0.047751
  -2.771994       10.772696     0.043548
  -2.648794        9.403544     0.042022
```

Finally, we can calculate the average error along the free-energy profile as a function of the block length:

```
for i in `seq 1 10 1000`; do a=`awk '{tot+=$3}END{print tot/NR}' fes.$i.dat`; echo $i $a; done > err.blocks
```

and visualize it using `gnuplot`:

```
gnuplot> p "err.blocks" u 1:2 w lp
```

As expected, the error increases with the block length until it reaches a plateau in correspondence of a dimension of the block that exceeds the correlation between data points (Fig. trieste-4-block-phi).
To complete this exercise, you should do the following:

- calculate the error associated to the free energy as a function of the collective variable ψ from Exercise 1
- calculate the error associated to the free energy as a function of the collective variable ψ from Exercise 2
- compare the different behaviors in Exercise 1 and 2

What can we learn from this analysis about the convergence of the two metadynamics simulations and the quality of the collective variables chosen?
At this time, the most important question of this lecture becomes:

- Could we distinguish the different behavior (in terms of convergence) of the simulations in Exercise 1 and 2 simply by looking at the time series of the Gaussian height?

11.38.4.8 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Setup and run a metadynamics calculation.
- Compute free energies from the metadynamics bias potential using the sum_hills utility.
- Calculate the error in the reconstructed free energy using block analysis.
- Discriminate between good and bad collective variables.
- Evaluate the convergence of a metadynamics simulation.

11.38.5 Trieste tutorial: Running and analyzing multi-replica simulations.

11.38.5.1 Aims

The aim of this tutorial is to show how to use PLUMED to run simulations with multiple replicas and how to analyze them. In particular, we will focus on cases where the replicas feel different biasing potentials.

11.38.5.2 Objectives

Once this tutorial is completed students will be able to:

- Write plumed input files suitable for multi-replica simulations.
- Run replica exchange simulations with gromacs and plumed using different bias potentials in each replica.
- Analyze replica exchange simulations using WHAM so as to obtain the weight of each snapshot.

11.38.5.3 Resources

The TARBALL for this project contains the following files:

- `topol0.tpr`, `topol1.tpr`, `topol2.tpr`, and `topol3.tpr`, gromacs input files for analine dipeptide. Notice that two of them (0 and 2) are initialized in one free-energy well and two of them (1 and 3) in the other free-energy well.
- A directory SCRIPT that contains a `wham.py` script to perform WHAM. Notice that this required python 3 (does not work with python 2).
- A directory SETUP with the gromacs input file that can be used to generate new tpr files.

This tutorial has been tested on a pre-release version of version 2.4. In particular, it takes advantage of a special syntax for setting up multi-replica simulations that is only available since version 2.4. Exercise could be done also with version 2.3 but using a different syntax with respect to the one suggested.
Also notice that in the `.solutions` directory of the tarball you will find correct input files. Please only look at these files after you have tried to solve the problems yourself.

11.38.5.4 Introduction

So far we always used PLUMED to run one simulation at a time. However, PLUMED can also be used in multi-replica algorithms. When coupled with GROMACS (at least) it is also possible to run replica exchange simulations, where coordinates are swapped from time to time. In this case, PLUMED is going to take into account the different bias potentials applied to different replicas so as to compute correctly the acceptance.
Similarly to what we did before, we will first use the driver to understand how to prepare multi-replica input files. However, the very same holds when you run multi-replica MD simulations with MD codes that support them. For instance, in GROMACS that would be using the `-multi` option of `mdrun`.
Notice that this tutorial was tested using a pre-release version of PLUMED 2.4. In particular, we will use a special syntax for multi-replica input that is only available starting with PLUMED 2.4.

11.38.5.5 Multi replica input files

Imagine that you are in a directory with these files

```
traj.0.xtc
traj.1.xtc
traj.2.xtc
plumed.dat
```

That is: three trajectories and a PLUMED input files. Let's say that the content of `plumed.dat` is

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-5.txt
d: DISTANCE ATOMS=1,2
PRINT ARG=d FILE=COLVAR
```

You can use the following command to process the three trajectories simultaneously:

```
mpirun -np 3 plumed driver --multi 3 --plumed plumed.dat --mf_xtc traj.xtc
```

This command will produce three COLVAR files, namely `COLVAR.0`, `COLVAR.1`, and `COLVAR.2`.
How does this work?
You are here running three replicas of the `plumed` executable. When PLUMED opens a file for **writing** it adds a a suffix corresponding to the replica number. This is done for all the possible files written by PLUMED and allows you to distinguish the output of different replicas redirecting it to different files.

Attention

This is true also for input files that are opened for **reading**. However, when PLUMED does not find the input file with the replica suffix, it looks for the file without the suffix. That's why here it is sufficient to provide a single `plumed.dat` file. If by mistake you include an additional `plumed.0.dat` file in the same directory, PLUMED will use that file for replica 0 (and `plumed.dat` for replicas 1 and 2). To be precise, this is true for all the files read by PLUMED, with the exception of PDB files where the suffix is never added.

The last comment implies that if you need to process your trajectories with *different* input files you might do it creating files named `plumed.0.dat`, `plumed.1.dat`, and `plumed.2.dat`. This is correct, and this was the only way to do multi-replica simulations with different input files up to PLUMED 2.3. However, in the following we will see how to obtain the same effect with a new special command that has been introduced in PLUMED 2.4.

11.38.5.6 Using special syntax for multiple replicas

In many cases, we need to run multiple replicas with almost identical PLUMED files. These files might be prepared with cut-and-paste, which is very error prone, or could be set up with some smart bash or python script. Additionally, one can take advantage of the INCLUDE keyword so as to have a shared input file with common definitions and specific input files with replica-dependent keywords. However, as of PLUMED 2.4, we introduced a simpler manner to manipulate multiple replica inputs with tiny differences. Look at the following example:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-5.txt
#SETTINGS NREPLICAS=3
# Compute a distance
d: DISTANCE ATOMS=1,2

# Apply a restraint.
RESTRAINT ARG=d AT=@replicas:1.0,1.1,1.2 KAPPA=1.0
# On replica 0, this means:
#   RESTRAINT ARG=d AT=1.0 KAPPA=1.0
# On replica 1, this means:
#   RESTRAINT ARG=d AT=1.1 KAPPA=1.0
# On replica 2, this means:
#   RESTRAINT ARG=d AT=1.2 KAPPA=1.0
```

If you prepare a single `plumed.dat` file like this one and feeds it to PLUMED while using 3 replicas, the 3 replicas will see the very same input except for the `AT` keyword, that sets the position of the restraint. Replica 0 will see a restraint centered at 1.0, replica 1 centered at 1.1, and replica 2 centered at 1.2.
The `@replicas:` keyword is not special for RESTRAINT or for the `AT` keyword. Any keyword in PLUMED can accept that syntax. For instance, the following single input file can be used to setup a bias exchange metadynamics [132] simulations:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-5.txt
#SETTINGS NREPLICAS=2
# Compute distance between atoms 1 and 2
d: DISTANCE ATOMS=1,2

# Compute a torsional angle
t: TORSION ATOMS=30,31,32,33

# Metadynamics.
METAD ...
  ARG=@replicas:d,t
  HEIGHT=1.0
  PACE=100
  SIGMA=@replicas:0.1,0.3
  GRID_MIN=@replicas:0.0,-pi
  GRID_MAX=@replicas:2.0,pi
...
# On replica 0, this means:
#  METAD ARG=d HEIGHT=1.0 PACE=100 SIGMA=0.1 GRID_MIN=0.0 GRID_MAX=2.0
# On replica 1, this means:
#  METAD ARG=t HEIGHT=1.0 PACE=100 SIGMA=0.3 GRID_MIN=-pi GRID_MAX=pi
```

This would be a typical setup for a bias exchange simulation. Notice that even though variables `d` and `t` are both read in both replicas, `d` is only computed on replica 0 (and `t` is only computed on replica 1). This is because variables that are defined but not used are never actually calculated by PLUMED.
If the value that should be provided for each replica is a vector, you should use curly braces as delimiters. For instance, if the restraint acts on two variables, you can use the following input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-5.txt
#SETTINGS NREPLICAS=3
# Compute distance between atoms 1 and 2
d: DISTANCE ATOMS=10,20

# Compute a torsional angle
t: TORSION ATOMS=30,31,32,33

# Apply a restraint:
RESTRAINT ...
  ARG=d,t
  AT=@replicas:{{1.0,2.0} {3.0,4.0} {5.0,6.0}}
  KAPPA=1.0,3.0
...
# On replica 0 this means:
#  RESTRAINT ARG=d AT=1.0,2.0 KAPPA=1.0,3.0
# On replica 1 this means:
#  RESTRAINT ARG=d AT=3.0,4.0 KAPPA=1.0,3.0
# On replica 2 this means:
#  RESTRAINT ARG=d AT=5.0,6.0 KAPPA=1.0,3.0
```

Notice the double curly braces. The outer ones are used by PLUMED to know there the argument of the `AT` keyword ends, whereas the inner ones are used to group the values corresponding to each replica. Also notice that the last example can be split in multiple lines exploiting the fact that within multi-line statements (enclosed by pairs of `...`) newlines are replaced with simple spaces:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-5.txt
#SETTINGS NREPLICAS=3
d: DISTANCE ATOMS=10,20
t: TORSION ATOMS=30,31,32,33
RESTRAINT ...
  ARG=d,t
# indentation is not required (this is not python!)
# but makes the input easier to read
  AT=@replicas:{
    {1.0,2.0}
    {3.0,4.0}
    {5.0,6.0}
  }
  KAPPA=1.0,1.0
...
```

In short, whenever there are keywords that should vary across replicas, you should set them using the `@replicas:` keyword. As mentioned above, you can always use the old syntax with separate input file, and this is recommended when the number of keywords that are different is large.

11.38.5.7 Exercise 1: Running multi-replica simulations

Write a plumed file that allows you to run a multi-replica simulation of alanine dipeptide where the following four replicas are simulated:

- Two replicas with no bias potential.
- A replica with metadynamics on ϕ.
- A replica with metadynamics on ψ.

With PLUMED 2.3 you should use four `plumed.dat` files (named `plumed.0.dat`, `plumed.1.dat`, etc). In this case, the first two replicas feel no potential, so you could just use empty files. For the third and fourth replica you could recycle the files from Trieste tutorial: Metadynamics simulations with PLUMED.
However, we recommend you to take the occasion to make practice with the special replica syntax of PLUMED 2.4. Use the following input file as a starting point

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-5.txt
# load the pdb to better select torsions later
MOLINFO __FILL__
# Compute the backbone dihedral angle phi, defined by atoms C-N-CA-C
phi: TORSION ATOMS=__FILL__
# Compute the backbone dihedral angle psi, defined by atoms N-CA-C-N
psi: TORSION ATOMS=__FILL__
```

```
metad: METAD ...
# Activate well-tempered metadynamics
# Deposit a Gaussian every 500 time steps, with initial height equal
# to 1.2 kJoule/mol, biasfactor equal to 10.0
# Remember: replica 0 and 1: no bias
# replica 2, bias on phi
# replica 3, bias on psi
  ARG=__FILL__
  HEIGHT=__FILL__ # make sure that replicas 0 and 1 feel no potential!
  PACE=500
  BIASFACTOR=10.0
  SIGMA=0.35
# Gaussians will be written to file and also stored on grid
  FILE=HILLS GRID_MIN=-pi GRID_MAX=pi
...

# Print both collective variables and the value of the bias potential on COLVAR file
PRINT ARG=__FILL__ FILE=COLVAR STRIDE=10
```

Now check the resources provided with this tutorial. There are 4 `tpr` files available (`topol0.tpr`, `topol1.tpr`, `topol2.tpr`, and `topol3.tpr`). You can run a replica exchange simulation with gromacs using the following command

```
> mpirun -np 4 gmx_mpi mdrun -plumed plumed.dat -nsteps 1000000 -replex 120 -multi 4
```

Notice that coordinates swaps will be attempted between different replicas every 120 steps. When computing the acceptance, gromacs will take into account that different replicas might be subject to different bias potential. In this example, replicas 0 and 1 are actually subject to the same potential, so all the exchanges will be accepted. You can verify this using the command

```
> grep "Repl ex" md0.log
```

It is interesting to see what happens at different stages of the simulation. At the beginning of the trajectory you should see something like this

```
> grep "Repl ex" md0.log | head
Repl ex  0 x  1    2 x  3
Repl ex  0    1 x  2    3
Repl ex  0 x  1    2 x  3
Repl ex  0    1 x  2    3
Repl ex  0 x  1    2 x  3
Repl ex  0    1    2    3
Repl ex  0 x  1    2 x  3
Repl ex  0    1 x  2    3
Repl ex  0 x  1    2 x  3
Repl ex  0    1 x  2    3
```

Notice that all the exchanges between replicas 0 and 1 are accepted and also almost all the exchanges between replicas 1, 2, and 3. At a later stage you will more likely find something like this

```
> grep "Repl ex" md0.log | tail
Repl ex  0    1 x  2    3
Repl ex  0 x  1    2    3
Repl ex  0    1 x  2    3
Repl ex  0 x  1    2    3
Repl ex  0    1 x  2    3
Repl ex  0 x  1    2 x  3
Repl ex  0    1 x  2    3
Repl ex  0 x  1    2 x  3
Repl ex  0    1    2    3
Repl ex  0 x  1    2    3
```

Notice that all the exchanges between replicas 0 and 1 are still accepted. This is because the potential energy felt by replicas 0 and 1 are identical. However, exchanges between replicas 1, 2, and 3 are sometime rejected.

Note

The setup above is very close to the one used in bias exchange simulations. However, in bias exchange simulations usually one would use at most one neutral (non-biased) replica. In addition, the RANDOM_EXCHANGES command is often used.

11.38.5.8 Exercise 2: Analyzing a multiple-restraint simulation

In the following we will analyze the result of the simulation above. To this aim we will have to use the WHAM method. The WHAM procedure described here is binless and allows one to reweight arbitrary bias potentials without the need to discretize collective variables. As a first step it is necessary to create a concatenated trajectory that includes all the conformations that you produced during your simulations.

```
> gmx_mpi trjcat -f traj_comp*.xtc -cat -o fulltraj.xtc
```

Notice that a new trajectory file `fulltraj.xtc` will be in your working directory now, approximately four times bigger than the individual files `traj_comp*.xtc`.
Now we will process that directory computing the bias potentials associated with the replicas that we simulated. Similarly to what we did in Trieste tutorial: Metadynamics simulations with PLUMED, we will assume that reweighting is done with final bias potential [50]. Notice that also here it would be possible to use [3] instead.

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-5.txt
# this will be file plumed_reweight.dat
# include here exactly the same file you used to run the simulation
# however, you should make some change to METAD:
# - replace PACE with a huge number
# - add TEMP=300 to set the temperature of the simulation
# (normally this is inherited from the MD code, but driver has no idea about it)
# - add RESTART=YES to trigger restart
__FILL__
__FILL__
__FILL__
__FILL__
# you should also change the PRINT command so as not to overwrite
# the files you wrote before.
# here omit STRIDE, so that you will print
# for all the frames in your concatenated trajectory
PRINT ARG=phi,psi FILE=COLVAR_CONCAT

# Then write the bias potential (as we did for reweighting)
# As a good practice, remember that there might be multiple bias
# potential applied (e.g. METAD and a RESTRAINT).
# to be sure that you include all of them, just combine them:
bias: COMBINE ARG=*.bias PERIODIC=NO
# here omit STRIDE, so that you will print the bias
# for all the frames in your concatenated trajectory
PRINT FILE=COLVAR_WHAM ARG=bias
```

Then analyze the concatenated trajectory with this command

```
> mpirun -np 4 plumed driver --mf_xtc fulltraj.xtc --plumed plumed_reweight.dat --multi 4
```

Notice that since `fulltraj.xtc` the four concurrent copies of PLUMED will all analyze the same trajectory, which is what we want to do now. The result will be a number of COLVAR_WHAM files, one per replica, with a number of lines corresponding to the whole concatenated trajectory.

```
# put the bias columns in a single file ALLBIAS
> paste COLVAR_WHAM.* | grep -v \# | awk '{for(i=1;i<=NF/2;i++) printf($(i*2)"  ");printf("\n");}' > ALLBIAS
```

Have a look at `ALLBIAS`. It should look more or less like this

```
0.000000  0.000000  75.453800  68.172206
0.000000  0.000000  74.973726  68.143015
0.000000  0.000000  66.948910  56.295899
0.000000  0.000000  70.383309  60.164381
0.000000  0.000000  65.595489  61.754951
0.000000  0.000000  67.689166  60.869430
```

The first two columns correspond to the bias felt in replicas 0 and 1, that are unbiased, and thus should be zero. In the other replicas on the other hand we will have large potentials accumulated by metadynamics. The actual numbers might very depending on the length of your simulation.
Now you can use the provided python script to analyze the `ALLBIAS` file. `SCRIPTS/wham.py` should be provided three arguments: name of the file with bias potentials, number of replicas (that is the number of columns) and value of k_BT (2.5 in kj/mol).

```
# use the python script to compute weights
> python3 SCRIPTS/wham.py ALLBIAS 4 2.5
```

This will generate a file named `weights.dat` that contains the weights for each of the frame. The initial values reported in the file should be similar to these ones:

```
> head weights.dat
 3.688530731178e-05
3.899877363004e-05
7.861996862568e-07
3.663355487180e-06
1.647251040152e-06
2.689614526429e-06
3.254124396415e-06
1.511057459698e-05
2.342149322226e-05
5.032261109943e-06
```

Weights are very small since we have a long trajectory with many frames. You can check that they are normalized using the command 'awk '{a+=$1}END{print a}' weights.dat`.
Now have a look at the weights. Remember that we are concatenating trajectories.
Can you understand why weights in the first half are systematically different from weights in the second half? The first part correspond to the unbiased trajectories. All the snapshots sampled here are reasonably correct. However, in the second part we have heavily biased snapshots, which also sample transition states. Those points are assigned a low weight by WHAM so that they would be discarded in the calculation of any unbiased average.
Now we have the weight for each frame. We would like to compute the relative stability of the two free-energy wells that correspond roughly to positive and negative ϕ.

```
> paste <(grep -v \# COLVAR_CONCAT.0) weights.dat | awk '{if($2>0)wb+=$NF; else wa+=$NF}END{print wa,wb}'
```

This will tell you the unbiased population of the two minima. You should expect values close to these ones

```
0.968885 0.0311148
```

that tells you that the first minimum has a population of roughly 97%.

Note

This approach can be used to reweight any replica exchange simulation, irrespective of how different were the employed bias potentials. This includes bias-exchange metadynamics [132] when using well-tempered metadynamics [49]. Notice that for non-well-tempered bias exchange metadynamics one can use a very similar approach based on binning that is described in [141]. In addition, the approach illustrated here can be used directly in bias-exchange metadynamics simulations with additional restraints which are different in different replicas (as in [137]) as well as other flavors of biased replica exchange simulations (e.g. [142] [53]). With some modification to take into account the force field energy it can be used to analyze parallel tempering simulations [143] and simulated tempering simulations [139] (notice that they can be implemented with PLUMED + GROMACS using [138]).

11.38.5.9 Exercise 3: What if a variable is missing?

Repeat the exercise above (that is: running replica exchange MD simulation and analyze the result) but using only three replicas:

- two unbiased replicas.
- one biased replica along psi.

Create a file `plumed3.dat` aimed at this. If you are not able you can find a working one in the `solutions` directory. Then use the following commands similarly to before:

```
> mpirun -np 3 gmx_mpi mdrun -plumed plumed3.dat -nsteps 1000000 -replex 120 -multi 3
> gmx_mpi trjcat -f traj_comp{0,1,2}.xtc -cat -o fulltraj.xtc
> mpirun -np 3 plumed driver --mf_xtc fulltraj.xtc --plumed plumed3_reweight.dat --multi 3
> paste COLVAR_WHAM.{0,1,2} | grep -v \# | awk '{for(i=1;i<=NF/2;i++) printf($(i*2)"  ");printf("\n");}' > ALL
> python3 SCRIPTS/wham.py ALLBIAS 3 2.5
```

Compute as before the relative population of the two minima.

```
> paste <(grep -v \# COLVAR_CONCAT.0) weights.dat | awk '{if($2>0)wb+=$NF; else wa+=$NF}END{print wa,wb}'
```

Which is the result? What can you learn from this example?
If you did things correctly, here you should find a population close to 2/3 for the first minimum and 1/3 for the second one. Notice that this population just reflects the way we initialized the simulations (2 of them in one minimum and

1 in the other minimum) and does not take into account which is the relative stability of the two minima according to the real potential energy function. In other words, we are just obtaining the relative population that we used to initialize the simulation.
Also plot the colvar files and look at them with attention. An excerpt is shown below.
always due to accepted exchanges between replicas."
How many transitions between the two free-energy wells can you observe? Remember that replica exchange involves coordinate swaps that do not correspond to the real system dynamics. It is very useful to look at "demuxed" trajectories.

11.38.5.10 Exercise 4: "demuxing" your trajectories

Use the following commands

```
> demux.pl md0.log
```

This will produce two files: `replica_index.xvg` and `replica_temp.xvg`. The former allows to convert your trajectories to continuous ones with the following commands

```
> demux.pl md0.log
> gmx_mpi  trjcat -f traj_comp?.xtc -demux replica_index.xvg
```

You will now find new trajectories `0_trajout.xtc`, `1_trajout.xtc`, `2_trajout.xtc`, and `3_↩ trajout.xtc`. These files can be in turn analyzed with `plumed driver`. Try to recompute collective variables on the trajectories obtained in Exercise 1: Running multi-replica simulations and with those obtained in Exercise 3: What if a variable is missing?.
As you can see, the trajectories from Exercise 1: Running multi-replica simulations show a number of transitions. On the other hand, the trajectories from Exercise 3: What if a variable is missing? are stuck.
As a general rule, notice that there is no way to estimate correctly the relative population of two metastable states if there is not a continuous "demuxed" trajectory joining them, possibly exhibiting multiple transitions.

11.38.5.11 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Run multi replica simulations.
- Analyze them using WHAM.

Notice that the setup used is similar to the ones typically used in bias exchange simulations but can be easily adapted to any case where multiple replicas should be combined that feel different bias potentials. Moreover, the problematic example discussed shows that one should be careful and check if real transitions are observed during a replica exchange simulation.

11.38.6 Trieste tutorial: Real-life applications with complex CVs

11.38.6.1 Aims

The aim of this tutorial is to train users to learn the syntax of complex collective variables and use them to analyze MD trajectories of realistic biological systems and bias them with metadynamics.

11.38.6.2 Objectives

Once this tutorial is completed students will be able to:

- Write the PLUMED input file to use complex CVs for analysis
- Analyze trajectories and calculate the free energy of complex biological systems
- Perform error analysis and evaluate convergence in realistic situations
- Setup, run, and analyze metadynamics simulations of a complex system

11.38.6.3 Resources

The reference trajectories and input files for the exercises proposed in this tutorial can be downloaded from `github` using the following command:

```
wget https://github.com/plumed/trieste2017/raw/master/tutorial/data/tutorial_6.tgz
```

This tutorial has been tested on a pre-release version of version 2.4. However, it should not take advantage of 2.4-only features, thus should also work with version 2.3.

11.38.6.4 Introduction

In this tutorial we propose exercises on the following biological systems:

- the BRCA1-associated RING domain protein 1 (BARD1 complex)
- the cmyc peptide in presence of urea at low concentration (cmyc-urea)
- the protein G B1 domain

The exercise are of increasing difficulties, inputs are partially provided for the first and second cases while for the last one the user is expected to be autonomous.

11.38.6.5 Exercise 1: analysis of the BARD1 complex simulation

The BARD1 complex is a hetero-dimer composed by two domains of 112 and 117 residues each. The system is represented at coarse-grained level using the MARTINI force field.
In the TARBALL of this exercise, we provide a long MD simulation of the BARD1 complex in which the two domains explore multiple different conformations.

Note

We encourage the users to get familiar with the system by visualizing the MD trajectory using `VMD`.

The users are expected to:

- calculate the values of different CVs on the trajectory
- estimate the free energies as a function of the CVs tested (mono- and multi-dimensional)
- extracting from the trajectories the configurations corresponding to relevant free-energy minima
- calculate the error in the associated free energy using the block analysis technique
- evaluate the convergence of the original trajectory

The users are free to choose his/her favorite CVs and they are encouraged to use the on-line manual to create their own PLUMED input file. However, we encourage all the users to experiment at least with the following CVs:
1) RMSD and/or DRMSD from the native state
The following line can be added to the `plumed.dat` file to calculate the RMSD on the beads representing the backbone amino acids:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-trieste-6.txt
rmsd: RMSD REFERENCE=bard1_rmsd.pdb TYPE=OPTIMAL
```

while this one can be used to calculate the DRMSD between chains:

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/a-trieste-6.txt
drmsd: DRMSD REFERENCE=bard1_drmsd.pdb TYPE=INTER-DRMSD LOWER_CUTOFF=0.1 UPPER_CUTOFF=1.5
```

2) Number of inter-chains contacts (specific, i.e. native, or all).
This can be achieved with the COORDINATION CVs, as follows:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-6.txt
# backbone beads index for chain A
chainA: GROUP ATOMS=1,3,5,7,9,10,12,15,17,19,21,23,25,27,29,31,33,34,36,38,41,43,45,47,49,51,53,55,57,59,61,63

# backbone beads index for chain B
chainB: GROUP ATOMS=226,228,230,232,234,235,238,239,240,245,246,250,252,255,256,257,259,261,264,266,268,271,27

coord: COORDINATION GROUPA=chainA GROUPB=chainB NOPBC R_0=1.0
```

3) A CV describing the relative orientation of the two chains.
This can be achieved, for example, by defining suitable virtual atoms with the CENTER keyword and the TORSION CV:

```
BEGIN_PLUMED_FILE incomplete DATADIR=example-check/a-trieste-6.txt
# virtual atom representing the first half of chain A
chainA_1: CENTER ATOMS=__FILL__
# virtual atom representing the second half of chain A
chainA_2: CENTER ATOMS=__FILL__

# virtual atom representing the first half of chain B
chainB_1: CENTER ATOMS=__FILL__
# virtual atom representing the second half of chain B
chainB_2: CENTER ATOMS=__FILL__

# torsion CV
dih: TORSION ATOMS=__FILL__
```

11.38.6.6 Exercise 2: analysis of the cmyc-urea simulation

Cmyc is a small disordered peptide made of 11 amino acid. In solution, cmyc adopts a variety of different, but equally populated conformations. In the TARBALL of this exercise, we provide a long MD simulation of cmyc in presence of a single molecule of urea.

Note

We encourage the users to get familiar with the system by visualizing the MD trajectory using `VMD`.

The users are expected to:

- characterize the conformational ensemble of cmyc by calculating free-energies as a function of different CVs.
- calculate the fraction of bound and unbound molecules of urea by defining suitable CVs to measure the position of urea relative to cmyc.
- find the cmyc amino acids that bind urea the most and the least.
- calculate the ensemble averages of different experimental CVs.

Warning

Be careful that the original trajectory might be broken by PBC!

The users are free to choose his/her favorite CVs and they are encouraged to use the on-line manual to create their own PLUMED input file. However, we encourage all the users to experiment at least with the following CVs to characterize the conformational landscape of cmyc:
1) the radius of gyration of cmyc (GYRATION)
2) the content of alpha (ALPHARMSD) and beta (ANTIBETARMSD) secondary structure
The fraction of bound and unbound molecules of urea can be computed after evaluating the minimum distance among all the distances between heavy atoms of cmyc and urea, as follows:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-6.txt
# cmyc heavy atoms
cmyc: GROUP ATOMS=5,6,7,9,11,14,15,17,19,22,24,26,27,28,30,32,34,38,41,45,46,47,49,51,54,56,60,64,65,66,68,70,

# urea heavy atoms
urea: GROUP ATOMS=192,193,194,197

# minimum distance cmyc-urea
mindist: DISTANCES GROUPA=cmyc GROUPB=urea MIN={BETA=50.}
```

For estimating the cmyc amino acid that bind the most and the least urea, we leave the users the choice of the best strategy.
For the calculation of ensemble averages of experimental CVs, we suggest to use:
1) 3J scalar couplings (JCOUPLING)
2) the FRET intensity between termini (FRET)

and we encourage the users to look at the examples provided in the manual for the exact syntax.

11.38.6.7 Exercise 3: Protein G folding simulations

GB1 is a small protein domain with a simple beta-alpha-beta fold. It is a well studied protein that folds on the millisecond time scale. Here we use a structure based potential and well-tempered metadynamics to study the free energy of folding and unfolding. In the TARBALL of this exercise we provide the files needed to run the simulation, the user should write the plumed input file needed to bias the sampling.
The users are expected to:

- setup and perform a well-tempered metadynamics simulation
- evaluate convergence and error calculation of the metadynamics simulation
- calculate the free energy difference between the folded and unfolded state of this protein
- evaluate the robustness of the former by reweighting the resulting free energy as function of different CVs

The users are free to choose his/her favorite CVs and they are encouraged to use the on-line manual to create their own PLUMED input file. However, we encourage all the users to experiment at least with the following CVs to characterize the free-energy landscape of GB1:

- RMSD with respect to the folded state
- GYRATION
- ALPHABETA
- DIHCOR

The users should select two of them for the METAD simulation. Once you are satisfied by the convergence of your simulation, you can use one of the reweighting algorithms proposed to evaluate the free-energy difference between folded and unfolded state as a function of multiple collective variables.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/a-trieste-6.txt
#SETTINGS MOLFILE=regtest/basic/rt32/helix.pdb
#this allows you to use short-cut for dihedral angles
MOLINFO STRUCTURE=GB1_native.pdb

#add here the collective variables you want to bias

#add here the METAD bias, remember that you need to set: one SIGMA per CV, one single HEIGHT, one BIASFACTOR a
#Using GRIDS can increase the performances, so set a as many GRID_MIN and GRID_MAX as the number of CVs with r
#(i.e an RMSD will range between 0 and 3, while ALPHABETA and DIHCOR will range between 0 and N of dihedrals).

#add here the printing
```

11.38.6.8 Conclusions

In summary, in this tutorial you should have learned how to use PLUMED to:

- Analyze trajectories of realistic biological systems using complex CVs
- Extract conformations that correspond to local free-energy minima
- Apply block analysis to estimate error in the reconstructed free-energy profiles
- Calculate ensemble averages of experimental CVs
- Reweight well-tempered metadynamics simulations

11.38.7 Belfast tutorial: Analyzing CVs

11.38.7.1 Aims

The aim of this tutorial is to introduce the users to the plumed syntax. We will go through the writing of simple collective variable and we will use them to analyze a trajectory in terms of probability distributions and free energy.

11.38.7.2 Learning Outcomes

Once this tutorial is completed students will:

- Know how to write a simple plumed input file
- Know how to analyze a trajectory using plumed

11.38.7.3 Resources

The `tarball` for this project contains the following files:

- trajectory-short.xyz : a (short) trajectory for a 16 residue protein in xyz format. All calculations with plumed driver use this trajectory.
- template.pdb : a single frame from the trajectory that can be used in conjunction with the MOLINFO command

11.38.7.4 Instructions

PLUMED is a library that can be accessed by multiple codes adding a relatively simple and well documented interface. Once PLUMED is installed you can run a plumed executable that can be used for multiple purposes:

```
plumed --help
```

some of the listed options report about features that are available in plumed, others can be used to tell plumed to do something: from analyzing a trajectory to patch the source code of a MD code and so on. All the commands have further options that can be seen using plumed command –help, i.e.:

```
plumed driver --help
```

In the following we are going to see how to write an input file for plumed2 that can be used to analyze a trajectory.

11.38.7.4.1 A note on units By default the PLUMED inputs and outputs quantities in the following units:

- Energy - kJ/mol
- Length - nanometers
- Time - picoseconds

If you want to change these units you can do this using the UNITS keyword.

11.38.7.4.2 Introduction to the PLUMED input file A typical input file for PLUMED input is composed by specification of one or more CVs, the printout frequency and a termination line. Comments are denoted with a # and the termination of the input for PLUMED is marked with the keyword ENDPLUMED. Whatever it follows is ignored by PLUMED. You can introduce blank lines. They are not interpreted by PLUMED.
In the following input we will analyze the DISTANCE between the two terminal carbons of a 16 residues peptide, and we will PRINT the results in file named COLVAR.

```
#my first plumed input:
DISTANCE ATOMS=2,253 LABEL=e2edist

#printout frequency
PRINT ARG=e2edist STRIDE=1 FILE=COLVAR

#endofinput
ENDPLUMED
here I can write what I want it won't be read.
```

Now we can use this simple input file to analyze the trajectory included in the RESOURCES:

```
plumed driver --plumed plumed.dat --ixyz trajectory-short.xyz --length-units 0.1
```

NOTE: –length-units 0.1, xyz files, as well as pdb files, are in Angstrom.
You should have a file COLVAR, if you look at it (i.e. more COLVAR) the first two lines should be:

```
#! FIELDS time e2edist
 0.000000 2.5613161
```

NOTE: the first line of the file COLVAR tells you what is the content of each column.
In PLUMED the commands defined in the input files are executed in the same order in which they are written, this means that the following input file is wrong:

```
#printout frequency
PRINT ARG=e2edist STRIDE=1 FILE=COLVAR
#my first plumed input:
DISTANCE ATOMS=2,253 LABEL=e2edist
#endofinput
ENDPLUMED
here I can write what I want it won't be read.
```

Try to run it.
Sometimes, when calculating a collective variable, you may not want to use the positions of a number of atoms directly. Instead you may wish to use the position of a virtual atom whose position is generated based on the positions of a collection of other atoms. For example you might want to use the center of a group of atoms (CENTER):
Since PLUMED executes the input in order you need to define the new Virtual Atom before using it:

```
first: CENTER ATOMS=1,2,3,4,5,6
last: CENTER ATOMS=251-256

e2edist: DISTANCE ATOMS=2,253
comdist: DISTANCE ATOMS=first,last

PRINT ARG=e2edist,comdist STRIDE=1 FILE=COLVAR

ENDPLUMED
```

NOTE: an action (i.e. CENTER or DISTANCE here) can be either labeled using LABEL as we did before or as label: ACTION as we have just done here.
With the above input this is what happen inside PLUMED with a STRIDE=1:

1. calculates the position of the Virtual Atom 'first' as the CENTER of atoms from 1 to 6;
2. calculates the position of the Virtual Atom 'last' as the CENTER of atoms from 251 to 256;
3. calculates the distance between atoms 2 and 253 and saves it in 'e2edist';
4. calculates the distance between the two atoms 'first' and 'last' and saves it in 'comdist';
5. print the content of 'e2edist' and 'comdist' in the file COLVAR

In the above input we have used two different ways of writing the atoms used in CENTER calculation:

1. ATOMS=1,2,3,4,5,6 is the explicit list of the atoms we need
2. ATOMS=251-256 is the range of atoms needed

ranges of atoms can be defined with a stride which can also be negative:

1. ATOMS=from,to:by (i.e.: 251-256:2)
2. ATOMS=to,from:-by (i.e.: 256-251:-2)

Now by plotting the content of the COLVAR file we can compare the behavior in this trajectory of both the terminal carbons as well as of the center of masses of the terminal residues.

```
gnuplot
```

What do you expect to see now by looking at the trajectory? Let's have a look at it

```
vmd template.pdb trajectory-short.xyz
```

Virtual atoms can be used in place of standard atoms everywhere an atom can be given as input, they can also be used together with standard atoms. So for example we can analyze the TORSION angle for a set of Virtual and Standard atoms:

```
first: CENTER ATOMS=1-6
last: CENTER ATOMS=251-256
cvtor: TORSION ATOMS=first,102,138,last

PRINT ARG=cvtor STRIDE=1 FILE=COLVAR

ENDPLUMED
```

The above CV don't look smart to learn something about the system we are looking at. In principle CV are used to reduce the complexity of a system by looking at a small number of properties that could be enough to rationalize its behavior.
Now try to write a collective variable that measures the Radius of Gyration of the system: GYRATION.
NOTE: if what you need for one or more variables is a long list of atoms and not a virtual atom one can use the keyword GROUP. A GROUP can be defined using ATOMS in the same way we saw before, in addition it is also possible to define a GROUP by reading a GROMACS index file.

```
ca: GROUP ATOMS=9,16,31,55,69,90,102,114,124,138,160,174,194,208,224,238
```

Now 'ca' is not a virtual atom but a simple list of atoms.

11.38.7.4.3 MULTICOLVAR Sometimes it can be useful to calculate properties of many similar collective variables at the same time, for example one can be interested in calculating the properties of the distances between a group of atoms, or properties linked to the distribution of the dihedral angles of a chain and so on. In PLUMED this kind of collective variables fall under the name of MULTICOLVAR (cf. MultiColvar.) Here we are going to analyze the distances between CA carbons along the chain:

```
ca: GROUP ATOMS=9,16,31,55,69,90,102,114,124,138,160,174,194,208,224,238
dd: DISTANCES GROUP=ca MEAN MIN={BETA=50} MAX={BETA=0.02} MOMENTS=2

PRINT ARG=dd.mean,dd.min,dd.max,dd.moment-2 STRIDE=1 FILE=COLVAR

ENDPLUMED
```

The above input tells PLUMED to calculate all the distances between CA carbons and then look for the mean distance, the minimum distance, the maximum distance and the variance. In this way we have defined four collective variables that are calculated using the distances. These four collective variables are stored as components of the defined action 'dd': dd.mean, dd.min, dd.max, dd.moment-2.
The infrastructure of multicolvar has been used to develop many PLUMED collective variables as for example the set of Secondary Structure CVs (ANTIBETARMSD, PARABETARMSD and ALPHARMSD).

```
MOLINFO STRUCTURE=template.pdb
abeta: ANTIBETARMSD RESIDUES=all TYPE=DRMSD LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} STRANDS_CUTOFF=1

PRINT ARG=abeta.lessthan STRIDE=1 FILE=COLVAR

ENDPLUMED
```

We have now seen how to write the input some of the many CVs available in PLUMED. More complex CVs will be discussed in the next workshop, Belfast tutorial: Adaptive variables I.

11.38.7.4.4 Analysis of Collective Variables Collective variables are usually used to visualize the Free Energy of a system. Given a system evolving at fixed temperature, fixed number of particles and fixed volume, it will explore different conformations with a probability

$$P(q) \propto e^{-\frac{U(q)}{k_B T}}$$

where q are the microscopic coordinates and k_B is the Boltzmann constant.

It is possible to analyze the above probability as a function of one or more collective variable $s(q)$:

$$P(s) \propto \int dq e^{-\frac{U(q)}{k_B T}} \delta(s - s(q))$$

where the δ function means that to for a given value s of the collective variable are counted only those conformations for which the CV is s. The probability can be recast to a free energy by taking its logarithm:

$$F(s) = -k_B T \log P(s)$$

This means that by estimating the probability distribution of a CV it is possible to know the free energy of a system along that CV. Estimating the probability distribution of the conformations of a system is what is called 'sampling'.
In order to estimate a probability distribution one needs to make HISTOGRAM from the calculated CVs. PLUMED includes the possibility of constructing a histogram from data both on the fly as well as a posteriori as we are going to do now.

```
MOLINFO STRUCTURE=template.pdb
abeta: ANTIBETARMSD RESIDUES=all TYPE=DRMSD LESS_THAN={RATIONAL R_0=0.08 NN=8 MM=12} STRANDS_CUTOFF=1
ca: GROUP ATOMS=9,16,31,55,69,90,102,114,124,138,160,174,194,208,224,238
DISTANCES ...
GROUP=ca MEAN MIN={BETA=50} MAX={BETA=0.02} MOMENTS=2 LABEL=dd
... DISTANCES

PRINT ARG=abeta.lessthan,dd.mean,dd.min,dd.max,dd.moment-2 STRIDE=1 FILE=COLVAR

HISTOGRAM ...
ARG=abeta.lessthan,dd.mean
LABEL=hh
KERNEL=DISCRETE
GRID_MIN=0,0.8
GRID_MAX=4,1.2
GRID_BIN=40,40
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo

ENDPLUMED
```

NOTE: HISTOGRAM ... means that what follow is part of the HISTOGRAM function, the same can be done for any action in PLUMED.
The above input tells PLUMED to accumulate the two collective variables on a GRID. In addition the probability can be converted to a free-energy using the CONVERT_TO_FES method and then use DUMPGRID to write it. Histograms can be accumulated in a smoother way by using a KERNEL function, a kernel is a normalized function, for example a normalized gaussian is the default kernel in PLUMED, that is added to the histogram centered in the position of the data. Estimating a probability density using kernels can in principle give more accurate results, on the other hand in addition to the choice of the binning one has to choose a parameter that is the WIDTH of the kernel function. As a rule of thumb: the grid spacing should be smaller (i.e. one half or less) than the BANDWIDTH and the BANDWIDTH should be smaller (i.e. one order of magnitude) than the variance observed/expected for the variable.

```
HISTOGRAM ...
LABEL=hh
ARG=abeta.lessthan,dd.mean
GRID_MIN=0,0.8
GRID_MAX=4,1.2
GRID_SPACING=0.04,0.004
BANDWIDTH=0.08,0.008
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo

ENDPLUMED
```

If you have time less at the end of the session read the manual and look for alternative collective variables to analyze the trajectory. Furthermore try to play with the HISTOGRAM parameters to see the effect of using KERNEL in analyzing data.

11.38.8 Belfast tutorial: Adaptive variables I

11.38.8.1 Aim

In this section we want to introduce the concept of adaptive collective variables. These are special variables that are knowledge-based in that are built from a pre-existing notion of the mechanism of the transition under study.

11.38.8.2 Resources

Here is the `tarball with the files referenced in the following`.

11.38.8.3 What happens when in a complex reaction?

When you deal with a complex conformational transition that you want to analyze (or bias), very often you cannot just describe it with a single order parameter.
As an example in Figure belfast-2-cdk-fig I consider a large conformational transition like those involved in activating the kinase via open-close transition of the activation loop. In sticks you see the part involved in the large conformational change, the rest is either keeping the structure and just moving a bit or is a badly resolved region in the X-ray structure. This is a complex transition and it is hard to tell which is the order parameter that best describes the transition.

One could identify a distance that can distinguish open from close but

- the plasticity of the loop is such that the same distance can correspond to an almost closed loop and almost open loop. One would like to completely divide these two situations with something which is discriminating what intuitively one would think as open and closed
- the transition state is an important point where one would like to see a certain crucial interaction forming/breaking so to better explain what is really happening. If you capture then hypothetically you would be able to say what is dictating the rate of this conformational transition. A generic distance is a very hybrid measure that is unlikely to capture a salt-bridge formation and a concerted change of many dihedral angles or the change in solvation state that are happening while the transition is happening. All these things are potentially important in such transition but none of them is explaining the whole thing.

So basically in these cases you have to deal with an intrinsic multidimensional collective variable where you would need many dimensions. How would you visualize a 10 dimensional CV where you use many distances, coordination numbers and dihedral angles (ouch, they're periodic too!) ?
Another typical case is the docking of a small molecule in a protein cleft or gorge, which is the mechanism of drug action. This involves specific conformational transition from both the small molecule and the protein as the small molecule approaches the protein cavity. This also might imply a specific change in the behavior when the solvation state changes.

Other typical examples are chemical reactions. Nucleophillic attacks typically happen with a role from the solvent (see some nice paper from Nobel-prize winner Arieh Warshel) and sizable geometric distortions of the neighboring groups.

11.38.8.4 Path collective variables

One possibility to describe many different things that happen in a single reaction is to use a dimensional reduction technique and in plumed the simplest example that may show its usefulness can be considered that of the path collective variables.
In a nutshell, your reaction might be very complex and happening in many degree of freedom but intuitively is a sort of track along which the reaction proceeds. So what we need is a coordinate that, given a conformation, just tells which point along the "reactive track" is closest.
For example, in Fig. belfast-2-ab-fig, you see a typical chemical reaction (hydrolysis of methyl phosphate) with the two end-points denoted by A and B. If you are given a third point, just by looking at it, you might find that this resembles the reactant more than the product, so, hypothetically, if you would intuitively give a parameter that would be 1 for a configuration in the A state and 2 for a configuration in the B state, you probably would give it something like 1.3, right?

Path collective variables are the extension to this concept in the case you have many conformation that describe your path, and therefore, instead of an index that goes from 1 to 2 you have an index that goes from 1 to N , where N is the number of conformation that you use in input to describe your path.
From a mathematical point of view, that's rather simple. The progress along the path is calculated with the following equation:

$$S(X) = \frac{\sum_{i=1}^{N} i \ \exp^{-\lambda|X-X_i|}}{\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|}}$$

where in belfast-2-s-eq the $|X - X_i|$ represents a distance between one configuration X which is analyzed and another from the set that compose the path X_i. The parameter λ is a positive value that is tuned in a way explained later. here are a number of things to note to make you think that this is exactly what you want.

- The negative exponential function is something that is 1 whenever the value at the exponent is zero, and is progressively smaller when the value is larger than zero (trivially, the case with the value at the exponent larger than zero never occurs since lambda is a positive quantity and the distance is by definition positive).
- Whenever you sit exactly on a specific images X_j then all the other terms in the sum disappear (if λ is large enough) and only the value j survives returning exactly $S(X) = j$.

In order to provide a value which is continuous, the parameter λ should be correctly tuned. As a rule of thumb I use the following formula

$$\lambda = \frac{2.3(N-1)}{\sum_{i=1}^{N-1} |X_i - X_{i+1}|}$$

which imply that one should calculate the average distance between consecutive frames composing the path. Note also that this distance should be more or less similar between the frames. Generally I tolerate fluctuation of the order of 10/15 percent tops. If you have larger, then it is better to have a smaller value of λ.
It is important to note that in principle one could even have a specific λ value associated to each frame of the path but this would provide some distortion in the diffusion coefficient which could potentially harm a straightforward interpretation of the free energy landscape.
So, at this point it is better to understand what is meant with "distance" since a distance between two conformations can be calculated in very many ways. The way we refer here is by using mean square deviation after optimal alignment. This means that at each step in which the analysis is performed, a number N of optimal alignments is performed. Namely what is calculated is $|X - X_i| = d(X, X_i)^2$ where $d(X, X_i)$ is the RMSD as defined in what you see here RMSD.
Using the MSD instead of RMSD is sometimes more convenient and more stable (you do not have a denominator that goes to zero in the derivatives when biasing.
Anyway this is a matter of choice. Potentially one could equally employ other metrics like a set of coordination numbers (this was done in the past), and then you would avoid the problem of rototranslations (well, which is not a problem since you have it already in plumed) but for some applications that might become appealing. So in path collective variables (and in all the dimensional reduction based collective variables) the problem is converted from the side of choosing the collective variable in choosing the right way to calculate distances, also called "metrics".
The discussion of this issue is well beyond the topic of this tutorial, so we can move forward in how to tell plumed to calculate the progress along the path whenever the MSD after optimal alignment is used as distance.

```
p1: PATHMSD REFERENCE=all.pdb LAMBDA=50.0
PRINT ARG=p1.sss,p1.zzz STRIDE=100 FILE=colvar FMT=%8.4f
```

Note that reference contains a set of PDB, appended one after the other, with a END field. Note that there is no need to place all the atoms of the system in the PDB reference file you provide. Just put the atoms that you think might be needed. You can leave out big domains, solvent and ions if you think that is not important for your use.
Additionally, note that the measure units of LAMBDA are in the units of the code. In gromacs they are in nm^2 while NAMD is Angstrom^2. PATHMSD produces two arguments that can be printed or used in other ActionWith↩ Arguments. One is the progress along the path of belfast-2-s-eq, the other is the distance from the closest point along the path, which is denoted with the zzz component. This is defined as

$$Z(X) = -\frac{1}{\lambda} \log(\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|})$$

It is easy to understand that in case of perfect match of $X = X_i$ this equation gives back the value of $-X_i|$ provided that the lambda is adjusted correctly.
So, the two variables, put together can be visualized as
This variable is important because whenever your simulation is running close to the path (low Z values), then you know that you are reproducing reliably the path you provided in input but if by chance you find some other path that goes, say, from $S = 1, Z = 0$ to $S = N, Z = 0$ via large Z values, then it might well be that you have just discovered a good alternative pathway. If your path indeed is going from $S = 1, Z = large$ to $S = N, Z = large$ then it might well be that you do not have your reaction accomplished, since your reaction, by definition should go from the reactant which is located at $S = 1, Z = 0$ to the product, which is located at $S = 1, Z = N$ so you should pay attention. This case is exemplified in Fig. belfast-2-ab-sz-nowhere-fig

11.38.8.5 A note on the path topology

A truly important point is that if you get a trajectory from some form of accelerated dynamics (e.g. simply by heating) this cannot simply be converted into a path. Since it is likely that your trajectory is going back and forth randomly (not in the case of SMD or US, discussed later), your trajectory might be not topologically suitable. To understand that, suppose you simply collect a reactive trajectory of 100 ps into the reference path you give to the PATHMSD and simply you assign the frame of 1 ps to index 1 (first frame occurring in the reference file provided to PATHMSD), the frame of 2 ps to index 2 and so on : it might be that you have two points which are really similar but one is associated to step, say 5 and the other is associated with frame 12. When you analyze the same trajectory, when you are sitting on any of those points then the calculation of S will be an average like $S(X) = (5 + 12)/2 = 8.5$ which is an average of the two indexes and is completely misleading since it let you think that you are moving between point 8 and 9, which is not the case. So this evidences that your reference frames should be "topologically consecutive". This means that frame 1 should be the closest to frame 2 and all the other frames should be farther apart. Similarly frame 2 should be equally close (in an RMSD sense) to 1 and 3 while all the others should be farther apart. Same for frame 3: this should be closest to frame 2 and 4 and farther apart from all the others and so on. This is equivalent to calculate an "RMSD matrix" which can be conveniently done in vmd (this is a good exercise for a number of reasons) with RMSD Trajectory tools, by choosing different reference system along the set of reference frames.
This is shown in Fig. belfast-2-good-matrix-fig where the matrix has a typical gull-wing shape.
On the contrary, whenever you extract the frames from a pdb that you produced via free MD or some biased methods (SMD or Targeted MD for example) then your frame-to-frame distance is rather inhomogeneous and looks something like
Aside from the general shape, which is important to keep the conformation-to-index relation (this, as we will see in the next part is crucial in the multidimensional scaling) the crucial thing is the distance between neighbors.
As a matter of fact, this is not much important in the analysis but is truly crucial in the bias. When biasing a simulation, you generally want to introduce a force that push your system somewhere. In particular, when you add a bias which is based on a path collective variable, most likely you want that your system goes back and forth along your path. The bias is generally applied by an additional term in the Hamiltonian, this can be a spring term for Umbrella Sampling, a Gaussian function for Metadynamics or whatever term which is a function of the collective variable s. Therefore the Hamiltonian $H(X)$ where X is the point of in the phase space where your system is takes the following form

$$H'(X) = H(X) + U(S(X))$$

where $U(S(X))$ is the force term which depends on the collective variable that ultimately is a function of the X. Now, when you use biased dynamics you need to evolve according the forces that this term produces (this only holds for MD, while not in MC) and therefore you need

$$F_i = -\frac{dH'(X)}{dx_i} = -\frac{dH'(X)}{dx_i} - \frac{\partial U(S(X))}{\partial S}\frac{\partial S(X)}{\partial x_i}$$

This underlines the fact that, whenever $\frac{\partial S(X)}{\partial x_i}$ is zero, then you have no force on the system. Now the derivative of the progress along the path is

$$\frac{\partial S(X)}{\partial x_i} = \frac{\sum_{i=1}^{N} -\lambda\, i\, \frac{\partial |X-X_i|}{\partial x_i} \exp^{-\lambda|X-X_i|}}{\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|}} - \frac{(\sum_{i=1}^{N} i\, \exp^{-\lambda|X-X_i|})(\sum_{j=1}^{N} -\lambda \frac{\partial |X-X_j|}{\partial x_i} \exp^{-\lambda|X-X_j|})}{(\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|})^2} = \frac{\sum_{i=1}^{N} -\lambda\, i\, \frac{\partial |X-X_i}{\partial x_i}}{\sum_{i=1}^{N} \exp^{-}}$$

which can be rewritten as

$$\frac{\partial S(X)}{\partial x_i} = \lambda \frac{\sum_{i=1}^{N} \frac{\partial |X-X_i|}{\partial x_i} \exp^{-\lambda|X-X_i|}[s(X) - i]}{\sum_{i=1}^{N} \exp^{-\lambda|X-X_i|}}$$

It is interesting to note that whenever the λ is too small the force will vanish. Additionally, when λ is too large, then it one single exponential term will dominate over the other on a large part of phase space while the other will vanish. This means that the $S(X)$ will assume almost discrete values that produce zero force. Funny, isn't it?

11.38.8.6 How many frames do I need?

A very common question that comes is the following: "I have my reaction or a model of it. how many frames do I need to properly define a path collective variable?" This is a very important point that requires a bit of thinking. It all depends on the limiting scale in your reaction. For example, if in your process you have a torsion, as the smallest event that you want to capture with path collective variable, then it is important that you mimic that torsion in the path and that this does not contain simply the initial and final point but also some intermediate. Similarly, if you have a concerted bond breaking, it might be that all takes place in the range of an Angstrom or so. In this case you should have intermediate frames that cover the sub-Angstrom scale. If you have both in the same path, then the smallest dominates and you have to mimic also the torsion with sub-Angstrom accuracy.

11.38.8.7 Some tricks of the trade: the neighbors list.

If it happens that you have a very complex and detailed path to use, say that it contains 100 frames with 200 atoms each, then the calculation of a 100 alignment is required every time you need the CV. This can be quite expensive but you can use a trick. If your trajectory is continuous and you are sure that your trajectory does not show jumps where your system suddenly move from the reactant to the product, then you can use a so-called neighbor list. The plumed input shown before then becomes

```
p1: PATHMSD REFERENCE=all.pdb LAMBDA=50.0 NEIGH_STRIDE=100 NEIGH_SIZE=10
PRINT ARG=p1.sss,p1.zzz STRIDE=100 FILE=colvar FMT=%8.4f
```

and in this case only the closest 10 frames from the path will be used for the CV. Then the list of the frames to use is updated every 100 steps. If you are using a biased dynamics this may introduce sudden change in the derivatives, therefore it is better to check the stability of the setup before running production-quality calculations.

11.38.8.8 The molecule of the day: alanine dipeptide

Here and probably in other parts of the tutorial a simple molecule is used as a test case. This is alanine dipeptide in vacuum. This rather simple molecule is useful to make many benchmark that are around for data analysis and free energy methods. It is a rather nice example since it presents two states separated by a high (free) energy barrier.
In Fig. belfast-2-ala-fig its structure is shown.
The two main metastable states are called C_7eq and C_7ax.
Here metastable states are intended as states which have a relatively low free energy compared to adjacent conformation. At this stage it is not really useful to know what is the free energy, just think in term of internal energy. This is almost the same for such a small system which so few degrees of freedom.
It is conventional use to show the two states in terms of Ramachandran dihedral angles, which are denoted with Φ and Ψ in Fig. belfast-2-transition-fig .

11.38.8.9 Examples

Now as a simple example, I want to show you that plotting some free dynamics trajectories shoot from the saddle point, you get a different plot in the path collective variables if you use the right path or if you use the wrong path.
In Fig. belfast-2-good-bad-path-fig I show you two example of possible path that join the C_7eq and C_7ax metastable states in alanine dipeptide. You might clearly expect that real (rare) trajectories that move from one basin to the other would rather move along the black line than on the red line.
So, in this example we do a sort of "commmittor analysis" where we start shooting a number of free molecular dynamics from the saddle point located at $\Phi = 0$ and $\Psi = -1$ and we want to see which way do they go. Intuitively, by assigning random velocities every time we should find part of the trajectories that move toward C_7eq and part that move towards C_7ax.
I provided you with two directories, each containing a bash script script.sh whose core (it is a bit more complicated in practice...) consists in:

```
#
# set how many runs you want to do
#
ntests=50
```

```
for i in `seq 1 $ntests`
do
        #
        # assign a random velocity at each timestep by initializing the
        #
        sed s/SEED/$RANDOM/ md.mdp >newmd.mdp
        #
        # do the topology: this should write a topol.tpr
        #
        $GROMPP -c start.gro -p topol.top -f newmd.mdp
        $GROMACS_BIN/$MDRUN -plumed plumed.dat
        mv colvar colvar_$i
done
```

This runs 50 short MD runs (few hundreds steps) and just saves the colvar file into a labeled colvar file. In each mdrun plumed is used to plot the collective variables and it is something that reads like the following:

```
# Phi
t1: TORSION ATOMS=5,7,9,15
# Psi
t2: TORSION ATOMS=7,9,15,17
# The right path
p1: PATHMSD REFERENCE=right_path.dat LAMBDA=15100.
# The wrong path
p2: PATHMSD REFERENCE=wrong_path.dat  LAMBDA=8244.4
# Just a printout of all the stuff calculated so far
PRINT ARG=* STRIDE=2 FILE=colvar FMT=%12.8f
```

where I just want to plot Φ , Ψ and the two path collective variables. Note that each path has a different LAMBDA parameters. Here the Ramachandran angles are plotted so you can realize which path is the system walking in a more comfortable projection. This is of course fine in such a small system but whenever you have to deal with larger systems and control hundreds of CVs at the same time, I think that path collective variables produce a more intuitive description for what you want to do.
If you run the script simply with

```
pd@plumed:~> ./script.sh
```

then after a minute or so, you should have a directory which is full of colvar files. Let's revise together how the colvar file is formatted:

```
#! FIELDS time t1 t2 p1.sss p1.zzz p2.sss p2.zzz
#! SET min_t1 -pi
#! SET max_t1 pi
#! SET min_t2 -pi
#! SET max_t2 pi
 0.000000  -0.17752998  -1.01329788  13.87216908   0.00005492  12.00532256   0.00233905
 0.004000  -0.13370142  -1.10611136  13.87613508   0.00004823  12.03390658   0.00255806
 0.008000  -0.15633049  -1.14298481  13.88290617   0.00004511  12.07203319   0.00273764
 0.012000  -0.23856451  -1.12343958  13.89969608   0.00004267  12.12872544   0.00284883
...
```

In first column you have the time, then t1 (Φ) , then t2 (Ψ) and the path collective variables p1 and p2. Note that the action PATHMSD calculates both the progress along the path (p1.sss) and the distance from it (p1.zzz) . In PLUMED jargon, these are called "components". So a single Action (a line in plumed input) can calculate many components at the same time. This is not always the case: sometimes by default you have one component but specific flags may enable more components to be calculated (see DISTANCE for example). Note that the header (all the part of a colvar file that contains # as first character) is telling already what it inside the file and eventually also tells you if a variable is contained in boundaries (for example torsion angles, are periodic and their co-domain is defined in -Pi and Pi).
At the end of the script, you also have two additional scripts. One is named script_rama.gplt and the other is named script_path.gplt. They contain some gnuplot commands that are very handy to visualize all the colvar files without making you load one by one, that would be a pain.
Now, let's visualize the result from the wrong path directory. In order to do so, after having run the calculation, then do

```
pd@plumed:~>gnuplot
gnuplot> load "script_rama.gplt"
```

what you see is that all the trajectories join the reactant and the product state along the low free energy path depicted before.
Now if you try to load the same bunch of trajectories as a function of the progress along the path and the distance from the path in the case of the wrong path then simply do

```
gnuplot> load "script_path_wrong.gplt"
```

What do you see? A number of trajectories move from the middle towards the right bottom side at low distance from the path. In the middle of the progress along the path, you have higher distance. This is expected since the distance zero corresponds to a unlikely, highly-energetic path which is unlikely to occur. Differently, if you now do

```
gnuplot> load "script_path_right.gplt"
```

You see that the path, compared to what happened before, run much closer to small distance from the path. This means that the provided path is right and more representative of what happens in the reactive path.
Note that going at high distances can be also beneficial. It might help you to explore alternative paths that you have not explored before. But beware, the more you get far from the path, the more states are accessible, in a similar way as the fact that the surface of a sphere increase with its radius. The surface ramps up much faster than the radius therefore you have a lots of states there. This means also high entropy, so many systems actually tend to drift to high distances while, on the contrary, zero distance is never reached in practice (zero entropy system is impossible to reach at finite temperature). So you can see by yourself that this can be a big can of worms. In particular, my experience with path collective variables and biological systems tells me that most of time is hopeless to go to high distances to find new path in many cases (for example, in folding). While this is worth whenever you think that the paths are not too many (alternative routes in chemical reaction or enzymatic catalysis).

11.38.8.10 How to format my input?

Very often it is asked how to format a set of pdb to be suitably used with path collective variables. Here are some tricks.

- When you dump the files with vmd or (for gromacs users, using trjcat), the pdb you obtain is re-indexed from 1. This is also the case when you select a sub-ensemble of atoms of the path (e.g. the heavy atoms only or the backbone atoms). This is rather unfortunate and you have to fix is someway. My preference is to dump the whole pdb but water (when I do not need it) and use some awk script to select the atoms I am interested in.
- Pay attention to the last two column. These are occupancy and beta. With the first (occupancy) you set the atoms which are used to perform the alignment. The atoms which have zero occupancy will not be used in the alignment. The second column is beta and controls which atoms are used for the calculation of the distance after having performed the alignment on the set of atoms which have nonzero occupancy column. In this way you can align all your system by using a part of the system and calculate the distance respect to another set. This is handy in case of protein-ligand. You set the alignment of the protein and you calculate the distance based on the ligand and the part of the protein which is in contact with the protein. This is done for example in this article.
- Plumed-GUI (version $>$ 2.0) provides the *Structure->Build reference structure...* function to generate inputs that conform to the above rules from within VMD.
- Note that all the atoms contained in the REFERENCE must be the same. You cannot have a variable number of atoms in each pdb contained in the reference.
- The reference is composed as a set of concatenated PDB files that are interrupted by a TER/END/ENDMDL card. Both HETATM and ATOM cards denote the atoms of the set.
- Include in the reference frames only the needed atoms. For example, if you have a methyl group involved in a conformational transition, it might be that you do not want to include the hydrogen atoms of the methyl since these rotate fast and probably they do not play ant relevant role.

11.38.8.11 Fast forward: metadynamics on the path

This section is actually set a bit forward but I included here for completeness now. It is recommended to be read after you have an introduction on Metadynamics and to well-tempered Metadynamics in particular. Here I want to show a couple of concept together.

- Path collective variables can be used for exploring alternative routes. It is effective in highly structure molecules, while it is tricky on complex molecules whenever you have many competing routes
- Path collective variables suffer from problems at the endpoints (as the highly popular coordinates COORDINATION for example) that can be cured with flexible hills and an appropriate reweighting procedure within the well-tempered Metadynamics scheme.

Let's go to the last problem. All comes from the derivative belfast-2-sder-eq. Whenever you have a point of phase space which is similar to one of the endpoint than one of the points in the center then you get a $s(X)$ which is 1 or N (where N is the number of frames composing the path collective variable). In this case that exponential will dominate the others and you are left with a constant (since the derivative of RMSD is a constant since it is linear in space). This means that, no matter what happens here, you have small force. Additionally you have small motion in the CV space. You can move a lot in configuration space but if the closest point is one of the endpoint, your CV value will always be one of the endpoint itself. So, if you use a fixed width of your CV which you retrieve from a middle point in your path, this is not suitable at all at the endpoints where your CV fluctuates much less. On the contrary if you pick the hills width by making a free dynamics on the end states you might pick some sigmas that are smaller than what you might use in the middle of the path. This might give a rough free energy profile and definitely more time to converge. A possible solution is to use the adaptive gaussian width scheme. In this scheme you adapt the hills to their fluctuation in time. You find more details in [50] Additionally you also adopt a non spherical shape taking into account variable correlation. So in this scheme you do not have to fix one sigma per variable sigma, but just the time in which you calculate this correlation (another possibility is to calculate it from the compression of phase space but will not be covered here). The input of metadynamics might become something like this

```
t1: TORSION ATOMS=5,7,9,15
t2: TORSION ATOMS=7,9,15,17
p1: PATHMSD REFERENCE=right_path.dat LAMBDA=15100.
p2: PATHMSD REFERENCE=wrong_path.dat  LAMBDA=8244.4
#
# do a metadynamics on the right path, use adaptive hills whose decay time is 125 steps (250 fs)
# and rather standard WT parameters
#
meta: METAD ARG=p1.sss,p1.zzz  ADAPTIVE=DIFF SIGMA=125 HEIGHT=2.4 TEMP=300 BIASFACTOR=12 PACE=125
PRINT ARG=* STRIDE=10 FILE=colvar FMT=%12.8f
```

You can find this example in the directory BIASED_DYNAMICS. After you run for a while it is interesting to have a feeling for the change in shape of the hills. That you can do with

```
pd@plumed:~> gnuplot
gnuplot>  p "<head -400 HILLS" u 2:3:4:5 w xyer
```

that plots the hills in the progress along the path and the distance from the path and add an error bar which reflects the diagonal width of the flexible hills for the first 400 hills (Hey note the funny trick in gnuplot in which you can manipulate the data like in a bash script directly in gnuplot. That's very handy!).
There are a number of things to observe: first that the path explores high distance since the metadynamics is working also in the distance from the path thus accessing the paths that were not explored before, namely the one that goes from the upper left corner of the Ramachandran plot and the one that passes through the lower left corner. So in this way one can also explore other paths. Additionally you can see that the hills are changing size rather considerably. This helps the system to travel faster since at each time you use something that has a nonzero gradient and your forces act on your system in an effective way. Another point is that you can see broad hills covering places which you have not visited in practice. For example you see that hills extend so wide to cover point that have negative progress along the path, which is impossible by using the present definition of the progress along the path. This introduced a problem in calculating the free energy. You actually have to correct for the point that you visited in reality.
You can actually use sum_hills to this purpose in a two-step procedure. First you calculate the negative bias on a given range:

```
pd@plumed:~> plumed sum_hills --hills HILLS --negbias  --outfile negative_bias.dat --bin 100,100 --min -5,-0.0
```

and then calculate the correction. You can use the same hills file for that purpose. The initial transient time should not matter if your simulation is long enough to see many recrossing and, secondly, you should check that the hills height in the well tempered are small compared to the beginning.

```
pd@plumed:~> plumed sum_hills --histo HILLS --bin 100,100 --min -5,-0.005 --max 25,0.05 --kt 2.5 --sigma 0.5,0
```

Note that in the correction you should assign a sigma, that is a "trust radius" in which you think that whenever you have a point somewhere, there in the neighborhood you assign a bin (it is done with Gaussian in reality, but this does not matter much). This is the important point that compensates for the issues you might encounter putting excessive large hills in places that you have not visited. It is nice to have a look to the correction and compare with the hills in the same range.

```
gnuplot> set pm3d
gnuplot> spl "correction.dat" u 1:2:3 w l
gnuplot> set contour
gnuplot> set cntrp lev incremental -20,4.,1000.
gnuplot> set view map
gnuplot> unset clabel
gnuplot> replot
```

You might notice that there are no contour in the unrealistic range, this means that the free energy correction is impossible to calculate since it is too high (see Fig. belfast-2-metadpath-correction-fig).

Now the last thing that one has to do to have the most plausible free energy landscape is to sum both the correction and the negative bias produced. This can be easily done in gnuplot as follows:

```
gnuplot> set pm3d
gnuplot> spl "<paste negative_bias.dat correction.dat " u 1:2:($3+$8) w pm3d
gnuplot> set view map
gnuplot> unset key
gnuplot> set xr [-2:23]
gnuplot> set contour
gnuplot> unset clabel
gnuplot> set cbrange [-140:-30]
gnuplot> set cntrp lev incr -140,6,-30
```

So now we can comment a bit on the free energy surface obtained and note that there is a free energy path that connects the two endpoints and runs very close to zero distance from the path. This means that our input path is resembles what is really happening in the system. Additionally you can see that there are many comparable routes different from the straight path. Can you make a sense of it just by looking at the free energy on the Ramachandran plot?

11.38.9 Belfast tutorial: Adaptive variables II

11.38.9.1 Aims

The aim of this tutorial is to consolidate the material that was covered during Belfast tutorial: Analyzing CVs and Belfast tutorial: Adaptive variables I on analyzing trajectories using collective variables and path collective variables. We will then build on this material by showing how you can use the multidimensional scaling algorithm to automate the process of finding collective variables.

11.38.9.2 Learning Outcomes

Once this tutorial is completed students will:

- Know how to load colvar data into the GISMO plugin
- Know how to run the multidimensional scaling algorithms on a trajectory
- Be able to explain how we can automate the process of finding collective variables by seeking out an isometry between a high-dimensional and low-dimensional space

11.38.9.3 Resources

The `tarball` for this project contains the following files:

- trajectory-short.xyz : a (short) trajectory for a 16 residue protein in xyz format. All calculations with plumed driver use this trajectory.
- trajectory-short.pdb : the same trajectory in pdb format, this can be loaded with VMD
- template.pdb : a single frame from the trajectory that can be used in conjunction with the MOLINFO command

11.38.9.4 Instructions

11.38.9.4.1 Visualizing the trajectory The aim of this tutorial is to understand the data contained in the trajectory called trajectory-short.pdb. This file contains some frames from a simulation of a 16 residue protein. As a start point then lets load this trajectory with vmd and have a look at it. Type the following command into the command line:

```
vmd trajectory-short.pdb
```

Look at it with the various representations that vmd offers. If you add the instructions below to your .vmdrc file you can make the new cartoon representation update automatically for each frame of the trajectory by pressing the m key - cool huh!

```
proc structure_trace {name index op} {
      vmd_calculate_structure $index
}

user add key m {
  puts "Automatic update of secondary structure, and alignment to first frame"
  trace variable vmd_frame w structure_trace
  rmsdtt
  rmsdtt::doAlign
  destroy $::rmsdtt::w
  clear_reps top
  mol color Structure
  mol selection backbone
  mol representation NewCartoon
  mol addrep top
}
```

What are your impressions about this trajectory based on looking at it with VMD? How many basins in the free energy landscape is this trajectory sampling from? What can we tell from looking at this trajectory that we could perhaps put in a paper?
If your answers to the questions at the end of the above paragraph are I don't know that is good. We can tell very little by just looking at a trajectory. In fact the whole point of the tutorials I referenced at the start of this one was to find ways of analyzing trajectories precisely so that we are not put in this position of staring at trajectories mystified!

11.38.9.4.2 Installing GISMO You can download and install the GISMO plugin by following the instructions on the page below:
http://epfl-cosmo.github.io/sketchmap/index.html?page=code
(you don't need to compile the sketch-map code) Once it is installed you should have an option to open it when you open vmd. The option to open GISMO can be found under Extensions>Analysis>GISMO.

11.38.9.4.3 Finding collective variables Right so lets come up with some CVs to analyze this trajectory with. As some of you may know we can understand the conformation of proteins by looking at the Ramachandran angles. For those of you who don't know here is a Wikipedia article:
http://en.wikipedia.org/wiki/Ramachandran_plot
Our protein has 32 Ramachandran angles. We'll come back to that. For the time being pick out a couple that you think might be useful and construct a plumed input to calculate them and print them to a file. You will need to use the TORSION and PRINT commands in order to do this. Once you have created your plumed input use driver to calculate the torsional angles using the following command:

```
plumed driver --plumed plumed.dat --ixyz trajectory-short.xyz
```

If you have done this correctly you should have an output file containing your torsional angles. We can use vmd+GISMO to visualize the relationship between the Ramachandran angles and the atomic configurations. To do this first load the trajectory in VMD:

```
vmd trajectory-short.pdb
```

Then click on Extensions>Analysis>GISMO. A new window should open in this window click on File>Load colvars. You will be asked to select a colvar file. Select the file that was output by the plumed calculation above. Once the file is loaded you should be able to select the labels that you gave to the Ramachandran angles you calculated with plumed. If you do so you will see that this data is plotted in the GISMO window so that you can interact with it and the trajectory.

What can you conclude from this exercise. Do the CV values of the various frames appear in clusters in the plane? Do points in different clusters correspond to structures that look the same or different? Are there similar looking structures clustered together or are they always far apart? What can we conclude about the various basins in the free energy landscape that have been explored in this trajectory? How many are there? Would your estimate be the same if you tried the above estimate with a different pair of Ramachandran angles?

11.38.9.4.4 Dimensionality reduction What we have done in most of the other exercises here is seek out a function that takes as input the position of all the atoms in the system - a $3N$ dimensional vector, where N is the number of atoms. This function then outputs a single number - the value of the collective variable - that tells us where in a low dimensional space we should project that configuration. Problems can arise because this collective-variable function is many-to-one. As you have hopefully seen in the previous exercise markedly different configurations of the protein can actually have quite similar values of a particular Ramachandran angle.
We are going to spend the rest of this session introducing an alternative approach to this business of finding collective variables. In this alternative approach we are going to stop trying to seek out a function that can take any configuration of the atoms (any $3N$-dimensional vector) and find its low dimensional projection on the collective variable axis. Instead we are going to take a set of configurations of the atoms (a set of $3N$-dimensional vectors of atom positions) and try to find a sensible set of projections for these configurations. We already touched on this idea earlier when we looked at paths. Our assumption, when we introduced this idea, was that we could find an ordered set of high-dimensional configurations that represented the transition pathway the system takes as it crossed a barrier and changed between two particularly interesting configurations. Lets say we have a path defined by four reference configurations - this implies that to travel between the configurations at the start and the end of this path you have to pass through configuration 1, then configuration 2, then configuration 3 and then configuration 4. This ordering means that the numbers 1 through 4 constitute sensible projections of these high-dimensional configurations. The numbers 1 through 4 all lie on a single Cartesian axis - a low-dimensional space.
The problem when it comes to applying this idea to the data that we have in the trajectory-short trajectory is that we have no information on the "order" of these points. We have not been told that this trajectory represents the transition between two interesting points in phase space and thus we cannot apply the logic of paths. Hence, to seek out a low dimensional representation we are going to try and find a representation of this data we are going to seek out an isometry between the space containing the $3N$-dimensional vectors of atom positions and some lower-dimensional space. This idea is explained in more detail in the following video .
Let's now generate our isometric embedding. You will need to create a plumed input file that contains the following instructions:

```
CLASSICAL_MDS ...
  ATOMS=1-256
  METRIC=OPTIMAL-FAST
  NLOW_DIM=2
  OUTPUT_FILE=rmsd-embed
... CLASSICAL_MDS
```

You should then run this calculation using the following command:

```
plumed driver --ixyz trajectory-short.xyz --plumed plumed.dat
```

This should generate an output file called rmsd-embed. You should now be able to use VMD+GISMO to visualize this output.
Do the CV values of the various frames appear in clusters in the plane? Do points in different clusters correspond to structures that look the same or different? Are there similar looking structures clustered together or are they always far apart? What can we conclude about the various basins in the free energy landscape that have been explored in this trajectory? How many are there? Do you think this gives you a fuller picture of the trajectory than the ones you obtained by considering Ramachandran angles?

11.38.9.5 Extensions

As discussed in the previous section this approach to trajectory analysis works by calculating distances between pairs of atomic configurations. Projections corresponding to these configurations are then generated in the low dimensional space in a way that tries to preserve these pairwise distances. There are, however, an infinite number of ways of calculating the distance between two high-dimensional configurations. In the previous section we used the RMSD distance but you could equally use the DRMSD distance. You could even try calculating a large number of collective variables for each of the high-dimensional points and seeing how much these all changed. You can use these different types of distances with the CLASSICAL_MDS action that was introduced in the previous section.

If you have time less at the end of the session read the manual for the CLASSICAL_MDS action and see if you can calculate an MDS projection using an alternative definition of the distances between configurations. Some suggestions to try in order of increasing difficulty: DRMSD, how much the full set of 32 Ramachandran angles change and the change in the contact map

11.38.9.6 Further Reading

There is a growing community of people using these ideas to analyze trajectory data. Some start points for investigating their work in more detail are:

- http://epfl-cosmo.github.io/sketchmap/index.html?page=main
- http://www.annualreviews.org/doi/abs/10.1146/annurev-physchem-040412-110006

11.38.10 Belfast tutorial: Umbrella sampling

11.38.10.1 Aims

In the previous lectures we learned how to compute collective variables (CVs) from atomic positions. We will now learn how one can add a bias potential to enforce the exploration of a particular region of the space. We will also see how it is possible to bias CVs so as to enhance the sampling of events hindered by large free-energy barriers and how to analyze this kind of simulation. This technique is known as "umbrella sampling" and can be used in combination with the weighted-histogram analysis method to compute free-energy landscapes. In this tutorial we will use simple collective variables, but the very same approach can be used with any kind of collective variable.

11.38.10.2 Summary of theory

11.38.10.2.1 Biased sampling A system at temperature T samples conformations from the canonical ensemble:

$$P(q) \propto e^{-\frac{U(q)}{k_B T}}$$

. Here q are the microscopic coordinates and k_B is the Boltzmann constant. Since q is a highly dimensional vector, it is often convenient to analyze it in terms of a few collective variables (see Belfast tutorial: Analyzing CVs , Belfast tutorial: Adaptive variables I , and Belfast tutorial: Adaptive variables II). The probability distribution for a CV s is

$$P(s) \propto \int dq e^{-\frac{U(q)}{k_B T}} \delta(s - s(q))$$

This probability can be expressed in energy units as a free energy landscape $F(s)$:

$$F(s) = -k_B T \log P(s)$$

.
Now we would like to modify the potential by adding a term that depends on the CV only. That is, instead of using $U(q)$, we use $U(q) + V(s(q))$. There are several reasons why one would like to introduce this potential. One is to avoid that the system samples some un-desired portion of the conformational space. As an example, imagine that you want to study dissociation of a complex of two molecules. If you perform a very long simulation you will be able to see association and dissociation. However, the typical time required for association will depend on the size of the simulation box. It could be thus convenient to limit the exploration to conformations where the distance between the two molecules is lower than a given threshold. This could be done by adding a bias potential on the distance between the two molecules. Another example is the simulation of a portion of a large molecule taken out from its initial context. The fragment alone could be unstable, and one might want to add additional potentials to keep the fragment in place. This could be done by adding a bias potential on some measure of the distance from the experimental structure (e.g. on root-mean-square deviation).
Whatever CV we decide to bias, it is very important to recognize which is the effect of this bias and, if necessary, remove it a posteriori. The biased distribution of the CV will be

$$P'(s) \propto \int dq e^{-\frac{U(q)+V(s(q))}{k_B T}} \delta(s - s(q)) \propto e^{-\frac{V(s(q))}{k_B T}} P(s)$$

and the biased free energy landscape

$$F'(s) = -k_B T \log P'(s) = F(s) + V(s) + C$$

Thus, the effect of a bias potential on the free energy is additive. Also notice the presence of an undetermined constant C. This constant is irrelevant for what concerns free-energy differences and barriers, but will be important later when we will learn the weighted-histogram method. Obviously the last equation can be inverted so as to obtain the original, unbiased free-energy landscape from the biased one just subtracting the bias potential

$$F(s) = F'(s) - V(s) + C$$

Additionally, one might be interested in recovering the distribution of an arbitrary observable. E.g., one could add a bias on the distance between two molecules and be willing to compute the unbiased distribution of some torsional angle. In this case there is no straightforward relationship that can be used, and one has to go back to the relationship between the microscopic probabilities:

$$P(q) \propto P'(q) e^{\frac{V(s(q))}{k_B T}}$$

The consequence of this expression is that one can obtained any kind of unbiased information from a biased simulation just by weighting every sampled conformation with a weight

$$w \propto e^{\frac{V(s(q))}{k_B T}}$$

That is, frames that have been explored in spite of a high (disfavoring) bias potential V will be counted more than frames that has been explored with a less disfavoring bias potential.

11.38.10.2.2 Umbrella sampling Often in interesting cases the free-energy landscape has several local minima. If these minima have free-energy differences that are on the order of a few times $k_B T$ they might all be relevant. However, if they are separated by a high saddle point in the free-energy landscape (i.e. a low probability region) than the transition between one and the other will take a lot of time and these minima will correspond to metastable states. The transition between one minimum and the other could require a time scale which is out of reach for molecular dynamics. In these situations, one could take inspiration from catalysis and try to favor in a controlled manner the conformations corresponding to the transition state.
Imagine that you know since the beginning the shape of the free-energy landscape $F(s)$ as a function of one CV s. If you perform a molecular dynamics simulation using a bias potential which is exactly equal to $-F(s)$, the biased free-energy landscape will be flat and barrier less. This potential acts as an "umbrella" that helps you to safely cross the transition state in spite of its high free energy.
It is however difficult to have an a priori guess of the free-energy landscape. We will see later how adaptive techniques such as metadynamics (Belfast tutorial: Metadynamics) can be used to this aim. Because of this reason, umbrella sampling is often used in a slightly different manner.
Imagine that you do not know the exact height of the free-energy barrier but you have an idea of where the barrier is located. You could try to just favor the sampling of the transition state by adding a harmonic restraint on the CV, e.g. in the form

$$V(s) = \frac{k}{2}(s - s_0)^2$$

. The sampled distribution will be

$$P'(q) \propto P(q) e^{\frac{-k(s(q)-s_0)^2}{2k_B T}}$$

For large values of k, only points close to s_0 will be explored. It is thus clear how one can force the system to explore only a predefined region of the space adding such a restraint. By combining simulations performed with different values of s_0, one could obtain a continuous set of simulations going from one minimum to the other crossing the transition state. In the next section we will see how to combine the information from these simulations.

11.38.10.2.3 Weighted histogram analysis method Let's now consider multiple simulations performed with restraints located in different positions. In particular, we will consider the i-th bias potential as V_i. The probability to observe a given value of the collective variable s is:

$$P_i(s) = \frac{P(s) e^{-\frac{V_i(s)}{k_B T}}}{\int ds' P(s') e^{-\frac{V_i(s')}{k_B T}}} = \frac{P(s) e^{-\frac{V_i(s)}{k_B T}}}{Z_i}$$

where

$$Z_i = \sum_q e^{-(U(q)+V_i(q))}$$

The likelihood for the observation of a sequence of snapshots $q_i(t)$ (where i is the index of the trajectory and t is time) is just the product of the probability of each of the snapshots. We use here the minus-logarithm of the likelihood (so that the product is converted to a sum) that can be written as

$$\mathcal{L} = -\sum_i \int dt \log P_i(s_i(t)) = \sum_i \int dt \left(-\log P(s_i(t)) + \frac{V_i(s_i(t))}{k_B T} + \log Z_i \right)$$

One can then maximize the likelihood by setting $\frac{\delta \mathcal{L}}{\delta P(\mathbf{s})} = 0$. After some boring algebra the following expression can be obtained

$$0 = \sum_i \int dt \left(-\frac{\delta_{\mathbf{s}_i(t),\mathbf{s}}}{P(\mathbf{s})} + \frac{e^{-\frac{V_i(\mathbf{s})}{k_B T}}}{Z_i} \right)$$

In this equation we aim at finding $P(s)$. However, also the list of normalization factors Z_i is unknown, and they should be found self consistently. Thus one can find the solution as

$$P(\mathbf{s}) \propto \frac{N(\mathbf{s})}{\sum_i \int dt \frac{e^{-\frac{V_i(\mathbf{s})}{k_B T}}}{Z_i}}$$

where Z is self consistently determined as

$$Z_i \propto \int ds' P(\mathbf{s}') e^{-\frac{V_i(\mathbf{s}')}{k_B T}}$$

These are the WHAM equations that are traditionally solved to derive the unbiased probability $P(s)$ by the combination of multiple restrained simulations. To make a slightly more general implementation, one can compute the weights that should be assigned to each snapshot, that turn out to be:

$$w_i(t) \propto \frac{1}{\sum_j \int dt \frac{e^{-\beta V_j(\mathbf{s}_i(t))}}{Z_j}}$$

The normalization factors can in turn be found from the weights as

$$Z_i \propto \frac{\sum_j \int dt e^{-\beta V_i(\mathbf{s}_j(t))} w_j(t)}{\sum_j \int dt w_j(t)}$$

This allows to straightforwardly compute averages related to other, non-biased degrees of freedom, and it is thus a bit more flexible. It is sufficient to pre-compute this factors w and use them to weight every single frame in the trajectory.

11.38.10.3 Learning Outcomes

Once this tutorial is completed students will know how to:

- Setup simulations with restraints.
- Use multiple-restraint umbrella sampling simulations to enhance the transition across a free-energy barrier.
- Analyze the results and compute weighted averages and free-energy profiles.

11.38.10.4 Resources

The tarball for this project contains the following files:

- A gromacs topology (topol.top), configuration (conf.gro), and control file (grompp.mdp). They should not be needed.
- A gromacs binary file (topol.tpr). This is enough for running this system.
- A small C++ program that computes WHAM (wham.cpp) and a script that can be used to feed it (wham.sh)

By working in the directory where the topol.tpr file is stored, one can launch gromacs with the command

```
gmx_mpi mdrun -plumed plumed.dat -nsteps 100000
```

(notice that the -nsteps flag allows the number of steps to be changed).

11.38.10.5 Instructions

11.38.10.5.1 The model system We here use a a model system alanine dipeptide with CHARM27 all atom force field already seen in the previous section.

11.38.10.5.2 Restrained simulations The simplest way in which one might influence a CV is by forcing the system to stay close to a chosen value during the simulation. This is achieved with a restraining potential that PLUMED provides via the directive RESTRAINT. In the umbrella sampling method a bias potential is added so as to favor the exploration of some regions of the conformational space and to disfavor the exploration of other regions [126] . A properly chosen bias potential could allow for example to favor the transition state sampling thus enhancing the transition state for a conformational transition. However, choosing such a potential is not trivial. In a later section we will see how metadynamics can be used to this aim. The simplest way to use umbrella sampling is that to apply harmonic constraints to one or more CVs.
We will now see how to enforce the exploration of a the neighborhood of a selected point the CV space using a RESTRAINT potential.

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Impose an umbrella potential on CV 1 and CV 2
# with a spring constant of 500 kjoule/mol
# at fixed points on the Ramachandran plot
#
restraint-phi: RESTRAINT ARG=phi KAPPA=500 AT=-0.3
restraint-psi: RESTRAINT ARG=psi KAPPA=500 AT=+0.3

# monitor the two variables and the bias potential from the two restraints
PRINT STRIDE=10 ARG=phi,psi,restraint-phi.bias,restraint-psi.bias FILE=COLVAR
```

(see TORSION, RESTRAINT, and PRINT).
The syntax for the command RESTRAINT is rather trivial. The directive is followed by a keyword ARG followed by the label of the CV on which the umbrella potential has to act. The keyword KAPPA determines the hardness of the spring constant and its units are [Energy units]/[Units of the CV]. The additional potential introduced by the UMBRELLA takes the form of a simple harmonic term:

$$U(s) = \frac{k}{2}(x - x_0)^2$$

.
where x_0 is the value specified following the AT keyword. The choice of AT (x_0) is obviously depending on the specific case. KAPPA (k) is typically chosen not to affect too much the intrinsic fluctuations of the system. A typical recipe is $k \approx \frac{k_B T}{\sigma^2}$, where σ^2 is the variance of the CV in a free simulation). In real applications, one must be careful with values of k larger than $\frac{k_B T}{\sigma^2}$ because they could break down the molecular dynamics integrator.
The CVs as well as the two bias potentials are shown in the COLVAR file. For this specific input the COLVAR file has in first column the time, in the second the value of ϕ, in the third the value of ψ, in the fourth the the additional potential introduced by the restraint on ϕ and in the fifth the additional potential introduced by the restraint on ψ.
It may happen that one wants that a given CV just stays within a given range of values. This is achieved in plumed through the directives UPPER_WALLS and LOWER_WALLS that act on specific collective variables and limit the exploration within given ranges.

11.38.10.5.3 Reweighting the results Now consider a simulation performed restraining the variable ϕ:

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
restraint-phi: RESTRAINT ARG=phi KAPPA=10.0 AT=-2
PRINT STRIDE=10 ARG=phi,psi,restraint-phi.bias FILE=COLVAR10
```

and compare the result with the one from a single simulation with no restraint

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
# we use a "dummy" restraint with strength zero here
restraint-phi: RESTRAINT ARG=phi KAPPA=0.0 AT=-2
PRINT STRIDE=10 ARG=phi,psi,restraint-phi.bias FILE=COLVAR0
```

Plot the time dependence of ϕ in the two cases and try to understand the difference.
Now let's try to compute the probability that ψ falls within a given range, say between 1 and 2. This can be done e.g. with this shell script

```
> grep -v \# COLVAR0 | tail -n 80000 |
  awk '{if($3>1 && $3<2)a++; else b++;}END{print a/(a+b)}'
```

Notice that we here considered only the last 80000 frames in the average. Look at the time series for ψ and guess why. Also notice that the script is removing the initial comments. After this trivial pre-processing, the script is just counting how many times the third column (ψ) lies between 1 and 2 and how many times it doesn't. At the end it prints the number of times the variable is between 1 and 2 divided by the total count. The result should be something around 0.40. Now try to do it on trajectories generated with different values of AT. Does the result depend on AT?
We can now try to reweight the result so as to get rid of the bias introduced by the restraint. Since the reweighting factor is just $\exp(\frac{V}{k_BT}$ the script should be modified as

```
> grep -v \# COLVAR10 | tail -n 80000 |
awk '{w=exp($4/2.5); if($3>1 && $3<2)a+=w; else b+=w;}END{print a/(a+b)}'
```

Notice that 2.5 is just k_BT in kj/mol units.
Repeat this calculation for different values of AT. Does the result depend on AT?

11.38.10.5.4 A free-energy landscape One can also count the probability of an angle to be in a precise bin. The logarithm of this quantity, in k_BT units, is the free-energy associated to that bin. There are several ways to compute histograms, either with PLUMED or with external programs. Here I decided to use awk.

```
grep -v \# COLVAR10 | tail -n 80000 |
awk 'BEGIN{
  min1=-3.14159265358979
  max1=+3.14159265358979
  min2=-3.14159265358979
  max2=+3.14159265358979
  nb1=100;
  nb2=100;
  for(i1=0;i1<nb1;i1++) for(i2=0;i2<nb2;i2++) f[i1,i2]=0.0;
}{
  i1=int(($2-min1)*nb1/(max1-min1));
  i2=int(($3-min2)*nb2/(max2-min2));
# we assume the potential is in the last column, and kbT=2.5 kj/mol
  w=exp($NF/2.5);
  f[i1,i2]+=w;
}
END{
  for(i1=0;i1<nb1;i1++){
  for(i2=0;i2<nb2;i2++) print min1+i1/100.0*(max1-min1), min2+i2/100.0*(max2-min2), -2.5*log(f[i1,i2]);
  print "";
}}' > plotme
```

You can then plot the "plotme" file with

```
gnuplot> set pm3d map
gnuplot> splot "plotme"
```

11.38.10.5.5 Combining multiple restraints In the last paragraph you have seen how to reweight simulations done with restraints in different positions to obtain virtually the same result. Let's now see how to combine data from multiple restraint simulations. A possible choice is to download and use the WHAM software here, which is well documented. This is probably the best idea for analyzing a real simulation.
For the sake of learning a bit, we will use a different approach here, namely we will use a short C++ program that implements the weight calculation. Notice that whereas people typically use harmonic restraints in this framework, PLUMED offers a very large variety of bias potentials. For this reason we will keep things as general as possible and use an approach that can be in principle used also to combine simulation with restraint on different variables or with complicated bias potential.
The first step is to generate several simulations with different positions of the restraint, gradually going from say -2 to +2. You can obtain them using e.g. the following script:

```
for AT in -2.0 -1.5 -1.0 -0.5 +0.0 +0.5 +1.0 +1.5 +2.0
do

cat >plumed.dat << EOF
```

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Impose an umbrella potential on CV 1 and CV 2
# with a spring constant of 500 kjoule/mol
# at fixed points on the Ramachandran plot
#
restraint-phi: RESTRAINT ARG=phi KAPPA=40.0 AT=$AT
# monitor the two variables and the bias potential from the two restraints
PRINT STRIDE=10 ARG=phi,psi,restraint-phi.bias FILE=COLVAR$AT
EOF

gmx_mpi mdrun -plumed plumed.dat -nsteps 100000 -x traj$AT.xtc

done
```

Notice that we are here saving separate trajectories for the separate simulation, as well as separate colvar files. In each simulation the restraint is located in a different position. Have a look at the plot of (phi,psi) for the different simulations to understand what is happening.
An often misunderstood fact about WHAM is that data of the different trajectories can be mixed and it is not necessary to keep track of which restraint was used to produce every single frame. Let's get the concatenated trajectory

```
gmx_mpi trjcat -cat -f traj*.xtc -o alltraj.xtc
```

Now we should compute the value of each of the bias potentials on the entire (concatenated) trajectory

```
for AT in -2.0 -1.5 -1.0 -0.5 +0.0 +0.5 +1.0 +1.5 +2.0
do

cat >plumed.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
restraint-phi: RESTRAINT ARG=phi KAPPA=40.0 AT=$AT

# monitor the two variables and the bias potential from the two restraints
PRINT STRIDE=10 ARG=phi,psi,restraint-phi.bias FILE=ALLCOLVAR$AT
EOF

plumed driver --mf_xtc alltraj.xtc --trajectory-stride=10 --plumed plumed.dat

done
```

It is very important that this script is consistent with the one used to generate the multiple simulations above. Now, single files named ALLCOLVARXX will contain on the fourth column the value of the bias centered in XX computed on the entire concatenated trajectory.
Next step is to compile the C++ program that computes weights self-consistently solving the WHAM equations. This is named wham.cpp and can be compiled with

```
g++ -O3 wham.cpp -o wham.x
```

and can be then used through a wrapper script wham.sh as

```
./wham.sh ALLCOLVAR* > colvar
```

The resulting colvar file will contain 3 columns: time, phi, and psi, plus the weights obtained from WHAM written in logarithmic scale. That is, the file will contain $k_B T \log w$.
Try now to use this file to compute the unbiased free-energy landscape as a function of phi and psi. You can use the script that you used earlier to compute histogram.

11.38.10.6 Comments

11.38.10.6.1 How does PLUMED work The fact that when you add a force on the collective variable PLUMED can force the atoms to do something depends on the fact that the collective variables implemented in PLUMED has analytical derivatives. By biasing the value of a single CV one turns to affect the time evolution of the system itself. Notice that some of the collective variables could be implemented without derivatives (either because the developers were lazy or because the CVs cannot be derived). In this case you might want to have a look at the NUMERICAL_DERIVATIVES option.

11.38.10.7 Further Reading

Umbrella sampling method is a widely used technique. You can find several resources on the web, e.g.:

- http://en.wikipedia.org/wiki/Umbrella_sampling

11.38.11 Belfast tutorial: Out of equilibrium dynamics

In plumed you can bring a system in a specific state in a collective variable by means of the MOVINGRESTRAINT directive. This directive is very flexible and allows for a programmed series moving restraints. This technique can be used also to sample multiple events within a single simulation. Here I will explain the concepts of it and show some examples

11.38.11.1 Resources

Here is the tarball with the files referenced in the following .

11.38.11.2 Steered MD

Steered MD (SMD) is often used to drag the system from an initial configuration to a final one by pulling one or more CVs. Most of time the aim of such simulations is to prepare the system in a particular state or produce nice snapshots for a cool movie. All the CVs present in PLUMED can be used in SMD.
In SMD the Hamiltonian of the system H is modified into
H_λ. This new Hamiltonian contains now another new term which now depends on time only via a Harmonic potential centered on a point which moves linear with time

$$\begin{aligned} H_\lambda(X,t) &= H(X) + U_\lambda(X,t) \\ &= H(X) + \frac{k(t)}{2}(s(X) - \lambda(t))^2 \\ &= H(X) + \frac{k(t)}{2}(s(X) - s_0 - vt)^2. \end{aligned}$$

This means that if the k is tight enough the system will follow closely the center of the moving harmonic spring. But be careful, if the spring constant is too hard your equations of motion will now keep up since they are tuned to the fastest motion in your system so if you artificially introduce a higher frequency in your system you'll screw up the dynamics. The same is true for the pulling speed v. As a matter of fact I never encountered the case where I had to lower the time step and I could all the time be happy just by making a softer spring constant or a slower steering speed. Generally, integrators of equations of motion like velocity-Verlet are very tolerant. Note that one can also make the spring constant depend on time and this, as we will see later in the examples is particularly useful to prepare your state.
In simulations, it is more convenient to adopt a situation where you specify only the starting point, the final point of collective variables and the time in which you want to cover the transition. That's why the plumed input is done in such a way.
For example, let's go back to the alanine dipeptide example encountered in The molecule of the day: alanine dipeptide. Let's say that now we want to steer from C_7eq to C_7ax. If you think, just by dragging along the Φ dihedral angle from a value of -1 to a value 1 should be enough to the state of interest. Additionally, it might be important to you not to stress the system too much, so you might want first to increase the k first so to lock the system in $\Phi = -1$, then move it gently to $\Phi = 1$ and then release again your spring constant so to end up to an equilibrated and unconstrained state. This you can program in PLUMED like this

```
# set up two variables for Phi and Psi dihedral angles
# drag this
phi: TORSION ATOMS=5,7,9,15
# this is just to monitor that you end up in the interesting state
psi: TORSION ATOMS=7,9,15,17
# the movingrestraint
restraint: ...
        MOVINGRESTRAINT
        ARG=phi
        AT0=-1.0 STEP0=0      KAPPA0=0
        AT1=-1.0 STEP1=2000   KAPPA1=1000
        AT2=1.0  STEP2=4000   KAPPA2=1000
```

```
        AT3=1.0  STEP3=6000   KAPPA3=0
...
# monitor the two variables and various restraint outputs
PRINT STRIDE=10 ARG=* FILE=COLVAR
```

Please note the syntax of MOVINGRESTRAINT : You need one (or more) argument(s) and a set of steps denote by AT x, STEP x, KAPPA x where x is a incremental starting from 0 that assign the center and the harness of the spring at step STEP x. What happens in between is a linear interpolation of the AT and KAPPA parameters. If those are identical in two consecutive steps then nothing is happening to that parameter. So if you put the same KAPPA and AT value in two different STEP keywords then this will give you an umbrella potential placed in the same point between the two time intervals defined by STEP. Note that you need to run a bit more than 6000 steps because after this your system has no more restraints so the actual thermalization period starts here.
The COLVAR file produced has the following shape

```
#! FIELDS time phi psi restraint.bias restraint.force2 restraint.phi_cntr restraint.phi_work
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
 0.000000 -1.409958 1.246193 0.000000 0.000000 -1.000000 0.000000
 0.020000 -1.432352 1.256545 0.467321 4.673211 -1.000000 0.441499
 0.040000 -1.438652 1.278405 0.962080 19.241592 -1.000000 0.918101
 0.060000 -1.388132 1.283709 1.129846 33.895372 -1.000000 1.344595
 0.080000 -1.360254 1.275045 1.297832 51.913277 -1.000000 1.691475
 ...
```

So we have time, phi, psi and the bias from the moving restraint. Note that at step 0 is zero since we imposed this to start from zero and ramp up in the first 2000 steps up to a value of 2000 kJ/mol/rad^2. It increases immediately since already at step 1 the harmonic potential is going to be increased in bits towards the value of 1000 which is set by KAPPA. The value of restraint.force2 is the squared force (which is a proxy of the force magnitude, despite the direction) on the CV.

$$-\frac{\partial H_\lambda(X,t)}{\partial s} = -(s(X) - s_0 - vt)$$

Note that the actual force on an atom of the system involved in a CV is instead

$$\begin{aligned} -\frac{\partial H_\lambda(X,t)}{\partial x_i} &= -\frac{\partial H_\lambda(X,t)}{\partial s}\frac{\partial s}{\partial x_i} \\ &= -(s(X) - s_0 - vt)\frac{\partial s}{\partial x_i} \end{aligned}$$

This is important because in CVs that have a derivative that change significantly with space then you might have regions in which no force is exerted while in others you might have an enormous force on it. Typically this is the case of sigmoid functions that are used in coordination numbers in which, in the tails, they are basically flat as a function of particle positions. Additionally note that this happens on any force-based enhanced-sampling algorithm so keep it in mind. Very often people miss this aspect and complain either that a CV or a enhanced-sampling method does not work. This is another good reason to use tight spring force so to compensate in the force the lack of derivative from the CV.
The other argument in colvar is restraint.phi_cntr which is the center of the harmonic potential. This is a useful quantity so you may know how close the system is following the center of harmonic potential (more on this below). The last parameter is restraint.phi_work. The work is defined as

$$W = \int_0^{t_s} dt\, \frac{\partial H_\lambda(t)}{\partial t}$$

so this is changing only when the Hamiltonian is changing with time. There are two time dependent contributions in this integral: one can come from the fact that $k(t)$ changes with time and another from the fact that the center of the spring potential is changing with time.

$$W = \int_0^{t_s} dt\, \frac{\partial H_\lambda(t)}{\partial t}$$

$$
\begin{aligned}
&= \int_0^{t_s} dt\, \frac{\partial H_\lambda(t)}{\partial \lambda}\frac{\partial \lambda(t)}{\partial t} + \int_0^{t_s} dt\, \frac{\partial H_\lambda(t)}{\partial k}\frac{\partial k}{\partial t} \\
&= \int_0^{t_s} -k(t)(s(X)-\lambda(t))\frac{\partial \lambda(t)}{\partial t}dt + \int_0^{t_s} \frac{(s(X)-\lambda(t))^2}{2}\frac{\partial k}{\partial t}dt \\
&= \int_0^{t_s} -k(t)(s(X)-\lambda(t))v dt + \int_0^{t_s} \frac{(s(X)-\lambda(t))^2}{2}\frac{\partial k}{\partial t}dt \\
&\simeq \sum_i -k(t_i)(s(X(t_i))-\lambda(t_i))\Delta\lambda(t_i) + \sum_i \frac{(s(X(t_i))-\lambda(t_i))^2}{2}\Delta k(t_i)
\end{aligned}
$$

where we denoted $\Delta\lambda(t_i)$ the difference of the center of the harmonic potential respect to the step before and $\Delta k(t_i)$ is the difference in spring constant respect to the step before. So in the exercised proposed in the first phase you see only the second part of the work since this is the part connected with the spring constant increase. After this phase you see the increase due to the motion of the center and then you later the release of the spring constant.

The work profile as function of time when steering ala dipeptide along the Φ variable.

This you get with gnuplot:

```
pd@plumed:~>gnuplot
gnuplot>  p "COLVAR" u 1:7 w lp
```

Another couple of interesting thing that you can check is

- Is my system finally in the $C7ax$? Plot the two dihedral to have a sense if we are in the right state. You know the target position what should look like, right?

- Is my system moving close to the center of the harmonic potential? This is important and we will see why in a while.

11.38.11.3 Moving on a more complex path

Very often it is useful to use this moving restraint to make a fancier schedule by using nice properties of MOVINGRESTRAINT. For example you can plan a schedule to drive multiple CVs at the same time in specific point of the phase space and also to stop for a while in specific using a fixed harmonic potential. This can be handy in case of an umbrella sampling run where you might want to explore a 1-dimensional landscape by acquiring some statistics in one point and then moving to the next to acquire more statistics. With MOVINGRESTRAINT you can do it in only one file. To give an example of such capabilities, let's say that we want to move from $C7eq$ vertically toward $\Phi = -1.5; \Psi = -1.3$, stop by for a while (e.g. to acquire a statistics that you might need for an umbrella sampling), then moving toward $\Phi = 1.3; \Psi = -1.3$ which roughly corresponds to $C7ax$.
This can be programmed conveniently with MOVINGRESTRAINT by adopting the following schedule

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
# the movingrestraint
restraint: ...
        MOVINGRESTRAINT
        ARG=phi,psi
        AT0=-1.5,1.3  STEP0=0      KAPPA0=0,0
        AT1=-1.5,1.3  STEP1=2000   KAPPA1=1000,1000
        AT2=-1.5,-1.3 STEP2=4000   KAPPA2=1000,1000
        AT3=-1.5,-1.3 STEP3=4000   KAPPA3=1000,1000
        AT4=1.3,-1.3  STEP4=6000   KAPPA4=1000,1000
        AT5=1.3,-1.3  STEP5=8000   KAPPA5=0,0
...
# monitor the two variables and various restraint outputs
PRINT STRIDE=10 ARG=* FILE=COLVAR
```

Note that by adding two arguments for moving restraint, now I am allowed to put two values (separated by comma, as usual for multiple values in PLUMED) and correspondingly two KAPPA values. One for each variable. Please note that no space must be used between the arguments! This is a very common fault in writing the inputs.

By plotting the instantaneous value of the variables and the value of the center of the harmonic potentials we can inspect the pathways that we make the system walk on the Ramachandran plot. (How to do this? Have a look to the header of COLVAR file to plot the right fields)
.png

Plot of the double steering schedule using MOVINGRESTRAINT

11.38.11.4 Why work is important?

The work as we have seen is the cumulative change of the Hamiltonian in time. So it is connected with the change in energy of your system while you move it around. It has also a more important connection with the free energy via the Jarzynski equation which reads

$$\Delta F = -\beta^{-1} \ln \langle \exp^{-\beta W} \rangle$$

This is important and says that potentially you can calculate the free energy even by driving your system very fast and out of equilibrium between two states of your interest. Unfortunately this in practice not possible since the accurate calculation of the quantity $\langle \exp^{-\beta W} \rangle$ has a huge associated error bar since it involves the average of a noisy quantity (the work) that appears in the exponential. So, before going wild with SMD, I want to make a small exercise on how tricky that is even for the smallest system.
Now we run, say 30 SMD run and we calculate the free energy difference by using Jarzynski equality and see how this differs from the average. First note that the average $\langle \exp^{-\beta W} \rangle$ is an average over a number of steered MD runs which start from the same value of CV and reach the final value of CV. So it is important to create initially an ensemble of states which are compatible with a given value of CVs. Let's assume that we can do this by using a restrained MD in a point (say at $Phi = -1.5$). In practice the umbrella biases a bit your distribution and the best situation would be to do this with a flat bottom potential and then choosing the snapshot that correspond to the wanted starting value and start from them.
In the directory JARZ/MAKE_ENSEMBLE you find the script to run. After you generate the constrained ensemble this needs to be translated from xtc format to something that GROMACS is able to read in input, typically a more convenient gro file. To do so just to

```
pd@plumed:~> echo 0 | trjconv_mpi-dp-pl -f traj.xtc -s topol.tpr -o all.gro
pd@plumed:~> awk 'BEGIN{i=1}{if($1=="Generated"){outfile=sprintf("start_%d.gro",i);i++}print >>outfile; if(NF=
```

This will generate a set of numbered gro files. Now copy them in the parallel directory MAKE_STEER. There you will find a script (script.sh) where you can set the number of runs that you want to go for. Just try 20 and let it run. Will take short time. The script will also produce a script_rama.gplt that you can use to visualize all the work performed in a single gnuplot session. Just do:

```
pd@plumed:~>gnuplot
gnuplot> load "script_work.gplt"
```

What you see is something like in Fig.
There are a number of interesting fact here. First you see that different starting points may end with very different work values. Not bad, one would say, since Jarzyinsi equality is saying that we can make an average and everything will be OK. So now calculate the average work by using the following bash one-liner:

```
pd@plumed:~> ntest=20; for i in `seq 1 $ntest` ; do tail -1 colvar_$i | awk '{print $7 }' ; done | awk '{g+=ex
```

For my test, what I get is a value of

```
FREE ENERGY ESTIMATE  17.482  STDEV  7.40342
```

and what this is saying is that the only thing that matters is the lowest work that I sampled. This has such an enormous weight over all the other trajectories that will do so that it will be the only other to count, and all the other do not matter much. So it is a kind of a waste of time. Also the standard deviation is rather high and probably it might well be that you obtain a much better result by using a standard umbrella sampling where you can use profitably most of the statistics. Here you waste most of the statistics indeed, since only the lowest work sampled will matter.
Some important point for doing some further exercises:

- How does the work distribution change if you increase the simulation time? Note that you have to increase both the time in the md.mdp file and in the plumed.dat file.
- How the work change if you now use a softer spring constant? And a harder one?
- In particular, what happens when you have softer spring constant, say 10? This does not look like working? Can you guess what is going on there from an analysis of COLVAR files only?
- Have a look of the trajectories in the Ramachandran plot in case of fast simulations and slow simulation. What can you observe? Is there a correlation between steering speed and how often you can go on the low energy path?

11.38.11.5 Targeted MD

Targeted MD can be seen as a special case of steered MD where the RMSD from a reference structure is used as a collective variable. It can be used for example if one wants to prepare the system so that the coordinates of selected atoms are as close as possible to a target pdb structure.
As an example we can take alanine dipeptide again

```
# set up two variables for Phi and Psi dihedral angles
# these variables will be just monitored to see what happens
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
# creates a CV that measures the RMSD from a reference pdb structure
# the RMSD is measured after OPTIMAL alignment with the target structure
rmsd: RMSD REFERENCE=c7ax.pdb TYPE=OPTIMAL
# the movingrestraint
restraint: ...
        MOVINGRESTRAINT
        ARG=rmsd
        AT0=0.0 STEP0=0      KAPPA0=0
        AT1=0.0 STEP1=5000   KAPPA1=10000
...
# monitor the two variables and various restraint outputs
PRINT STRIDE=10 ARG=* FILE=COLVAR
```

(see TORSION, RMSD, MOVINGRESTRAINT, and PRINT).
Note that RMSD should be provided a reference structure in pdb format and can contain part of the system but the second column (the index) must reflect that of the full pdb so that PLUMED knows specifically which atom to drag where. The MOVINGRESTRAINT bias potential here acts on the rmsd, and the other two variables (phi and psi) are untouched. Notice that whereas the force from the restraint should be applied at every step (thus rmsd is computed at every step) the two torsion angles are computed only every 10 steps. PLUMED automatically detect which variables are used at every step, leading to better performance when complicated and computationally expensive variables are monitored - this is not the case here, since the two torsion angles are very fast to compute. Note that here the work always increase with time and never gets lower which is somewhat surprising if you think that we are moving in another metastable state. One would expect this to bend and give a signal of approaching a minimum like before. Nevertheless consider what you we are doing: we are constraining the system in one specific conformation and this is completely unnatural for a system at 300 kelvin so, even for this small system adopting a specific conformation in which all the heavy atoms are in a precise position is rather unrealistic. This means that this state is an high free energy state.

11.38.12 Belfast tutorial: Metadynamics

11.38.12.1 Aims

The aim of this tutorial is to introduce the users to running a metadynamics simulation with PLUMED. We will set up a simple simulation of alanine dipeptide in vacuum, analyze the output, and estimate free energies from the simulation. We will also learn how to run a well-tempered metadynamics simulation and detect issues related to a bad choice of collective variables.

11.38.12.2 Summary of theory

In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from any local minimum and into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t \to \infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t \to \infty) = -\frac{\Delta T}{T + \Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard molecular dynamics, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T + \Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135].

11.38.12.3 Learning Outcomes

Once this tutorial is completed students will know how to:

- run a metadynamics simulation using PLUMED
- analyze the output of the simulation
- restart a metadynamics simulation
- calculate free energies from a metadynamics simulation
- run a well-tempered metadynamics simulation using PLUMED
- detect issues with the choice of the collective variables

11.38.12.4 Resources

The `tarball` for this project contains the following directories:

- TOPO: it contains the gromacs topology and configuration files to simulate alanine dipeptide in vacuum
- Exercise_1: run a metadynamics simulation with 2 CVs, dihedral angles phi and psi, and analyze the output
- Exercise_2: restart a metadynamics simulation
- Exercise_3: calculate free energies from a metadynamics simulation and monitor convergence
- Exercise_4: run a well-tempered metadynamics simulation with 2 CVs, dihedral angles phi and psi
- Exercise_5: run a well-tempered metadynamics simulation with 1 CV, dihedral psi

11.38.12.5 Instructions

11.38.12.5.1 The model system Here we use as model system alanine dipeptide in vacuum with AMBER99↩ SB-ILDN all-atom force field.

11.38.12.5.2 Exercise 1. Setup and run a metadynamics simulation In this exercise, we will run a metadynamics simulation on alanine dipeptide in vacuum, using as CVs the two backbone dihedral angles phi and psi. In order to run this simulation we need to prepare the PLUMED input file (plumed.dat) as follows.

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in phi and psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJ/mol,
# and width 0.35 rad for both CVs.
#
metad: METAD ARG=phi,psi PACE=500 HEIGHT=1.2 SIGMA=0.35,0.35 FILE=HILLS

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see TORSION, METAD, and PRINT).
The syntax for the command METAD is simple. The directive is followed by a keyword ARG followed by the labels of the CVs on which the metadynamics potential will act. The keyword PACE determines the stride of Gaussian deposition in number of time steps, while the keyword HEIGHT specifies the height of the Gaussian in kJ/mol. For each CVs, one has to specified the width of the Gaussian by using the keyword SIGMA. Gaussian will be written to the file indicated by the keyword FILE.
Once the PLUMED input file is prepared, one has to run Gromacs with the option to activate PLUMED and read the input file:

```
gmx_mpi mdrun -plumed
```

During the metadynamics simulation, PLUMED will create two files, named COLVAR and HILLS. The COLVAR file contains all the information specified by the PRINT command, in this case the value of the CVs every 10 steps of simulation, along with the current value of the metadynamics bias potential. The HILLS file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content:

```
#! FIELDS time phi psi sigma_phi sigma_psi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
```

The line starting with FIELDS tells us what is displayed in the various columns of the HILLS file: the time of the simulation, the value of phi and psi, the width of the Gaussian in phi and psi, the height of the Gaussian, and the bias factor. This quantity is relevant only for well-tempered metadynamics simulation (see Exercise 4. Setup and run a well-tempered metadynamics simulation, part I) and it is equal to 1 in standard metadynamics simulations. We will use the HILLS file later to calculate free-energies from the metadynamics simulation and assess its convergence. For the moment, we can plot the behavior of the CVs during the simulation.
By inspecting Figure belfast-6-metad-fig, we can see that the system is initialized in one of the two metastable states of alanine dipeptide. After a while (t=0.3 ns), the system is pushed by the metadynamics bias potential to visit the other local minimum. As the simulation continues, the bias potential fills the underlying free-energy landscape, and the system is able to diffuse in the entire phase space.
If we use the PLUMED input file described above, the expense of a metadynamics simulation increases with the length of the simulation as one has to evaluate the values of a larger and larger number of Gaussian kernels at every step. To avoid this issue you can store the bias on a grid. In order to use grids, we have to add some additional information to the line of the METAD directive, as follows.

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in phi and psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 0.35 rad for both CVs.
# The bias potential will be stored on a grid
```

```
# with bin size equal to 0.1 rad for both CVs.
# The boundaries of the grid are -pi and pi, for both CVs.
#
METAD ...
LABEL=metad
ARG=phi,psi
PACE=500
HEIGHT=1.2
SIGMA=0.35,0.35
FILE=HILLS
GRID_MIN=-pi,-pi
GRID_MAX=pi,pi
GRID_SPACING=0.1,0.1
... METAD

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

The bias potential will be stored on a grid, whose boundaries are specified by the keywords GRID_MIN and GR↩ ID_MAX. Notice that you should provide either the number of bins for every collective variable (GRID_BIN) or the desired grid spacing (GRID_SPACING). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither GRID_BIN nor GRID_SPACING) and if Gaussian width is fixed PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.

11.38.12.5.3 Exercise 2. Restart a metadynamics simulation If we try to run again a metadynamics simulation using the script above in a directory where a COLVAR and HILLS files are already present, PLUMED will create a backup copy of the old files, and run a new simulation. Instead, if we want to restart a previous simulation, we have to add the keyword RESTART to the PLUMED input file (plumed.dat), as follows.

```
# restart previous simulation
RESTART

# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in phi and psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 0.35 rad for both CVs.
#
metad: METAD ARG=phi,psi PACE=500 HEIGHT=1.2 SIGMA=0.35,0.35 FILE=HILLS

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see RESTART, TORSION, METAD, and PRINT).
In this way, PLUMED will read the old Gaussian kernels from the HILLS file and append the new information to both COLVAR and HILLS files.

11.38.12.5.4 Exercise 3. Calculate free-energies and monitor convergence One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills should be used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.

To calculate the two-dimensional free energy as a function of phi and psi, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called fes.dat in which the free-energy surface as function of phi and psi is calculated on a regular grid. One can modify the default name for the free energy file, as well as the boundaries and bin size of the grid, by using the following options of sum_hills :

```
--outfile - specify the outputfile for sumhills
```

```
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

It is also possible to calculate one-dimensional free energies from the two-dimensional metadynamics simulation. For example, if one is interested in the free energy as a function of the phi dihedral alone, the following command line should be used:

```
plumed sum_hills --hills HILLS --idw phi --kt 2.5
```

The result should look like this:
To assess the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar, apart from a constant offset. The option –stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option –mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --idw phi --kt 2.5 --stride 500 --mintozero
```

one free energy is calculated every 500 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. The resulting plot should look like the following:
To assess the convergence of the simulation more quantitatively, we can calculate the free-energy difference between the two local minimums in the one-dimensional free energy along phi as a function of simulation time. We can use the bash script analyze_FES.sh to integrate the multiple free-energy profiles in the two basins defined by the following intervals in phi space: basin A, -3<phi<-1, basin B, 0.5<phi<1.5.

```
./analyze_FES.sh NFES -3.0 -1.0 0.5 1.5 KBT
```

where NFES is the number of profiles (free-energy estimates at different times of the simulation) generated by the option –stride of sum_hills, and KBT is the temperature in energy units (in this case KBT=2.5).
This analysis, along with the observation of the diffusive behavior in the CVs space, suggest that the simulation is converged.

11.38.12.5.5 Exercise 4. Setup and run a well-tempered metadynamics simulation, part I

In this exercise, we will run a well-tempered metadynamics simulation on alanine dipeptide in vacuum, using as CVs the two backbone dihedral angles phi and psi. To activate well-tempered metadynamics, we need to add two keywords to the line of METAD, which specifies the bias factor and temperature of the simulation. For the first example, we will try a bias factor equal to 6. Here how the PLUMED input file (plumed.dat) should look like:

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in phi and psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 0.35 rad for both CVs.
# Well-tempered metadynamics is activated,
# and the bias factor is set to 6.0
#
metad: METAD ARG=phi,psi PACE=500 HEIGHT=1.2 SIGMA=0.35,0.35 FILE=HILLS BIASFACTOR=6.0 TEMP=300.0

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see TORSION, METAD, and PRINT).
After running the simulation using the instruction described above, we can have a look at the HILLS file. At variance with standard metadynamics, the last two columns of the HILLS file report more useful information. The last column contains the value of the bias factor used, while the last but one the height of the Gaussian, which is rescaled during the simulation following the well-tempered recipe.

```
#! FIELDS time phi psi sigma_phi sigma_psi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
```

```
#! SET min_psi -pi
#! SET max_psi pi
      1.0000     -1.3100    0.0525          0.35           0.35      1.4400      6
      2.0000     -1.4054    1.9742          0.35           0.35      1.4400      6
      3.0000     -1.9997    2.5177          0.35           0.35      1.4302      6
      4.0000     -2.2256    2.1929          0.35           0.35      1.3622      6
```

If we carefully look at the height column, we will notice that in the beginning the value reported is higher than the initial height specified in the input file, which should be 1.2 kJoule/mol. In fact, this column reports the height of the Gaussian rescaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy. In this way, when we will use sum_hills, the sum of the Gaussian kernels deposited will directly provide the free-energy, without further rescaling needed.
We can plot the time evolution of the CVs along with the height of the Gaussian.
The system is initialized in one of the local minimum where it starts accumulating bias. As the simulation progresses and the bias added grows, the Gaussian height is progressively reduced. After a while (t=0.8 ns), the system is able to escape the local minimum and explore a new region of the phase space. As soon as this happens, the Gaussian height is restored to the initial value and starts to decrease again. In the long time, the Gaussian height becomes smaller and smaller while the system diffuses in the entire CVs space.
We can now try a different bias factor and see the effect on the simulation. If we choose a bias factor equal to 1.5, we can notice a faster decrease of the Gaussian height with simulation time, as expected by the well-tempered recipe. We will also conclude from the plot below that this bias factor is not large enough to allow for the system to escape from the initial local minimum in the time scale of this simulation.
Following the procedure described for standard metadynamics in the previous example, we can estimate the free energy as a function of time and monitor the convergence of the simulations using the analyze_FES.sh script. We will do this for the simulation in which the bias factor was set to 6.0. In this case we will notice that the oscillations observed in standard metadynamics are here damped, and the bias potential converges more smoothly to the underlying free-energy landscape, provided that the bias factor is sufficiently high for the free-energy barriers of the system under study to be crossed.

11.38.12.5.6 Exercise 5. Setup and run a well-tempered metadynamics simulation, part II In this exercise, we will study the effect of neglecting a relavant degree of freedom in the choice of metadynamics CVs. We are going to run a well-tempered metadynamics simulation with the psi dihedral alone as CV, using the following PLUMED input file (plumed.dat):

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 0.35 rad.
# Well-tempered metadynamics is activated,
# and the bias factor is set to 10.0
#
metad: METAD ARG=psi PACE=500 HEIGHT=1.2 SIGMA=0.35 FILE=HILLS BIASFACTOR=10.0 TEMP=300.0

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see TORSION, METAD, and PRINT).
Let's look at the HILLS file, in particular at the time series of the CV psi and of the Gaussian height.
From this plot, we observe a nice diffusive behavior of the CV psi when the Gaussian height is already quite small. This happens until t=3 ns, when the CV seems to be stuck for a while in a small region of the CV space. This behavior is typical of a situation in which a slow variable is not included in the set of CV. When something happens in this hidden degree of freedom, the biased CVs typically cannot access anymore regions of the phase space previously visited. To understand this behavior, we need to visualize the time evolution of both phi and psi stored in the COLVAR file.
It is clear from the plot above that what happened around t=3 ns is a jump of the neglected, slow degree of freedom phi from one free-energy basin to another. The dynamics of phi is not biased by any potential, so we need to equilibrate this degree of freedom, i.e. to observe multiple transitions from the two basins, before declaring convergence of our simulation. Or alternatively we can add phi to the set of CVs. This example demonstrates how to declare convergence of a well-tempered metadynamics simulation it is necessary but not sufficient to observe: 1) Gaussian

kernels with very small height, 2) a diffusive behavior in the CV space (as in the first 3 ns of this example).
What we should do is repeating the simulation multiple times starting from different initial conformations. If in all simulations, we observe a diffusive behavior in the biased CV when the Gaussian height is very small, and we obtain very similar free-energy surfaces, then we can be quite confident that our simulations are converged to the right value. If this is not the case, a manual inspection of the runs can help us identifying the missing slow degrees of freedom to add to the set of biased CVs.

11.38.13 Belfast tutorial: Replica exchange I

11.38.13.1 Aims

The aim of this tutorial is to introduce the users to running a parallel tempering (PT) simulation using PLUMED. We will set up a simple simulation of alanine dipeptide in vacuum, analyze the output, calculate free energies from the simulation, and detect problems. We will also learn how to run a combined PT-metadynamics simulation (PTMetaD) and introduce the users to the Well-Tempered Ensemble (WTE).

11.38.13.2 Summary of theory

In Replica Exchange Methods [143] (REM), sampling is accelerated by modifying the original Hamiltonian of the system. This goal is achieved by simulating N non-interacting replicas of the system, each evolving in parallel according to a different Hamiltonian. At fixed intervals, an exchange of configurations between two replicas is attempted. One popular case of REM is PT, in which replicas are simulated using the same potential energy function, but different temperatures. By accessing high temperatures, replicas are prevented from being trapped in local minima. In PT, exchanges are usually attempted between adjacent temperatures with the following acceptance probability:

$$p(i \to j) = min\{1, e^{\Delta_{i,j}^{PT}}\},$$

with

$$\Delta_{i,j}^{PT} = \left(\frac{1}{k_B T_i} - \frac{1}{k_B T_j}\right)\left(U(R_i) - U(R_j)\right),$$

where R_i and R_j are the configurations at temperature T_i and T_j, respectively. The equation above suggests that the acceptance probability is ultimately determined by the overlap between the energy distributions of two replicas. The efficiency of the algorithm depends on the benefits provided by sampling at high-temperature. Therefore, an efficient diffusion in temperature space is required and configurational sampling is still limited by entropic barriers. Finally, PT scales poorly with system size. In fact, a sufficient overlap between the potential energy distributions of neighboring temperatures is required in order to obtain a significant diffusion in temperature. Therefore, the number of temperatures needed to cover a given temperature range scales as the square root of the number of degrees of freedom, making this approach prohibitively expensive for large systems.
PT can be easily combined with metadynamics [144]. In the resulting PTMetaD algorithm (16), N replicas performed in parallel a metadynamics simulation at different temperatures, using the same set of CVs. The PT acceptance probability must be modified in order to account for the presence of a bias potential:

$$\Delta_{i,j}^{PTMetaD} = \Delta_{i,j}^{PT} + \frac{1}{k_B T_i}\left[V_G^i(s(R_i),t) - V_G^i(s(R_j),t)\right] + \frac{1}{k_B T_j}\left[V_G^j(s(R_j),t) - V_G^j(s(R_i),t)\right],$$

where V_G^i and V_G^j are the bias potentials acting on the ith and jth replicas, respectively.
PTMetaD is particularly effective because it compensates for some of the weaknesses of each method alone. The effect of neglecting a slow degree of freedom in the choice of the metadynamics CVs is alleviated by PT, which allows the system to cross moderately high free-energy barriers on all degrees of freedom. On the other hand, the metadynamics bias potential allows crossing higher barriers on a few selected CVs, in such a way that the sampling efficiency of PTMetaD is greater than that of PT alone.
PTMetaD still suffers from the poor scaling of computational resources with system size. This issue may be circumvented by including the potential energy of the system among the set of well-tempered metadynamics CVs. The well-tempered metadynamics bias leads to the sampling of a well-defined distribution called Well-Tempered Ensemble (WTE) [11]. In this ensemble, the average energy remains close to the canonical value but its fluctuations are enhanced in a way that can be tuned, thus improving sampling.
In the so-called PTMetaD-WTE scheme [145], each replica diffusion in temperature space is enhanced by the increased energy fluctuations at all temperatures.

11.38.13.3 Learning Outcomes

Once this tutorial is completed students will know how to:

- run a PT simulation
- analyze the output of the PT simulation and detect problems
- run a PTMetaD simulation
- run a PT and PTMetaD in the WTE

11.38.13.4 Resources

The `tarball` for this project contains the following directories:

- Exercise_1: run a PT simulation using 2 replicas and analyze the output
- Exercise_2: run a PT simulation using 4 replicas and analyze the output
- Exercise_3: run a PTMetaD simulation
- Exercise_4: run a PT, PT-WTE and PTMetaD-WTE simulations

Each directory contains a TOPO sub-directory where topology and configuration files for Gromacs are stored.

11.38.13.5 Instructions

11.38.13.5.1 The model system Here we use as model systems alanine dipeptide in vacuum and water with AMBER99SB-ILDN all-atom force field.

11.38.13.5.2 Exercise 1. Setup and run a PT simulation, part I In this exercise, we will run a PT simulation of alanine dipeptide in vacuum, using only two replicas, one at 300K, the other at 305K. During this simulation, we will monitor the time evolution of the two dihedral angles phi and psi. In order to do that, we need the following PLUMED input file (plumed.dat).

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

# monitor the two variables
PRINT STRIDE=10 ARG=phi,psi FILE=COLVAR
```

(see TORSION, and PRINT).
To submit this simulation with Gromacs, we need the following command line.

```
mpirun -np 2 gmx_mpi mdrun -s TOPO/topol -plumed -multi 2 -replex 100
```

This command will execute two MPI processes in parallel, using the topology files topol0.tpr and topol1.tpr stored in the TOPO sub-directory. These two binary files have been created using the usual Gromacs procedure (see Gromacs manual for further details) and setting the temperature of the two simulations at 300K and 305K in the configuration files. An exchange between the configurations of the two simulations will be attempted every 100 steps.
When Gromacs is executed using the -multi option and PLUMED is activated, the output files produced by PLU↩MED will be renamed and a suffix indicating the replica id will be appended. We will start inspecting the output file COLVAR.0, which reports the time evolution of the CVs at 300K.
The plot above suggests that during the PT simulation the system is capable to access both the relevant basins in the free-energy surface. This seems to suggest that our simulation is converged. We can use the COLVAR.0 and COLVAR.1 along with the tool sum_hills to estimate the free energy as a function of the CV phi. We will do this a function of simulation time to assess convergence more quantitatively, using the following command line:

```
plumed sum_hills --histo COLVAR.0 --idw phi --sigma 0.2 --kt 2.5 --outhisto fes_ --stride 1000
```

As we did in our previous tutorial, we can now use the script analyze_FES.sh to calculate the free-energy difference between basin A (-3.0<phi<-1.0) and basin B (0.5<phi<1.5), as a function of simulation time.
The estimate of the free-energy difference between these two basins seems to be converged. This consideration, along with the observation that the system is exploring all the relevant free-energy basins, might lead us to declare convergence and to state that the difference in free energy between basin A and B is roughly 0 kJoule/mol. Unfortunately, in doing so we would make a big mistake.
In PT simulations we have to be a little bit more careful, and examine the time evolution of each replica diffusing in temperature space, before concluding that our simulation is converged. In order to do that, we need to reconstruct the continuos trajectories of the replicas in temperature from the two files (typically traj0.trr and traj1.trr) which contain the discontinuous trajectories of the system at 300K and 305K. To demux the trajectories, we need to use the following command line:

```
demux.pl md0.log
```

which will create two files, called replica_temp.xvg and replica_index.xvg. We can plot the first file, which reports the temperature index of each configuration at a given time of the simulation. This file is extremely useful, because it allows us monitoring the replicas diffusion in temperature. As we discussed in Summary of theory, in order for the PT algorithm to be effective, we need an efficient diffusion of the replicas in temperature space. In this case, both replicas are rapidly accessing the highest temperature, so there seems not to be any problem on this side.
We can use the second file to reconstruct the continuous trajectories of each replica in temperature:

```
gmx_mpi trjcat -f traj0.trr traj1.trr -demux replica_index.xvg
```

and the following PLUMED input file (plumed_demux.dat) to recalculate the value of the CVs on the demuxed trajectories, typically called 0_trajout.xtc and 1_trajout.xtc.

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

# monitor the two variables
PRINT STRIDE=1 ARG=phi,psi FILE=COLVAR_DEMUX
```

(see TORSION, and PRINT).
For the analysis of the demuxed trajectories, we can use the -rerun option of Gromacs, as follows:

```
# rerun Gromacs on replica 0 trajectory
gmx_mpi mdrun -s TOPO/topol0.tpr -plumed plumed_demux.dat -rerun 0_trajout.xtc
# rename the output
mv COLVAR_DEMUX COLVAR_DEMUX.0
# rerun Gromacs on replica 1 trajectory
gmx_mpi mdrun -s TOPO/topol1.tpr -plumed plumed_demux.dat -rerun 1_trajout.xtc
# rename the output
mv COLVAR_DEMUX COLVAR_DEMUX.1
```

We can now plot the time evolution of the two CVs in the two demuxed trajectories.
This plot shows clearly that each replica is sampling only one of the two basins of the free-energy landscape, and it is never able to cross the high barrier that separates them. This means that what we considered an exhaustive exploration of the free energy landscape at 300K (Figure belfast-7-pt-fig), was instead caused by an efficient exchange of configurations between replicas that were trapped in different free-energy basins. The results of the present simulation were then influenced by the initial conformations of the two replicas. If we had initialized both of them in the same basin, we would have never observed "transitions" to the other basin at 300K.
To declare convergence of a PT simulation, it is thus mandatory to examine the behavior of the replicas diffusing in temperature and check that these are exploring all the relevant basins. Another good practice is repeating the PT simulation starting from different initial conformations, and check that the results are consistent.

11.38.13.5.3 Exercise 2. Setup and run a PT simulation, part II We will now repeat the previous exercise, but with a different setup. The problem with the previous exercise was that replicas were never able to interconvert between the two metastable states of alanine dipeptide in vacuum. The reason was that the highest temperature used (305K) was too low to accelerate barrier crossing in the time scale of the simulation. We will now use 4 replicas at the following temperatures: 300K, 400K, 600K, and 1000K.
We can use the same PLUMED input file described above (plumed.dat), and execute Gromacs using the following command line:

```
mpirun -np 4 gmx_mpi mdrun -s TOPO/topol -plumed -multi 4 -replex 100
```

At the end of the simulation, we first monitor the diffusion in temperature space of each replica. We need to create the replica_temp.xvg and replica_index.xvg:

```
demux.pl md0.log
```

and plot the content of replica_temp.xvg. Here how it should look for Replica 0:
From this analysis, we can conclude that replicas are diffusing effectively in temperature. Now, we need to monitor the space sampled by each replica while diffusing in temperature space and verify that they are interconverting between the different basins of the free-energy landscape. The demux is carried out as in the previous example:

```
trjcat_mpi -f traj0.trr traj1.trr traj2.trr traj3.trr -demux replica_index.xvg
```

and so is the analysis of the demuxed trajectories using the -rerun option of Gromacs and the plumed_demux.dat input file. Here is the space sampled by two of the four replicas:
It is clear that in this case replicas are able to interconvert between the two metastable states, while efficiently diffusing in temperature. We can then calculate the free-energy difference between basin A and B as a function of simulation time at 300K:
and conclude that in this case the PT simulation is converged.

11.38.13.5.4 Exercise 3. Setup and run a PTMetaD simulation In this exercise we will learn how to combine PT with metadynamics. We will use the setup of the previous exercise, and run a PT simulations with 4 replicas at the following temperatures: 300K, 400K, 600K, and 1000K. Each simulation will perform a well-tempered metadynamics calculation, using the dihedral psi alone as CV and a biasfactor equal to 10 (see Exercise 5. Setup and run a well-tempered metadynamics simulation, part II).
Previously, we prepared a single PLUMED input file to run a PT simulation. This was enough, since in that case the same task was performed at all temperatures. Here instead we need to have a slightly different PLUMED input file for each simulation, since we need to use the keyword TEMP to specify the temperature on the line of the METAD directory. We will thus prepare 4 input files, called plumed.0.dat, plumed.1.dat, plumed.2.dat, and plumed.3.dat, with a different value for the keyword TEMP.

Warning

> Notice that the rules for replica suffix are changed with version 2.2. With PLUMED versions 2.0 and 2.1 these files should have been named plumed.dat.0, plumed.dat.1, etc.

Here how plumed.3.dat should look like:

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 0.35 rad.
# Well-tempered metadynamics is activated,
# and the biasfactor is set to 10.0
#
metad: METAD ARG=psi PACE=500 HEIGHT=1.2 SIGMA=0.35 FILE=HILLS BIASFACTOR=10.0 TEMP=1000.0

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see TORSION, METAD, and PRINT).
The PTMetaD simulation is executed in the same way as the PT:

```
mpirun -np 4 gmx_mpi mdrun -s TOPO/topol -plumed -multi 4 -replex 100
```

and it will produce one COLVAR and HILLS file per temperature (COLVAR.0, HILLS.0, ...). The analysis of the results requires what we have learned in the previous exercise for the PT case (analysis of the replica diffusion in temperature and demuxing of each replica trajectory), and the post-processing of a well-tempered metadynamics simulation (FES calculation using sum_hills and convergence analysis).

Since in the previous tutorial we performed the same well-tempered metadynamics simulation without the use of PT (see Exercise 5. Setup and run a well-tempered metadynamics simulation, part II), here we can focus on the differences with the PTMetaD simulation. Let's compare the behavior of the biased variable psi in the two simulations:
In well-tempered metadynamics (left panel), the biased variable psi looked stuck early in the simulation (t=3 ns). The reason was the transition of the other hidden degree of freedom phi from one free-energy basin to the other. In the PTMetaD case (right panel), this seems not to happen. To better appreciate the difference, we can plot the time evolution of the hidden degree of freedom phi in the two cases.
Thanks to the excursions at high temperature, in the PTMetaD simulation the transition of the CV phi between the two basins is accelerate. As a result, the convergence of the reconstructed free energy in psi will be accelerated. This simple exercise demonstrates how PTMetaD can be used to cure a bad choice of metadynamics CVs and the neglecting of slow degrees of freedom.

11.38.13.5.5 Exercise 4. The Well-Tempered Ensemble In this exercise we will learn how to run a PT-WTE and PTMetaD-WTE simulations of alanine dipeptide in water. We will start by running a short PT simulation using 4 replicas in the temperature range between 300K and 400K. We will use a geometric distribution of temperatures, which is valid under the assumption that the specific heat of the system is constant across temperatures. Replicas will thus be simulated at T=300, 330.2, 363.4, and 400K. In this simulation, we will just monitor the two dihedral angles and the total energy of the system, by preparing the following PLUMED input file (plumed_PT.dat):

```
# set up three variables for Phi and Psi dihedral angles
# and total energy
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
ene: ENERGY

# monitor the three variables
PRINT STRIDE=10 ARG=phi,psi,ene FILE=COLVAR_PT
```

(see TORSION, ENERGY, and PRINT).
As usual, the simulation is run for 400ps using the following command:

```
 mpirun -np 4 gmx_mpi mdrun -s TOPO/topol -plumed plumed_PT.dat -multi 4 -replex 100
```

At the end of the run, we want to analyze the acceptance rate between exchanges. This quantity is reported at the end of the Gromacs output file, typically called md.log, and it can be extracted using the following bash command line:

```
grep -A2 "Repl  average probabilities" md0.log
```

From the line above, we will find out that none of the attempted exchanges has been accepted. The reason is that the current setup (4 replicas to cover the temperature range 300-400K) resulted in a poor overlap of the energy distributions at different temperatures. We can easily realize this by plotting the time series of the total energy in the different replicas:
To improve the overlap of the potential energy distributions at different temperatures, we enlarge the fluctuations of the energy by sampling the Well-Tempered Ensemble (WTE). In order to do that, we need to setup a well-tempered metadynamics simulation using energy as CV. In WTE, fluctuations - the standard deviation of the energy time series measured above - will be enhanced by a factor equal to the square root of the biasfactor. In this exercise, we will enhance fluctuations of a factor of 4, thus we will set the biasfactor equal to 16. We need to prepare 4 PLUMED input files (plumed_PTWTE.dat.0, plumed_PTWTE.dat.1,...), which will be identical to the following but for the temperature specified in the line of the METAD directive:

```
# set up three variables for Phi and Psi dihedral angles
# and total energy
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
ene: ENERGY

# Activate metadynamics in ene
# depositing a Gaussian every 250 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 140 kJoule/mol.
# Well-tempered metadynamics is activated,
# and the biasfactor is set to 16.0
#
wte: METAD ARG=ene PACE=250 HEIGHT=1.2 SIGMA=140.0 FILE=HILLS_PTWTE BIASFACTOR=16.0 TEMP=300.0
```

```
# monitor the three variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,ene,wte.bias FILE=COLVAR_PTWTE
```

(see TORSION, ENERGY, METAD, and PRINT).
Here, we use a Gaussian width larger than usual, and of the order of the fluctuations of the potential energy at 300K, as calculated from the preliminary PT run.
We run the simulation following the usual procedure:

```
 mpirun -np 4 gmx_mpi mdrun -s TOPO/topol -plumed plumed_PTWTE.dat -multi 4 -replex 100
```

If we analyze the average acceptance probability in this run:

```
grep -A2 "Repl  average probabilities" md0.log
```

we will notice that now on average 18% of the exchanges are accepted. To monitor the diffusion of each replica in temperature, we can examine the file replica_temp.xvg created by the following command line:

```
demux.pl md0.log
```

This analysis assures us that the system is efficiently diffusing in the entire temperature range and no bottlenecks are present.
Finally, as done in the previous run, we can visualize the time series of the energy CV at all temperatures:
If we compare this plot with the one obtained in the PT run, we can notice that now the enlarged fluctuations caused by the use of WTE lead to a good overlap between energy distributions at different temperatures, thus increasing the exchange acceptance probability. At this point, we can extend our PT-WTE simulation and for example converge the free energy as a function of the dihedral angles phi and psi. Alternatively, we can accelerate sampling of the phi and psi dihedrals by combining PT-WTE with a metadynamics simulation using phi and psi as CVs (PTMetaD-WTE [145]). This can be achieved by preparing 4 PLUMED input files (plumed_PTMetaDWTE.dat.0, plumed_PTMeta↩ DWTE.dat.1,...), which will be identical to the following but for the temperature specified in the two lines containing the METAD directives:

```
# reload WTE bias
RESTART

# set up three variables for Phi and Psi dihedral angles
# and total energy
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
ene: ENERGY

# Activate metadynamics in ene
# Old Gaussians will be reloaded to perform
# the second metadynamics run in WTE.
#
wte: METAD ARG=ene PACE=99999999 HEIGHT=1.2 SIGMA=140.0 FILE=HILLS_PTWTE BIASFACTOR=16.0 TEMP=300.0

# Activate metadynamics in phi and psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJoule/mol,
# and width 0.35 rad for both CVs.
# Well-tempered metadynamics is activated,
# and the biasfactor is set to 6.0
#
metad: METAD ARG=phi,psi PACE=500 HEIGHT=1.2 SIGMA=0.35,0.35 FILE=HILLS_PTMetaDWTE BIASFACTOR=6.0 TEMP=300.0

# monitor the three variables, the wte and metadynamics bias potentials
PRINT STRIDE=10 ARG=phi,psi,ene,wte.bias,metad.bias FILE=COLVAR_PTMetaDWTE
```

(see TORSION, ENERGY, METAD, and PRINT).
These scripts activate two metadynamics simulations. One will use the energy as CV and will reload the Gaussians deposited during the preliminary PT-WTE run. No additional Gaussians on this variable will be deposited during the PTMetaD-WTE simulation, due to the large deposition stride. A second metadynamics simulation will be activated on the dihedral angles. Please note the different parameters and biasfactors in the two metadynamics runs.
The simulation is carried out using the usual procedure:

```
 mpirun -np 4 gmx_mpi mdrun -s TOPO/topol -plumed plumed_PTMetaDWTE.dat -multi 4 -replex 100
```

11.38.14 Belfast tutorial: Replica exchange II and Multiple walkers

11.38.14.1 Aims

The aim of this tutorial is to introduce the users to the use of Bias-Exchange Metadynamics. We will go through the writing of the input files for BEMETA for a simple case of three peptide and we will use METAGUI to to analyze them. We will compare the results of WT-BEMETA and STANDARD-BEMETA with four independent runs on the four Collective Variables. Finally we will use a simplified version of BEMETA that is Multiple Walkers Metadynamics.

11.38.14.1.1 Learning Outcomes Once this tutorial is completed students will:

- Know how to run a Bias-Exchange simulation using PLUMED and GROMACS
- Know how to analyze the results of BEMETA with the help of METAGUI
- Know how to run a Multiple Walker simulation

11.38.14.2 Resources

The `tarball` for this project contains the following files:

- system folder: a starting structure for Val-Ile-Leu system
- WTBX: a run of Well-Tempered Bias-Exchange Metadynamics ready for the analysis

11.38.14.3 Instructions

11.38.14.3.1 Bias-Exchange Metadynamics In all variants of metadynamics the free-energy landscape of the system is reconstructed by gradually filling the local minima with gaussian hills. The dimensionality of the landscape is equal to the number of CVs which are biased, and typically a number of CVs smaller than three is employed. The reason for this is that qualitatively, if the CVs are not correlated among them, the simulation time required to fill the free-energy landscape grows exponentially with the number of CVs. This limitation can be severe when studying complex transformations or reactions in which more than say three relevant CVs can be identified.
A possible technique to overcome this limitation is parallel-tempering metadynamics, Belfast tutorial: Replica exchange I.
A different solution is performing a bias-exchange simulation: in this approach a relatively large number N of CVs is chosen to describe the possible transformations of the system (e.g., to study the conformations of a peptide one may consider all the dihedral angles between amino acids). Then, N metadynamics simulations (replicas) are run on the same system at the same temperature, biasing a different CV in each replica.
Normally, in these conditions, each bias profile would converge very slowly to the equilibrium free-energy, due to hysteresis. Instead, in the bias-exchange approach every fixed number of steps (say 10,000) an exchange is attempted between a randomly selected pair of replicas a and b. The probability to accept the exchange is given by a Metropolis rule:

$$\min\left(1, \exp\left[\beta(V_G^a(x^a,t) + V_G^b(x^b,t) - V_G^a(x^b,t) - V_G^b(x^a,t))\right]\right)$$

where x^a and x^b are the coordinates of replicas a and b and $V_G^{a(b)}(x,t)$ is the metadynamics potential acting on the replica $a(\ b)$. Each trajectory evolves through the high dimensional free energy landscape in the space of the CVs sequentially biased by different metadynamics potentials acting on one CV at each time. The results of the simulation are N one-dimensional projections of the free energy.
In the following example, a bias-exchange simulation is performed on a VIL peptide (zwitterionic form, in vacuum with $\epsilon = 80$, force field amber03), using the four backbone dihedral angles as CVs.
Four replicas of the system are employed, each one biased on a different CV, thus four similar Plumed input files are prepared as follows:

- a common input file in which all the collective variables are defined:

```
MOLINFO STRUCTURE=VIL.pdb
RANDOM_EXCHANGES

cv1: TORSION ATOMS=@psi-1
cv2: TORSION ATOMS=@phi-2
cv3: TORSION ATOMS=@psi-2
cv4: TORSION ATOMS=@phi-3
```

NOTE:

1. By using MOLINFO we can use shortcut to select atoms for dihedral angles (currently @phi, @psi, @omega and @chi1 are available).
2. We use cv# as labels in order to make the output compatible with METAGUI.
3. RANDOM_EXCHANGES generates random exchanges list that are sent back to GROMACS.

- four additional input files that INCLUDE the common input and define the four METAD along the four CVs, respectively.

```
INCLUDE FILE=plumed-common.dat
be: METAD ARG=cv1 HEIGHT=0.2 SIGMA=0.2 PACE=100 GRID_MIN=-pi GRID_MAX=pi
PRINT ARG=cv1,cv2,cv3,cv4 STRIDE=1000 FILE=COLVAR
```

NOTE:

1. in COLVAR we PRINT only the four collective variables, always in the same order in such a way that COLVAR files are compatible with METAGUI
2. if you want to print additional information, like the METAD bias it is possible to use additional PRINT keyword

```
PRINT ARG=cv1,be.bias STRIDE=xxx FILE=BIAS
```

The four replicas start from the same GROMACS topology file replicated four times: topol0.tpr, topol1.tpr, topol2.tpr, topol3.tpr. Finally, GROMACS is launched as a parallel run on 4 cores, with one replica per core, with the command

```
mpirun -np 4 gmx_mpi mdrun -s topol -plumed plumed -multi 4 -replex 2000 >& log &
```

where -replex 2000 indicates that every 2000 molecular-dynamics steps all replicas are randomly paired (e.g. 0-2 and 1-3) and exchanges are attempted between each pair (as printed in the GROMACS $*$.log files).
The same simulation can be run using WELLTEMPERED metadynamics.

11.38.14.3.2 Convergence of the Simulations In the resources for this tutorial you can find the results for a 40ns long Well-Tempered Bias Exchange simulation. First of all we can try to assess the convergence of the simulations by looking at the profiles. In the "convergence" folder there is a script that calculates the free energy from the HILLS.0 file at increasing simulation lengths (i.e. every more 0.8 ns of simulation). The scripts also generate two measures of the evolution of the profiles in time:

1. time-diff.HILLS.0: where it is stored the average deviation between two successive profiles
2. KL.HILLS.0: where it is stored the average deviation between profiles correctly weighted for the free energy of the profiles themselves (Symmetrized Kullback-Lieber divergence)

From both plots one can deduce that after 8 ns the profiles don't change significantly thus suggesting that averaging over the range 8-40ns should result in a accurate profile (we will test this using metagui). Another test is that of looking at the fluctuations of the profiles in a time window instead of looking at successive profiles:

11.38.14.3.3 Bias-Exchange Analysis with METAGUI In principle Bias-Exchange Metadynamics can give as a results only N 1D free energy profiles. But the information contained in all the replicas can be used to recover multidimensional free energy surfaces in $>$=N dimensions. A simple way to perform this analysis is to use METAGUI. METAGUI performs the following operations:

1. Clusters the frames in the trajectories on a multidimensional GRID defined by at least the biased coordinates.
2. By using the 1D free energy profiles and the clustering assigns a free energy to the cluster using a WHAM procedure.
3. Lets the user visualize the clusters.
4. Approximates the kinetics among clusters.

METAGUI (Biarnes et. al) is a plugin for VMD that implements the approach developed by Marinelli et. al 2009. It can be downloaded from the PLUMED website.
In order for the colvar and hills file to be compatible with METAGUI their header must be formatted as following:
COLVAR.#:

```
#! FIELDS time cv1 cv2 cv3 cv4
#! ACTIVE 1 1 A
#! ..
...
```

NOTE:

1. the COLVAR.# files should contain ALL the collective variables employed (all those biased in at least one replica plus those additionally analyzed). They MUST be named cv1 ... cv

   $N.theCOLVAR.\#filesmustbesynchronizedwiththetrajectories, thismeansthatforeachframeinthetrajectoryatt$

2. a keyword #! ACTIVE NBIASEDCV BIASEDCV LABEL is needed, where NBIASEDCV is the number of biased collective variables in that replica (not overall), BIASEDCV is the index of the biased collective variables in that replica (i.e. 1 for the first replica and so on); LABEL is a letter that identify the replica (usually is simply A for the first, B for the second and so on) this is useful if two replicas are biasing the same collective variable:

```
COLVAR.0:
#! FIELDS time cv1 cv2 cv3
#! ACTIVE 1 1 A
#! ..
...
COLVAR.1:
#! FIELDS time cv1 cv2 cv3
#! ACTIVE 1 2 B
#! ..
...
COLVAR.2:
#! FIELDS time cv1 cv2 cv3
#! ACTIVE 1 2 C
#! ..
...
COLVAR.3:
#! FIELDS time cv1 cv2 cv3
#! ACTIVE 0
#! ..
...
```

In the above case Replica 0 biases cv1; replicas 1 and 2 biases cv2 while replica 3 is a neutral (unbiased) replica. cv3 is unbiased in all the replicas.
The ACTIVE keyword must be the FIRST LINE in the HILLS.# files:
HILLS.#:

```
#! ACTIVE 1 1 A
#! FIELDS time cv1 sigma_cv1 height biasf
#! ..
...
```

The above notes hold for the HILLS files as well. In the folder metagui the script check_for_metagui.sh checks if the header of your file is compatible with METAGUI, but remember that this is not enough! Synchronization of COLVAR and trajectory files is also needed. HILLS files can be written with a different frequency but times must be consistent.
NOTE: It is important to copy HILLS files in the metagui folder.

```
./check_for_metagui.sh ../COLVAR.0
```

will tell you that the ACTIVE keyword is missing, you need to modify all the header BEFORE proceeding with the tutorial!!
In the metagui folder there is a metagui.input file:

```
WHAM_EXE        wham_bemeta.x
BASINS_EXE      kinetic_basins.x
KT 2.4900
HILLS_FILE   HILLS.0
```

```
HILLS_FILE   HILLS.1
HILLS_FILE   HILLS.2
HILLS_FILE   HILLS.3
GRO_FILE     VIL.pdb
COLVAR_FILE COLVAR.0 ../traj0.xtc "psi-1"
COLVAR_FILE COLVAR.1 ../traj1.xtc "phi-2"
COLVAR_FILE COLVAR.2 ../traj2.xtc "psi-2"
COLVAR_FILE COLVAR.3 ../traj3.xtc "phi-3"
TRAJ_SKIP 10
CVGRID 1  -3.1415 3.1415 15 PERIODIC
CVGRID 2  -3.1415 3.1415 15 PERIODIC
CVGRID 3  -3.1415 3.1415 15 PERIODIC
CVGRID 4  -3.1415 3.1415 15 PERIODIC
ACTIVE 4 1 2 3 4
T_CLUSTER 0.
T_FILL    8000.
DELTA 4
GCORR 1
TR_N_EXP 5
```

where are defined the temperature in energy units, the place where to find COLVAR, HILLS and trajectory files. A reference gro or pdb file is needed to load the trajectories. The definition of the ranges and the number of bins for the available collective variables.
Now let's start with the analysis:

1. run VMD and load metagui
2. in metagui load the metagui.input file belfast-8-mg1-fig
3. In the left section of the interface "load all" the trajectories
4. Find the Microstates

In order to visualize the microstate it is convenient to align all the structures using the VMD RMSD Trajectory tool that can be found in Extensions->Analysis.
One or more microstates can be visualized by selecting them and clicking show.
You can sort the microstates using the column name tabs, for example by clicking on size the microstates will be ordered from the larger to the smaller. If you look at the largest one it is possible to observe that by using the four selected collective variables the backbone conformation of the peptide is well defined while the side chains can populate different rotameric states.
The equilibrium time in the analysis panel should be such that by averaging over the two halves of the remind of the simulation the profiles are the same (i.e the profile averaged between Teq and Teq+(Ttot-Teq)/2 should be the same of that averaged from Teq+(Ttot-Teq)/2 and Ttot). By clicking on COMPUTE FREE ENERGIES, the program will first generate the 1D free energy profiles from the HILLS files and then run the WHAM analysis on the microstates. Once the analysis is done it is possible to visually check the convergence of the 1D profiles one by one by clicking on the K buttons next to the HILLS.# files. The BLUE and the RED profiles are the two profiles just defined, while the GREEN is the average of the two. Now it is possible for example to sort the microstates as a function of the free energy and save them by dumping the structures for further analysis.
If you look in the metagui folder you will see a lot of files, some of them can be very useful:
metagui/MICROSTATES: is the content of the microstates list table metagui/WHAM_RUN/VG_HILLS.#: are the opposite of the free energies calculated from the hills files metagui/WHAM_RUN/*.gnp: are gnuplot input files to plot the VG_HILLS.# files (i.e. gnuplot -> load "convergence..") metagui/WHAM_RUN/FES: is the result of the WHAM, for each cluster there is its free energy and the error estimate from WHAM

```
gnuplot> plot [0:40]'FES' u 2:3
```

plots the microstate error in the free energy estimate as a function of the microstates free energy. Finally in the folder metagui/FES there is script to integrate the multidimensional free energy contained in the MICROSTATES files to a 2D FES as a function of two of the used CV. To use it is enough to copy the MICROSTATES file in FES:

```
cp MICROSTATES FES/FES.4D
```

and edit the script to select the two columns of MICROSTATES on which show the integrated FES.

11.38.14.3.4 Multiple Walker Metadynamics Multiple Walker metadynamics is the simplest way to parallelize a metadynamics calculation: multiple simulation of the same system are run in parallel using metadynamics on the same set of collective variables. The deposited bias is shared among the replicas in such a way that the history dependent potential depends on the whole history.
We can use the same common input file defined above and then we can define four metadynamics bias in a similar way of what was done above for bias-exchange but now all the biases are defined on the same collective variables:

```
plumed.#.dat
INCLUDE FILE=plumed-common.dat

METAD ...
LABEL=mw
ARG=cv2,cv3
SIGMA=0.3,0.3
HEIGHT=0.2
PACE=100
BIASFACTOR=8
GRID_MIN=-pi,-pi
GRID_MAX=pi,pi
WALKERS_MPI
... METAD

PRINT ARG=cv1,cv2,cv3,cv4 STRIDE=1000 FILE=COLVAR
```

and the simulation can be run in a similar way without doing exchanges:

```
mpirun -np 4 gmx_mpi mdrun -s topol -plumed plumed -multi 4  >& log &
```

alternatively Multiple Walkers can be run as independent simulations sharing via the file system the biasing potential, this is useful because it provides a parallelization that does not need a parallel code. In this case the walkers read with a given frequency the Gaussian kernels deposited by the others and add them to their own METAD.

11.38.14.4 Reference

This tutorial is freely inspired to the work of Biarnes et al.
More materials can be found in

1. Marinelli, F., Pietrucci, F., Laio, A. & Piana, S. A kinetic model of TRP-cage folding from multiple biased molecular dynamics simulations. PLoS Comput. Biol. 5, e1000452 (2009).
2. Biarnes, X., Pietrucci, F., Marinelli, F. & Laio, A. METAGUI. A VMD interface for analyzing metadynamics and molecular dynamics simulations. Comput. Phys. Commun. 183, 203–211 (2012).
3. Baftizadeh, F., Cossio, P., Pietrucci, F. & Laio, A. Protein folding and ligand-enzyme binding from bias-exchange metadynamics simulations. Current Physical Chemistry 2, 79–91 (2012).
4. Granata, D., Camilloni, C., Vendruscolo, M. & Laio, A. Characterization of the free-energy landscapes of proteins by NMR-guided metadynamics. Proc. Natl. Acad. Sci. U.S.A. 110, 6817–6822 (2013).
5. Raiteri, P., Laio, A., Gervasio, F. L., Micheletti, C. & Parrinello, M. Efficient reconstruction of complex free energy landscapes by multiple walkers metadynamics. J. Phys. Chem. B 110, 3533–3539 (2006).

11.38.15 Belfast tutorial: NMR restraints

11.38.15.1 Aims

This tutorial is about the use of experimental data, in particular NMR data, either as collective variables or as replica-averaged restraints in MD simulations. While the first is a just a simple extension of what we have been already doing in previous tutorials, the latter is an approach that can be used to increase the quality of a force-field in describing the properties of a specific system.

11.38.15.1.1 Learning Outcomes Once this tutorial is completed students will:

- know why and how to use experimental data to define a collective variable
- know why and how to use experimental data as replica-averaged restraints in MD simulations

11.38.15.2 Resources

The tarball for this project contains the following:

- system: the files use to generate the topol?.tpr files of the first and second example (the setup is for simulations in vacuum)
- first: an example on the use of chemical shifts as a collective variable
- second: an example on the use of chemical shifts as replica-averaged restraints
- third: an example on the use of RDCs (calculated with the theta-method) as replica-averaged restraints

11.38.15.3 Instructions

11.38.15.3.1 Experimental data as Collective Variables In the former tutorials it has been often discussed the possibility of measuring a distance with respect to a structure representing some kind of state for a system, i.e. Belfast tutorial: Out of equilibrium dynamics. An alternative possibility is to use as a reference a set of experimental data that represent a state and measure the current deviation from the set. In plumed there are currently implemented the following NMR experimental observables: Chemical Shifts (only for proteins) CS2BACKBONE, NOE distances, JCOUPLING, PRE intensities, and Residual Dipolar couplings/pseudo-contact shifts RDC.
In the following we will write the CS2BACKBONE collective variable similar to the one used in Granata et al. (2013) (while the collective variable is still proportional to the square sum of the deviation of the calculated and experimental chemical shifts divided by a typical error, the exact definition is not the same. The sum is not done anymore with a flat bottom difference and the error used are not the one published, so the exact result of the scoring function can be different).
As a general rule, when using CS2BACKBONE or other experimental restraints it is better to increase the accuracy of the constraint algorithm due to the increased strain on the bonded structure. In the case of GROMACS it is safer to use lincs-iter=2 and lincs-order=6.

```
prot: GROUP ATOMS=1-862
cs: CS2BACKBONE ATOMS=prot DATA=data NRES=56 CAMSHIFT
PRINT ARG=cs FILE=COLVAR STRIDE=100

ENDPLUMED
```

In this case the chemical shifts are those measured for the native state of the protein and can be used, together with other CVs and Bias-Exchange Metadynamics, to guide the system back and forth from the native structure. The experimental chemical shifts are in six files inside the "data/" folder (see first example in the resources tarball), one file for each nucleus. A 0 chemical shift is used where a chemical shift doesn't exist (i.e. CB of GLY) or where it has not been assigned. Additionally the data folder contains:

- camshift.db: this file is a parameter file for camshift, it is a standard file needed to calculate the chemical shifts from a structure
- template.pdb: this is a pdb file for the protein we are simulating (i.e. editconf -f conf.gro -o template.pdb) where atoms are ordered in the same way in which are included in the main code and again it is used to map the atom in plumed with those in almost.

This example can be executed as

```
gmx_mpi mdrun -s topol -plumed plumed
```

11.38.15.3.2 Replica-Averaged Restrained Simulations NMR data, as all the equilibrium experimental data, are the result of a measure over an ensemble of structures and over time. In principle a "perfect" molecular dynamics simulations, that is a simulations with a perfect force-field and a perfect sampling can predict the outcome of an experiments in a quantitative way. Actually in most of the cases obtaining a qualitative agreement is already a fortunate outcome. In order to increase the accuracy of a force field in a system dependent manner it is possible to add to the force-field an additional term based on the agreement with a set of experimental data. This agreement is not enforced as a simple restraint because this would mean to ask the system to be always in agreement with all the experimental data at the same time, instead the restraint is applied over an AVERAGED COLLECTIVE VARIABLE where the average is performed over multiple independent simulations of the same system in the same conditions. In this way the is not a single replica that must be in agreement with the experimental data but they should be

in agreement on average. It has been shown that this approach is equivalent to solving the problem of finding a modified version of the force field that will reproduce the provided set of experimental data without any additional assumption on the data themselves.
The second example included in the resources show how the amber force field can be improved in the case of protein domain GB3 using the native state chemical shifts a replica-averaged restraint. By the fact that replica-averaging needs the use of multiple replica simulated in parallel in the same conditions it is easily complemented with BIAS-EXCHANGE or MULTIPLE WALKER metadynamics to enhance the sampling.

```
prot: GROUP ATOMS=1-862
cs: CS2BACKBONE ATOMS=prot DATA=data NRES=56
enscs: ENSEMBLE ARG=(cs\.hn_.*),(cs\.nh_.*),(cs\.ca_.*),(cs\.cb_.*),(cs\.co_.*),(cs\.ha_.*)
stcs: STATS ARG=enscs.* SQDEVSUM PARARG=(cs\.exphn_.*),(cs\.expnh_.*),(cs\.expca_.*),(cs\.expcb_.*),(cs\.expco
res: RESTRAINT ARG=stcs.sqdevsum AT=0. KAPPA=0. SLOPE=12

PRINT ARG=(cs\.hn_.*),(cs\.nh_.*),(cs\.ca_.*),(cs\.cb_.*),(cs\.co_.*),(cs\.ha_.*) FILE=CS STRIDE=1000
PRINT ARG=res.bias FILE=COLVAR STRIDE=10

ENDPLUMED
```

with respect to the case in which chemical shifts are used to define a standard collective variable, in this case CS2BACKBONE is a collective variable with multiple components, that are all the back calculated chemical shifts, plus all the relative experimental values. The keyword function ENSEMBLE tells plumed to calculate the average of the arguments over the replicas (i.e. 4 replicas) and the function STATS compare the averaged back calculated chemical shifts with the experimental values and calculates the sum of the squared deviation. On this latter number it is possible to apply a linear RESTRAINT (because the variable is already a sum of squared differences) that is the new term we are adding to the underlying force field.
This example can be executed as

```
mpiexec -np 4 gmx_mpi mdrun -s topol -plumed plumed -multi 4
```

The third example show how RDC (calculated with the theta-methods) can be employed in the same way, in this case to describe the native state of Ubiquitin. In particular it is possible to observe how the RDC averaged restraint applied on the correlation between the calculated and experimental N-H RDCs result in the increase of the correlation of the RDCs for other bonds already on a very short time scale.

```
RDC ...
GYROM=-72.5388
SCALE=0.001060
ADDCOUPLINGS
LABEL=nh
ATOMS1=20,21 COUPLING1=8.17
ATOMS2=37,38 COUPLING2=-8.271
ATOMS3=56,57 COUPLING3=-10.489
ATOMS4=76,77 COUPLING4=-9.871
#continue....
```

In this input the first four N-H RDCs are defined.
This example can be executed as

```
mpiexec -np 8 gmx_mpi mdrun -s topol -plumed plumed -multi 8
```

11.38.15.4 Reference

1. Granata, D., Camilloni, C., Vendruscolo, M. & Laio, A. Characterization of the free-energy landscapes of proteins by NMR-guided metadynamics. Proc. Natl. Acad. Sci. U.S.A. 110, 6817–6822 (2013).
2. Cavalli, A., Camilloni, C. & Vendruscolo, M. Molecular dynamics simulations with replica-averaged structural restraints generate structural ensembles according to the maximum entropy principle. J. Chem. Phys. 138, 094112 (2013).
3. Camilloni, C., Cavalli, A. & Vendruscolo, M. Replica-Averaged Metadynamics. JCTC 9, 5610–5617 (2013).
4. Roux, B. & Weare, J. On the statistical equivalence of restrained-ensemble simulations with the maximum entropy method. J. Chem. Phys. 138, 084107 (2013).
5. Boomsma, W., Lindorff-Larsen, K. & Ferkinghoff-Borg, J. Combining Experiments and Simulations Using the Maximum Entropy Principle. PLoS Comput. Biol. 10, e1003406 (2014).
6. Camilloni, C. & Vendruscolo M. A Tensor-Free Method for the Structural and Dynamical Refinement of Proteins using Residual Dipolar Couplings. J. PHYS. CHEM. B 119, 653 (2015).

11.38.16 Belfast tutorial: Steinhardt Parameters

11.38.16.1 Aims

This tutorial is concerned with the collective variables that we use to study order/disorder transitions such as the transition between the solid and liquid phase. In this tutorial we will look at the transition from solid to liquid as this is easier to study using simulation. The CVs we will introduce can, and have, been used to study the transition from liquid to solid. More information can be found in the further reading section.
You will need to ensure that you have compiled PLUMED with the crystallisation module enabled in order to complete this tutorial. To learn how to activate this module consult the information in the manual about the List of modules.

11.38.16.1.1 Learning Outcomes

Once this tutorial is completed students will:

- Know of the existence of the Steinhardt Parameters and know how to create plumed input files for calculating them.
- Appreciate that the Steinhardt Parameter for a particular atom is a vector quantity and that you can thus measure local order in a material by taking dot products of the Steinhardt Parameter vectors of adjacent atoms. Students will know how to calculate such quantities using plumed.

11.38.16.2 Resources

The `tarball` for this project contains the following files:

- in : An input file for the simplemd code that forms part of plumed
- input.xyz : A configuration file for Lennard-Jones solid with an FCC solid structure

11.38.16.3 Instructions

11.38.16.3.1 Simplemd

For this tutorial we will be using the MD code that is part of plumed - simplemd. This code allows us to do the simulations of `Lennard-Jones` that we require here but not much else. We will thus start this tutorial by doing some simulations with this code. You should have two files from the tarball, the first is called input.xyz and is basically an xyz file containing the positions of the atoms. The second meanwhile is called in and is the input to simplemd. If you open the file the contents should look something like this:

```
inputfile input.xyz
outputfile output.xyz
temperature 0.2
tstep 0.005
friction 1
forcecutoff 2.5
listcutoff  3.0
nstep 50000
nconfig 100 trajectory.xyz
nstat   10 energies.dat
```

Having run some molecular dynamics simulations in the past it should be pretty obvious what each line of the file does. One thing that might be a little strange is the units. Simplemd works with Lennard-Jones units so energy is in units of ϵ and length is in units of σ. This means that temperature is in units of $\frac{k_B T}{\epsilon}$, which is why the temperature in the above file is apparently so low.
Use simplemd to run 50000 step calculations at 0.2, 0.8 and 1.2 $\frac{k_B T}{\epsilon}$. (N.B. You will need an empty file called plumed.dat in order to run these calculations.) Visualize the trajectory output by each of your simulations using V↩
MD. Please be aware that simplemd does not wrap the atoms into the cell box automatically. If you are at the tutorial we have resolved this problem by making it so that if you press w when you load all the atoms they will be wrapped into the box. At what temperatures did the simulations melt? What then is the melting point of the Lennard-Jones potential at this density?

11.38.16.3.2 Coordination Numbers

At the end of the previous section you were able to make very a sophisticated judgement about whether or not the arrangement of atoms in your system was solid-like or liquid-like by simply looking at the configuration. The aim in the rest of this tutorial is to see if we can derive collective variables that are able to make an equally sophisticated judgement. For our first attempt lets use a CV which we have encountered

elsewhere, the COORDINATIONNUMBER. Rerun the calculations from the previous section but at variance with the previous section use the plumed input file below instead of a empty file. This input will ensure that the average COORDINATIONNUMBER of the atoms in your system is computed and printed.

```
cc: COORDINATIONNUMBER SPECIES=1-108 SWITCH={RATIONAL D_0=1.3 R_0=0.2 D_MAX=3.0} MEAN
PRINT ARG=cc.* FILE=colvar STRIDE=100
```

Rerun the simplemd simulations described at the end of the previous section. Is the average coordination number good at differentiating between solid and liquid configurations? Given your knowledge of physics/chemistry is this result surprising?

11.38.16.3.3 Steinhardt parameter The solid and liquid phases of a material are both relatively dense so the result at the end of the last section - the fact that the coordination number is not particularly good at differentiating between them - should not be that much of a surprise. As you will have learned early on in your scientific education when solids melt the atoms rearrange themselves in a much less ordered fashion. The bonds between them do not necessarily break it is just that, whereas in a the solid the bonds were all at pretty well defined angles to each other, in a liquid the spread of bond angles is considerably larger. To detect the transition from solid to liquid what we need then is a coordinate that is able to recognize whether or not the geometry in the coordination spheres around each of the atoms in the system are the same or different. If these geometries are the same the system is in a solid-like configuration, whereas if they are different the system is liquid-like. The Steinhardt parameters Q3, Q4 and Q6 are coordinates that allow us to examine the coordination spheres of atoms in precisely this way. The way in which these coordinates are able to do this is explained in the video .
Repeat the calculations from the end of the previous section but using the plumed.dat file below.

```
cc: COORDINATIONNUMBER SPECIES=1-108 SWITCH={RATIONAL D_0=1.3 R_0=0.2 D_MAX=3.0} MEAN
q6: Q6 SPECIES=1-108 SWITCH={RATIONAL D_0=1.3 R_0=0.2 D_MAX=3.0} MEAN
PRINT ARG=cc.*,q6.* FILE=colvar STRIDE=100
```

This will output the average Q6 parameter and the average COORDINATIONNUMBER. In the Steinhardt parameter implementation in plumed the set of atoms in the coordination sphere of a particular atom are defined using a continuous switching function. Is the average Q6 parameter able to differentiate between the solid and liquid states?

11.38.16.3.4 Local versus Global At the end of the previous section you showed that the average Q6 parameter for a system of atoms is able to tell you whether or not that collection of atoms is in a solid-like or liquid-like configuration. In this section we will now ask the question - can the Steinhardt parameter always, unambiguously tell us whether particular tagged atoms are in a solid-like region of the material or whether they are in a liquid-like region of the material? This is an important question that might come up if we are looking at nucleation of a solid from the melt. Our question in this context reads - how do we unambiguously identify those atoms that are in the crystalline nucleus? A similar question would also come up when studying an interface between the solid and liquid phases. In this guise we would be asking about the extent of interface; namely, how far from the interface do we have to go before we can start thinking of the atoms as just another atom in the solid/liquid phase?
With these questions in mind our aim is to look at the distribution of values for the Q6 parameters of atoms in our system of Lennard-Jones have. If Q6 is able to unambiguously tell us whether or not an atom is in a solid-like pose or a liquid-like pose then there should be no overlap in the distributions obtained for the solid and liquid phases. If there is overlap, however, then we cannot use these coordinates for the applications described in the previous paragraph. To calculate these distributions you will need to run two simulations - one at 0.2 $\frac{k_B T}{\epsilon}$ and one at 1.2 $\frac{k_B t}{\epsilon}$ using now the following PLUMED input file:

```
q6: Q6 SPECIES=1-108 SWITCH={RATIONAL D_0=1.3 R_0=0.2 D_MAX=3.0}
hh: HISTOGRAM DATA=q6 GRID_MIN=0 GRID_MAX=1.0 GRID_BIN=100 BANDWIDTH=0.05 STRIDE=100
DUMPGRID GRID=hh FILE=histo STRIDE=50000
```

Do the distributions of Q6 parameters in the solid and liquid phase overlap?

11.38.16.3.5 Local Steinhardt parameters Hopefully you saw that the distribution of Q6 parameters for the solid and liquid phase overlap at the end of the previous section. Again this is not so surprising - as you go from solid to liquid the distribution of the geometries of the coordination spheres widens. The system is now able to arrange the atoms in the coordination spheres in a much wider variety of different poses. Importantly, however, the fact that an atom is in a liquid does not preclude it from having a very-ordered, solid-like coordination environment. As

such in the liquid state we will find the occasional atom with a high value for the Q6 parameter. Consequently, an ordered coordination environment does not always differentiate solid-like atoms from liquid-like atoms. The major difference is in the liquid the ordered atoms will always be isolated. That is to say in the liquid atoms with an ordered coordination will always be surrounded by atoms with a disordered coordination sphere. By contrast in the solid each ordered atom will be surrounded by further ordered atoms. This observation forms the basis of the local Steinhardt parameters and the locally averaged Steinhardt parameters that are explained in this video .
Lets use plumed to calculate the distribution of LOCAL_Q6 parameters in the solid and liquid phases. We can do this using the input file shown below:

```
q6: Q6 SPECIES=1-108 SWITCH={RATIONAL D_0=1.3 R_0=0.2 D_MAX=3.0}
lq6: LOCAL_Q6 SPECIES=q6 SWITCH={RATIONAL D_0=1.3 R_0=0.2 D_MAX=3.0}
hh: HISTOGRAM DATA=lq6 GRID_MIN=0 GRID_MAX=1.5 GRID_BIN=150 BANDWIDTH=0.05 STRIDE=10
DUMPGRID GRID=hh FILE=histo STRIDE=50000
```

Do the distributions of LOCAL_Q6 parameter for the solid and liquid phases overlap?
Lectner and Dellago have designed an alternative to the LOCAL_Q6 parameter that is based on taking the LOCAL_AVERAGE of the Q6 parameter around a central atom. This quantity can be calculated using plumed. If you have time try to use the manual to work out how.

11.38.16.4 Further Reading

There is a substantial literature on simulation of nucleation. Some papers are listed below but this list is far from exhaustive.

- F. Trudu, D. Donadio and M. Parrinello Freezing of a Lennard-Jones Fluid: From Nucleation to Spinodal Regime , Phys. Rev. Lett. 97 105701 (2006)
- D. Quigley and P. M. Rodger A metadynamics-based approach to sampling crystallization events , Mol. Simul. 2009
- W. Lechner and C. Dellago Accurate determination of crystal structures based on averaged local bond order parameters J. Chem. Phys 129 114707 (2008)
- B. Cheng, G. A. Tribello and M. Ceriotti Solid-liquid interfacial free energy out of equilibrium

11.38.17 Cambridge tutorial

Authors

Max Bonomi and Carlo Camilloni, part of the tutorial is based on other tutorials.

Date

March 4, 2015

This document describes the PLUMED tutorial held for the Computational Biology MPhil, University of Cambridge, March 2015. The aim of this tutorial is to learn how to use PLUMED to run well-tempered and bias-exchange simulations and how to run replica-averaged metadynamics to incorporate experimental data into your simulation. Although the presented input files are correct, the users are invited to refer to the literature to understand how the parameters of enhanced sampling methods should be chosen in a real application.
Users are also encouraged to follow the links to the full PLUMED reference documentation and to wander around in the manual to discover the many available features and to do the other, more complete, tutorials. Here we are going to present only a very narrow selection of methods.
Users are expected to write PLUMED input files based on the instructions below.

11.38.17.1 Alanine dipeptide: our toy model

In this tutorial we will play with alanine dipeptide (see Fig. cambridge-1-ala-fig). This rather simple molecule is useful to make many benchmark that are around for data analysis and free energy methods. It is a rather nice example since it presents two metastable states separated by a high (free) energy barrier. Here metastable states are intended as states which have a relatively low free energy compared to adjacent conformation. It is conventional use to show the two states in terms of Ramachandran dihedral angles, which are denoted with Φ and Ψ in Fig. cambridge-1-transition-fig .

11.38.17.2 Exercise 1. Metadynamics

11.38.17.2.1 Resources The tarball for this project contains all the files needed to run this exercise.

11.38.17.2.2 Summary of theory In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t\to\infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t\to\infty) = -\frac{\Delta T}{T+\Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard molecular dynamics, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T+\Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135].

11.38.17.2.3 Setup, run, and analyze a well-tempered metadynamics simulation In this exercise, we will run a well-tempered metadynamics simulation on alanine dipeptide in vacuum, using as CVs the two backbone dihedral angles phi and psi.
In order to run this simulation we need to prepare the PLUMED input file (plumed.dat) as follows.

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate metadynamics in phi and psi
# depositing a Gaussian every 500 time steps,
# with height equal to 1.2 kJ/mol,
# and width 0.35 rad for both CVs.
# Well-tempered metadynamics is activated,
# and the biasfactor is set to 6.0
#
metad: METAD ARG=phi,psi PACE=500 HEIGHT=1.2 SIGMA=0.35,0.35 FILE=HILLS BIASFACTOR=6.0

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see TORSION, METAD, and PRINT).
The syntax for the command METAD is simple. The directive is followed by a keyword ARG followed by the labels of the CVs on which the metadynamics potential will act. The keyword PACE determines the stride of Gaussian deposition in number of time steps, while the keyword HEIGHT specifies the height of the Gaussian in kJ/mol. To run well-tempered metadynamics, we need two additional keywords BIASFACTOR and TEMP, which specify how fast the bias potential is decreasing with time and the temperature of the simulation, respectively. For each CVs, one has to specify the width of the Gaussian by using the keyword SIGMA. Gaussian will be written to the file indicated by the keyword FILE.
Once the PLUMED input file is prepared, one has to run Gromacs with the option to activate PLUMED and read the input file:

```
gmx_mpi mdrun -plumed
```

During the metadynamics simulation, PLUMED will create two files, named COLVAR and HILLS. The COLVAR file contains all the information specified by the PRINT command, in this case the value of the CVs every 10 steps of simulation, along with the current value of the metadynamics bias potential. The HILLS file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content. The last column contains the value of the bias factor used, while the last but one the height of the Gaussian, which is scaled during the simulation following the well-tempered recipe.

```
#! FIELDS time phi psi sigma_phi sigma_psi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
      1.0000     -1.3100    0.0525          0.35            0.35      1.4400      6
      2.0000     -1.4054    1.9742          0.35            0.35      1.4400      6
      3.0000     -1.9997    2.5177          0.35            0.35      1.4302      6
      4.0000     -2.2256    2.1929          0.35            0.35      1.3622      6
```

If we carefully look at the height column, we will notice that in the beginning the value reported is higher than the initial height specified in the input file, which should be 1.2 kJ/mol. In fact, this column reports the height of the Gaussian scaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy. In this way, when we will use sum_hills, the sum of the Gaussian kernels deposited will directly provide the free-energy, without further rescaling needed.
We can plot the time evolution of the CVs along with the height of the Gaussian.
The system is initialized in one of the local minimum where it starts accumulating bias. As the simulation progresses and the bias added grows, the Gaussian height is progressively reduced. After a while (t=0.8 ns), the system is able to escape the local minimum and explore a new region of the phase space. As soon as this happens, the Gaussian height is restored to the initial value and starts to decrease again. In the long time, the Gaussian height becomes smaller and smaller while the system diffuses in the entire CVs space.

11.38.17.2.4 Calculate free-energies and monitor convergence One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills should be used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.

To calculate the two-dimensional free energy as a function of phi and psi, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called fes.dat in which the free-energy surface as function of phi and psi is calculated on a regular grid. One can modify the default name for the free energy file, as well as the boundaries and bin size of the grid, by using the following options of sum_hills :

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

It is also possible to calculate one-dimensional free energies from the two-dimensional metadynamics simulation. For example, if one is interested in the free energy as a function of the phi dihedral alone, the following command line should be used:

```
plumed sum_hills --hills HILLS --idw phi --kt 2.5
```

The result should look like this:
To assess the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The option –stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option –mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --idw phi --kt 2.5 --stride 500 --mintozero
```

one free energy is calculated every 500 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. Try to visualize the different estimates of the free energy as a function of time.
To assess the convergence of the simulation more quantitatively, we can calculate the free-energy difference between the two local minima in the one-dimensional free energy along phi as a function of simulation time. We can use the bash script analyze_FES.sh to integrate the multiple free-energy profiles in the two basins defined by the following intervals in phi space: basin A, -3<phi<-1, basin B, 0.5<phi<1.5.

```
./analyze_FES.sh NFES -3.0 -1.0 0.5 1.5 KBT
```

where NFES is the number of profiles (free-energy estimates at different times of the simulation) generated by the option –stride of sum_hills, and KBT is the temperature in energy units (in this case KBT=2.5).
This analysis, along with the observation of the diffusive behavior in the CVs space, suggest that the simulation is converged.

11.38.17.3 Exercise 2. Bias-Exchange Metadynamics

11.38.17.3.1 Resources The `tarball` for this project contains all the files needed to run this exercise.

11.38.17.3.2 Summary of theory In well-tempered metadynamics the free-energy landscape of the system is reconstructed by gradually filling the local minima with gaussian hills. The dimensionality of the landscape is equal to the number of CVs which are biased, and typically a number of CVs smaller than four is employed. The reason for this is that qualitatively, if the CVs are not correlated among them, the simulation time required to fill the free-energy landscape grows exponentially with the number of CVs. This limitation can be severe when studying complex transformations or reactions in which more than say three relevant CVs can be identified.
Alternative solutions employ metadynamics together with other enhanced sampling techniques (i.e. Parallel Tempering or more generally Hamiltonian Replica Exchange). In particular in Bias-Exchange metadynamics the problem of sampling a multi-dimensional free-energy surface is recast in that of sampling many one-dimensional free-energies. In this approach a relatively large number N of CVs is chosen to describe the possible transformations of the system (e.g., to study the conformations of a peptide one may consider all the dihedral angles between amino acids). Then, N metadynamics simulations (replicas) are run on the same system at the same temperature, biasing a different CV in each replica. Normally, in these conditions, each bias profile would converge very slowly to the equilibrium free-energy, due to hysteresis.
In order to tackle this problem, in the bias-exchange approach every fixed number of steps (say 10,000) an exchange is attempted between a randomly selected pair of replicas a and b. The exchanges allow the replicas to gain from the sampling of the other replicas and enable the system to explore the conformational space along a large number of different directions.
The probability to accept the exchange is given by a Metropolis rule:

$$\min\left(1, \exp\left[\beta(V_G^a(x^a,t) + V_G^b(x^b,t) - V_G^a(x^b,t) - V_G^b(x^a,t))\right]\right)$$

where x^a and x^b are the coordinates of replicas a and b and $V_G^{a(b)}(x,t)$ is the metadynamics potential acting on the replica $a(\ b)$. Each trajectory evolves through the high dimensional free energy landscape in the space of the CVs sequentially biased by different metadynamics potentials acting on one CV at each time. The results of the simulation are N one-dimensional projections of the free energy.

11.38.17.3.3 Setup, run, and analyze a well-tempered bias-exchange metadynamics simulation In the following example, a bias-exchange simulation is performed on Alanine di-peptide. A typical input file for bias exchange well tempered metadynamics is the following:

- a common input file in which all the collective variables are defined:

```
plumed-common.dat:

# read a pdb file to get topology info
MOLINFO STRUCTURE=ALAD.pdb

# tell PLUMED to generate random exchange trials
RANDOM_EXCHANGES

# set up four variables for Phi, Psi, Theta, Xi dihedral angles
phi: TORSION ATOMS=@phi-2
psi: TORSION ATOMS=@psi-2
theta: TORSION ATOMS=6,5,7,9
xi: TORSION ATOMS=16,15,17,19

PRINT ARG=phi,psi,theta,xi STRIDE=250 FILE=COLVAR
```

(see MOLINFO and RANDOM_EXCHANGES).

- additional input files that INCLUDE the common input and define the METAD along the selected CVs:

```
plumed.dat.0:

INCLUDE FILE=plumed-common.dat
METAD ARG=phi HEIGHT=1.2 SIGMA=0.25 PACE=100 GRID_MIN=-pi GRID_MAX=pi BIASFACTOR=10.0
ENDPLUMED


plumed.dat.1:

INCLUDE FILE=plumed-common.dat
METAD ARG=psi HEIGHT=1.2 SIGMA=0.25 PACE=100 GRID_MIN=-pi GRID_MAX=pi BIASFACTOR=10.0
ENDPLUMED


plumed.dat.2:

INCLUDE FILE=plumed-common.dat
METAD ARG=theta HEIGHT=1.2 SIGMA=0.50 PACE=100 GRID_MIN=-pi GRID_MAX=pi BIASFACTOR=10.0
ENDPLUMED


plumed.dat.3:

INCLUDE FILE=plumed-common.dat
METAD ARG=xi HEIGHT=1.2 SIGMA=0.50 PACE=100 GRID_MIN=-pi GRID_MAX=pi BIASFACTOR=10.0
ENDPLUMED
```

The, in this case, two replicas start from the same GROMACS topology file replicated twice: topol0.tpr, topol1.tpr, topol2.tpr and topol3.tpr. Finally, GROMACS is launched as a parallel run on 4 cores, with one replica per core, with the command

```
mpirun -np 4 gmx_mpi mdrun -s topol -plumed plumed -multi 4 -replex 10000 >& log &
```

where -replex 10000 indicates that every 10000 molecular-dynamics steps exchanges are attempted (as printed in the GROMACS $*$.log files).
In order to have an idea of how Bias-Exchange Metadynamics works we can compare the sampling of the four replicas along the first two collective variables with respect to a free energy surface obtained by sampling in two dimensions.
The main features are that all the replicas can sample all the relevant free energy minima even if their collective variable is not meant to sample them, only the replicas with a collective variable meant to pass a barrier can sample that transition, so high energy regions are sampled only locally. This can seem a weakness in the case of alanine dipeptide but is the real strength of bias exchange metadynamics, indeed by not knowing anything of a system it is possible to choose as many collective variables as possible and even if most of them are not particularly useful a posteriori, they all gain from the good ones and nothing is lost.

11.38.17.3.4 Calculate free-energies and monitor convergence As for the former case, one can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. This approach can only be used to recover one-dimensional free-energy profiles:

```
plumed sum_hills --hills HILLS.0 --mintozero
```

The command above generates a file called fes.dat in which the free-energy surface as function of phi is calculated on a regular grid.
It is also possible to calculate one-dimensional free energies from the two-dimensional metadynamics simulation. For example, if one is interested in the free energy as a function of the phi dihedral alone, the following command line should be used:
The result should look like this (compared with the one obtained before):
As above we can assess the convergence of a metadynamics simulation by calculating the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The option –stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option –mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS.0 --stride 500 --mintozero
```

Try to visualize the different estimates of the free energy as a function of time.
To assess the convergence of the simulation more quantitatively, we can calculate the free-energy difference between the two local minima in the one-dimensional free energy along phi as a function of simulation time. We can use the bash script analyze_FES.sh to integrate the multiple free-energy profiles in the two basins defined by the following intervals in phi space: basin A, -3<phi<-1, basin B, 0.5<phi<1.5.

```
./analyze_FES.sh NFES -3.0 -1.0 0.5 1.5 KBT
```

where NFES is the number of profiles (free-energy estimates at different times of the simulation) generated by the option –stride of sum_hills, and KBT is the temperature in energy units (in this case KBT=2.5).
In addition to check for the convergence of the one dimesional profiles, it is important to verify the rate of exchange among the replicas. In fact if the rate is too low or if it is not enough homegenues then the replicas can still suffer from histeris problems (ie with one replica trapped somewhere in the conformational space). Generally speaking a good probability of exchange is between 10 and 30%.
It is possible to check the exchange statistics and the detailed diffusion of the replicas among the different biases using a script provided:

```
./demux_m.pl md0.log
```

which will create two files, called replica_temp.xvg and replica_index.xvg. We can plot the first file, which reports the bias index of each configuration at a given time of the simulation. This file is extremely useful, because it allows us monitoring the replicas diffusion in bias.

11.38.17.4 Exercise 3. Replica-Average Metadynamics

11.38.17.4.1 Resources The `tarball` for this project contains all the files needed to run this exercise.

11.38.17.4.2 Summary of theory Equilibrium experimental data are the result of a measure over an ensemble of structures and over time. In principle a "perfect" molecular dynamics simulations, that is a simulations with a perfect force-field and a perfect sampling can predict the outcome of an experiment in a quantitative way. Actually in most of the cases obtaining a qualitative agreement is already a fortunate outcome. In order to increase the accuracy of a force field in a system dependent manner it is possible to add to the force-field an additional term based on the agreement with a set of experimental data. This agreement is not enforced as a simple restraint because this would mean to ask the system to be always in agreement with all the experimental data at the same time, instead the restraint is applied over an AVERAGED COLLECTIVE VARIABLE where the average is performed over multiple identical simulations (replicas). In this way there is not a single replica that must be in agreement with the experimental data but they should be in agreement on average. It has been shown that this approach is equivalent to solve the problem of finding a modified version of the force field that reproduces the provided set of experimental data without any additional assumption on the data themselves [146] [147] [148] [149] .

11.38.17.4.3 The system: Chignolin In this exercise we will model the equilibrium ensemble of Chignolin by combining an implicit solvent force-field (Amber99sb) with synthetic experimental data derived from an experimentally determined ensemble of structures.
The experimental data are the following distances among CA carbons that have been averaged over the whole ensemble.

```
#RES  RES  DISTANCE(nm)
 1    3    0.656214
 1    4    0.963703
 1    5    1.197160
 1    6    1.219970
 1    7    1.206760
 1    8    1.060110
 1    9    0.858911
 1   10    0.872988
 2    4    0.656913
 2    5    0.931405
 2    6    0.953046
 2    7    0.901493
 2    8    0.809623
 2    9    0.716444
 2   10    0.882070
 3    5    0.588322
 3    6    0.660023
 3    7    0.704165
 3    8    0.720063
 3    9    0.799961
 3   10    0.981838
 4    6    0.550374
 4    7    0.572299
 4    8    0.764540
 4    9    0.939752
 4   10    1.186040
 5    7    0.613088
 5    8    0.847734
 5    9    1.084080
 5   10    1.296630
 6    8    0.591571
 6    9    0.888388
 6   10    1.102920
 7    9    0.701717
 7   10    0.987368
 8   10    0.654310
```

11.38.17.4.4 Setup, run and analysis Here we will give partial information on how to setup the simulations. Users should refer to the manual and the literature cited on how to complete the information provided and thus successfully perform the exercise.

Step 1: Add the collective variables that describe the distances averaged over the ensemble to the template input files provided.

For each of the above experimental data one should define:

```
#this is the distance between CA atoms 5 and 33 belonging to residues 1 and 3, respectively.
d1: DISTANCE ATOMS=5,33
# this is the averaging of the distance among the replicas
ed1: ENSEMBLE ARG=d1
# this is the restraint applied on the averaged distance to match the experimental one
RESTRAINT ARG=ed1.d1 KAPPA=100000 AT=0.656214
```

(see DISTANCE, ENSEMBLE, and RESTRAINT).

Step 2: Setup the Bias-Exchange simulations

We will run Bias-Exchange simulations using four CVs, two of them have already been chosen (see plumed-common.dat) while the others should be selected by you. Additionally Metadynamics parameters METAD for the two selected collective variables should be added in the plumed.dat.0 and plumed.dat.1 files. In particular while the PACE, HEIGHT and BIASFACTOR can be kept the same of those defined for the selected CVs, the SIGMA should be half of the fluctuations of the chosen CV in an unbiased run ($>$1 ns).

Step 3: Run

The simulation should be run until convergence of the four one dimensional free energies, this will typically take more than 200 ns per replica.

Step 4: Analysis

The user should check the diffusion of the replicas among the biases (see above) and send the converged free energy profiles.

Step 5: Only for the brave!

The same simulation without experimental restraints could be performed in such a way to compare the free energy profiles obtained with and without the inclusion of experimental data.

11.38.18 CINECA tutorial

Authors

Max Bonomi, stealing most of the material from other tutorials. Alejandro Gil Ley is acknowledged for beta-testing this tutorial.

Date

November 20, 2015

This document describes the PLUMED tutorial held at CINECA, November 2015. The aim of this tutorial is to learn how to use PLUMED to analyze molecular dynamics (MD) simulations on-the-fly, to analyze existing trajectories, and to perform enhanced sampling. Although the presented input files are correct, the users are invited to **refer to the literature to understand how the parameters of enhanced sampling methods should be chosen in a real application.**
Users are also encouraged to follow the links to the full PLUMED reference documentation and to wander around in the manual to discover the many available features and to do the other, more complete, tutorials. Here we are going to present only a very narrow selection of methods.
All the tests here are performed on a toy system, alanine dipeptide, simulated in vacuum using the AMBER99SB force field. Simulations are carried out with GROMACS 5.0.4, which is here assumed to be already patched with PLUMED 2.2 and properly installed. However, these examples could be easily converted to other MD software.

11.38.18.1 Resources

All the GROMACS input files and analysis scripts are provided in this `tarball`. Users are expected to write PLUMED input files based on the instructions below.
We can start by downloading the tarball and uncompressing it:

```
> tar xzvf cineca.tar.gz
```

Warning

Exercises should be run inside the newly-created cineca directory, by creating new sub-directories, one per exercise.

11.38.18.2 Alanine dipeptide: our toy model

In this tutorial we will play with alanine dipeptide (see Fig. cineca-1-ala-fig). This rather simple molecule is useful to benchmark data analysis and free-energy methods. This system is a nice example because it presents two metastable states separated by a high free-energy barrier. It is conventional use to characterize the two states in terms of Ramachandran dihedral angles, which are denoted with Φ and Ψ in Fig. cineca-1-transition-fig .

11.38.18.3 Monitoring collective variables

The main goal of PLUMED is to compute collective variables, which are complex descriptors of a system than can be used to analyze a conformational change or a chemical reaction. This can be done either on-the-fly, that is during MD, or a posteriori, using PLUMED as a post-processing tool. In both cases one should create an input file with a specific PLUMED syntax. A sample input file is below:

```
# compute distance between atoms 1 and 10
d: DISTANCE ATOMS=1,10
# create a virtual atom in the center between atoms 20 and 30
center: CENTER ATOMS=20,30
# compute torsional angle between atoms 1,10,20 and center
phi: TORSION ATOMS=1,10,20,center
# print d and phi every 10 step
PRINT ARG=d,phi STRIDE=10
```

(see DISTANCE, CENTER, TORSION, and PRINT)
PLUMED works using kJ/nm/ps as energy/length/time units. This can be personalized using UNITS. Notice that variables should be given a name (in the example above, `d`, and `phi`), which is then used to refer to these variables. Lists of atoms should be provided as comma separated numbers, with no space. Virtual atoms can be created and assigned a name for later use. You can find more information on the PLUMED syntax at Getting Started page of the manual. The complete documentation for all the supported collective variables can be found at the Collective Variables page.

11.38.18.3.1 Exercise 1: on-the-fly analysis Here we will run a plain MD simulation on alanine dipeptide and compute two torsional angles on-the-fly. GROMACS needs a .tpr file, which is a binary file containing initial positions as well as force-field parameters. For this tutorial, it is sufficient to use the .tpr files provided in the SETUP directory of the package:

- topolA.tpr - setup in vacuum, initialized in state A
- topolB.tpr - setup in vacuum, initialized in state B

However, we also provide .gro, .mdp, and .top files, that can be modified and used to generate a new .tpr file. GROMACS can be run (interactively) using the following command:

```
> gmx_mpi mdrun -s ../SETUP/topolA.tpr -nsteps 10000 -x traj.xtc
```

The nsteps flags can be used to change the number of time steps and topolA.tpr is the name of the tpr file. While running, GROMACS will produce a md.log file, with log information, and a traj.xtc file, with a binary trajectory. The trajectory can be visualized with VMD using:

```
> vmd confout.gro traj.xtc
```

To activate PLUMED during a GROMACS MD simulation, you need to add the -plumed flag

```
> gmx_mpi mdrun -s ../SETUP/topolA.tpr -nsteps 10000 -plumed plumed.dat -x traj.xtc
```

Here plumed.dat is the name of the PLUMED input file. Notice that PLUMED will write information in the md.log that could be useful to verify if the simulation has been set up properly.
In this exercise, we will run a plain MD simulation and monitor the Φ and Ψ dihedral angles on-the-fly. In order to do so, you need to prepare the following PLUMED input file:

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
PRINT ARG=phi,psi STRIDE=100 FILE=COLVAR
```

(see TORSION and PRINT)
PLUMED is going to compute the collective variables only when necessary, that is, in this case, every 100 steps. This is not very relevant for simple variables such as torsional angles, but provides a significant speed-up when using expensive collective variables.
PLUMED will write a textual file named COLVAR containing three columns: simulation time, Φ and Ψ. Results can be plotted using gnuplot:

```
> gnuplot
# this shows phi as a function of time
gnuplot> plot "COLVAR" u 2
# this shows psi as a function of time
gnuplot> plot "COLVAR" u 3
# this shows psi as a function of phi
gnuplot> plot "COLVAR" u 2:3
```

Now try to do the same using the two different initial configurations that we provided (`topolA.tpr` and `topol↩B.tpr`). Results from 200ps (100000 steps) trajectories in vacuum are shown in Figure cineca-ala-traj.
Notice that the result depends heavily on the starting structure. For the simulation in vacuum, the two free-energy minima are separated by a large barrier so that the system cannot cross it in such short simulation time. Also notice that the two clouds are well separated, indicating that these two collective variables are appropriate to properly distinguish the two minima.
As a final comment, consider that if you run twice the same calculation in the same directory, you might overwrite the output files. GROMACS takes automatic backup of these files, and PLUMED does it as well. In case you are restarting a simulation, you can add the keyword RESTART at the beginning of the PLUMED input file. This will tell PLUMED to *append* files instead of taking a backup copy.

11.38.18.3.2 Exercise 2: analysis with the driver tool Imagine you have already run a simulation, with or without PLUMED. You might want to compute the collective variables a posteriori, i.e. from the trajectory file. You can do this by using the PLUMED executable on the command line. Type

```
> plumed driver --help
```

to have an idea of the possible options. See driver for the full documentation.
Here we will use the driver the compute Φ and Ψ on the trajectory previously generated. Let's assume the trajectory is named `traj.xtc`. You should prepare a PLUMED input file named `analysis.dat` as:

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
PRINT ARG=phi,psi FILE=COLVAR_ANALYSIS
```

(see TORSION and PRINT) Notice that typically when using the driver we do not provide a STRIDE keyword to PRINT. This implies "print at every step" which, analyzing a trajectory, means "print for all the available snapshots". Then, you can use the following command:

```
> plumed driver --mf_xtc traj.xtc --plumed analysis.dat
```

Notice that PLUMED has no way to now the value of physical time from the trajectory. If you want physical time to be printed in the output file you should give more information to the driver, e.g.:

```
> plumed driver --mf_xtc traj.xtc --plumed analysis.dat --timestep 0.002 --trajectory-stride 1000
```

(see driver)
In this case we inform the driver that the `traj.xtc` file was produced in a run with a timestep of 0.002 ps and saving a snapshot every 1000 time steps.
You might want to analyze a different collective variable, such as the gyration radius. The gyration radius tells how extended is a molecule in space. You can do it with the following PLUMED input file

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

heavy: GROUP ATOMS=1,5,6,7,9,11,15,16,17,19
gyr: GYRATION ATOMS=heavy

# the same could have been achieved with
# gyr: GYRATION ATOMS=1,5,6,7,9,11,15,16,17,19

PRINT ARG=phi,psi,gyr FILE=analyze
```

(see TORSION, GYRATION, GROUP, and PRINT)
Now try to compute the time series of the gyration radius.

11.38.18.4 Biasing collective variables

We have seen that PLUMED can be used to compute collective variables. However, PLUMED is most often use to add forces on the collective variables. To this aim, we have implemented a variety of possible biases acting on collective variables. A bias works in a manner conceptually similar to the PRINT command, taking as argument one or more collective variables. However, here the STRIDE is usually omitted (that is equivalent to setting it to 1), which means that forces are applied at every timestep. Starting from PLUMED 2.2 you can change the STRIDE also for bias potentials, but that's another story. In the following we will see how to build an adaptive bias potential with metadynamics and how to apply harmonic restraints. The complete documentation for all the biasing methods available in PLUMED can be found at the Bias page.

11.38.18.4.1 Metadynamics

11.38.18.4.1.1 Summary of theory In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t \to \infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy

calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t \to \infty) = -\frac{\Delta T}{T + \Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard MD, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T + \Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135].
If you do not know exactly where you would like your collective variables to go, and just know (or suspect) that some variables have large free-energy barriers that hinder some conformational rearrangement or some chemical reaction, you can bias them using metadynamics. In this way, a time dependent, adaptive potential will be constructed that tends to disfavor visited configurations in the collective-variable space. The bias is usually built as a sum of Gaussian deposited in the already visited states.

11.38.18.4.1.2 Exercise 3 In this exercise, we will run a well-tempered metadynamics simulation on alanine dipeptide in vacuum, using as CV the backbone dihedral angle phi. In order to run this simulation we need to prepare the PLUMED input file (plumed.dat) as follows.

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate well-tempered metadynamics in phi depositing
# a Gaussian every 500 time steps, with initial height equal
# to 1.2 kJ/mol, bias factor equal to 10.0, and width to 0.35 rad

METAD ...
LABEL=metad
ARG=phi
PACE=500
HEIGHT=1.2
SIGMA=0.35
FILE=HILLS
BIASFACTOR=10.0
TEMP=300.0
GRID_MIN=-pi
GRID_MAX=pi
GRID_SPACING=0.1
... METAD

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see TORSION, METAD, and PRINT).
The syntax for the command METAD is simple. The directive is followed by a keyword ARG followed by the labels of the CVs on which the metadynamics potential will act. The keyword PACE determines the stride of Gaussian deposition in number of time steps, while the keyword HEIGHT specifies the height of the Gaussian in kJ/mol. For each CVs, one has to specified the width of the Gaussian by using the keyword SIGMA. Gaussian will be written to the file indicated by the keyword FILE.

The bias potential will be stored on a grid, whose boundaries are specified by the keywords GRID_MIN and GR↩ ID_MAX. Notice that you should provide either the number of bins for every collective variable (GRID_BIN) or the desired grid spacing (GRID_SPACING). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither GRID_BIN nor GRID_SPACING) and if Gaussian width is fixed PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.
Once the PLUMED input file is prepared, one has to run GROMACS with the option to activate PLUMED and read the input file:

```
gmx_mpi mdrun -s ../SETUP/topolA.tpr -plumed plumed.dat -nsteps 5000000 -x traj.xtc
```

During the metadynamics simulation, PLUMED will create two files, named COLVAR and HILLS. The COLVAR file contains all the information specified by the PRINT command, in this case the value of the CVs every 10 steps of simulation, along with the current value of the metadynamics bias potential. We can use COLVAR to visualize the behavior of the CV during the simulation:
By inspecting Figure cineca-metad-phi-fig, we can see that the system is initialized in one of the two metastable states of alanine dipeptide. After a while (t=0.1 ns), the system is pushed by the metadynamics bias potential to visit the other local minimum. As the simulation continues, the bias potential fills the underlying free-energy landscape, and the system is able to diffuse in the entire phase space.
The HILLS file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content:

```
#! FIELDS time phi psi sigma_phi sigma_psi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
```

The line starting with FIELDS tells us what is displayed in the various columns of the HILLS file: the time of the simulation, the value of phi and psi, the width of the Gaussian in phi and psi, the height of the Gaussian, and the bias factor. We can use the HILLS file to visualize the decrease of the Gaussian height during the simulation, according to the well-tempered recipe:
If we look carefully at the scale of the y-axis, we will notice that in the beginning the value of the Gaussian height is higher than the initial height specified in the input file, which should be 1.2 kJ/mol. In fact, this column reports the height of the Gaussian scaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy. In this way, when we will use sum_hills, the sum of the Gaussian kernels deposited will directly provide the free-energy, without further rescaling needed (see below).
One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills should be used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.
To calculate the free energy as a function of phi, it is sufficient to use the following command line:

```
plumed sum_hills --hills HILLS
```

The command above generates a file called fes.dat in which the free-energy surface as function of phi is calculated on a regular grid. One can modify the default name for the free energy file, as well as the boundaries and bin size of the grid, by using the following options of sum_hills :

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

The result should look like this:
To assess the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The option –stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option –mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
plumed sum_hills --hills HILLS --stride 100 --mintozero
```

one free energy is calculated every 100 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. The resulting plot should look like the following:
To assess the convergence of the simulation more quantitatively, we can calculate the free-energy difference between the two local minima of the free energy along phi as a function of simulation time. We can use following script to integrate the multiple free-energy profiles in the two basins defined by the following regions in phi space: basin A, -3<phi<-1, basin B, 0.5<phi<1.5.

```
# number of free-energy profiles
nfes= # put here the number of profiles
# minimum of basin A
minA=-3
# maximum of basin A
maxA=1
# minimum of basin B
minB=0.5
# maximum of basin B
maxB=1.5
# temperature in energy units
kbt=2.5

for((i=0;i<nfes;i++))
do
 # calculate free-energy of basin A
 A=`awk 'BEGIN{tot=0.0}{if($1!="#!" && $1>min && $1<max)tot+=exp(-$2/kbt)}END{print -kbt*log(tot)}' min=${minA
 # and basin B
 B=`awk 'BEGIN{tot=0.0}{if($1!="#!" && $1>min && $1<max)tot+=exp(-$2/kbt)}END{print -kbt*log(tot)}' min=${minB
 # calculate difference
 Delta=$(echo "${A} - ${B}" | bc -l)
 # print it
 echo $i $Delta
done
```

Please notice that nfes should be set to the number of profiles (free-energy estimates at different times of the simulation) generated by the option –stride of sum_hills.
This analysis, along with the observation of the diffusive behavior in the CVs space, suggest that the simulation is converged.

11.38.18.4.1.3 Exercise 4 In this exercise, we will run a well-tempered metadynamics simulation on alanine dipeptide in vacuum, using as CV the backbone dihedral angle psi. In order to run this simulation we need to prepare the PLUMED input file (plumed.dat) as follows.

```
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate well-tempered metadynamics in psi depositing
# a Gaussian every 500 time steps, with initial height equal
# to 1.2 kJ/mol, bias factor equal to 10.0, and width to 0.35 rad

METAD ...
LABEL=metad
ARG=psi
PACE=500
HEIGHT=1.2
SIGMA=0.35
FILE=HILLS
BIASFACTOR=10.0
TEMP=300.0
GRID_MIN=-pi
GRID_MAX=pi
GRID_SPACING=0.1
... METAD

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

(see TORSION, METAD, and PRINT).
Once the PLUMED input file is prepared, one has to run GROMACS with the option to activate PLUMED and read the input file. We will submit this job using the PBS queue system, as done for the previous exercise.

As we did in the previous exercise, we can use COLVAR to visualize the behavior of the CV during the simulation. Here we will plot at the same time the evolution of the metadynamics CV psi and of the other dihedral phi:
By inspecting Figure cineca-metad-psi-phi-fig, we notice that something different happened compared to the previous exercise. At first the behavior of psi looks diffusive in the entire CV space. However, around t=1 ns, psi seems trapped in a region of the CV space in which it was previously diffusing without problems. The reason is that the non-biased CV phi after a while has jumped into a different local minima. Since phi is not directly biased, one has to wait for this (slow) degree of freedom to equilibrate before the free energy along psi can converge. Try to repeat the analysis done in the previous exercise (calculate the estimate of the free energy as a function of time and monitor the free-energy difference between basins) to assess the convergence of this metadynamics simulation.

11.38.18.4.2 Restraints

11.38.18.4.2.1 Biased sampling theory A system at temperature T samples conformations from the canonical ensemble:

$$P(q) \propto e^{-\frac{U(q)}{k_B T}}$$

. Here q are the microscopic coordinates and k_B is the Boltzmann constant. Since q is a highly dimensional vector, it is often convenient to analyze it in terms of a few collective variables (see Belfast tutorial: Analyzing CVs , Belfast tutorial: Adaptive variables I , and Belfast tutorial: Adaptive variables II). The probability distribution for a CV s is

$$P(s) \propto \int dq e^{-\frac{U(q)}{k_B T}} \delta(s - s(q))$$

This probability can be expressed in energy units as a free energy landscape $F(s)$:

$$F(s) = -k_B T \log P(s)$$

.
Now we would like to modify the potential by adding a term that depends on the CV only. That is, instead of using $U(q)$, we use $U(q) + V(s(q))$. There are several reasons why one would like to introduce this potential. One is to avoid that the system samples some un-desired portion of the conformational space. As an example, imagine that you want to study dissociation of a complex of two molecules. If you perform a very long simulation you will be able to see association and dissociation. However, the typical time required for association will depend on the size of the simulation box. It could be thus convenient to limit the exploration to conformations where the distance between the two molecules is lower than a given threshold. This could be done by adding a bias potential on the distance between the two molecules. Another example is the simulation of a portion of a large molecule taken out from its initial context. The fragment alone could be unstable, and one might want to add additional potentials to keep the fragment in place. This could be done by adding a bias potential on some measure of the distance from the experimental structure (e.g. on root-mean-square deviation).
Whatever CV we decide to bias, it is very important to recognize which is the effect of this bias and, if necessary, remove it a posteriori. The biased distribution of the CV will be

$$P'(s) \propto \int dq e^{-\frac{U(q)+V(s(q))}{k_B T}} \delta(s - s(q)) \propto e^{-\frac{V(s(q))}{k_B T}} P(s)$$

and the biased free energy landscape

$$F'(s) = -k_B T \log P'(s) = F(s) + V(s) + C$$

Thus, the effect of a bias potential on the free energy is additive. Also notice the presence of an undetermined constant C. This constant is irrelevant for what concerns free-energy differences and barriers, but will be important later when we will learn the weighted-histogram method. Obviously the last equation can be inverted so as to obtain the original, unbiased free-energy landscape from the biased one just subtracting the bias potential

$$F(s) = F'(s) - V(s) + C$$

Additionally, one might be interested in recovering the distribution of an arbitrary observable. E.g., one could add a bias on the distance between two molecules and be willing to compute the unbiased distribution of some torsional angle. In this case there is no straightforward relationship that can be used, and one has to go back to the relationship between the microscopic probabilities:

$$P(q) \propto P'(q) e^{\frac{V(s(q))}{k_B T}}$$

The consequence of this expression is that one can obtained any kind of unbiased information from a biased simulation just by weighting every sampled conformation with a weight

$$w \propto e^{\frac{V(s(q))}{k_B T}}$$

That is, frames that have been explored in spite of a high (disfavoring) bias potential V will be counted more than frames that has been explored with a less disfavoring bias potential.

11.38.18.4.2.2 Umbrella sampling theory Often in interesting cases the free-energy landscape has several local minima. If these minima have free-energy differences that are on the order of a few times $k_B T$ they might all be relevant. However, if they are separated by a high saddle point in the free-energy landscape (i.e. a low probability region) than the transition between one and the other will take a lot of time and these minima will correspond to metastable states. The transition between one minimum and the other could require a time scale which is out of reach for MD. In these situations, one could take inspiration from catalysis and try to favor in a controlled manner the conformations corresponding to the transition state.
Imagine that you know since the beginning the shape of the free-energy landscape $F(s)$ as a function of one CV s. If you perform a MD simulation using a bias potential which is exactly equal to $-F(s)$, the biased free-energy landscape will be flat and barrierless. This potential acts as an "umbrella" that helps you to safely cross the transition state in spite of its high free energy.
It is however difficult to have an a priori guess of the free-energy landscape. We will see later how adaptive techniques such as metadynamics (Belfast tutorial: Metadynamics) can be used to this aim. Because of this reason, umbrella sampling is often used in a slightly different manner.
Imagine that you do not know the exact height of the free-energy barrier but you have an idea of where the barrier is located. You could try to just favor the sampling of the transition state by adding a harmonic restraint on the CV, e.g. in the form

$$V(s) = \frac{k}{2}(s - s_0)^2$$

. The sampled distribution will be

$$P'(q) \propto P(q) e^{\frac{-k(s(q)-s_0)^2}{2k_B T}}$$

For large values of k, only points close to s_0 will be explored. It is thus clear how one can force the system to explore only a predefined region of the space adding such a restraint. By combining simulations performed with different values of s_0, one could obtain a continuous set of simulations going from one minimum to the other crossing the transition state. In the next section we will see how to combine the information from these simulations.
If you want to just bring a collective variables to a specific value, you can use a simple restraint. Let's imagine that we want to force the Φ angle to visit a region close to $\Phi = \pi/2$. We can do it adding a restraint in Φ, with the following input

```
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
res: RESTRAINT ARG=phi AT=0.5pi KAPPA=5
PRINT ARG=phi,psi,res.bias
```

(see TORSION, RESTRAINT, and PRINT).
Notice that here we are printing a quantity named `res.bias`. We do this because RESTRAINT does not define a single value (that here would be theoretically named `res`) but a structure with several components. All biasing methods (including METAD) do so, as well as many collective variables (see e.g. DISTANCE used with COMPONENTS keyword). Printing the bias allows one to know how much a given snapshot was penalized. Also notice that PLUMED understands numbers in the form `{number}pi`. This is convenient when using torsions, since they are expressed in radians.
Now you can plot your trajectory with gnuplot and see the effect of KAPPA. You can also try different values of KAPPA. The stiffer the restraint, the less the collective variable will fluctuate. However, notice that a too large kappa could make the MD integrator unstable.

11.38.18.4.3 Using multiple replicas

Warning

Notice that multireplica simulations with PLUMED are fully supported with GROMACS, but only partly supported with other MD engines.

Some free-energy methods are intrinsically parallel and requires running several simultaneous simulations. This can be done with GROMACS using the multi replica framework. That is, if you have 4 tpr files named topol0.tpr, topol1.tpr, topol2.tpr, topol3.tpr you can run 4 simultaneous simulations.

```
> mpirun -np 4 gmx_mpi mdrun -s topol.tpr -plumed plumed.dat -multi 4 -nsteps 500000
```

Each of the 4 replicas will open a different topol file, and GROMACS will take care of adding the replica number before the .tpr suffix. PLUMED deals with the extra number in a slightly different way. In this case, for example, PLUMED first look for a file named `plumed.X.dat`, where X is the number of the replica. In case the file is not found, then PLUMED looks for `plumed.dat`. If also this is not found, PLUMED will complain. As a consequence, if all the replicas should use the same input file it is sufficient to put a single `plumed.dat` file, but one has also the flexibility of using separate files named `plumed.0.dat`, `plumed.1.dat` etc.
Also notice that providing the flag `-replex` one can instruct GROMACS to perform a replica exchange simulation. Namely, from time to time GROMACS will try to swap coordinates among neighboring replicas and accept of reject the exchange with a Monte Carlo procedure which also takes into account the bias potentials acting on the replicas, even if different bias potentials are used in different replicas. That is, PLUMED allows to easily implement many forms of Hamiltonian replica exchange.

11.38.18.4.4 Using multiple restraints with replica exchange

11.38.18.4.4.1 Weighted histogram analysis method theory Let's now consider multiple simulations performed with restraints located in different positions. In particular, we will consider the i-th bias potential as V_i. The probability to observe a given value of the collective variable s is:

$$P_i(s) = \frac{P(s)e^{-\frac{V_i(s)}{k_BT}}}{\int ds' P(s')e^{-\frac{V_i(s')}{k_BT}}} = \frac{P(s)e^{-\frac{V_i(s)}{k_BT}}}{Z_i}$$

where

$$Z_i = \sum_q e^{-(U(q)+V_i(q))}$$

The likelihood for the observation of a sequence of snapshots $q_i(t)$ (where i is the index of the trajectory and t is time) is just the product of the probability of each of the snapshots. We use here the minus-logarithm of the likelihood (so that the product is converted to a sum) that can be written as

$$\mathcal{L} = -\sum_i \int dt \log P_i(s_i(t)) = \sum_i \int dt \left(-\log P(s_i(t)) + \frac{V_i(s_i(t))}{k_BT} + \log Z_i \right)$$

One can then maximize the likelihood by setting $\frac{\delta \mathcal{L}}{\delta P(\mathbf{s})} = 0$. After some boring algebra the following expression can be obtained

$$0 = \sum_i \int dt \left(-\frac{\delta_{\mathbf{s}_i(t),\mathbf{s}}}{P(\mathbf{s})} + \frac{e^{-\frac{V_i(\mathbf{s})}{k_BT}}}{Z_i} \right)$$

In this equation we aim at finding $P(s)$. However, also the list of normalization factors Z_i is unknown, and they should be found selfconsistently. Thus one can find the solution as

$$P(\mathbf{s}) \propto \frac{N(\mathbf{s})}{\sum_i \int dt \frac{e^{-\frac{V_i(\mathbf{s})}{k_BT}}}{Z_i}}$$

where Z is selfconsistently determined as

$$Z_i \propto \int ds' P(\mathbf{s}') e^{-\frac{V_i(\mathbf{s}')}{k_BT}}$$

These are the WHAM equations that are traditionally solved to derive the unbiased probability $P(s)$ by the combination of multiple restrained simulations. To make a slightly more general implementation, one can compute the weights that should be assigned to each snapshot, that turn out to be:

$$w_i(t) \propto \frac{1}{\sum_j \int dt \frac{e^{-\beta V_j(\mathbf{s}_i(t))}}{Z_j}}$$

The normalization factors can in turn be found from the weights as

$$Z_i \propto \frac{\sum_j \int dt e^{-\beta V_i(\mathbf{s}_j(t))} w_j(t)}{\sum_j \int dt w_j(t)}$$

This allows to straighforwardly compute averages related to other, non-biased degrees of freedom, and it is thus a bit more flexible. It is sufficient to precompute this factors w and use them to weight every single frame in the trajectory.

11.38.18.4.4.2 Exercise 5 In this exercise we will run multiple restraint simulations and learn how to reweight and combine data with WHAM to obtain free-energy profiles. We start with running in a replica-exchange scheme 32 simulations with a restraint on phi in different positions, ranging from -3 to 3. We will instruct GROMACS to attempt an exchange between different simulations every 1000 steps.
First we need to create the 32 PLUMED input files and .tpr files, using the following script:

```
nrep=32
dx=`echo "6.0 / ( $nrep - 1 )" | bc -l`

for((i=0;i<nrep;i++))
do
# center of the restraint
AT=`echo "$i * $dx - 3.0" | bc -l`

cat >plumed.${i}.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Impose an umbrella potential on phi
# with a spring constant of 200 kjoule/mol
# and centered in phi=AT
#
restraint-phi: RESTRAINT ARG=phi KAPPA=200.0 AT=$AT
# monitor the two variables and the bias potential
PRINT STRIDE=100 ARG=phi,psi,restraint-phi.bias FILE=COLVAR
EOF

# we initialize some replicas in A and some in B:
if((i%2==0)); then
  cp ../SETUP/topolA.tpr topol$i.tpr
else
  cp ../SETUP/topolB.tpr topol$i.tpr
fi
done
```

And then run GROMACS with the following command:

```
mpirun -np 32 gmx_mpi mdrun -plumed plumed.dat -s topol.tpr -multi 32 -replex 1000 -nsteps 500000 -x traj.xtc
```

To be able to combine data from all the simulations, it is necessary to have an overlap between statistics collected in two adjacent umbrellas.
Have a look at the plots of (phi,psi) for the different simulations to understand what is happening. To visualize them all at once as in Fig. cineca-usrem-phi-all, you can prepare the following script:

```
awk 'BEGIN{print ""; ORS=""; print "p \"COLVAR.0\" u 2:3 notitle,"; for(i=1;i<=30;i++) print "\"COLVAR."i"\" u
```

and then open gnuplot and load it:

```
gnuplot> load "plot.plt"
```

An often misunderstood fact about WHAM is that data of the different trajectories can be mixed and it is not necessary to keep track of which restraint was used to produce every single frame. Let's get the concatenated trajectory

```
trjcat_mpi -cat -f traj*.xtc -o alltraj.xtc
```

Now we should compute the value of each of the bias potentials on the entire (concatenated) trajectory.

```
nrep=32
dx=`echo "6.0 / ( $nrep - 1 )" | bc -l`

for i in `seq 0 $(( $nrep - 1 ))`
do
# center of the restraint
AT=`echo "$i * $dx - 3.0" | bc -l`

cat >plumed.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
restraint-phi: RESTRAINT ARG=phi KAPPA=200.0 AT=$AT

# monitor the two variables and the bias potential
PRINT STRIDE=100 ARG=phi,psi,restraint-phi.bias FILE=ALLCOLVAR.$i
EOF

plumed driver --mf_xtc alltraj.xtc --trajectory-stride=10 --plumed plumed.dat

done
```

It is very important that this script is consistent with the one used to generate the multiple simulations above. Now, single files named ALLCOLVAR.XX will contain on the fourth column the value of the bias centered in a given position, computed on the entire concatenated trajectory.
Next step is to compute the weights self-consistently solving the WHAM equations, using the python script "wham.↩ py" contained in the SCRIPTS directory. We will run this script as follows:

```
bash ../SCRIPTS/wham.sh ALLCOLVAR.*
```

This script will produce several files. Let's visualize "phi_fes.dat", which contains the free energy as a function of phi, and compare this with the result previously obtained with metadynamics.

11.38.18.4.4.3 Exercise 6 In the previous exercise, we use multiple restraint simulations to calculate the free energy as a function of the dihedral phi. The resulting free energy was in excellent agreement with our previous metadynamics simulation. In this exercise we will repeat the same procedure for the dihedral psi. At the end of the steps defined above, we can plot the free energy "psi_fes.dat" and compare it with the profile calculated from a metadynamics simulations using both phi and psi as CVs.
We can easily spot from the plot above that something went wrong in this multiple restraint simulations, despite we used the very same approach we adopted for the phi dihedral. The problem here is that psi is a "bad" collective variable, and the system is not able to equilibrate the missing slow degree of freedom phi in the short time scale of the umbrella simulation (1 ns). In the metadynamics exercise in which we biased only psi, we detect problems by observing the behavior of the CV as a function of simulation time. How can we detect problems in multiple restraint simulations? This is slightly more complicated, but running this kind of simulation in a replica-exchange scheme offers a convenient way to detect problems.
The first thing we need to do is to demux the replica-exchange trajectories and reconstruct the continuous trajectories of the replicas across the different restraint potentials. In order to do so, we can use the following script:

```
demux.pl md0.log
trjcat_mpi -f traj?.xtc traj??.xtc  -demux replica_index.xvg
```

This commands will generate 32 continuous trajectories, named XX_trajout.xtc. We will use the driver to calculate the value of the CVs phi and psi on these trajectories.

```
nrep=32

for i in `seq 0 $(( $nrep - 1 ))`
do

cat >plumed.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

# monitor the two variables
PRINT STRIDE=100 ARG=phi,psi FILE=COLVARDEMUX.$i
EOF

plumed driver --mf_xtc ${i}_trajout.xtc --trajectory-stride=10 --plumed plumed.dat

done
```

The COLVARDEMUX.XX files will contain the value of the CVs on the demuxed trajectory. If we visualize these files (see previous exercise for instructions) we will notice that replicas sample the CVs space differently. In order for each umbrella to equilibrate the slow degrees of freedom phi, the continuous replicas must be ergodic and thus sample the same distribution in phi and psi.

11.38.19 Moving from PLUMED 1 to PLUMED 2

Syntax in PLUMED 2 has been completely redesigned based on our experience using PLUMED 1, hopefully making it clearer, more flexible, and less error prone. The main difference is that whereas in PLUMED 1 lines could be inserted in any order, in PLUMED 2 the order of the lines matters. This is due to a major change in the internal architecture of PLUMED. In version 2, commands (or "actions") are executed in the order they are found in the input file. Because of this, you must e.g. first compute a collective variable and then print it later. More information can be found in the Section about Getting Started.
Other important changes are in the way groups and units are used, as discussed below. Finally, many features appear under a different name in the new version.

11.38.19.1 New syntax

We know that changing the input syntax requires a lot of work from the user side to update their input files. However, we believe that the new syntax is easier to read and that it allows for more flexibility. As an example, something that in PLUMED 1.3 was:

```
HILLS HEIGHT 0.4 W_STRIDE 600
WELLTEMPERED SIMTEMP 300  BIASFACTOR 15
RGYR LIST <all>
all->
1   5  6  7   9  11  15  16  17 19
all<-
TORSION LIST 5 7 9 15 SIGMA  0.1
TORSION LIST 7 9 15 17
PRINT W_STRIDE 100
```

in PLUMED 2.x becomes:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/moving.txt
all:  GROUP     ATOMS=1,5,6,7,9,11,15,16,17,19
rg:   GYRATION  ATOMS=all
t1:   TORSION ATOMS=5,7,9,15
METAD ...
  LABEL=meta
  ARG=t1 SIGMA=0.1 HEIGHT=0.4 PACE=600
  BIASFACTOR=15 TEMP=300
... METAD
t2:   TORSION ATOMS=7,9,15,17
PRINT ARG=t1,t2,meta.bias  STRIDE=100   FILE=COLVAR
```

Giving all quantities an explicit name makes the input easier to interpret. Additionally, all the parameters related to METAD are placed on a single line (actually, this is done here exploiting continuation lines). Also notice that one can customize the name of the COLVAR file. By specifying to PRINT which collective variables should be printed, one can easily decide what to print exactly and using which stride. By repeating the PRINT line one can also monitor very expensive variables with a larger stride, just putting the result on a separate file.
You might have noticed that ideas that were very difficult to implement in PLUMED 1.3 now become immediately available. As an example, one can now apply concurrently several METAD potentials [53] .

11.38.19.2 Groups

In PLUMED 1 groups (lists) were used for two tasks:

- To provide centers of masses to collective variables such as distances, angles, etc. This is now done by defining virtual atoms using either CENTER or COM
- To provide lists of atoms to collective variables such as coordination, gyration radius, etc. This is now done directly in the line that defines the collective variable.

If you would still like to use groups you can use the GROUP commands. Whenever the label for a GROUP action appears in the input it is replaced by the list of atoms that were specified in the GROUP.
A restraint on the distance between centers of mass in PLUMED 1 was something like:

```
DISTANCE LIST <g1> <g2>
g1->
17 20 22 30
g1<-
g2->
LOOP 37 40
g2<-
UMBRELLA CV 1 KAPPA 200 AT 1.0
PRINT W_STRIDE 100
```

The same in PLUMED 2.x reads:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/moving.txt
g1: COM ATOMS=17,20,22,30
g2: COM ATOMS=37-40
d:  DISTANCE ATOMS=g1,g2
r:  RESTRAINT ARG=d KAPPA=200 AT=1.0
PRINT STRIDE=100 FILE=COLVAR ARG=d,r.*
```

Notice that virtual atoms are very powerful tools in PLUMED 2. Actually, they can be used in any collective variable where normal atoms can be used, just by calling them by name. This allows to straightforwardly define variables such as coordination between centers of mass, which would have required an ad hoc implementation in PLUMED 1.
In the example above you can also appreciate the advantage of calling collective variables by name. It is obvious here that RESTRAINT is acting on distance `d`, whereas in PLUMED 1 one had to keep track of the number of the collective variables. This was easy for a single collective variable, but could become cumbersome for complex input files.

11.38.19.3 Names in output files

Another advantage of having names is that when PLUMED produces an output file it can insert explicit names in the file. Consider for example this PLUMED 1 input

```
DISTANCE LIST 1 2
ANGLE    LIST 3 4 5
PRINT W_STRIDE 100
```

The first line of the COLVAR file was then

```
#!  FIELDS time cv1 cv2
```

The equivalent input file in PLUMED 2 is

```
BEGIN_PLUMED_FILE working DATADIR=example-check/moving.txt
d: DISTANCE ATOMS=1,2
a: ANGLE    ATOMS=3,4,5
PRINT ARG=d,a FILE=COLVAR STRIDE=100
```

The first line of the COLVAR file now is

```
#! FIELDS time d a
```

This makes it easy to remember what's the meaning of each column of the COLVAR file without the need to always go back to the PLUMED input to check in which order the variables were declared.

11.38.19.4 Units

In PLUMED 1 the input file of PLUMED was expected to be written using the same units of the MD code. This choice was made to allow users to adopt the same units they were used to. However, we realized later that this choice was not allowing the PLUMED input files to be ported between different MD code. Let's say that one was using this keyword with GROMACS (kj/mol - nm)

```
UMBRELLA CV 1 KAPPA 200 AT 1.0
```

Let's assume the variable CV 1 was a distance. The same keyword in NAMD (kcal/mol - A) should have been converted to

```
UMBRELLA CV 1 KAPPA .4780 AT 10.0
```

The conversion of `AT` is straightforward (1nm=10A), but the conversion on `KAPPA` is more error prone. This is because `KAPPA` is measured in units of energy divided by distance squared. Notice that a different factor should have been used if the CV was an angle.

Learning from this, we designed PLUMED 2 in a way that units in the PLUMED input are independent of the MD code. Technically, this is achieved by doing the conversion when coordinates and forces are passed back and forth. In this way, the same PLUMED input could be used with GROMACS and NAMD.
We decided to use as standard units in PLUMED kj/mol, nm, and ps. Perhaps this is because most of the developers are using GROMACS, which also adopts these units. However, we still allow users to change units by means of the UNITS keyword. Since this keyword is included directly in the PLUMED input file, even when using personalized units the input files remains perfectly portable across different MD engines.

11.38.19.5 Directives

What follows is a list of all the documented directives of PLUMED 1 together with their plumed 2 equivalents. Be aware that the input syntax for these directives are not totally equivalent. You should read the documentation for the PLUMED 2 Action.

HILLS	METAD
WELLTEMPERED	METAD with BIASFACTOR
GRID	METAD with GRID_MIN, GRID_MAX, and GRID_BIN
WRITE_GRID	METAD with GRID_WFILE, GRID_WSTRIDE
READ_GRID	METAD with GRID_RFILE
MULTIPLE_WALKERS	METAD with options WALKERS_ID, WALKERS_N, WALKERS_DIR, and WALKERS_RSTRIDE
NOHILLS	not needed (collective variables are not biased by default)
INTERVAL	METAD with INTERVAL
INVERT	**currently missing**
PTMETAD	not needed (replica exchange detected from MD engine)
BIASXMD	not needed (replica exchange detected from MD engine); one should anyway use RANDOM_EXCHANGES to get the normal behavior
UMBRELLA	RESTRAINT
STEER	MOVINGRESTRAINT
STEERPLAN	MOVINGRESTRAINT
ABMD	ABMD
UWALL	UPPER_WALLS
LWALL	LOWER_WALLS
EXTERNAL	EXTERNAL
COMMITMENT	COMMITTOR
PROJ_GRAD	DUMPPROJECTIONS
DAFED	a similar method using a Langevin thermostat, with EXTENDED_LAGRANGIAN
DISTANCE	DISTANCE - POINT_FROM_AXIS and PROJ_ON_AXIS can be reproduced with DISTANCE and MATHEVAL
POSITION	POSITION
MINDIST	DISTANCES with keyword MIN
ANGLE	ANGLE
TORSION	TORSION
COORD	COORDINATION
HBOND	**currently missing** , can be emulated with COORDINATION
WATERBRIDGE	BRIDGE
RGYR	GYRATION
DIPOLE	DIPOLE
DIHCOR	DIHCOR
ALPHABETA	ALPHABETA
ALPHARMSD	ALPHARMSD
ANTIBETARMSD	ANTIBETARMSD
PARABETARMSD	PARABETARMSD
ELSTPOT	currently missing, but a related quantity can be obtained with DHENERGY
PUCKERING	PUCKERING

S_PATH	PATHMSD, s component
Z_PATH	PATHMSD, z component
TARGETED	RMSD
ENERGY	ENERGY
HELIX	**currently missing**
PCA	PCARMSD
SPRINT	SPRINT
RDF	DISTANCES, used in combination with HISTOGRAM / BETWEEN keyword
ADF	ANGLES, used in combination with HISTOGRAM / BETWEEN keyword
POLY	COMBINE
FUNCTION	MATHEVAL
ALIGN_ATOMS	WHOLEMOLECULES

11.38.20 Munster tutorial

Authors

Max Bonomi and Giovanni Bussi, stealing a lot of material from other tutorials. Richard Cunha is acknowledged for beta-testing this tutorial.

Date

March 11, 2015

This document describes the PLUMED tutorial held in Munster, March 2015. The aim of this tutorial is to learn how to use PLUMED to analyze molecular dynamics simulations on the fly, to analyze existing trajectories, and to perform enhanced sampling. Although the presented input files are correct, the users are invited to **refer to the literature to understand how the parameters of enhanced sampling methods should be chosen in a real application.**
Users are also encouraged to follow the links to the full PLUMED reference documentation and to wander around in the manual to discover the many available features and to do the other, more complete, tutorials. Here we are going to present only a very narrow selection of methods.
We here use PLUMED 2.1 syntax and we explicitly note if some syntax is expected to change in PLUMED 2.2, which will be released later in 2015. All the tests here are performed on a toy system, alanine dipeptide, simulated using the AMBER99SB force field. We provide both a setup that includes explicit water, which is more realistic but slower, and a setup in gas phase, which is much faster. Simulations are made using GROMACS 4.6.7, which is here assumed to be already patched with PLUMED and properly installed. However, these examples could be easily converted to other MD software.
All the gromacs input files and analysis scripts are provided in this TARBALL.
Users are expected to write PLUMED input files based on the instructions below.

11.38.20.1 Alanine dipeptide: our toy model

In this tutorial we will play with alanine dipeptide (see Fig. munster-1-ala-fig). This rather simple molecule is useful to make benchmark that are around for data analysis and free energy methods. It is a nice example since it presents two metastable states separated by a high (free) energy barrier. Here metastable states are intended as states which have a relatively low free energy compared to adjacent conformations. It is conventional use to show the two states in terms of Ramachandran dihedral angles, which are denoted with Φ and Ψ in Fig. munster-1-transition-fig .

11.38.20.2 Monitoring collective variables

The main goal of PLUMED is to compute collective variables, which are complex descriptors than can be used to analyze a conformational change or a chemical reaction. This can be done either on the fly, that is during molecular dynamics, or a posteriori, using PLUMED as a post-processing tool. In both cases one should create an input file with a specific PLUMED syntax. A sample input file is below:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
# compute distance between atoms 1 and 10
```

```
d: DISTANCE ATOMS=1,10
# create a virtual atom in the center between atoms 20 and 30
center: CENTER ATOMS=20,30
# compute torsional angle between atoms 1,10,20 and center
phi: TORSION ATOMS=1,10,20,center
# compute some function of previously computed variables
d2: MATHEVAL ARG=phi FUNC=cos(x) PERIODIC=NO
# print both of them every 10 step
PRINT ARG=d,phi,d2 STRIDE=10
```

PLUMED works using kJ/nm/ps as energy/length/time units. This can be personalized using UNITS. Notice that variables should be given a name (in the example above, `d`, `phi`, and `d2`), which is then used to refer to these variables. Lists of atoms should be provided as comma separated numbers, with no space. Virtual atoms can be created and assigned a name for later use. You can find more information on the PLUMED syntax at Getting Started page of the manual. The complete documentation for all the supported collective variables can be found at the Collective Variables page.

11.38.20.2.1 Analyze on the fly Here we will run a plain MD on alanine dipeptide and compute two torsional angles on the fly. GROMACS needs a .tpr file, which is a binary file containing initial positions as well as force-field parameters. We also provide .gro, .mdp, and .top files, that can be modified and used to generate a new .tpr file. For this tutorial, it is sufficient to use the provided .tpr files. You will find several tpr files, namely:

- topolAwat.tpr - setup in water, initialized in state A
- topolBwat.tpr - setup in water, initialized in state B
- topolA.tpr - setup in vacuum, initialized in state A
- topolB.tpr - setup in vacuum, initialized in state B

Gromacs md can be run using on the command line:

```
> gmx_mpi mdrun -s topolA.tpr -nsteps 10000
```

The nsteps flags can be used to change the number of time steps and topolA.tpr is the name of the tpr file. While running, gromacs will produce an md.log file, with log information, and a traj.xtc file, with a binary trajectory. The trajectory can be visualized with VMD using a command such as

```
> vmd confout.gro tra.xtc
```

To run a simulation with gromacs+plumed you just need to add a -plumed flag

```
> gmx_mpi mdrun -s topolA.tpr -nsteps 10000 -plumed plumed.dat
```

Here plumed.dat is the name of the plumed input file. Notice that PLUMED will write information in the md.log that could be useful to verify if the simulation has been set up properly.

11.38.20.2.1.1 Exercise 0 In this exercise, we will run a plain molecular dynamics simulation and monitor the Φ and Ψ dihedral angles on the fly. Using the following PLUMED input file you can monitor Φ and Ψ angles during the MD simulation

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
PRINT ARG=phi,psi STRIDE=100 FILE=colvar
```

Notice that PLUMED is going to compute the collective variables only when necessary, that is, in this case, every 100 steps. This is not very relevant for simple variables such as torsional angles, but provides a significant speedup when using expensive collective variables.
PLUMED will write a textual file named `colvar` containing three columns: physical time, Φ and Ψ. Results can be plotted using gnuplot:

```
> gnuplot
# this shows phi as a function of time
gnuplot> plot "colvar" u 2
# this shows psi as a function of time
gnuplot> plot "colvar" u 3
# this shows psi as a function of phi
gnuplot> plot "colvar" u 2:3
```

Now try to do the same using the two different initial configurations that we provided (`topolA.tpr` and `topolB.tpr`). You can try both setup (water and vacuum). Results from 200ps (100000 steps) trajectories in vacuum are shown in Figure munster-ala-traj.
Notice that the result depends heavily on the starting structure. For the simulation in vacuum, the two free-energy minima are separated by a large barrier and, in such a short simulation, the system cannot cross it. In water the barrier is smaller and you might see some crossing. Also notice that the two clouds are well separated, indicating that these two collective variables are good enough to properly distinguish among the two minima.
As a final comment, notice that if you run twice the same calculation in the same directory, you might overwrite the resulting files. GROMACS takes automatic backup of the output files, and PLUMED does it as well. In case you are restarting a simulation, you can add the keyword RESTART at the beginning of the PLUMED input file. This will tell PLUMED to *append* files instead of taking a backup copy.

11.38.20.2.2 Analyze using the driver Imagine you already made a simulation, with or without PLUMED. You might want to compute the collective variables a posteriori, from the trajectory file. You can do this by using the plumed executable on the command line. Type

```
> plumed driver --help
```

to have an idea of the possible options. See driver for the full documentation.
Here we will use the driver the compute Φ and Ψ on the already generated trajectory. Let's assume the trajectory is named `traj.xtc`. You should prepare an PLUMED input file named `analysis.dat` as:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
PRINT ARG=phi,psi FILE=analysis
```

Notice that typically when using the driver we do not provide a STRIDE keyword to PRINT. This implies "print at every step" which, analyzing a trajectory, means "print for all the available snapshots". Then, you can use the following command:

```
> plumed driver --mf_xtc traj.xtc --plumed analysis.dat
```

Notice that PLUMED has no way to now the value of physical time from the trajectory. If you want physical time to be printed in the `analysis` file you should give more information to the driver, e.g.:

```
> plumed driver --mf_xtc traj.xtc --plumed analysis.dat --timestep 0.002 --trajectory-stride 1000
```

(see driver)
In this case we inform the driver that the `traj.xtc` file was produced in a run with a timestep of 0.002 ps and saving a snap shot every 1000 time steps.
You might want to analyze a different collective variable, such as the gyration radius. The gyration radius tells how extended is the molecules in space. You can do it with the following plumed input file

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

heavy: GROUP ATOMS=1,5,6,7,9,11,15,16,17,19
gyr: GYRATION ATOMS=heavy

# the same could have been achieved with
# gyr: GYRATION ATOMS=1,5,6,7,9,11,15,16,17,19

PRINT ARG=phi,psi,gyr FILE=analyze
```

Now try to compute the time series of the gyration radius.

11.38.20.2.3 Periodic boundaries and explicit water In case you are running the simulation in water, you might see that at some point this variable shows some crazy jump. The reason is that the trajectory contains coordinates where molecules are broken across periodic-boundary conditions This happens with GROMACS and some other MD code. These codes typically have tools to process trajectories and restore whole molecules. This trick is OK for the a-posteriori analysis we are trying now, but cannot used when one needs to compute a collective variable on-the-fly or, as we will see later, one wants to add a bias to that collective variable. For this reason, we implemented a workaround in PLUMED, that is the molecule should be made whole using the WHOLEMOLECULES command. What this command is doing is making a loop over all the atoms (in the order they are provided) and set the coordinates of each of them in the periodic image which is as close as possible to the coordinates of the preceding atom. In most cases it is sufficient to list all atoms of a molecule in order. Look in the WHOLEMOLECULES page to get more information. Here this will be enough:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

# notice the 1-22 syntax, a shortcut for a list 1,2,3,...,22
WHOLEMOLECULES ENTITY0=1-22

heavy: GROUP ATOMS=1,5,6,7,9,11,15,16,17,19
gyr: GYRATION ATOMS=heavy

PRINT ARG=phi,psi,gyr FILE=analyze
```

This is a very important issue that should be kept in mind when using PLUMED. Notice that starting with version 2.2 PLUMED will make molecules used in GYRATION (as well as in other variables) whole automatically, so that this extra command will not be necessary.
Notice that you can instruct PLUMED to dump on a file not only the collective variables (as we are doing with PRINT) but also the atomic positions. This is a very good way to understand what WHOLEMOLECULES is actually doing. Try the following input

```
BEGIN_PLUMED_FILE broken DATADIR=example-check/munster.txt
#SETTINGS MOLFILE=user-doc/tutorials/munster/TOPO/reference.pdb
MOLINFO STRUCTURE=reference.pdb
DUMPATOMS FILE=test1.gro ATOMS=1-22
WHOLEMOLECULES ENTITY0=1-22
DUMPATOMS FILE=test2.gro ATOMS=1-22
```

DUMPATOMS writes on a gro file the coordinates of the alanine dipeptide atoms. Here PLUMED will produce two files, one with coordinates *before* the application of WHOLEMOLECULES, and one with coordinates after∗ the application of WHOLEMOLECULES. You can load both trajectories in VMD to see the difference. The MOLINFO command is here used to provide atom names to PLUMED so that the resulting gro file looks nicer in VMD.
Notice that PLUMED has several commands that manipulate atomic coordinates. One example is WHOLEMOLECULES, that fixes problems with periodic boundary conditions. COM and CENTER add new atoms, and FIT_TO_TEMPLATE can actually move atoms from their original position to align them on a template. DUMPATOMS is this very useful to check what these commands are doing and for using the PLUMED driver to manipulate MD trajectories.

11.38.20.2.4 Other analysis tools PLUMED also allows you to make some analysis on the collective variables you are calculating. For example, you can compute a histogram with an input like this one

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
heavy: GROUP ATOMS=1,5,6,7,9,11,15,16,17,19
gyr: GYRATION ATOMS=heavy
PRINT ARG=phi,psi,gyr FILE=analyze
myhist: HISTOGRAM ...
  ARG=gyr
  KERNEL=DISCRETE
  GRID_MIN=0
  GRID_MAX=1
  GRID_BIN=50
...
DUMPGRID GRID=myhist FILE=histogram
```

An histogram with 50 bins will be performed on the gyration radius. Try to compute the histogram for the Φ and Ψ angles.
PLUMED can do much more than a histogram, more information on analysis can be found at the page Analysis
Notice that the plumed driver can also be used directly from VMD taking advantage of the PLUMED collective variable tool developed by Toni Giorgino (http://multiscalelab.org/utilities/PlumedGUI). Just open a recent version of VMD and go to Extensions/Analysis/Collective Variable Analysis (PLUMED). This graphical interface can also be used to quickly build PLUMED input files based on template lines.

11.38.20.3 Biasing collective variables

We have seen that PLUMED can be used to compute collective variables. However, PLUMED is most often use to add forces on the collective variables. To this aim, we have implemented a variety of possible biases acting on collective variables. A bias works in a manner conceptually similar to the PRINT command, taking as argument

one or more collective variables. However, here the STRIDE is usually omitted (that is equivalent to setting it to 1), which means that forces are applied at every timestep. In PLUMED 2.2 you will be able to change the STRIDE also for bias potentials, but that's another story. In the following we will see how to apply harmonic restraints and how to build an adaptive bias potential with metadynamics. The complete documentation for all the biasing methods available in PLUMED can be found at the Bias page.

11.38.20.3.1 Metadynamics

11.38.20.3.1.1 Summary of theory In metadynamics, an external history-dependent bias potential is constructed in the space of a few selected degrees of freedom $\vec{s}(q)$, generally called collective variables (CVs) [47]. This potential is built as a sum of Gaussian kernels deposited along the trajectory in the CVs space:

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i(q(k\tau)))^2}{2\sigma_i^2}\right).$$

where τ is the Gaussian deposition stride, σ_i the width of the Gaussian for the ith CV, and $W(k\tau)$ the height of the Gaussian. The effect of the metadynamics bias potential is to push the system away from local minima into visiting new regions of the phase space. Furthermore, in the long time limit, the bias potential converges to minus the free energy as a function of the CVs:

$$V(\vec{s}, t\to\infty) = -F(\vec{s}) + C.$$

In standard metadynamics, Gaussian kernels of constant height are added for the entire course of a simulation. As a result, the system is eventually pushed to explore high free-energy regions and the estimate of the free energy calculated from the bias potential oscillates around the real value. In well-tempered metadynamics [49], the height of the Gaussian is decreased with simulation time according to:

$$W(k\tau) = W_0 \exp\left(-\frac{V(\vec{s}(q(k\tau)), k\tau)}{k_B \Delta T}\right),$$

where W_0 is an initial Gaussian height, ΔT an input parameter with the dimension of a temperature, and k_B the Boltzmann constant. With this rescaling of the Gaussian height, the bias potential smoothly converges in the long time limit, but it does not fully compensate the underlying free energy:

$$V(\vec{s}, t\to\infty) = -\frac{\Delta T}{T+\Delta T} F(\vec{s}) + C.$$

where T is the temperature of the system. In the long time limit, the CVs thus sample an ensemble at a temperature $T + \Delta T$ which is higher than the system temperature T. The parameter ΔT can be chosen to regulate the extent of free-energy exploration: $\Delta T = 0$ corresponds to standard molecular dynamics, $\Delta T \to \infty$ to standard metadynamics. In well-tempered metadynamics literature and in PLUMED, you will often encounter the term "bias factor" which is the ratio between the temperature of the CVs ($T + \Delta T$) and the system temperature (T):

$$\gamma = \frac{T+\Delta T}{T}.$$

The bias factor should thus be carefully chosen in order for the relevant free-energy barriers to be crossed efficiently in the time scale of the simulation.
Additional information can be found in the several review papers on metadynamics [133] [134] [135].
If you do not know exactly where you would like your collective variables to go, and just know (or suspect) that some variables have large free-energy barriers that hinder some conformational rearrangement or some chemical reaction, you can bias them using metadynamics. In this way, a time dependent, adaptive potential will be constructed that tends to disfavor visited configurations in the collective-variable space. The bias is usually built as a sum of Gaussian deposited in the already visited states.

11.38.20.3.1.2 Exercise 1 Now run a metadynamics simulation with the following input

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
METAD ARG=phi,psi HEIGHT=1.0 BIASFACTOR=10 SIGMA=0.35,0.35 PACE=100 GRID_MIN=-pi,-pi GRID_MAX=pi,pi
```

Thus, a single METAD line will contain all the metadynamics related options, such as Gaussian height (HEIGHT, here in kJ/mol), stride (PACE, here in number of time steps), bias factor (BIASFACTOR, here indicates that we are going to effectively boost the temperature of the collective variables by a factor 10), and width (SIGMA, an array with same size as the number of collective variables).
There are two additional keywords that are optional, namely GRID_MIN and GRID_MAX. These keywords sets the range of the collective variables and tell PLUMED to keep the bias potential stored on a grid. This affects speed but, in principle, not the accuracy of the calculation. You can try to remove those keywords and see the difference.
Now, run a metadynamics simulations and check the explored collective variable space. Results from a 200ps (100000 steps) trajectory in vacuum are shown in Figure munster-ala-traj-metad.
As you can see, exploration is greatly enhanced. Notice that the explored ensemble can be tuned using the bias factor γ. Larger γ implies that the system will explore states with higher free energy. As a rule of thumb, if you expect a barrier of the order of ΔG^*, a reasonable choice for the bias factor is $\gamma \approx \frac{\Delta G}{2k_B T}$.
Finally, notice that METAD potential depends on the previously visited trajectories. As such, when you restart a previous simulation, it should read the previously deposited HILLS file. This is automatically triggered by the RESTART keyword.

11.38.20.3.1.3 Exercise 2 In this exercise, we will run a well-tempered metadynamics simulation on alanine dipeptide in vacuum, using as CV the backbone dihedral angle phi. In order to run this simulation we need to prepare the PLUMED input file (plumed.dat) as follows.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate well-tempered metadynamics in phi depositing
# a Gaussian every 500 time steps, with initial height equal
# to 1.2 kJ/mol, bias factor equal to 10.0, and width to 0.35 rad

METAD ...
LABEL=metad
ARG=phi
PACE=500
HEIGHT=1.2
SIGMA=0.35
FILE=HILLS
BIASFACTOR=10.0
TEMP=300.0
GRID_MIN=-pi
GRID_MAX=pi
GRID_SPACING=0.1
... METAD

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

The syntax for the command METAD is simple. The directive is followed by a keyword ARG followed by the labels of the CVs on which the metadynamics potential will act. The keyword PACE determines the stride of Gaussian deposition in number of time steps, while the keyword HEIGHT specifies the height of the Gaussian in kJ/mol. For each CVs, one has to specified the width of the Gaussian by using the keyword SIGMA. Gaussian will be written to the file indicated by the keyword FILE.
The bias potential will be stored on a grid, whose boundaries are specified by the keywords GRID_MIN and GR↩ ID_MAX. Notice that you should provide either the number of bins for every collective variable (GRID_BIN) or the desired grid spacing (GRID_SPACING). In case you provide both PLUMED will use the most conservative choice (highest number of bins) for each dimension. In case you do not provide any information about bin size (neither GRID_BIN nor GRID_SPACING) and if Gaussian width is fixed PLUMED will use 1/5 of the Gaussian width as grid spacing. This default choice should be reasonable for most applications.
Once the PLUMED input file is prepared, one has to run Gromacs with the option to activate PLUMED and read the input file:

```
> gmx_mpi mdrun -s ../TOPO/topolA.tpr -plumed plumed.dat -nsteps 5000000
```

During the metadynamics simulation, PLUMED will create two files, named COLVAR and HILLS. The COLVAR file contains all the information specified by the PRINT command, in this case the value of the CVs every 10 steps of

simulation, along with the current value of the metadynamics bias potential. We can use COLVAR to visualize the behavior of the CV during the simulation:
By inspecting Figure munster-metad-phi-fig, we can see that the system is initialized in one of the two metastable states of alanine dipeptide. After a while (t=0.1 ns), the system is pushed by the metadynamics bias potential to visit the other local minimum. As the simulation continues, the bias potential fills the underlying free-energy landscape, and the system is able to diffuse in the entire phase space.
The HILLS file contains a list of the Gaussian kernels deposited along the simulation. If we give a look at the header of this file, we can find relevant information about its content:

```
#! FIELDS time phi psi sigma_phi sigma_psi height biasf
#! SET multivariate false
#! SET min_phi -pi
#! SET max_phi pi
#! SET min_psi -pi
#! SET max_psi pi
```

The line starting with FIELDS tells us what is displayed in the various columns of the HILLS file: the time of the simulation, the value of phi and psi, the width of the Gaussian in phi and psi, the height of the Gaussian, and the bias factor. We can use the HILLS file to visualize the decrease of the Gaussian height during the simulation, according to the well-tempered recipe:
If we look carefully at the scale of the y-axis, we will notice that in the beginning the value of the Gaussian height is higher than the initial height specified in the input file, which should be 1.2 kJ/mol. In fact, this column reports the height of the Gaussian scaled by the pre-factor that in well-tempered metadynamics relates the bias potential to the free energy. In this way, when we will use sum_hills, the sum of the Gaussian kernels deposited will directly provide the free-energy, without further rescaling needed (see below).
One can estimate the free energy as a function of the metadynamics CVs directly from the metadynamics bias potential. In order to do so, the utility sum_hills should be used to sum the Gaussian kernels deposited during the simulation and stored in the HILLS file.
To calculate the free energy as a function of phi, it is sufficient to use the following command line:

```
> plumed sum_hills --hills HILLS
```

The command above generates a file called fes.dat in which the free-energy surface as function of phi is calculated on a regular grid. One can modify the default name for the free energy file, as well as the boundaries and bin size of the grid, by using the following options of sum_hills :

```
--outfile - specify the outputfile for sumhills
--min - the lower bounds for the grid
--max - the upper bounds for the grid
--bin - the number of bins for the grid
--spacing - grid spacing, alternative to the number of bins
```

The result should look like this:
To assess the convergence of a metadynamics simulation, one can calculate the estimate of the free energy as a function of simulation time. At convergence, the reconstructed profiles should be similar. The option –stride should be used to give an estimate of the free energy every N Gaussian kernels deposited, and the option –mintozero can be used to align the profiles by setting the global minimum to zero. If we use the following command line:

```
> plumed sum_hills --hills HILLS --stride 100 --mintozero
```

one free energy is calculated every 100 Gaussian kernels deposited, and the global minimum is set to zero in all profiles. The resulting plot should look like the following:
To assess the convergence of the simulation more quantitatively, we can calculate the free-energy difference between the two local minima of the free energy along phi as a function of simulation time. We can use following script to integrate the multiple free-energy profiles in the two basins defined by the following intervals in phi space: basin A, -3<phi<-1, basin B, 0.5<phi<1.5.

```
# number of free-energy profiles
nfes= # put here the number of profiles
# minimum of basin A
minA=-3
# maximum of basin A
maxA=1
# minimum of basin B
minB=0.5
```

```
# maximum of basin B
maxB=1.5
# temperature in energy units
kbt=2.5

for((i=0;i<nfes;i++))
do
 # calculate free-energy of basin A
 A=`awk 'BEGIN{tot=0.0}{if($1!="#!" && $1>min && $1<max)tot+=exp(-$2/kbt)}END{print -kbt*log(tot)}' min=${minA
 # and basin B
 B=`awk 'BEGIN{tot=0.0}{if($1!="#!" && $1>min && $1<max)tot+=exp(-$2/kbt)}END{print -kbt*log(tot)}' min=${minB
 # calculate difference
 Delta=$(echo "${A} - ${B}" | bc -l)
 # print it
 echo $i $Delta
done
```

notice that nfes should be set to the number of profiles (free-energy estimates at different times of the simulation) generated by the option –stride of sum_hills.
This analysis, along with the observation of the diffusive behavior in the CVs space, suggest that the simulation is converged.

11.38.20.3.1.4 Exercise 3 In this exercise, we will run a well-tempered metadynamics simulation on alanine dipeptide in vacuum, using as CV the backbone dihedral angle psi. In order to run this simulation we need to prepare the PLUMED input file (plumed.dat) as follows.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
# set up two variables for Phi and Psi dihedral angles
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Activate well-tempered metadynamics in psi depositing
# a Gaussian every 500 time steps, with initial height equal
# to 1.2 kJ/mol, bias factor equal to 10.0, and width to 0.35 rad

METAD ...
LABEL=metad
ARG=psi
PACE=500
HEIGHT=1.2
SIGMA=0.35
FILE=HILLS
BIASFACTOR=10.0
TEMP=300.0
GRID_MIN=-pi
GRID_MAX=pi
GRID_SPACING=0.1
... METAD

# monitor the two variables and the metadynamics bias potential
PRINT STRIDE=10 ARG=phi,psi,metad.bias FILE=COLVAR
```

Once the PLUMED input file is prepared, one has to run Gromacs with the option to activate PLUMED and read the input file:

```
> gmx_mpi mdrun -s ../TOPO/topolA.tpr -plumed plumed.dat -nsteps 5000000
```

As we did in the previous exercise, we can use COLVAR to visualize the behavior of the CV during the simulation. Here we will plot at the same time the evolution of the metadynamics CV psi and of the other dihedral phi:
By inspecting Figure munster-metad-psi-phi-fig, we notice that something different happened compared to the previous exercise. At first the behavior of psi looks diffusive in the entire CV space. However, around t=1 ns, psi seems trapped in a region of the CV space in which it was previously diffusing without problems. The reason is that the non-biased CV phi after a while has jumped into a different local minima. Since phi is not directly biased, one has to wait for this (slow) degree of freedom to equilibrate before the free energy along psi can converge. Try to repeat the analysis done in the previous exercise (calculate the estimate of the free energy as a function of time and monitor the free-energy difference between basins) to assess the convergence of this metadynamics simulation.

11.38.20.3.2 Restraints

11.38.20.3.2.1 Biased sampling theory A system at temperature T samples conformations from the canonical ensemble:

$$P(q) \propto e^{-\frac{U(q)}{k_B T}}$$

. Here q are the microscopic coordinates and k_B is the Boltzmann constant. Since q is a highly dimensional vector, it is often convenient to analyze it in terms of a few collective variables (see Belfast tutorial: Analyzing CVs , Belfast tutorial: Adaptive variables I , and Belfast tutorial: Adaptive variables II). The probability distribution for a CV s is

$$P(s) \propto \int dq e^{-\frac{U(q)}{k_B T}} \delta(s - s(q))$$

This probability can be expressed in energy units as a free energy landscape $F(s)$:

$$F(s) = -k_B T \log P(s)$$

.
Now we would like to modify the potential by adding a term that depends on the CV only. That is, instead of using $U(q)$, we use $U(q) + V(s(q))$. There are several reasons why one would like to introduce this potential. One is to avoid that the system samples some un-desired portion of the conformational space. As an example, imagine that you want to study dissociation of a complex of two molecules. If you perform a very long simulation you will be able to see association and dissociation. However, the typical time required for association will depend on the size of the simulation box. It could be thus convenient to limit the exploration to conformations where the distance between the two molecules is lower than a given threshold. This could be done by adding a bias potential on the distance between the two molecules. Another example is the simulation of a portion of a large molecule taken out from its initial context. The fragment alone could be unstable, and one might want to add additional potentials to keep the fragment in place. This could be done by adding a bias potential on some measure of the distance from the experimental structure (e.g. on root-mean-square deviation).
Whatever CV we decide to bias, it is very important to recognize which is the effect of this bias and, if necessary, remove it a posteriori. The biased distribution of the CV will be

$$P'(s) \propto \int dq e^{-\frac{U(q)+V(s(q))}{k_B T}} \delta(s - s(q)) \propto e^{-\frac{V(s(q))}{k_B T}} P(s)$$

and the biased free energy landscape

$$F'(s) = -k_B T \log P'(s) = F(s) + V(s) + C$$

Thus, the effect of a bias potential on the free energy is additive. Also notice the presence of an undetermined constant C. This constant is irrelevant for what concerns free-energy differences and barriers, but will be important later when we will learn the weighted-histogram method. Obviously the last equation can be inverted so as to obtain the original, unbiased free-energy landscape from the biased one just subtracting the bias potential

$$F(s) = F'(s) - V(s) + C$$

Additionally, one might be interested in recovering the distribution of an arbitrary observable. E.g., one could add a bias on the distance between two molecules and be willing to compute the unbiased distribution of some torsional angle. In this case there is no straightforward relationship that can be used, and one has to go back to the relationship between the microscopic probabilities:

$$P(q) \propto P'(q) e^{\frac{V(s(q))}{k_B T}}$$

The consequence of this expression is that one can obtained any kind of unbiased information from a biased simulation just by weighting every sampled conformation with a weight

$$w \propto e^{\frac{V(s(q))}{k_B T}}$$

That is, frames that have been explored in spite of a high (disfavoring) bias potential V will be counted more than frames that has been explored with a less disfavoring bias potential.

11.38.20.3.2.2 Umbrella sampling theory Often in interesting cases the free-energy landscape has several local minima. If these minima have free-energy differences that are on the order of a few times k_BT they might all be relevant. However, if they are separated by a high saddle point in the free-energy landscape (i.e. a low probability region) than the transition between one and the other will take a lot of time and these minima will correspond to metastable states. The transition between one minimum and the other could require a time scale which is out of reach for molecular dynamics. In these situations, one could take inspiration from catalysis and try to favor in a controlled manner the conformations corresponding to the transition state.
Imagine that you know since the beginning the shape of the free-energy landscape $F(s)$ as a function of one CV s. If you perform a molecular dynamics simulation using a bias potential which is exactly equal to $-F(s)$, the biased free-energy landscape will be flat and barrierless. This potential acts as an "umbrella" that helps you to safely cross the transition state in spite of its high free energy.
It is however difficult to have an a priori guess of the free-energy landscape. We will see later how adaptive techniques such as metadynamics (Belfast tutorial: Metadynamics) can be used to this aim. Because of this reason, umbrella sampling is often used in a slightly different manner.
Imagine that you do not know the exact height of the free-energy barrier but you have an idea of where the barrier is located. You could try to just favor the sampling of the transition state by adding a harmonic restraint on the CV, e.g. in the form

$$V(s) = \frac{k}{2}(s - s_0)^2$$

. The sampled distribution will be

$$P'(q) \propto P(q) e^{\frac{-k(s(q)-s_0)^2}{2k_BT}}$$

For large values of k, only points close to s_0 will be explored. It is thus clear how one can force the system to explore only a predefined region of the space adding such a restraint. By combining simulations performed with different values of s_0, one could obtain a continuous set of simulations going from one minimum to the other crossing the transition state. In the next section we will see how to combine the information from these simulations.
If you want to just bring a collective variables to a specific value, you can use a simple restraint. Let's imagine that we want to force the Φ angle to visit a region close to $\Phi = \pi/2$. We can do it adding a restraint in Φ, with the following input

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
res: RESTRAINT ARG=phi AT=0.5pi KAPPA=5
PRINT ARG=phi,psi,res.bias
```

Notice that here we are printing a quantity named `res.bias`. We do this because RESTRAINT does not define a single value (that here would be theoretically named `res`) but a structure with several components. All biasing methods (including METAD) do so, as well as many collective variables (see e.g. DISTANCE used with COMP↩ ONENTS keyword). Printing the bias allows one to know how much a given snap shot was penalized. Also notice that PLUMED understands numbers in the for `{number}pi`. This is convenient when using torsions, since they are expressed in radians.
Now you can plot your trajectory with gnuplot and see the effect of KAPPA. You can also try different values of KAPPA. The stiffer the restraint, the less the collective variable will fluctuate. However, notice that a too large kappa could make the MD integrator unstable.

11.38.20.3.3 Moving restraints A restraint can also be modified as a function of time. For example, if you want to bring the system from one minimum to the other, you can use a moving restraint on Φ:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/munster.txt
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
# notice that a long line can be splitted with this syntax
MOVINGRESTRAINT ...
# also notice that a LABEL keyword can be used and is equivalent
# to adding the name at the beginning of the line with colon, as we did so far
  LABEL=res
  ARG=phi
  STEP0=0 AT0=-0.5pi KAPPA0=5
  STEP1=10000 AT1=0.5pi
...
PRINT ARG=phi,psi,res.work,res.phi_cntr FILE=colvar
```

Notice that here we are plotting a few new components, namely `work` and `phi_cntr`. The former gives the work performed in pulling the restraint, and the latter the position of the restraint. Notice that if pulling is slow enough one can compute free energy profile from the work. You can plot the putative free-energy landscape with

```
> gnuplot
# column 5 is res.phi_cntr
# column 4 is res.work
gnuplot> p "colvar" u 5:4
```

11.38.20.3.4 Using multiple replicas

Warning

Notice that multireplica simulations with PLUMED are fully supported with GROMACS, but only partly supported with other MD engines.

Some free-energy methods are intrinsically parallel and requires running several simultaneous simulations. This can be done with gromacs using the multi replica framework. That is, if you have 4 tpr files named topol0.tpr, topol1.tpr, topol2.tpr, topol3.tpr you can run 4 simultaneous simulations.

```
> mpirun -np 4 gmx_mpi mdrun -s topol.tpr -plumed plumed.dat -multi 4 -nsteps 500000
```

Each of the 4 replicas will open a different topol file, and GROMACS will take care of adding the replica number before the .tpr suffix. PLUMED deals with the extra number in a slightly different way. In this case, for example, PLUMED first look for a file named `plumed.dat.X`, where X is the number of the replica. In case the file is not found, then PLUMED looks for `plumed.dat`. If also this is not found, PLUMED will complain. As a consequence, if all the replicas should use the same input file it is sufficient to put a single `plumed.dat` file, but one has also the flexibility of using separate files named `plumed.dat.0`, `plumed.dat.1` etc. Finally, notice that the way PLUMED adds suffixes will change in version 2.2, and names will be `plumed.0.dat` etc.
Also notice that providing the flag `-replex` one can instruct gromacs to perform a replica exchange simulation. Namely, from time to time gromacs will try to swap coordinates among neighboring replicas and accept of reject the exchange with a Monte Carlo procedure which also takes into account the bias potentials acting on the replicas, even if different bias potentials are used in different replicas. That is, PLUMED allows to easily implement many forms of Hamiltonian replica exchange.

11.38.20.3.5 Using multiple restraints with replica exchange

11.38.20.3.5.1 Weighted histogram analysis method theory Let's now consider multiple simulations performed with restraints located in different positions. In particular, we will consider the i-th bias potential as V_i. The probability to observe a given value of the collective variable s is:

$$P_i(s) = \frac{P(s)e^{-\frac{V_i(s)}{k_BT}}}{\int ds' P(s')e^{-\frac{V_i(s')}{k_BT}}} = \frac{P(s)e^{-\frac{V_i(s)}{k_BT}}}{Z_i}$$

where

$$Z_i = \sum_q e^{-(U(q)+V_i(q))}$$

The likelihood for the observation of a sequence of snapshots $q_i(t)$ (where i is the index of the trajectory and t is time) is just the product of the probability of each of the snapshots. We use here the minus-logarithm of the likelihood (so that the product is converted to a sum) that can be written as

$$\mathcal{L} = -\sum_i \int dt \log P_i(s_i(t)) = \sum_i \int dt \left(-\log P(s_i(t)) + \frac{V_i(s_i(t))}{k_BT} + \log Z_i \right)$$

One can then maximize the likelihood by setting $\frac{\delta \mathcal{L}}{\delta P(\mathbf{s})} = 0$. After some boring algebra the following expression can be obtained

$$0 = \sum_i \int dt \left(-\frac{\delta_{\mathbf{s}_i(t),\mathbf{s}}}{P(\mathbf{s})} + \frac{e^{-\frac{V_i(\mathbf{s})}{k_BT}}}{Z_i} \right)$$

In this equation we aim at finding $P(s)$. However, also the list of normalization factors Z_i is unknown, and they should be found selfconsistently. Thus one can find the solution as

$$P(\mathbf{s}) \propto \frac{N(\mathbf{s})}{\sum_i \int dt \frac{e^{-\frac{V_i(\mathbf{s})}{k_BT}}}{Z_i}}$$

where Z is selfconsistently determined as

$$Z_i \propto \int ds' P(\mathbf{s}') e^{-\frac{V_i(\mathbf{s}')}{k_B T}}$$

These are the WHAM equations that are traditionally solved to derive the unbiased probability $P(s)$ by the combination of multiple restrained simulations. To make a slightly more general implementation, one can compute the weights that should be assigned to each snapshot, that turn out to be:

$$w_i(t) \propto \frac{1}{\sum_j \int dt \frac{e^{-\beta V_j(\mathbf{s}_i(t))}}{Z_j}}$$

The normalization factors can in turn be found from the weights as

$$Z_i \propto \frac{\sum_j \int dt e^{-\beta V_i(\mathbf{s}_j(t))} w_j(t)}{\sum_j \int dt w_j(t)}$$

This allows to straighforwardly compute averages related to other, non-biased degrees of freedom, and it is thus a bit more flexible. It is sufficient to precompute this factors w and use them to weight every single frame in the trajectory.

11.38.20.3.5.2 Exercise 4 In this exercise we will run multiple restraint simulations and learn how to reweight and combine data with WHAM to obtain free-energy profiles. We start with running in a replica-exchange scheme 32 simulations with a restraint on phi in different positions, ranging from -3 to 3. We will instruct gromacs to attempt an exchange between different simulations every 1000 steps.

```
nrep=32
dx=`echo "6.0 / ( $nrep - 1 )" | bc -l`

for((i=0;i<nrep;i++))
do
# center of the restraint
AT=`echo "$i * $dx - 3.0" | bc -l`

cat >plumed.dat.$i << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
#
# Impose an umbrella potential on phi
# with a spring constant of 200 kjoule/mol
# and centered in phi=AT
#
restraint-phi: RESTRAINT ARG=phi KAPPA=200.0 AT=$AT
# monitor the two variables and the bias potential
PRINT STRIDE=100 ARG=phi,psi,restraint-phi.bias FILE=COLVAR
EOF

# we initialize some replicas in A and some in B:
if((i%2==0)); then
  cp ../TOPO/topolA.tpr topol$i.tpr
else
  cp ../TOPO/topolB.tpr topol$i.tpr
fi
done

# run REM
mpirun -np $nrep gmx_mpi mdrun -plumed plumed.dat -s topol.tpr -multi $nrep -replex 1000 -nsteps 500000
```

To be able to combine data from all the simulations, it is necessary to have an overlap between statistics collected in two adjacent umbrellas.
Have a look at the plot of (phi,psi) for the different simulations to understand what is happening.
An often misunderstood fact about WHAM is that data of the different trajectories can be mixed and it is not necessary to keep track of which restraint was used to produce every single frame. Let's get the concatenated trajectory

```
> trjcat_mpi -cat -f traj*.xtc -o alltraj.xtc
```

Now we should compute the value of each of the bias potentials on the entire (concatenated) trajectory.

```
nrep=32
dx=`echo "6.0 / ( $nrep - 1 )" | bc -l`

for i in `seq 0 $(( $nrep - 1 ))`
do
# center of the restraint
AT=`echo "$i * $dx - 3.0" | bc -l`

cat >plumed.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17
restraint-phi: RESTRAINT ARG=phi KAPPA=200.0 AT=$AT

# monitor the two variables and the bias potential
PRINT STRIDE=100 ARG=phi,psi,restraint-phi.bias FILE=ALLCOLVAR.$i
EOF

plumed driver --mf_xtc alltraj.xtc --trajectory-stride=10 --plumed plumed.dat

done
```

It is very important that this script is consistent with the one used to generate the multiple simulations above. Now, single files named ALLCOLVAR.XX will contain on the fourth column the value of the bias centered in a given position, computed on the entire concatenated trajectory.
Next step is to compute the weights self-consistently solving the WHAM equations, using the python script "wham.↩ py" contained in the SCRIPTS directory. To use this code:

```
> ../SCRIPTS/wham.sh ALLCOLVAR.*
```

This script will produce several files. Let's visualize "phi_fes.dat", which contains the free energy as a function of phi, and compare this with the result previously obtained with metadynamics.

11.38.20.3.5.3 Exercise 5 In the previous exercise, we use multiple restraint simulations to calculate the free energy as a function of the dihedral phi. The resulting free energy was in excellent agreement with our previous metadynamics simulation. In this exercise we will repeat the same procedure for the dihedral psi. At the end of the steps defined above, we can plot the free energy "psi_fes.dat" and compare it with the reference profile calculated from a metadynamics simulations using both phi and psi as CVs.
We can easily spot from the plot above that something went wrong in this multiple restraint simulations, despite we used the very same approach we adopted for the phi dihedral. The problem here is that psi is a "bad" collective variable, and the system is not able to equilibrate the missing slow degree of freedom phi in the short time scale of the umbrella simulation (1 ns). In the metadynamics exercise in which we biased only psi, we detect problems by observing the behavior of the CV as a function of simulation time. How can we detect problems in multiple restraint simulations? This is slightly more complicated, but running this kind of simulation in a replica-exchange scheme offers a convenient way to detect problems.
The first thing we need to do is to demux the replica-exchange trajectories and reconstruct the continuous trajectories of the replicas across the different restraint potentials. In order to do so, we can use the following script:

```
> demux.pl md0.log
> trjcat_mpi -f traj*.xtc  -demux replica_index.xvg
```

This commands will generate 32 continuous trajectories, named XX_trajout.xtc. We will use the driver to calculate the value of the CVs phi and psi on these trajectories.

```
nrep=32

for i in `seq 0 $(( $nrep - 1 ))`
do

cat >plumed.dat << EOF
phi: TORSION ATOMS=5,7,9,15
psi: TORSION ATOMS=7,9,15,17

# monitor the two variables
PRINT STRIDE=100 ARG=phi,psi FILE=COLVARDEMUX.$i
EOF

plumed driver --mf_xtc ${i}_trajout.xtc --trajectory-stride=10 --plumed plumed.dat

done
```

The COLVARDEMUX.XX files will contain the value of the CVs on the demuxed trajectory. If we visualize these files we will notice that replicas sample the CVs space differently. In order for each umbrella to equilibrate the slow degrees of freedom phi, the continuous replicas must be ergodic and thus sample the same distribution in phi and psi.

Chapter 12

Performances

In this page we collect hints on how to use the features available in PLUMED to speed up your calculations. Please note that PLUMED performs many different tasks, it can calculate a number of different collective variables, functions of collective variables, bias, on-the-fly analysis, etc in a way that is compatible with a number of different molecular dynamics codes. This means that there cannot be a single strategy to speed up all the possible calculations.
Here you can find a step-by-step tutorial on optimizing PLUMED performances, discussing some of the topics below in more detail and using practical examples.
PLUMED makes use of MPI and OpenMP to parallelize some of its functions, try to always compile it with these features enabled. Furthermore, newer compilers with proper optimization flags can provide a dramatic boost to performances.
PLUMED collects atoms from an external code and sends back forces, so it is key to minimize the effect of PLUMED on highly parallel calculations to keep to the minimum the number of atoms used by PLUMED at every calculation step. The less is the number of atoms you need to send to PLUMED the less will be the overhead in the communication between PLUMED and the code.
In the following you can find specific strategies for specific calculations, these could help in taking the most by using PLUMED for your simulations.

- GROMACS and PLUMED with GPU
- Metadynamics
- Multiple time stepping
- Multicolvar
- Neighbor Lists
- OpenMP
- Secondary Structure
- Time your Input
- Making lepton library faster

Check also this tutorial: Optimizing PLUMED performance

12.1 GROMACS and PLUMED with GPU

Since version 4.6.x GROMACS can run in an hybrid mode making use of both your CPU and your GPU (either using CUDA or OpenCL for newer versions of GROMACS). The calculation of the short-range non-bonded interactions is performed on the GPU while long-range and bonded interactions are at the same time calculated on the CPU. By varying the cut-off for short-range interactions GROMACS can optimize the balance between GPU/CPU loading and obtain amazing performances.
GROMACS patched with PLUMED takes into account PLUMED in its load-balancing, adding the PLUMED timings to the one resulting from bonded interactions and long- range interactions. This means that the CPU/GPU balance will be optimized automatically to take into account PLUMED!

It is important to notice that the optimal setup to use GROMACS alone on the GPU or GROMACS + PLUMED can be different, try to change the number of MPI/OpenMP processes (OpenMP) used by GROMACS and PLUMED to find optimal performances. Remember that in GROMACS multiple MPI threads can use the same GPU:
i.e. if you have 4 cores and 2 GPU you can:

- use 2 MPI/2GPU/2OPENMP:

```
export PLUMED_NUM_THREADS=2
mpiexec -np 2 gmx_mpi mdrun -nb gpu -ntomp 2 -pin on -gpu_id 01
```

- use 4 MPI/2GPU:

```
export PLUMED_NUM_THREADS=1
mpiexec -np 4 gmx_mpi mdrun -nb gpu -ntomp 1 -pin on -gpu_id 0011
```

Of notice that since plumed 2.5 and gromacs 2018.3 the number of openMP threads can automatically set by gromacs (so PLUMED_NUM_THREADS is not needed, and the number of OpenMP threads used by plumed is set by -ntomp)

```
mpiexec -np 2 gmx_mpi mdrun -nb gpu -ntomp 2 -pin on -gpu_id 01
```

12.2 Metadynamics

Metadynamics can be sped up significantly using grids, which are activated setting the GRID_MIN and GRID_MAX keywords of METAD and PBMETAD. This makes addition of a hill to the list a bit slower (since the Gaussian has to be evaluated for many grid points) but the evaluation of the potential very fast. Since the former is usually done every few hundred steps, whereas the latter typically at every step, using grids will make the simulation faster in particular for long runs.
Notice that when restarting a simulation the history is read by default from a file and hills are added again to the grid. This allows one to change the grid boundaries upon restart. However, the first step after restart is usually very slow. Since PLUMED 2.3 you can also store the grid on a file and read it upon restart. This can be particularly useful if you perform many restarts and if your hills are large.
For the precise syntax, see METAD and PBMETAD

12.3 Multiple time stepping

By setting a STRIDE different from 1, you change how frequently an action is calculated. In the case of actions such as PRINT, this just means how frequently you dump some quantity on the disk. Notice that variables are only computed when necessary. Thus, if a variable is only appearing as the argument of a PRINT statement with STRIDE=10, it will be computed every 10 steps.
In a similar fashion, the STRIDE keyword can be used in a bias potential so as to apply the bias potential every few steps. In this case, forces from this bias potential are scaled up by a factor equal to STRIDE.
This technique can allow your simulation to run faster if you need the apply a bias potential on some very expensive collective variable. Consider the following input:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PerformancesPP.md
c1: COM ATOMS=1-1000
c2: COM ATOMS=1001-2000
d:  DISTANCE ATOMS=c1,c2
METAD ARG=d HEIGHT=1 SIGMA=0.1 BIASFACTOR=5 PACE=500
```

This performs a METAD simulation biasing the distance between two centers of mass. Since computing these centers requires a lot of atoms to be imported from the MD engine, it could slow down significantly the simulation. Notice that whereas the bias is changed every PACE=500 steps, it is applied every STRIDE step, where STRIDE=1 by default. The following input could lead to a significantly faster simulation at the price of a negligible systematic error

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PerformancesPP.md
c1: COM ATOMS=1-1000
c2: COM ATOMS=1001-2000
d:  DISTANCE ATOMS=c1,c2
METAD ARG=d HEIGHT=1 SIGMA=0.1 BIASFACTOR=5 PACE=500 STRIDE=2
```

Similarly, the STRIDE keyword can be used with other biases (e.g. RESTRAINT).
The technique is discussed in details here [140]. See also EFFECTIVE_ENERGY_DRIFT.

12.3.1 EFFECTIVE_ENERGY_DRIFT

This is part of the generic module

Print the effective energy drift
The method used to calculate the effective energy drift is described in Ref [140]

Examples

This is to monitor the effective energy drift for a metadynamics simulation on the Debye-Huckel energy. Since this variable is very expensive, it could be conveniently computed every second step.

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EFFECTIVE_ENERGY_DRIFT.tmp
dh: DHENERGY GROUPA=1-10 GROUPB=11-20 EPSILON=80.0 I=0.1 TEMP=300.0
METAD ARG=dh HEIGHT=0.5 SIGMA=0.1 PACE=500 STRIDE=2
EFFECTIVE_ENERGY_DRIFT PRINT_STRIDE=100 FILE=eff
```

This is to monitor if a restraint is too stiff

```
BEGIN_PLUMED_FILE working DATADIR=example-check/EFFECTIVE_ENERGY_DRIFT.tmp
d: DISTANCE ATOMS=10,20
RESTRAINT ARG=d KAPPA=100000 AT=0.6
EFFECTIVE_ENERGY_DRIFT PRINT_STRIDE=100 FILE=eff
```

Glossary of keywords and components

Compulsory keywords

STRIDE	(default=1) should be set to 1. Effective energy drift computation has to be active at each step.
FILE	file on which to output the effective energy drift.
PRINT_STRIDE	frequency to which output the effective energy drift on FILE

Options

ENSEMBLE	(default=off) Set to TRUE if you want to average over multiple replicas.
FMT	the format that should be used to output real numbers
RESTART	allows per-action setting of restart (YES/NO/AUTO)
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

12.4 Multicolvar

Whenever you have a multicolvar action such as:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PerformancesPP.md
COORDINATIONNUMBER SPECIES=1-100 SWITCH={RATIONAL R_0=1. D_MAX=3.0} MORE_THAN={RATIONAL R_0=6.0 NN=6 MM=12 D_0
```

You will get a colossal speedup by specifying the D_MAX keyword in all switching functions that act on distances. D_MAX tells PLUMED that the switching function is strictly zero if the distance is greater than this value. As a result PLUMED knows that it does not need to calculate these zero terms in what are essentially sums with a very large number of terms. In fact when D_MAX is set PLUMED uses linked lists when calculating these coordination numbers, which is what gives you such a dramatic increase in performance.

12.5 Neighbor Lists

Collective variables that can be speed up making us of neighbor lists:

- COORDINATION
- DHENERGY
- PATHMSD

By tuning the cut-off for the neighbor list and the frequency for the recalculation of the list it is possible to balance between accuracy and performances.
Notice that for COORDINATION and DHENERGY using a neighbor list could imply that a smaller number of atoms are requested to the host MD engine. This is typically true when considering COORDINATION of a small number of atoms (e.g. a ligand) again many atoms (e.g. water). When the neighbor list is used, only the water atoms close to the ligand will be requested at each step.

12.6 OpenMP

PLUMED is partly parallelized using OpenMP. This should be enabled by default if your compiler supports it, and can be disabled with `--disable-openmp`.. At runtime, you should set the environment variable PLUMED_N↩UM_THREADS to the number of threads you wish to use with PLUMED. The number of OpenMP threads can be set either by the MD code, if implemented in the patch, or generally by setting PLUMED_NUM_THREADS. If they are not set openmp will be disabled at runtime.
E.g., to run with gromacs you can do:

```
export PLUMED_NUM_THREADS=8
mdrun -plumed
```

or as well

```
mdrun -plumed -ntomp 8
```

In the first case the number of OpenMP threads used by plumed is 8 while the one used by gromacs can be 1 or something else, this is usually sub optimal. In the second case GROMACS and plumed will use the same number of OpenMP threads.
Notice that:

- This option is likely to improve the performance, but could also slow down the code in some case.
- Results could be slightly different because of numerical round off and different order in summations. This should be harmless.
- The optimum number of threads is not necessary "all of them", nor should be equal to the number of threads used to parallelize MD.
- Only a few CVs are parallelized with openMP (currently, COORDINATION and DHENERGY).
- You might want to tune also the environmental variable PLUMED_CACHELINE_SIZE, by default 512, to set the size of cache lines on your machine. This is used by PLUMED to decrease the number of threads to be used in each loop so as to avoid clashes in memory access. This variable is expected to affect performance only, not results.

12.7 Secondary Structure

Secondary Structure collective variables (ALPHARMSD, PARABETARMSD and ANTIBETARMSD) can be particularly demanding if you want to calculate them for all the residues of a protein. This is particularly true for the calculation of beta structures.
The FIRST thing to speed up PARABETARMSD and ANTIBETARMSD is to use the keyword STRANDS_CUTOFF (i.e. STRANDS_CUTOFF=1), in this way only a subset of possible fragments, the one less than 1.0 nm apart, are used in the calculation.
The metric used to calculate the distance from ideal secondary structure elements can also influence the performances, try to use TYPE=OPTIMAL or TYPE=OPTIMAL-FAST instead of TYPE=DRMSD.
At last, try to reduce the number of residues in the calculation.

12.8 Time your Input

Once you have prepared your plumed input file you can run a test simulation, or use driver, to see which collective variable, function, bias or analysis is consuming more time and can thus be the target for a different definition (use less atoms, change relevant parameters, or just use something else)
To have an accurate timing of your input you can use the DEBUG DETAILED_TIMERS.

12.9 Making lepton library faster

In case you are using a lot of CUSTOM functions or switching functions, notice that these commands depend on the lepton library that is included in PLUMED. This library replaces libmatheval since PLUMED 2.5, and by itself it is significantly faster than libmatheval. However, you can make it even faster using a just-in-time compiler. As of PLUMED 2.6, the correct version of ASMJIT is embedded in PLUMED. As of PLUMED 2.8, ASMJIT is enabled by default on supported architectures (X86/X64). You can disable it at runtime setting the environment variable PLUMED_USE_ASMJIT:

```
export PLUMED_USE_ASMJIT=no
```

In some case using a custom expression is almost as fast as using a hard-coded function. For instance, with an input that contained the following lines:

```
BEGIN_PLUMED_FILE working DATADIR=example-check/PerformancesPP.md
c: COORDINATION GROUPA=1-108 GROUPB=1-108 R_0=1
d_fast: COORDINATION GROUPA=1-108 GROUPB=1-108 SWITCH={CUSTOM FUNC=1/(1+x2^3) R_0=1}
```

I (GB) obtained the following timings (on a Macbook laptop):

```
...
PLUMED: 4A  1 c                                          108      0.126592      0.001172      0.000701      0.0025
PLUMED: 4A  2 d_fast                                      108      0.135210      0.001252      0.000755      0.002
...
```

Notice the usage of x2 as a variable for the switching function (see switchingfunction), which avoids an unnecessary square root calculation (this is done automatically by the hard-coded switching functions when you use only even powers). The asmjit calculation (d_fast) takes less than 10% more than the hard-coded one (c).

Chapter 13

Index of Actions

The following page contains an alphabetically ordered list of all the Actions and command line tools that are available in PLUMED 2. For lists of Actions classified in accordance with the particular tasks that are being performed see:

- Collective Variables tells you about the ways that you can calculate functions of the positions of the atoms.
- Analysis tells you about the various forms of analysis you can run on trajectories using PLUMED.
- Bias tells you about the methods that you can use to bias molecular dynamics simulations with PLUMED.

13.1 Full list of actions

ABMD	BIAS	Adds a ratchet-and-pawl like restraint on one or more variables.
ADAPTIVE_PATH	COLVAR	Compute path collective variables that adapt to the lowest free energy path connecting states A and B.
ALIGNED_MATRIX	MATRIX	Adjacency matrix in which two molecule are adjacent if they are within a certain cutoff and if they have the same orientation.
ALPHABETA	COLVAR	Measures a distance including pbc between the instantaneous values of a set of torsional angles and set of reference values.
ALPHARMSD	COLVAR	Probe the alpha helical content of a protein structure.
ANGLE	COLVAR	Calculate an angle.
ANGLES	MCOLVAR	Calculate functions of the distribution of angles .
ANN	ANNMOD_Function	Calculates the ANN-function.
ANTIBETARMSD	COLVAR	Probe the antiparallel beta sheet content of your protein structure.
AROUND	VOLUMES	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.
AVERAGE	GRIDCALC	Calculate the ensemble average of a collective variable
BF_CHEBYSHEV	VES_BASISF	Chebyshev polynomial basis functions.

BF_COMBINED	VES_BASISF	Combining other basis functions types
BF_COSINE	VES_BASISF	Fourier cosine basis functions.
BF_CUBIC_B_SPLINES	VES_BASISF	Cubic B spline basis functions.
BF_CUSTOM	VES_BASISF	Basis functions given by arbitrary mathematical expressions.
BF_FOURIER	VES_BASISF	Fourier basis functions.
BF_GAUSSIANS	VES_BASISF	Gaussian basis functions.
BF_LEGENDRE	VES_BASISF	Legendre polynomials basis functions.
BF_POWERS	VES_BASISF	Polynomial power basis functions.
BF_SINE	VES_BASISF	Fourier sine basis functions.
BF_WAVELETS	VES_BASISF	Daubechies Wavelets basis functions.
BIASVALUE	BIAS	Takes the value of one variable and use it as a bias
BOND_DIRECTIONS	MCOLVAR	Calculate the vectors connecting atoms that are within cutoff defined using a switching function.
BRIDGE	MCOLVAR	Calculate the number of atoms that bridge two parts of a structure
CALIBER	ISDB_BIAS	Add a time-dependent, harmonic restraint on one or more variables.
CAVITY	VOLUMES	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a box defined by the positions of four atoms.
CELL	COLVAR	Calculate the components of the simulation cell
CENTER	VATOM	Calculate the center for a group of atoms, with arbitrary weights.
CENTER_OF_MULTICOLVAR	VATOM	Calculate a a weighted average position based on the value of some multicolvar.
CLASSICAL_MDS	DIMRED	Create a low-dimensional projection of a trajectory using the classical multidimensional scaling algorithm.
CLUSTER_DIAMETER	CONCOMP	Print out the diameter of one of the connected components
CLUSTER_DISTRIBUTION	CONCOMP	Calculate functions of the distribution of properties in your connected components.
CLUSTER_NATOMS	CONCOMP	Gives the number of atoms in the connected component
CLUSTER_PROPERTIES	CONCOMP	Calculate properties of the distribution of some quantities that are part of a connected component
CLUSTER_WITHSURFACE	MATRIXF	Take a connected component that was found using a clustering algorithm and create a new cluster that contains those atoms that are in the cluster together with those atoms that are within a certain cutoff of the cluster.

COLLECT_FRAMES	ANALYSIS	This allows you to convert a trajectory and a dissimilarity matrix into a dissimilarity object
COLUMNSUMS	MATRIXF	Sum the columns of a contact matrix
COM	VATOM	Calculate the center of mass for a group of atoms.
COMBINE	FUNCTION	Calculate a polynomial combination of a set of other variables.
COMMITTOR	PRINTANALYSIS	Does a committor analysis.
CONSTANT	COLVAR	Return one or more constant quantities with or without derivatives.
CONTACTMAP	COLVAR	Calculate the distances between a number of pairs of atoms and transform each distance by a switching function.
CONTACT_MATRIX	MATRIX	Adjacency matrix in which two atoms are adjacent if they are within a certain cutoff.
CONVERT_TO_FES	GRIDANALYSIS	Convert a histogram, H(x), to a free energy surface using F(x) = -k_B T ln H(x).
COORDINATION	COLVAR	Calculate coordination numbers.
COORDINATIONNUMBER	MCOLVAR	Calculate the coordination numbers of atoms so that you can then calculate functions of the distribution of coordination numbers such as the minimum, the number less than a certain quantity and so on.
CS2BACKBONE	ISDB_COLVAR	Calculates the backbone chemical shifts for a protein.
CUSTOM	FUNCTION	Calculate a combination of variables using a custom expression.
DEBUG	GENERIC	Set some debug options.
DENSITY	MCOLVAR	Calculate functions of the density of atoms as a function of the box. This allows one to calculate the number of atoms in half the box.
DFSCLUSTERING	MATRIXF	Find the connected components of the matrix using the depth first search clustering algorithm.
DHENERGY	COLVAR	Calculate Debye-Huckel interaction energy among GROUPA and GR↩ OUPB.
DIHCOR	COLVAR	Measures the degree of similarity between dihedral angles.
DIMER	COLVAR	This CV computes the dimer interaction energy for a collection of dimers.
DIPOLE	COLVAR	Calculate the dipole moment for a group of atoms.
DISTANCE	COLVAR	Calculate the distance between a pair of atoms.

DISTANCES	MCOLVAR	Calculate the distances between one or many pairs of atoms. You can then calculate functions of the distribution of distances such as the minimum, the number less than a certain quantity and so on.
DISTANCE_FROM_CONTOUR	COLVAR	Calculate the perpendicular distance from a Willard-Chandler dividing surface.
DRMSD	DCOLVAR	Calculate the distance RMSD with respect to a reference structure.
DRR	EABFMOD_BIAS	Used to performed extended-system adaptive biasing force(e↩ ABF)
DUMPATOMS	PRINTANALYSIS	Dump selected atoms on a file.
DUMPCUBE	GRIDANALYSIS	Output a three dimensional grid using the Gaussian cube file format.
DUMPDERIVATIVES	PRINTANALYSIS	Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).
DUMPFORCES	PRINTANALYSIS	Dump the force acting on one of a values in a file.
DUMPGRAPH	CONCOMP	Write out the connectivity of the nodes in the graph in dot format.
DUMPGRID	GRIDANALYSIS	Output the function on the grid to a file with the PLUMED grid format.
DUMPMASSCHARGE	PRINTANALYSIS	Dump masses and charges on a selected file.
DUMPMULTICOLVAR	PRINTANALYSIS	Dump atom positions and multicolvar on a file.
DUMPPROJECTIONS	PRINTANALYSIS	Dump the derivatives with respect to the input parameters for one or more objects (generally CVs, functions or biases).
ECV_CUSTOM	OPES_EXPANSION_CV	Use some given CVs as a set of expansion collective variables (EC↩ Vs).
ECV_LINEAR	OPES_EXPANSION_CV	Linear expansion, according to a parameter lambda.
ECV_MULTITHERMAL	OPES_EXPANSION_CV	Expand a simulation to sample multiple temperatures simultaneously.
ECV_MULTITHERMAL_MULTIBARIC	OPES_EXPANSION_CV	Expand a simulation to sample multiple temperatures and pressures.
ECV_UMBRELLAS_FILE	OPES_EXPANSION_CV	Target a multiumbrella ensemble, by combining systems each with a parabolic bias potential at a different location.
ECV_UMBRELLAS_LINE	OPES_EXPANSION_CV	Target a multiumbrella ensemble, by combining systems each with a parabolic bias potential at a different location.
EDS	EDSMOD_BIAS	Add a linear bias on a set of observables.

EEFSOLV	COLVAR	Calculates EEF1 solvation free energy for a group of atoms.
EFFECTIVE_ENERGY_DRIFT	GENERIC	Print the effective energy drift
EMMI	ISDB_COLVAR	Calculate the fit of a structure or ensemble of structures with a cryo-EM density map.
ENDPLUMED	GENERIC	Terminate plumed input.
ENERGY	COLVAR	Calculate the total potential energy of the simulation box.
ENSEMBLE	FUNCTION	Calculates the replica averaging of a collective variable over multiple replicas.
ENVIRONMENTSIMILARITY	MCOLVAR	Measure how similar the environment around atoms is to that found in some reference crystal structure.
ERMSD	COLVAR	Calculate eRMSD with respect to a reference structure.
EUCLIDEAN_DISSIMILARITIES	ANALYSIS	Calculate the matrix of dissimilarities between a trajectory of atomic configurations.
EXTENDED_LAGRANGIAN	BIAS	Add extended Lagrangian.
EXTERNAL	BIAS	Calculate a restraint that is defined on a grid that is read during start up
EXTRACV	COLVAR	Allow PLUMED to use collective variables computed in the MD engine.
FAKE	COLVAR	This is a fake colvar container used by cltools or various other actions that supports input and period definitions
FCCUBIC	MCOLVAR	Measure how similar the environment around atoms is to that found in a FCC structure.
FIND_CONTOUR	GRIDANALYSIS	Find an isocontour in a smooth function.
FIND_CONTOUR_SURFACE	GRIDANALYSIS	Find an isocontour by searching along either the x, y or z direction.
FIND_SPHERICAL_CONTOUR	GRIDANALYSIS	Find an isocontour in a three dimensional grid by searching over a Fibonacci sphere.
FISST	FISSTMOD_BIAS	Compute and apply the optimal linear force on an observable to enhance sampling of conformational distributions over a range of applied forces.
FIT_TO_TEMPLATE	GENERIC	This action is used to align a molecule to a template.
FIXEDATOM	VATOM	Add a virtual atom in a fixed position.
FLUSH	GENERIC	This command instructs plumed to flush all the open files with a user specified frequency.Notice that all files are flushed anyway every 10000 steps.

FOURIER_TRANSFORM	GRIDANALYSIS	Compute the Discrete Fourier Transform (DFT) by means of FFTW of data stored on a 2D grid.
FRET	ISDB_COLVAR	Calculates the FRET efficiency between a pair of atoms.The efficiency is calculated using the Forster relation:
FUNCPATHGENERAL	FUNCTION	This function calculates path collective variables (PCVs) using an arbitrary combination of collective variables.
FUNCPATHMSD	FUNCTION	This function calculates path collective variables.
FUNCSUMHILLS	FUNCTION	This function is intended to be called by the command line tool sum_hills. It is meant to integrate a HILLS file or an HILLS file interpreted as a histogram in a variety of ways. It is, therefore, not expected that you use this during your dynamics (it will crash!)
FUNNEL	FUNNELMOD_BIAS	Calculate a funnel-shape restraint potential that is defined on a grid that is read during the setup.
FUNNEL_PS	FUNNELMOD_COLVAR	FUNNEL_PS implements the Funnel-Metadynamics (FM) technique in PLUMED 2.
FUSIONPOREEXPANSIONP	MEMBRANEFUSIONMOD_CO↩ LVAR	A CV for inducing the expansion of a fusion pore from a nucleated fusion pore.
FUSIONPORENUCLEATIONP	MEMBRANEFUSIONMOD_CO↩ LVAR	A CV for inducing the nucleation of the fusion pore from a hemifusion stalk.
GHBFIX	COLVAR	Calculate the GHBFIX interaction energy among GROUPA and G↩ ROUPBusing a potential defined in Kuhrova et al., Improving the performance of the AMBER RNA force field bytuning the hydrogen-bonding interactions, JCTC, 2019. Essentially it is a switching function being -1 for small distances and 0 for large distances with a smooth interpolation in the middle. This can be scaled as desired by specifying interaction scaling parameters and energy units.
GHOST	VATOM	Calculate the absolute position of a ghost atom with fixed coordinates in the local reference frame formed by three atoms.
GPROPERTYMAP	COLVAR	Property maps but with a more flexible framework for the distance metric being used.
GRADIENT	MCOLVARF	Calculate the gradient of the average value of a multicolvar value

GRID_TO_XYZ	GRIDANALYSIS	Output the function on the grid to an xyz file
GROUP	GENERIC	Define a group of atoms so that a particular list of atoms can be referenced with a single label in definitions of CVs or virtual atoms.
GYRATION	COLVAR	Calculate the radius of gyration, or other properties related to it.
HBOND_MATRIX	MATRIX	Adjacency matrix in which two atoms are adjacent if there is a hydrogen bond between them.
HBPAMM_MATRIX	MATRIX	Adjacency matrix in which two electronegative atoms are adjacent if they are hydrogen bonded
HBPAMM_SH	MCOLVAR	Number of HBPAMM hydrogen bonds formed by each hydrogen atom in the system
HISTOGRAM	GRIDCALC	Accumulate the average probability density along a few CVs from a trajectory.
INCLUDE	GENERIC	Includes an external input file, similar to #include in C preprocessor.
INCYLINDER	VOLUMES	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.
INENVELOPE	VOLUMES	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a region where the density of a certain type of atom is high.
INPLANEDISTANCES	MCOLVAR	Calculate distances in the plane perpendicular to an axis
INSPHERE	VOLUMES	This quantity can be used to calculate functions of the distribution of collective variables for the atoms that lie in a particular, user-specified part of of the cell.
INTEGRATE_GRID	GRIDANALYSIS	Calculate the total integral of the function on the input grid
INTERMOLECULARTORSIONS	MCOLVARF	Calculate torsion angles between vectors on adjacent molecules
INTERPOLATE_GRID	GRIDANALYSIS	Interpolate a smooth function stored on a grid onto a grid with a smaller grid spacing.
JCOUPLING	ISDB_COLVAR	Calculates ${}^3J$ coupling constants for a dihedral angle.
LANDMARK_SELECT_FPS	LANDMARKS	Select a set of landmarks using farthest point sampling.
LANDMARK_SELECT_RANDOM	LANDMARKS	Select a random set of landmarks from a large set of configurations.
LANDMARK_SELECT_STAGED	LANDMARKS	Select a set of landmarks using the staged algorithm.

LANDMARK_SELECT_STRIDE	LANDMARKS	Select every kth landmark from the trajectory.
LOAD	GENERIC	Loads a library, possibly defining new actions.
LOCALENSEMBLE	FUNCTION	Calculates the average over multiple arguments.
LOCAL_AVERAGE	MCOLVARF	Calculate averages over spherical regions centered on atoms
LOCAL_Q3	MCOLVARF	Calculate the local degree of order around an atoms by taking the average dot product between the q_3 vector on the central atom and the q_3 vector on the atoms in the first coordination sphere.
LOCAL_Q4	MCOLVARF	Calculate the local degree of order around an atoms by taking the average dot product between the q_4 vector on the central atom and the q_4 vector on the atoms in the first coordination sphere.
LOCAL_Q6	MCOLVARF	Calculate the local degree of order around an atoms by taking the average dot product between the q_6 vector on the central atom and the q_6 vector on the atoms in the first coordination sphere.
LOGMFD	LOGMFDMOD_BIAS	Used to perform LogMFD, LogPD, and TAMD/d-AFED.
LOWER_WALLS	BIAS	Defines a wall for the value of one or more collective variables, which limits the region of the phase space accessible during the simulation.
LWALLS	MCOLVARB	Add LOWER_WALLS restraints on all the multicolvar values
MATHEVAL	FUNCTION	An alias to the CUSTOM function.
MAXENT	BIAS	Add a linear biasing potential on one or more variables that satisfies a maximum entropy principle.
MAZE_LOSS	MAZE_LOSS	
MAZE_MEMETIC_SAMPLING	MAZE_OPTIMIZER	
MAZE_OPTIMIZER_BIAS	MAZE_BIAS	
MAZE_RANDOM_ACCELERATION_MD	MAZE_OPTIMIZER	
MAZE_RANDOM_WALK	MAZE_OPTIMIZER	
MAZE_SIMULATED_ANNEALING	MAZE_OPTIMIZER	
MAZE_STEERED_MD	MAZE_OPTIMIZER	
MCOLV_COMBINE	MCOLVARF	Calculate linear combinations of multiple multicolvars
MCOLV_PRODUCT	MCOLVARF	Calculate a product of multiple multicolvars
MEMFUSIONP	MEMBRANEFUSIONMOD_CO↩LVAR	Calculate a CV that can induce the formation of the hemifusion stalk between two initially flat and planar bilayers.
METAD	BIAS	Used to performed metadynamics on one or more collective variables.

METAINFERENCE	ISDB_BIAS	Calculates the Metainference energy for a set of experimental data.
MFILTER_BETWEEN	MFILTERS	This action can be used to filter the colvar values calculated by a mcolvso that one can compute the mean and so on for only those multicolvars within a certain range.
MFILTER_LESS	MFILTERS	This action can be used to filter the distribution of colvar values in a mcolvso that one can compute the mean and so on for only those multicolvars less than a tolerance.
MFILTER_MORE	MFILTERS	This action can be used to filter the distribution of colvar values in a mcolvso that one can compute the mean and so on for only those multicolvars more than a tolerance.
MOLECULES	MCOLVAR	Calculate the vectors connecting a pair of atoms in order to represent the orientation of a molecule.
MOLINFO	TOPOLOGY	This command is used to provide information on the molecules that are present in your system.
MOVINGRESTRAINT	BIAS	Add a time-dependent, harmonic restraint on one or more variables.
MTRANSFORM_BETWEEN	MTRANSFORMS	This action can be used to transform the colvar values calculated by a MultiColvar using a histogram bead
MTRANSFORM_LESS	MTRANSFORMS	This action can be used to transform the colvar values calculated by a multicovar using a switching function
MTRANSFORM_MORE	MTRANSFORMS	This action can be used to transform the colvar values calculated by a multicolvar using one minus a switching function
MULTICOLVARDENS	GRIDCALC	Evaluate the average value of a multicolvar on a grid.
MULTI_RMSD	DCOLVAR	Calculate the RMSD distance moved by a number of separated domains from their positions in a reference structure.
NLINKS	MCOLVARF	Calculate number of pairs of atoms/molecules that are linked
NOE	ISDB_COLVAR	Calculates NOE intensities as sums of 1/r^6, also averaging over multiple equivalent atoms or ambiguous NOE.
OPES_EXPANDED	OPES_BIAS	On-the-fly probability enhanced sampling with expanded ensembles for the target distribution.
OPES_METAD	OPES_BIAS	On-the-fly probability enhanced sampling with metadynamics-like target distribution.

OPES_METAD_EXPLORE	OPES_BIAS	On-the-fly probability enhanced sampling with well-tempered target distribution in exploreation mode.
OPT_ADAM	VES_OPTIMIZER	Adaptive moment estimation (AD↩ AM) optimizer.
OPT_AVERAGED_SGD	VES_OPTIMIZER	Averaged stochastic gradient decent with fixed step size.
OPT_DUMMY	VES_OPTIMIZER	Dummy optimizer for debugging.
OPT_ROBBINS_MONRO_SGD	VES_OPTIMIZER	Robbins-Monro stochastic gradient decent.
OUTPUT_ANALYSIS_DATA_TO_COLVAR	ANALYSIS	This can be used to output the data that has been stored in an Analysis object.
OUTPUT_ANALYSIS_DATA_TO_PDB	ANALYSIS	This can be used to output the data that has been stored in an Analysis object.
OUTPUT_CLUSTER	CONCOMP	Output the indices of the atoms in one of the clusters identified by a clustering object
OUTPUT_PCA_PROJECTION	DIMRED	This is used to output the projection calculated by principle component analysis
PAMM	MCOLVARF	Probabilistic analysis of molecular motifs.
PARABETARMSD	COLVAR	Probe the parallel beta sheet content of your protein structure.
PATH	COLVAR	Path collective variables with a more flexible framework for the distance metric being used.
PATHMSD	COLVAR	This Colvar calculates path collective variables.
PBMETAD	BIAS	Used to performed Parallel Bias metadynamics.
PCA	DIMRED	Perform principal component analysis (PCA) using either the positions of the atoms a large number of collective variables as input.
PCARMSD	DCOLVAR	Calculate the PCA components for a number of provided eigenvectors and an average structure.
PCAVARS	COLVAR	Projection on principal component eigenvectors or other high dimensional linear subspace
PCS	ISDB_COLVAR	Calculates the Pseudo-contact shift of a nucleus determined by the presence of a metal ion susceptible to anisotropic magnetization.
PIECEWISE	FUNCTION	Compute a piece wise straight line through its arguments that passes through a set of ordered control points.
PIV	PIVMOD_COLVAR	Calculates the PIV-distance.

PLANES	MCOLVAR	Calculate the plane perpendicular to two vectors in order to represent the orientation of a planar molecule.
PLUMED	GENERIC	Embed a separate PLUMED instance.
POLYMER_ANGLES	MCOLVARF	Calculate a function to investigate the relative orientations of polymer angles
POSITION	COLVAR	Calculate the components of the position of an atom.
PRE	ISDB_COLVAR	Calculates the Paramagnetic Resonance Enhancement intensity ratio between a spin label atom and a list of atoms .
PRINT	PRINTANALYSIS	Print quantities to a file.
PRINT_DISSIMILARITY_MATRIX	ANALYSIS	Print the matrix of dissimilarities between a trajectory of atomic configurations.
PROJECTION_ON_AXIS	COLVAR	Calculate a position based on the projection along and extension from a defined axis.
PROJECT_ALL_ANALYSIS_DATA	DIMRED	Find projections of all non-landmark points using the embedding calculated by a dimensionality reduction optimization calculation.
PROPERTYMAP	COLVAR	Calculate generic property maps.
PUCKERING	COLVAR	Calculate sugar pseudorotation coordinates.
PYTORCH_MODEL	PYTORCH_FUNCTION	Load a PyTorch model compiled with TorchScript.
Q3	MCOLVAR	Calculate 3rd order Steinhardt parameters.
Q4	MCOLVAR	Calculate fourth order Steinhardt parameters.
Q6	MCOLVAR	Calculate sixth order Steinhardt parameters.
RANDOM_EXCHANGES	GENERIC	Set random pattern for exchanges.
RDC	ISDB_COLVAR	Calculates the (Residual) Dipolar Coupling between two atoms.
READ	GENERIC	Read quantities from a colvar file.
READ_DISSIMILARITY_MATRIX	ANALYSIS	Read a matrix of dissimilarities between a trajectory of atomic configurations from a file.
RESCALE	ISDB_BIAS	Scales the value of an another action, being a Collective Variable or a Bias.
RESELECT_LANDMARKS	LANDMARKS	This allows us to use one measure in landmark selection and a different measure in dimensionality reduction
RESET_CELL	GENERIC	This action is used to rotate the full cell
RESTART	GENERIC	Activate restart.
RESTRAINT	BIAS	Adds harmonic and/or linear restraints on one or more variables.

REWEIGHT_BIAS	REWEIGHTING	Calculate weights for ensemble averages that negate the effect the bias has on the region of phase space explored
REWEIGHT_METAD	REWEIGHTING	Calculate the weights configurations should contribute to the histogram in a simulation in which a metadynamics bias acts upon the system.
REWEIGHT_TEMP_PRESS	REWEIGHTING	Calculate weights for ensemble averages at temperatures and/or pressures different than those used in your original simulation.
REWEIGHT_WHAM	REWEIGHTING	Calculate the weights for configurations using the weighted histogram analysis method.
RMSD	DCOLVAR	Calculate the RMSD with respect to a reference structure.
ROWSUMS	MATRIXF	Sum the rows of a adjacency matrix.
S2CM	S2CMMOD_COLVAR	S2 contact model CV.
SANS	ISDB_COLVAR	Calculates SANS intensity.
SASA_HASEL	SASAMOD_COLVAR	Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.
SASA_LCPO	SASAMOD_COLVAR	Calculates the solvent accessible surface area (SASA) of a protein molecule, or other properties related to it.
SAXS	ISDB_COLVAR	Calculates SAXS intensity.
SELECT	ISDB_FUNCTION	Selects an argument based on the value of a SELECTOR.
SELECTOR	ISDB_GENERIC	Defines a variable (of the type double) inside the PLUMED code that can be used and modified by other actions.
SIMPLECUBIC	MCOLVAR	Calculate whether or not the coordination spheres of atoms are arranged as they would be in a simple cubic structure.
SKETCHMAP_CONJGRAD	DIMRED	Optimize the sketch-map stress function using conjugate gradients.
SKETCHMAP_POINTWISE	DIMRED	Optimize the sketch-map stress function using a pointwise global optimization algorithm.
SKETCHMAP_READ	DIMRED	Read in a sketch-map projection from an input file
SKETCHMAP_SMACOF	DIMRED	Optimize the sketch-map stress function using the SMACOF algorithm.
SKETCH_MAP	DIMRED	This can be used to output the data that has been stored in an Analysis object.

SMAC	MCOLVARF	Calculate a variant on the SMAC collective variable
SMACOF_MDS	DIMRED	Optimize the multidimensional scaling stress function using the SMACOF algorithm.
SMAC_MATRIX	MATRIX	Adjacency matrix in which two molecules are adjacent if they are within a certain cutoff and if the angle between them is within certain ranges.
SORT	FUNCTION	This function can be used to sort colvars according to their magnitudes.
SPRINT	MATRIXF	Calculate SPRINT topological variables from an adjacency matrix.
STATS	FUNCTION	Calculates statistical properties of a set of collective variables with respect to a set of reference values.
TARGET	DCOLVAR	This function measures the Pythagorean distance from a particular structure measured in the space defined by some set of collective variables.
TD_CHI	VES_TARGETDIST	Chi distribution (static).
TD_CHISQUARED	VES_TARGETDIST	Chi-squared distribution (static).
TD_CUSTOM	VES_TARGETDIST	Target distribution given by an arbitrary mathematical expression (static or dynamic).
TD_EXPONENTIAL	VES_TARGETDIST	Exponential distribution (static).
TD_EXPONENTIALLY_MODIFIED_GAUSSIAN	VES_TARGETDIST	Target distribution given by a sum of exponentially modified Gaussian distributions (static).
TD_GAUSSIAN	VES_TARGETDIST	Target distribution given by a sum of Gaussian kernels (static).
TD_GENERALIZED_EXTREME_VALUE	VES_TARGETDIST	Generalized extreme value distribution (static).
TD_GENERALIZED_NORMAL	VES_TARGETDIST	Target distribution given by a sum of generalized normal distributions (static).
TD_GRID	VES_TARGETDIST	Target distribution from an external grid file (static).
TD_LINEAR_COMBINATION	VES_TARGETDIST	Target distribution given by linear combination of distributions (static or dynamic).
TD_MULTICANONICAL	VES_TARGETDIST	Multicanonical target distribution (dynamic).
TD_MULTITHERMAL_MULTIBARIC	VES_TARGETDIST	Multithermal-multibaric target distribution (dynamic).
TD_PRODUCT_COMBINATION	VES_TARGETDIST	Target distribution given by product combination of distributions (static or dynamic).
TD_PRODUCT_DISTRIBUTION	VES_TARGETDIST	Target distribution given by a separable product of one-dimensional distributions (static or dynamic).
TD_UNIFORM	VES_TARGETDIST	Uniform target distribution (static).

TD_VONMISES	VES_TARGETDIST	Target distribution given by a sum of Von Mises distributions (static).
TD_WELLTEMPERED	VES_TARGETDIST	Well-tempered target distribution (dynamic).
TEMPLATE	COLVAR	This file provides a template for if you want to introduce a new CV.
TETRAHEDRAL	MCOLVAR	Calculate the degree to which the environment about ions has a tetrahedral order.
TETRAHEDRALPORE	VOLUMES	This quantity can be used to calculate functions of the distribution of collective variables for the atoms lie that lie in a box defined by the positions of four atoms at the corners of a tetrahedron.
TIME	GENERIC	retrieve the time of the simulation to be used elsewhere
TOPOLOGY_MATRIX	MATRIX	Adjacency matrix in which two atoms are adjacent if they are connected topologically
TORSION	COLVAR	Calculate a torsional angle.
TORSIONS	MCOLVAR	Calculate whether or not a set of torsional angles are within a particular range.
UNITS	GENERIC	This command sets the internal units for the code.
UPDATE_IF	PRINTANALYSIS	Conditional update of other actions.
UPPER_WALLS	BIAS	Defines a wall for the value of one or more collective variables, which limits the region of the phase space accessible during the simulation.
UWALLS	MCOLVARB	Add UPPER_WALLS restraints on all the multicolvar values
VES_DELTA_F	VES_BIAS	Implementation of VES Delta F method
VES_LINEAR_EXPANSION	VES_BIAS	Linear basis set expansion bias.
VES_OUTPUT_BASISFUNCTIONS	VES_UTILS	Output basis functions to file.
VES_OUTPUT_FES	VES_UTILS	Tool to output biases and free energy surfaces for VES biases from previously obtained coefficients.
VES_OUTPUT_TARGET_DISTRIBUTION	VES_UTILS	Output target distribution to file.
VOLUME	COLVAR	Calculate the volume of the simulation box.
WHAM_HISTOGRAM	REWEIGHTING	This can be used to output the a histogram using the weighted histogram technique
WHAM_WEIGHTS	REWEIGHTING	Calculate and output weights for configurations using the weighted histogram analysis method.
WHOLEMOLECULES	GENERIC	This action is used to rebuild molecules that can become split by the periodic boundary conditions.
WRAPAROUND	GENERIC	Rebuild periodic boundary conditions around chosen atoms.

XANGLES	MCOLVAR	Calculate the angles between the vector connecting two atoms and the x axis.
XDISTANCES	MCOLVAR	Calculate the x components of the vectors connecting one or many pairs of atoms.You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.
XYDISTANCES	MCOLVAR	Calculate distance between a pair of atoms neglecting the z-component.
XYTORSIONS	MCOLVAR	Calculate the torsional angle around the x axis from the positive y direction.
XZDISTANCES	MCOLVAR	Calculate distance between a pair of atoms neglecting the y-component.
XZTORSIONS	MCOLVAR	Calculate the torsional angle around the x axis from the positive z direction.
YANGLES	MCOLVAR	Calculate the angles between the vector connecting two atoms and the y axis.
YDISTANCES	MCOLVAR	Calculate the y components of the vectors connecting one or many pairs of atoms.You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.
YXTORSIONS	MCOLVAR	Calculate the torsional angle around the y axis from the positive x direction.
YZDISTANCES	MCOLVAR	Calculate distance between a pair of atoms neglecting the x-component.
YZTORSIONS	MCOLVAR	Calculate the torsional angle around the y axis from the positive z direction.
ZANGLES	MCOLVAR	Calculate the angles between the vector connecting two atoms and the z axis.
ZDISTANCES	MCOLVAR	Calculate the z components of the vectors connecting one or many pairs of atoms.You can then calculate functions of the distribution of values such as the minimum, the number less than a certain quantity and so on.
ZXTORSIONS	MCOLVAR	Calculate the torsional angle around the z axis from the positive x direction.
ZYTORSIONS	MCOLVAR	Calculate the torsional angle around the z axis from the positive y direction.

completion	TOOLS	Dumps the body of a bash function to be used for auto completion.
config	TOOLS	inquire plumed about how it was configure
driver	TOOLS	driver is a tool that allows one to to use plumed to post-process an existing trajectory.
driver-float	TOOLS	Equivalent to driver, but using single precision reals.
drr_tool	EABFMOD_TOOLS	- Extract .grad and .count files from the binary output .drrstate - Merge windows
gen_example	TOOLS	gen_example is a tool that you can use to construct an example for the manual that users can interact with to understand
gen_json	TOOLS	gen_json constructs a json file that includes a dictionary of actions, the keywords for those actions and the components and outputs this to standard output
gentemplate	TOOLS	gentemplate is a tool that you can use to construct template inputs for the various actions
info	TOOLS	This tool allows you to obtain information about your plumed version
kt	TOOLS	Print out the value of k_BT at a particular temperature
manual	TOOLS	manual is a tool that you can use to construct the manual page fora particular action
mklib	TOOLS	compile a .cpp file into a shared library
newcv	TOOLS	create a new collective variable from a template
partial_tempering	TOOLS	scale parameters in a gromacs topology to implement solute or partial tempering
patch	TOOLS	patch an MD engine
pathtools	TOOLS	pathtools can be used to construct paths from pdb data
pdbrenumber	TOOLS	Modify atom numbers in a PDB, possibly using hybrid-36 coding.
pesmd	TOOLS	Pesmd allows one to do (biased) Langevin dynamics on a two-dimensional potential energy surface.
selector	TOOLS	create lists of serial atom numbers
simplemd	TOOLS	simplemd allows one to do molecular dynamics on systems of Lennard-Jones atoms.
sum_hills	TOOLS	sum_hills is a tool that allows one to to use plumed to post-process an existing hills/colvar file

ves_md_linearexpansion	VES_TOOLS	Simple MD code for dynamics on a potential energy surface given by a linear basis set expansion.
vim2html	TOOLS	convert plumed input file to colored html using vim syntax

Chapter 14

Todo List

Page PLUMED

Add support for multiple time stepping (`STRIDE` different from 1).

- Add the possibility to import CVs calculated in the host PLUMED instance into the guest PLUMED instance. Will be possible after #83 will be closed.
- Add the possibility to export CVs calculated in the guest PLUMED instance into the host PLUMED instance. Could be implemented using the `DataFetchingObject` class.

Chapter 15

Bug List

Page CENTER_OF_MULTICOLVAR

The virial contribution for this type of virtual atom is not currently evaluated so do not use in bias functions unless the volume of the cell is fixed

Page ENERGY

This ENERGY does not include long tail corrections. Thus when using e.g. LAMMPS "`pair_modify tail yes`" or GROMACS "`DispCorr Ener`" (or "`DispCorr EnerPres`"), the potential energy from ENERGY will be slightly different form the one of the MD code. You should still be able to use ENERGY and then reweight your simulation with the correct MD energy value.

Acceptance for replica exchange when ENERGY is biased is computed correctly only if all the replicas have the same potential energy function. This is for instance not true when using GROMACS with lambda replica exchange or with plumed-hrex branch.

Page MOLINFO

At the moment the HA1 atoms in a GLY residues are treated as if they are the CB atoms. This may or may not be true - GLY is problematic for secondary structure residues as it is achiral.

If you use WHOLEMOLECULES RESIDUES=1-10 for a 18 amino acid protein (18 amino acids + 2 terminal groups = 20 residues) the code will fail as it will not be able to interpret terminal residue 1.

Page namd-2.12

NAMD does not currently take into account virial contributions from PLUMED. Please use constant volume simulations only.

Page namd-2.13

NAMD does not currently take into account virial contributions from PLUMED. Please use constant volume simulations only.

Page namd-2.14

NAMD does not currently take into account virial contributions from PLUMED. Please use constant volume simulations only.

Page PCAVARS

It is not possible to use the DRMSD metric with this variable. You can get around this by listing the set of distances you wish to calculate for your DRMSD in the plumed file explicitly and using the EUCLIDEAN metric. MAHALONOBIS and NORM-EUCLIDEAN also do not work with this variable but using these options makes little sense when projecting on a linear subspace.

Bibliography

[1] Gareth A. Tribello, Massimiliano Bonomi, Davide Branduardi, Carlo Camilloni, and Giovanni Bussi. Plumed 2: New feathers for an old bird. *Comput. Phys. Commun.*, 185(2):604–613, 2014. 1

[2] Massimiliano Bonomi, Davide Branduardi, Giovanni Bussi, Carlo Camilloni, Davide Provasi, Paolo Raiteri, Davide Donadio, Fabrizio Marinelli, Fabio Pietrucci, Ricardo A Broglia, and Michele Parrinello. PLUMED: A portable plugin for free-energy calculations with molecular dynamics. *Computer Physics Communications*, 180(10):1961–1972, 2009. 1

[3] Pratyush Tiwary and Michele Parrinello. A time-independent free energy estimator for metadynamics. *The Journal of Physical Chemistry B*, 119(3):736–742, Jan 2015. 14, 499, 501, 668, 793, 886, 921, 940, 1028, 1034

[4] Grisell Díaz Leines and Bernd Ensing. Path finding on high-dimensional free energy landscapes. *Phys. Rev. Lett.*, 109:020601, Feb 2012. 26, 97

[5] Davide Branduardi, Francesco Luigi Gervasio, and Michele Parrinello. From A to B in free energy space. *J. Chem. Phys.*, 126(5):054103, Feb 2007. 96, 137, 140, 171

[6] F. Pietrucci and A. Laio. A collective variable for the efficient exploration of protein beta-structures with metadynamics: application to sh3 and gb1. *J. Chem. Theory Comput.*, 5(9):2197–2201, 2009. 100, 104, 134

[7] R. B. Best, G. Hummer, and W. A. Eaton. Native contacts determine protein folding mechanisms in atomistic simulations. *Proc. Natl. Acad. Sci. U.S.A.*, 110(44):17874–17879, 2013. 109

[8] Trang N. Do, Paolo Carloni, Gabriele Varani, and Giovanni Bussi. Rna/peptide binding driven by electrostatics—insight from bidirectional pulling simulations. *Journal of Chemical Theory and Computation*, 9(3):1720–1730, 2013. 113

[9] Marco Nava, Ferruccio Palazzesi, Claudio Perego, and Michele Parrinello. Dimer metadynamics. *Journal of Chemical Theory and Computation*, 13(2):425–430, 2017. 116

[10] C. Bartels and M. Karplus. Probability Distributions for Complex Systems: Adaptive Umbrella Sampling of the Potential Energy. *J. Phys. Chem. B*, 102(5):865–880, 1998. 126

[11] M. Bonomi and M. Parrinello. Enhanced sampling in the well-tempered ensemble. *Phys. Rev. Lett.*, 104:190601, 2010. 126, 1070

[12] Sandro Bottaro, Francesco Di Palma, and Giovanni Bussi. The role of nucleobase interactions in rna structure and dynamics. *Nucleic acids research*, 21(42):13306–13314, 2014. 127

[13] Vojtech Spiwok and Blanka Králová. Metadynamics in the conformational space nonlinearly dimensionally reduced by Isomap. *Journal of Chemical Physics*, 135(22):224504, 2011. 131, 147

[14] Jiří Vymětal and Jiří Vondrášek. Gyration- and Inertia-Tensor-Based Collective Coordinates for Metadynamics. Application on the Conformational Behavior of Polyalanine Peptides and Trp-Cage Folding. *J. Phys. Chem. A*, page 110930112611005, 2011. 133

[15] Grisell Díaz Leines and Bernd Ensing. Path finding on high-dimensional free energy landscapes. *Phys. Rev. Lett.*, 109:020601, 2012. 137

[16] Ming Huang, Timothy J Giese, Tai-Sung Lee, and Darrin M York. Improvement of dna and rna sugar pucker profiles from semiempirical quantum methods. *Journal of chemical theory and computation*, 10(4):1538–1545, 2014. 149

[17] D t Cremer and JA Pople. General definition of ring puckering coordinates. *Journal of the American Chemical Society*, 97(6):1354–1358, 1975. 149

[18] Xevi Biarnés, Albert Ardevol, Antoni Planas, Carme Rovira, Alessandro Laio, and Michele Parrinello. The conformational free energy landscape of β-d-glucopyranose. implications for substrate preactivation in β-glucoside hydrolases. *Journal of the American Chemical Society*, 129(35):10686–10693, 2007. 149

[19] L Sutto, M D Abramo, and F L Gervasio. Comparing the efficiency of biased and unbiased molecular dynamics in reconstructing the free energy landscape of met-enkephalin. *J. Chem. Theory Comput.*, 6(12):3640–3646, 2010. 157

[20] Vojtech Spiwok, Petra Lipovová, and Blanka Králová. Metadynamics in essential coordinates: free energy simulation of conformational changes. *The journal of physical chemistry B*, 111(12):3073–6, Mar 2007. 157

[21] S. K. Kearsley. On the orthogonal transformation used for structural comparison. *Acta Cryst. A*, 45:208–210, 1989. 160, 724

[22] Andrea Pérez-Villa, Maria Darvas, and Giovanni Bussi. Atp dependent ns3 helicase interaction with rna: insights from molecular simulations. *Nucleic Acids Research*, 43(18):8725, 2015. 167

[23] Ladislav Hovan, Federico Comitani, and Francesco L Gervasio. An Optimal Metric for the Path Collective Variables. *Journal of Chemical Theory and Computation*, 15(1):25–32, 2019. 170

[24] Wolfgang Lechner and Christoph Dellago. Accurate determination of crystal structures based on averaged local bond order parameters. *The Journal of Chemical Physics*, 129(11):–, 2008. 185, 321, 322

[25] Gareth A. Tribello, Jérôme Cuny, Hagai Eshet, and Michele Parrinello. Exploring the free energy surfaces of clusters using reconnaissance metadynamics. *J. Chem. Phys.*, 135(11):114109, 2011. 187

[26] Andrew D White and Gregory A Voth. An Efficient and Minimal Method to Bias Molecular Simulations with Experimental Data. *Journal of Chemical Theory and Computation*, 10:3023–3030, 2014. 193, 519, 591, 592

[27] Pablo M Piaggi and Michele Parrinello. Calculation of phase diagrams in the multithermal-multibaric ensemble. *The Journal of chemical physics*, 150(24):244119, 2019. 202, 425, 636, 638, 841

[28] Stefano Angioletti-Uberti, Michele Ceriotti, Peter D. Lee, and Mike W. Finnis. Solid-liquid interface free energy through metadynamics simulations. *Phys. Rev. B*, 81:125416, 2010. 207

[29] Bingqing Cheng, Gareth A. Tribello, and Michele Ceriotti. Solid-liquid interfacial free energy out of equilibrium. *Phys. Rev. B*, 92:180102, 2015. 207

[30] Federico Giberti, Gareth A. Tribello, and Michele Parrinello. Transient polymorphism in nacl. *Journal of Chemical Theory and Computation*, 9(2526-2530):null, 2013. 318

[31] Federico Giberti, Matteo Salvalaglio, Marco Mazzotti, and Michele Parrinello. Insight into the nucleation of urea crystals from the melt. *Chemical Engineering Science*, 121:51 – 59, 2015. 2013 Danckwerts Special Issue on Molecular Modelling in Chemical Engineering. 351

[32] Gareth A. Tribello, Federico Giberti, Gabriele C. Sosso, Matteo Salvalaglio, and Michele Parrinello. Analyzing and driving cluster formation in atomistic simulations. *Journal of Chemical Theory and Computation*, 13(3):1317–1327, 2017. 369, 371, 372, 385, 392, 396

[33] Gabriele C. Sosso, Gareth A. Tribello, Andrea Zen, Philipp Pedevilla, and Angelos Michaelides. Ice formation on kaolinite: Insights from molecular dynamics simulations. *The Journal of Chemical Physics*, 145(21):211927, 2016. 380

[34] Fabio Pietrucci and Wanda Andreoni. Graph theory MeetsAb InitioMolecular dynamics: Atomic structures and transformations at the nanoscale. *Physical Review Letters*, 107(8), August 2011. 389

[35] Pratyush Tiwary and Michele Parrinello. A time-independent free energy estimator for metadynamics. *The Journal of Physical Chemistry B*, 119(3):736–742, 2015. PMID: 25046020. 423

[36] F. G. Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, 86:2050–2053, 2001. 425, 636, 638

[37] Cristian Micheletti, Alessandro Laio, and Michele Parrinello. Reconstructing the density of states by history-dependent metadynamics. *Phys. Rev. Lett.*, 92(17):170601, April 2004. 425

[38] Pablo M. Piaggi and Michele Parrinello. Multithermal-multibaric molecular simulations from a variational principle. *Phys. Rev. Lett.*, 122:050601, Feb 2019. 425, 636, 638

[39] Adam P. Willard and David Chandler. Instantaneous liquid interfaces. *The Journal of Physical Chemistry B*, 114(5):1954–1958, 2010. 444, 445

[40] M. Marchi and P. Ballone. Adiabatic bias molecular dynamics: A method to navigate the conformational space of complex molecular systems. *J. Chem. Phys.*, 110(8):3697–3702, 1999. 486

[41] D. Provasi and M. Filizola. Putative active states of a prototypic g-protein-coupled receptor from biased molecular dynamics. *Biophys. J.*, 98:2347—2355, 2010. 486

[42] C. Camilloni, R. A. Broglia, and G. Tiana. Hierarchy of folding and unfolding events of protein g, ci2, and acbp from explicit-solvent simulations. *J. Chem. Phys.*, 134:045105, 2011. 486

[43] M. Iannuzzi, A. Laio, and M. Parrinello. Efficient exploration of reactive potential energy surfaces using car-parrinello molecular dynamics. *Phys. Rev. Lett.*, 90:238302, 2003. 489

[44] L. Maragliano and E. Vanden-Eijnden. A temperature-accelerated method for sampling free energy and determining reaction pathways in rare events simulations. *Chem. Phys. Lett.*, 426:168–175, 2006. 489, 585

[45] Jerry B. Abrams and Mark E. Tuckerman. Efficient and Direct Generation of Multidimensional Free Energy Surfaces via Adiabatic Dynamics without Coordinate Transformations. *J. Phys. Chem. B*, 112(49):15742–15757, DEC 11 2008. 489, 583, 585

[46] Alejandro Gil-Ley Andrea Cesari and Giovanni Bussi. Combining simulations and solution experiments as a paradigm for RNA force field refinement. *J Chem Theory Comput*, 12(12):6192–6200, dec 2016. 495

[47] A. Laio and M. Parrinello. Escaping free energy minima. *Proc. Natl. Acad. Sci. USA*, 99:12562–12566, 2002. 498, 508, 665, 723, 790, 796, 882, 918, 937, 1025, 1064, 1086, 1094, 1110

[48] V. Babin, C. Roland, and C. Sagui. Adaptively biased molecular dynamics for free energy calculations. *J. Chem. Phys.*, 128:134101, 2008. 498

[49] A Barducci, G Bussi, and M Parrinello. Well-tempered metadynamics: A smoothly converging and tunable free-energy method. *Phys. Rev. Lett.*, 100(2):020603, Jan 2008. 499, 509, 646, 790, 796, 883, 918, 937, 1025, 1035, 1065, 1086, 1095, 1110

[50] D Branduardi, G Bussi, and M PARRINELLO. Metadynamics with adaptive Gaussians. *J. Chem. Theory Comput.*, 8(7):2247–2254, 2012. 499, 509, 793, 886, 921, 940, 1028, 1034, 1050

[51] Fahimeh Baftizadeh, Pilar Cossio, Fabio Pietrucci, and Alessandro Laio. Protein folding and ligand-enzyme binding from bias-exchange metadynamics simulations. *Curr Phys Chem*, 2:79–91, 2012. 499, 509

[52] P. Raiteri, A. Laio, F.L. Gervasio, C. Micheletti, and M. Parrinello. Efficient reconstruction of complex free energy landscapes by multiple walkers metadynamics. *J. Phys. Chem. B*, 110:3533–3539, 2006. 499, 501, 509

[53] Alejandro Gil-Ley and Giovanni Bussi. Enhanced conformational sampling using replica exchange with collective-variable tempering. *Journal of chemical theory and computation*, 11(3):1077–1085, 2015. 499, 1035, 1103

[54] Pratyush Tiwary and Michele Parrinello. From metadynamics to dynamics. *Phys. Rev. Lett.*, 111:230602, 2013. 501, 804

[55] Yong Wang, Omar Valsson, Pratyush Tiwary, Michele Parrinello, and Kresten Lindorff-Larsen. Frequency adaptive metadynamics for the calculation of rare-event kinetics. *The Journal of Chemical Physics*, 149(7):072309, Aug 2018. 501

[56] Andrew D White, James F Dama, and Gregory A Voth. Designing free energy surfaces that match experimental data with metadynamics. *J. Chem. Theory Comput.*, 11(6):2451–2460, 2015. 502

[57] Fabrizio Marinelli and José D Faraldo-Gómez. Ensemble-biased metadynamics: A molecular simulation method to sample experimental distributions. *Biophys. J.*, 108(12):2779–2782, 2015. 502

[58] Alejandro Gil-Ley and Giovanni Bussi. Empirical corrections to the amber rna force field with target metadynamics, 2016. 502

[59] James F Dama, Michele Parrinello, and Gregory A Voth. Well-tempered metadynamics converges asymptotically. *Phys. Rev. Lett.*, 112(24):240602, 2014. 502

[60] H. Grubmüller, B. A. Heymann, and P. Tavan. *Science*, 271:997–999, 1996. 506

[61] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, 78:2690–2693, 1997. 506

[62] Jim Pfaendtner and Massimiliano Bonomi. Efficient sampling of high-dimensional free-energy landscapes with parallel bias metadynamics. *Journal of Chemical Theory and Computation*, 11(11):5062–5067, 2015. 508

[63] Arushi Prakash, Christopher D. Fu, Massimiliano Bonomi, and Jim Pfaendtner. Biasing smarter, not harder, by partitioning collective variables into families in parallel bias metadynamics. *Journal of Chemical Theory and Computation*, 14(10):4985–4990, 2018. 509

[64] Michael J Hartmann, Yuvraj Singh, Eric Vanden-Eijnden, and Glen M Hocky. Infinite switch simulated tempering in force (fisst). *arXiv:1910.14064*, 2019. 519, 522, 523, 853

[65] Massimiliano Bonomi and Carlo Camilloni. Integrative structural and dynamical biology with PLUMED-ISDB. *Bioinformatics*, 33:3999–4000, 2017. 519

[66] Luigi Bonati, Valerio Rizzi, and Michele Parrinello. Data-driven collective variables for enhanced sampling. *The Journal of Physical Chemistry Letters*, 11(8):2998–3004, 2020. 519, 580, 582, 816

[67] T. Morishita, S. G. Itoh, H. Okumura, and M. Mikami. Free-energy calculation via mean-force dynamics using a logarithmic energy landscape. *Physical Review E*, 85:066702, 2012. 519, 583, 585

[68] T. Morishita, Y Yonezawa, and A. M. Ito. Free energy reconstruction from logarithmic mean-force dynamics using multiple nonequilibrium trajectories. *Journal of Chemical Theory and Computation*, 13:3106, 2017. 519, 583, 586

[69] T. Morishita, T Nakamura, W Shinoda, and A. M. Ito. Isokinetic approach in logarithmic mean-force dynamics for on-the-fly free energy reconstruction. *Chemical Physics Letter*, 706:633, 2018. 519, 587

[70] Glen M. Hocky, Thomas Dannenhoffer-Lafage, and Gregory A. Voth. Coarse-grained directed simulation. *Journal of Chemical Theory and Computation*, 13(9):4593–4603, 2017. 519, 591, 592

[71] Dilnoza Amirkulova and Andrew D. White. Recent advances in maximum entropy biasing techniques for molecular dynamics. *arXiv preprint arXiv:1902.02252*, 2019. 519, 591, 592

[72] Haochuan Chen, Haohao Fu, Xueguang Shao, Christophe Chipot, and Wensheng Cai. ELF: An extended-lagrangian free energy calculation module for multiple molecular dynamics engines. *Journal of Chemical Information and Modeling*, 58:1315–1318, Jun 2018. 519, 596

[73] Tony Lelièvre, Mathias Rousset, and Gabriel Stoltz. Computation of free energy profiles with parallel adaptive dynamics. *The Journal of Chemical Physics*, 126(13):134111, apr 2007. 519, 596

[74] Adrien Lesage, Tony Lelièvre, Gabriel Stoltz, and Jérôme Hénin. Smoothed biasing forces yield unbiased free energies with the extended-system adaptive biasing force method. *The Journal of Physical Chemistry B*, 121(15):3676–3685, dec 2016. 519, 596

[75] Haohao Fu, Xueguang Shao, Christophe Chipot, and Wensheng Cai. Extended adaptive biasing force algorithm. an on-the-fly implementation for accurate free-energy calculations. *Journal of Chemical Theory and Computation*, 12(8):3506–3513, aug 2016. 519, 596

[76] Omar Valsson and Michele Parrinello. Variational approach to enhanced sampling and free energy calculations. *Phys. Rev. Lett.*, 113(9):090601, 2014. 519, 601, 670

[77] J Rydzewski and O Valsson. Finding multiple reaction pathways of ligand unbinding. *arXiv: 1808.08089*, 2018. 519

[78] Michele Invernizzi and Michele Parrinello. Rethinking metadynamics: From bias potentials to probability distributions. *The Journal of Physical Chemistry Letters*, 11(7):2731–2736, 2020. 519, 694, 697, 711, 713, 811, 814

[79] Michele Invernizzi, Pablo M. Piaggi, and Michele Parrinello. Unified approach to enhanced sampling. *Physical Review X*, 10:041034, 2020. 519, 694, 695, 703, 704, 706, 707, 708, 710, 811, 812

[80] Grégoire A. Gallet and Fabio Pietrucci. Structural cluster analysis of chemical reactions in solution. *The Journal of Chemical Physics*, 139(7):074101, 2013. 520, 714

[81] S. Pipolo, M. Salanne, G. Ferlat, S. Klotz, A. M. Saitta, and F. Pietrucci. Navigating at will on the water phase diagram. *Phys. Rev. Lett.*, 119:245701, Dec 2017. 520, 714, 716

[82] Ferruccio Palazzesi, Omar Valsson, and Michele Parrinello. Conformational Entropy as Collective Variable for Proteins. *The Journal of Physical Chemistry Letters*, 8(19):4752–4756, 2017. 520, 717

[83] W Hasel, T F Hendrickson, and W C Still. A rapid approximation to the solvent accessible surface areas of atoms. *Tetrahedron Computer Methodology*, 1:103–116, 1988. 520, 718, 719, 722, 847

[84] Jörg Weiser, Peter S. Shenkin, and W. Clark Still. Approximate atomic surfaces from linear combinations of pairwise overlaps (lcpo). *Journal of Computational Chemistry*, 20(2):217–230, 1999. 520, 718, 720, 721, 847

[85] Andrea Arsiccio, Pritam Ganguly, and Joan-Emma Shea. A transfer free energy based implicit solvent model for protein simulations in solvent mixtures: Urea-induced denaturation as a case study. *The Journal of Physical Chemistry B*, 0(0):null, 2022. 520, 718, 719, 720, 721, 722, 847, 848, 849

[86] Andrea Arsiccio and Joan-Emma Shea. Protein cold denaturation in implicit solvent simulations: A transfer free energy approach. *The Journal of Physical Chemistry B*, 125(20):5222–5232, 2021. 520, 718, 719, 720, 721, 722, 847, 848, 849

[87] Andrea Arsiccio and Joan-Emma Shea. Pressure unfolding of proteins: New insights into the role of bound water. *The Journal of Physical Chemistry B*, 125(30):8431–8442, 2021. 520, 718, 719, 720, 721, 722, 847, 848, 849

[88] Vittorio Limongelli, Massimiliano Bonomi, and Michele Parrinello. Funnel metadynamics as accurate binding free-energy method. *Proceedings of the National Academy of Sciences*, 110(16):6358–6363, 2013. 520, 723, 724

[89] Stefano Raniolo and Vittorio Limongelli. Ligand binding free-energy calculations with funnel metadynamics. *Nature Protocols*, 15(9):2837–2866, 2020. 520, 724, 796, 797

[90] Ary Lautaro Di Bartolo and Diego Masone. Synaptotagmin-1 c2b domains cooperatively stabilize the fusion stalk via a master-servant mechanism. *Chem. Sci.*, pages –, 2022. 520, 728, 732

[91] Jochen S. Hub and Neha Awasthi. Probing a continuous polar defect: A reaction coordinate for pore formation in lipid membranes. *Journal of Chemical Theory and Computation*, 13(5):2352–2366, 2017. PMID: 28376619. 520, 728, 731, 732

[92] Jochen S Hub. Joint reaction coordinate for computing the free-energy landscape of pore nucleation and pore expansion in lipid membranes. *Journal of Chemical Theory and Computation*, 17(2):1229–1239, 2021. 520, 728, 729

[93] Chetan S Poojari, Katharina C Scherer, and Jochen S Hub. Free energies of membrane stalk formation from a lipidomics perspective. *Nature communications*, 12(1):1–10, 2021. 520, 728

[94] KJ Kohlhoff, Paul Robustelli, Andrea Cavalli, Xavier Salvatella, and Michele Vendruscolo. Fast and accurate predictions of protein NMR chemical shifts from interatomic distances. *J. Am. Chem. Soc.*, 131(39):13894–13895, 2009. 526

[95] Daniele Granata, Carlo Camilloni, Michele Vendruscolo, and Alessandro Laio. Characterization of the free-energy landscapes of proteins by NMR-guided metadynamics. *Proc. Natl. Acad. Sci. U.S.A.*, 110(17):6817–6822, 2013. 526, 528

[96] Paul Robustelli, Kai Kohlhoff, Andrea Cavalli, and Michele Vendruscolo. Using NMR chemical shifts as structural restraints in molecular dynamics simulations of proteins. *Structure*, 18(8):923–933, 2010. 527

[97] Carlo Camilloni, Paul Robustelli, Alfonso De Simone, Andrea Cavalli, and Michele Vendruscolo. Characterization of the Conformational Equilibrium between the Two Major Substates of RNase A Using NMR Chemical Shifts. *J. Am. Chem. Soc.*, 134(9):3968–3971, 2012. 528

[98] Carlo Camilloni, Andrea Cavalli, and Michele Vendruscolo. Assessment of the Use of NMR Chemical Shifts as Replica-Averaged Structural Restraints in Molecular Dynamics Simulations to Characterize the Dynamics of Proteins. *J. Phys. Chem. B*, 117(6):1838–1843, 2013. 528

[99] Samuel Hanot, Massimiliano Bonomi, Charles H Greenberg, Andrej Sali, Michael Nilges, Michele Vendruscolo, and Riccardo Pellarin. Multi-scale bayesian modeling of cryo-electron microscopy density maps. *bioRxiv*, page doi: 10.1101/113951, 2017. 531

[100] Massimiliano Bonomi, Carlo Camilloni, Andrea Cavalli, and Michele Vendruscolo. Metainference: A Bayesian inference method for heterogeneous systems. *Science Advances*, 2(1):e1501177, 2016. 531, 564, 571

[101] Carlo Camilloni and Michele Vendruscolo. Using Pseudocontact Shifts and Residual Dipolar Couplings as Exact NMR Restraints for the Determination of Protein Structural Ensembles. *Biochemistry*, 54(51):7470–7476, 2015. 542

[102] Carlo Camilloni and Michele Vendruscolo. A Tensor-Free Method for the Structural and Dynamical Refinement of Proteins using Residual Dipolar Couplings. *J. Phys. Chem. B*, 119(3):653–661, 2015. 548

[103] Riccardo Capelli, Guido Tiana, and Carlo Camilloni. An implementation of the maximum-caliber principle by replica-averaged time-resolved restrained simulations. *J. Chem. Phys.*, 148(18):184114, May 2018. 562

[104] Massimiliano Bonomi, Carlo Camilloni, and Michele Vendruscolo. Metadynamic metainference: Enhanced sampling of the metainference ensemble using metadynamics. *Sci. Rep.*, 6:31232, 2016. 565, 571, 576

[105] Thomas Löhr, Alexander Jussupow, and Carlo Camilloni. Metadynamic metainference: Convergence towards force field independent structural ensembles of a disordered peptide. *J. Chem. Phys.*, 146(16):165102–11, 2017. 565, 571, 573

[106] Massimiliano Bonomi, Gabriella T Heller, Carlo Camilloni, and Michele Vendruscolo. Principles of protein structural ensemble determination. *Curr. Opin. Struct. Biol.*, 42:106–116, 2017. 571

[107] Cristina Paissoni, Alexander Jussupow, and Carlo Camilloni. Martini bead form factors for nucleic acids and their application in the refinement of protein–nucleic acid complexes against SAXS data. *J Appl Crystallogr*, 52(2):394–402, April 2019. 576

[108] Erica Valentini, Alexey G Kikhney, Gianpietro Previtali, Cy M Jeffries, and Dmitri I Svergun. SASBDB, a repository for biological small-angle scattering data. *Nucleic Acids Res*, 43(Database issue):D357–63, January 2015. 577

[109] Stephan Niebling, Alexander Björling, and Sebastian Westenhoff. MARTINI bead form factors for the analysis of time-resolved X-ray scattering of proteins. *J Appl Crystallogr*, 47(4):1190–1198, August 2014. 577

[110] Luigi Bonati, GiovanniMaria Piccini, and Michele Parrinello. Deep learning the slow modes for rare events sampling. *Proceedings of the National Academy of Sciences*, 118(44), 2021. 580, 582, 816

[111] Lianqing Zheng and Wei Yang. Practically efficient and robust free energy calculations: Double-integration orthogonal space tempering. *Journal of Chemical Theory and Computation*, 8(3):810–823, mar 2012. 596

[112] Tanfeng Zhao, Haohao Fu, Tony Lelièvre, Xueguang Shao, Christophe Chipot, and Wensheng Cai. The extended generalized adaptive biasing force algorithm for multidimensional free-energy calculations. *Journal of Chemical Theory and Computation*, 13(4):1566–1576, 2017. 597

[113] Michele Invernizzi and Michele Parrinello. Making the best of a bad situation: a multiscale approach to free energy calculation. *J. Chem. Theory Comput.*, 15(4):2187–2194, 2019. 601, 602

[114] James McCarty, Omar Valsson, Pratyush Tiwary, and Michele Parrinello. Variationally optimized free-energy flooding for rate calculation. *Phys. Rev. Lett.*, 115(7):070601, 2015. 605

[115] Christian Habermann and Fabian Kindermann. Multidimensional spline interpolation: Theory and applications. *Computational Economics*, 30(2):153–169, September 2007. 611

[116] Benjamin Pampel and Omar Valsson. Improving the Efficiency of Variationally Enhanced Sampling with Wavelet-Based Bias Potentials. *J. Chem. Theory Comput.*, 2022. 611, 615, 619

[117] Ingrid Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992. 619

[118] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA, 1997. 619

[119] Bernd A. Berg and Thomas Neuhaus. Multicanonical ensemble: A new approach to simulate first-order phase transitions. *Phys. Rev. Lett.*, 68:9–12, Jan 1992. 636, 638

[120] Hisashi Okumura and Yuko Okamoto. Molecular dynamics simulations in the multibaric–multithermal ensemble. *Chemical Physics Letters*, 391(4):248 – 253, 2004. 638

[121] Omar Valsson and Michele Parrinello. Well-Tempered Variational Approach to Enhanced Sampling. *J. Chem. Theory Comput.*, 11(5):1996–2002, 2015. 639, 647, 650, 675, 676, 834, 835

[122] Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o(1/n). In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 773–781. Curran Associates, Inc., Red Hook, NY, 2013. 650, 670, 834

[123] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler. Transition path sampling: throwing ropes over dark mountain passes. *Ann. Rev. Phys. Chem.*, 54:20, 2002. 665

[124] B Montgomery Pettitt and Peter J Rossky. Alkali halides in water: Ion–solvent correlations and ion–ion potentials of mean force at infinite dilution. *The Journal of chemical physics*, 84(10):5836–5844, 1986. 665

[125] Daniel J Price and Charles L Brooks III. A modified tip3p water potential for simulation with ewald summation. *The Journal of chemical physics*, 121(20):10096–10103, 2004. 665

[126] G.M. Torrie and J.P. Valleau. Nonphysical sampling distributions in monte carlo free energy estimation: Umbrella sampling. *J. Comput. Phys.*, 23:187–199, 1977. 674, 1057

[127] Patrick Shaffer, Omar Valsson, and Michele Parrinello. Enhanced, targeted sampling of high-dimensional free-energy landscapes using variationally enhanced sampling, with an application to chignolin. *Proc. Natl. Acad. Sci. USA*, 113(5):1150–1155, 2016. 675

[128] Michele Invernizzi and Michele Parrinello. Exploration vs convergence speed in adaptive-bias enhanced sampling. *Journal of Chemical Theory and Computation*, 18(6):3988–3996, 2022. 694, 701, 811, 814

[129] Fabio Pietrucci. Strategies for the exploration of free energy landscapes: Unity in diversity and challenges ahead. *Reviews in Physics*, 2:32–45, 2017. 697

[130] Fengli Zhang and Rafael Brüschweiler. Contact Model for the Prediction of NMR NH Order Parameters in Globular Proteins. *Journal of the American Chemical Society*, 124(43):12654–12655, 2002. 717

[131] Dengming Ming and Rafael Brüschweiler. Prediction of methyl-side Chain Dynamics in Proteins. *Journal of Biomolecular NMR*, 29(3):363–368, 2004. 717

[132] Stefano Piana and Alessandro Laio. A bias-exchange approach to protein folding. *J. Phys. Chem. B*, 111(17):4553–9, 2007. 770, 773, 1031, 1035

[133] Alessandrio Laio and Francesco Luigi Gervasio. Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science. *Rep. Prog. Phys.*, 71:126601, 2008. 790, 797, 882, 883, 919, 938, 1025, 1065, 1086, 1095, 1110

[134] Alessandro Barducci, Massimiliano Bonomi, and Michele Parrinello. Metadynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(5):826–843, 2011. 790, 797, 882, 883, 919, 938, 1025, 1065, 1086, 1095, 1110

[135] Ludovico Sutto, Simone Marsili, and Francesco Luigi Gervasio. New advances in metadynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(5):771–779, 2012. 790, 797, 882, 883, 919, 938, 1025, 1065, 1086, 1095, 1110

[136] Giovanni Bussi, Davide Branduardi, et al. Free-energy calculations with metadynamics: Theory and practice. *Rev. Comput. Chem*, 28:1–49, 2015. 790, 797, 882, 883, 919, 938

[137] Richard A Cunha and Giovanni Bussi. Unraveling mg2+–rna binding with atomistic molecular dynamics. *RNA*, 23(5):628–638, 2017. 960, 996, 1035

[138] Giovanni Bussi. Hamiltonian replica-exchange in gromacs: a flexible implementation. *Mol. Phys.*, 2013. DOI: 10.1080/00268976.2013.824126. 965, 1035

[139] Lingle Wang, Richard A Friesner, and BJ Berne. Replica exchange with solute scaling: A more efficient version of replica exchange with solute tempering (rest2). *The Journal of Physical Chemistry B*, 115(30):9431–9438, 2011. 965, 1035

[140] Marco Jacopo Ferrarotti, Sandro Bottaro, Andrea Perez-Villa, and Giovanni Bussi. Accurate multiple time step in biased molecular simulations. *J. Chem. Theory Comput.*, 11(1):139–146, 2015. 995, 1122, 1124

[141] Fabrizio Marinelli, Fabio Pietrucci, Alessandro Laio, and Stefano Piana. A kinetic model of trp-cage folding from multiple biased molecular dynamics simulations. *PLoS Comput. Biol.*, 5(8):e100045, 2009. 1035

[142] Jeremy Curuksu and Martin Zacharias. Enhanced conformational sampling of nucleic acids by a new hamiltonian replica exchange molecular dynamics approach. *The Journal of chemical physics*, 130(10):03B610, 2009. 1035

[143] Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314(1–2):141–151, November 1999. 1035, 1070

[144] Giovanni Bussi, Francesco Luigi Gervasio, Alessandro Laio, and Michele Parrinello. Free-energy landscape for beta hairpin folding from combined parallel tempering and metadynamics. *J. Am. Chem. Soc.*, 128(41):13435–41, 2006. 1070

[145] Michael Deighan, Massimiliano Bonomi, and Jim Pfaendtner. Efficient simulation of explicitly solvated proteins in the well-tempered ensemble. *Journal of Chemical Theory and Computation*, 8(7):2189–2192, 2012. 1070, 1075

[146] Andrea Cavalli, Carlo Camilloni, and Michele Vendruscolo. Molecular dynamics simulations with replica-averaged structural restraints generate structural ensembles according to the maximum entropy principle. *J. Chem. Phys.*, 138(9):094112, 2013. 1090

[147] Carlo Camilloni, Andrea Cavalli, and Michele Vendruscolo. Replica-Averaged Metadynamics. *J. Chem. Theory Comput.*, 9(12):5610–5617, 2013. 1090

[148] Carlo Camilloni and Michele Vendruscolo. Statistical mechanics of the denatured state of a protein using replica-averaged metadynamics. *J. Am. Chem. Soc.*, 136(25):8982–8991, 2014. 1090

[149] Wouter Boomsma, Kresten Lindorff-Larsen, and Jesper Ferkinghoff-Borg. Combining Experiments and Simulations Using the Maximum Entropy Principle. *PLoS Comput. Biol.*, 10(2):e1003406, 2014. 1090