PLUMED

2.4.6

Generated by Doxygen 1.8.14

Contents

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 analyse 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 **2.4.6** (git version: **9c7b1e0**). Manual built on Travis CI for branch v2.4-ves.

Regtest results for this version can be found `here`.

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 analyse 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:

- `AmberTools`, sander module, since version 15.
- `CP2K`, since Feb 2015.
- `ESPResSo`, in a Plumedized version that can be found `here`.
- `PINY-MD`, in its plumed branch.

- IPHIGENIE.
- AceMD, see this link.
- OpenMM, using the openmmp-plumed plugin.
- DL_POLY4.
- VNL-ATK, see this link.
- ABIN.
- LAMMPS since Nov 2018.

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:

- amber14
- gromacs-2016-6
- gromacs-2018-6
- gromacs-4-5-7
- gromacs-5-1-4
- lammps-6Apr13
- namd-2-12
- namd-2-13
- qespresso-5-0-2
- qespresso-6-2

Alternatively, one can use PLUMED as a Command Line Tools for postprocessing 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.

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

Version 2.0.0 (Sep 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).

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 setMPIFIntercomm" and "GREX setMPIFIntracomm").
- cmd("setStopFlag") should now be called after PLUMED initialization.
- Several small fixes in documentation.

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.

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.

Version 2.0.4 (Sep 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

Version 2.0.5 (Dec 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 lamdba-dynamics or plumed XX-hrex branch) possibly occuring 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

Version 2.1.0 (Sep 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 : parallelised and added weights.

- New features in MD patches (require repatch):

  - 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 repatch, and that all combinations should work correctly (new/old PLUMED with repatched/non-repatched 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.
- Optimizations:
  - Root-mean-square devations with align weights different from displace weights are now considerably faster. This will affect RMSD calculations plus other variables based on RMSD.
  - WHOLEMOLECULES is slighlty faster.
  - COORDINATION is slighlty 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 referenceable quantities outputted by each action
  - 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 (QuantumEspresso 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 subdirectories (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 incompleted, if you are developing in PLUMED you are encouraged to follow changes on github.

Version 2.1.1 (Dec 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 contactmaps 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.

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 repatching.
- 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 availabity 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.

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 repatched 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.

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 repatching 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 repatching.
- 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 normalisation of kernel functions. This affects HISTOGRAM If these actions were used with previous versions of the code care should be taken when analysing the results.
- Fixed a bug in derivatives of kernel functions with non-diagonal covariances. This affects the derivatives output by sum_hills

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

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 previusly 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 relevent 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 amonst those that have a magnitude greater than some tolerance
  - New way of calculating the minimum value in multicolvars (ALT_MIN). This is less succetible 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 multcolvar, 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.

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 usefull 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.

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 syncronise 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`.

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.

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.

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

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 repatch):
  - 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 multple 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 infrastracture, 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 parallelised 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 constructure as a componentWithDerivatives, derivatives are automaticcaly 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.

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 symmetrise 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` subdirectory.
- 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

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 apear 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 PLU↩ MED 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 SPE↩ CIESA 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.

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.

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.

Version 2.3.5 (Mar 2, 2018)

For users:

- Fixed `plumed partial_tempering` to agree with GROMACS conventions for the choice of dihedrals (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.

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).

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).

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 hardcoded so as to facilitate conda packaging (see #416).

*/

2.5 Version 2.4

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 biasfactor 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 gaussians
  - PBMETAD : default names for GRID and FILE (usefull 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.

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 averged gradient and output quantities related to it.
- RMSD variable and other collective variables using reference PDBs 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.

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.

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.

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 threadsafety 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

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).

Version 2.4.6 (Jul 19, 2019)

Warning

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

For users:

- Fixed a bug in COORDINATIONNUMBER where derivatives were wrong when using R_POWER $>$ 2, thanks to 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).

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 regularily 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 pre-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 functionalities of 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-matheval
```

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 matheval 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 matheval without any additional flag, and then in case of failure will retry adding "-lmatheval" to the LIBS options. If also this does not work, the matheval library will be disabled and some features will not be available. This procedure allows you to use libraries with custom names. So, if your matheval library is called /opt/local/lib/libmymatheval.so you can link it with

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

In this example, the linker will directly try to link `/opt/local/lib/libmymatheval.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 matheval example, it would be `-lmatheval`) 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=-lmatheval
```

will make sure that only matheval 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 hardcode 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 hardcoded). On the other hand, on Linux it is common pratice not to hardcode 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 hardcode 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 hardcode 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 slighty 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/matheval -lmatheval 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_MATHEVAL to enable the matheval 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 only static executables 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 (e.g. Matheval) 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 unsetting 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

If you configure PLUMED with VMD's plugins you will be able to read many more trajectory formats. 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/pathtovmdplugi
```

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.

3.3 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 optimizations or even architecture dependent features trigger bugs that did not show up on travis. So please always perform 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 hardcoded 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. The standard way to do it is during the configure step:

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

However, you can also change it after compilation setting the variable `prefix`.

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

If you didn't specify the `--prefix` option during configure, and you did not set the `prefix` variable when installing, 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.

Warning

> Please **do not** set prefix to the current directory (`./configure --prefix=$PWD`). PLUMED expects the installation directory to be a different one! You might want to use something like `./configure --prefix=$PWD/install` instead.

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, executables are 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 amongst 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 the PLUMED internal functionalities (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:

- amber14
- gromacs-2016-6
- gromacs-2018-6
- gromacs-4-5-7
- gromacs-5-1-4
- lammps-6Apr13
- namd-2-12
- namd-2-13
- qespresso-5-0-2
- qespresso-6-2

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 executible 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 enginge (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 PLUMED_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 it is not currently possible to link PLUMED as a static library (something like 'libplumed.a'). The reason for this is that PLUMED heavily relies on C++ static constructors that do not behave well in static libraries. For this reason, to produce a static executable with an MD code + PLUMED we link PLUMED as a collection of object files.

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 precompiled 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 hardcoded. 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 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 additionaly features if specific libraries are available upon compiling it. Install libmatheval first and check if PLUMED `./configure` is detecting it. Libmatheval is a must have with PLUMED. If someone uses gromacs, install libxdrfile first and check if PLUMED `./configure` is detecting it. PLUMED will be able to write trr/xtc file, simplifying analysis.

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. libmatheval).

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 `libmatheval`, 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 libmatheval is, without the need to load libmatheval 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.

3.9 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.

3.10 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.

3.10.1 amber14

PLUMED can be incorporated into amber (sander module) using the standard patching procedure. Patching must be done in the root directory of amber *before* compilation.

To enable PLUMED in a sander simulation one should use add to the cntrl input namelist these two fields:

plumed=1 , plumedfile='plumed.dat'

The first is switching plumed on, the second is specifying the name of the plumed input file.

This patch is compatible with the MPI version of sander and support multisander. However, replica exchange is not supported. Multisander can thus only be used for multiple walkers metadynamics or for ensemble restraints.

For more information on amber you should visit `http://ambermd.org`

3.10.2 gromacs-2016.6

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`

3.10.3 gromacs-2018.6

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

3.10.4 gromacs-4.5.7

PLUMED can be incorporated into gromacs using the standard patching procedure. Patching must be done in the gromacs source directory *after* gromacs has been configured but *before* gromacs is compiled. Gromcas should be configured with ./configure (not cmake).

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.:

mdrun -plumed plumed.dat

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

3.10.5 gromacs-5.1.4

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

This patch also implements the -hrex keyword for gromacs. See Using Hamiltonian replica exchange with GROMACS

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

3.10.6 lammps-6Apr13

PLUMED can be incorporated into LAMMPS using a simple patching procedure. Patching must be done *before* LAMMPS is configured. After patching, one should enable PLUMED using the command make yes-user-plumed In the same way, before reverting one should disable PLUMED using the command make no-user-plumed

Also notice that command "fix plumed" should be used in lammps input file *after* the relevant input parameters have been set (e.g. after "timestep" command)

Bug The provided patch does not pass the virial correctly to PLUMED (see https://github.com/plumed/plumed2/issues/377). If you need to perform constant pressure simulations you should download a LAMMPS version newer than November 2018, where the mentioned bug has been fixed, that natively supports PLUMED 2.4.

See also http://lammps.sandia.gov/doc/Section_commands.html for further info on processing LAMMPS input, as well as this discussion on github: http://github.com/plumed/plumed2/issues/67.

For more information on LAMMPS you should visit http://lammps.sandia.gov/

3.10.7 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/

3.10.8 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/

3.10.9 qespresso-5.0.2

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

3.10.10 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

Chapter 4

Getting Started

To run PLUMED you need to provide one input file. In this 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 biassing 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 bounadry 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 commma 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
DISTANCE ATOMS=1,2
```

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

```
BEGIN_PLUMED_FILE
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
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 analisys 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 how to convert from the plumed default unit into that new unit or by using the shortcuts described below. This directive MUST appear at the BEGINNING of the plumed.dat file. The same units must be used througout 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.

Options

NATURAL	(default=off) use natural units
LENGTH	the units of lengths. Either specify a conversion factor from the default, nm, or A (for angstroms) or um
ENERGY	the units of energy. Either specify a conversion factor from the default, kj/mol, or use j/mol or kcal/mol
TIME	the units of time. Either specify a conversion factor from the default, ps, or use ns or fs
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

Examples

```
BEGIN_PLUMED_FILE
# this is using nm - 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, *irrespectively* of which units you are using the 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
UNITS LENGTH=0.1 TIME=0.001
```

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 postions directly. Consquentially, 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 encourntered. 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 implementing having an unmanigiably large ammount 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=10,11,12,13,14,15,16,17,18,19,20 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
- all the above methods together. For example GROUP ATOMS=1,2,10-20,40-60:5,100-70:-2 LABEL=g5.

Some collective variable must accept a fixed number of atoms, for example a DISTANCE is calculated using two atoms only, an ANGLE is calcuated 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 probem - 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
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
# 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_OF_MULTICOLVAR	Calculate a a weighted average position based on the value of some multi-colvar.
CENTER	Calculate the center for a group of atoms, with arbitrary weights.
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 coordinatesin the local reference frame formed by three atoms.The computed ghost atom is stored as a virtual atom that can be accessed inan atom list through the the label for the GHOST action that creates it.

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
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.
- If `NDX_FILE` is present, then append to it the list obtained from the gromacs group.
- If `REMOVE` is present, then remove the first occurence 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 resuling 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.

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

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
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
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
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 can also be imported

```
BEGIN_PLUMED_FILE
# 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 hydrogens in this manner

```
BEGIN_PLUMED_FILE
# 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
```

5.1.4 MOLINFO

This is part of the setup 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.

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 whenenver 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 keyoword.

Using MOLINFO with a protein's or nucleic acid's pdb extends the possibility of atoms selection using the @ special symbol in the form

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

So for example

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

In the following are listed the current available definitions:

For protein residues, the following groups are available:

```
@phi-#
@psi-#
@omega-#
@chi1-#
```

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-chainresiduenum` or `@atom-residuenum`.

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.

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

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
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
MOLINFO STRUCTURE=reference.pdb
hb1: DISTANCE ATOMS=@N2-1,@O2-14
hb2: DISTANCE ATOMS=@N1-1,@N3-14
hb3: DISTANCE ATOMS=@O6-1,@N4-14
PRINT ARG=hb1,hb2,hb3
```

This example use MOLINFO to calculate torsions angles

```
BEGIN_PLUMED_FILE
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
t1: TORSION ATOMS=@phi-3
t2: TORSION ATOMS=@psi-4
PRINT ARG=t1,t2 FILE=colvar STRIDE=10
```

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 consecute atoms (1-100), but it is also possible to skip some atoms, provided consecute chosen atoms are close enough.

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

MOLTYPE	the type of molecule that is under study. This is used to define the backbone atoms

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
```

```
# 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
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
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES RESIDUES=all MOLTYPE=protein
```

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 correclty 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.

Warning

The molecule used for alignment should be whole. In case it is broken by the host MD code, please use WHOLEMOLECULES to reconstruct it before FIT_TO_TEMPLATE .

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.

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
# 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
# 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 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. It could be for instance the center of mass of a ligand with respect to a protein

```
BEGIN_PLUMED_FILE
# center of the ligand:
ce: CENTER ATOMS=100-110

FIT_TO_TEMPLATE REFERENCE=protein.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=ce,fix

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

Notice that you could have obtained an (almost) identical result adding a fictitious atom to `ref.pdb` with the serial number corresponding to the `ce` atom (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 PBCs (such as DISTANCE without `NOPBC`, as in the example above) are allowed after FIT_TO_TEMPLATE, whereas RMSD expects PBCs to be already solved. The latter means that before the RMSD statement one should use WRAPAROUND or WHOLEMOLECULES to properly place the ligand.

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 analyse 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).

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

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
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
# 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
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 oxygens will be brought close to the solute, and all the hydrogens will be kept close to their related oxygen.

```
BEGIN_PLUMED_FILE
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
```

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.

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

Examples

Reset cell to be triangular after a rototranslational fit

```
BEGIN_PLUMED_FILE
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
```

5.1.9 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

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

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 recognise 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 recognise 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
c1: COORDINATIONNUMBER SPECIES=1-512 SWITCH={EXP D_0=4.0 R_0=0.5}
cc: CENTER_OF_MULTICOLVAR DATA=c1
```

The first line here calclates 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
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 coordinations 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.

5.1.10 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 PBCs 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.

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
WEIGHTS	Center is computed as a weighted average.

Examples

```
BEGIN_PLUMED_FILE
# 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
```

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 PBCs 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.

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

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
c1: COM ATOMS=1-7
c2: COM ATOMS=15,20
d1: DISTANCE ATOMS=c1,c2
PRINT ARG=d1
```

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 coordinats (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 to 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.

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

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
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 on a template and then compute the distance of one of its atom from the point (10,20,30).

```
BEGIN_PLUMED_FILE
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
```

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.

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

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
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
```

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 quantitieswith 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_FROM_CONTOUR	Calculate the perpendicular distance from a Willard-Chandler dividing surface.
DISTANCE	Calculate the distance between a pair of atoms.
EEFSOLV	Calculates EEF1 solvation free energy for a group of atoms.
ENERGY	Calculate the total energy of the simulation box.
ERMSD	Calculate eRMSD with respect to a reference structure.
FAKE	This is a fake colvar container used by cltools or various other actionsand just support input and period definition
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.
PATHMSD	This Colvar calculates path collective variables.
PATH	Path collective variables with a more flexible framework for the distance metric being used.
PCAVARS	Projection on principal component eigenvectors or other high dimensional linear subspace
POSITION	Calculate the components of the position of an atom.
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:
JCOUPLING	(from PLUMED-ISDB module) Calculates ^3J coupling constants for a dihedral angle.
NOE	(from PLUMED-ISDB module) Calculates NOE intensities as sums of 1/r^6, also averaging over multiple equivalent atomsor ambiguous NOE.
PCS	(from PLUMED-ISDB module) Calculates the Pseudocontact shift of a nucleus determined by the presence of a metal ion susceptible to anisotropic magnetization.
PRE	(from PLUMED-ISDB module) Calculates the Paramegnetic Resonance Enhancement intensity ratio between a spinlabel atom and a list of atoms .
RDC	(from PLUMED-ISDB module) Calculates the (Residual) Dipolar Coupling between two atoms.
SAXS	(from PLUMED-ISDB module) Calculates SAXS scattered intensity using the Debye equation.

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.

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
UPDATE	the frequency with which the path should be updated
TOLERANCE	(default=1E-6) the tolerance to use for the path updating algorithm that makes all frames equidistant
FMT	(default=f) the format to use for output files

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
SERIAL	(default=off) do the calculation in serial. Do not parallelize
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

Examples

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

```
BEGIN_PLUMED_FILE
d1: DISTANCE ATOMS=1,2 COMPONENTS
pp: ADAPTIVE_PATH TYPE=EUCLIDEAN FIXED=5,15 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
```

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

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.

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 torsions 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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

Examples

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

```
BEGIN_PLUMED_FILE
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
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 torsions 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
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 4th residue of the protein.

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 interatomic 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 anthing other than TYPE=DRMSD. For more details as to how to do this see WHOLEMOLECULES.

Description of components

By default this Action calculates the number of structural units that are within a certain distance of a idealised 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 referened 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 are customizable. In particular, by using the LABEL keyword in the description of you LESS_THAN function you can set name of the component that you are calculating

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 lowest of the quantitities 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 quantitities 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
VERBOSE	(default=off) write a more detailed output
SERIAL	(default=off) do the calculation in serial. Do not parallelize

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

Examples

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

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=helix.pdb
hh: ALPHARMSD RESIDUES=all
```

Here the same is done use RMSD instead of DRMSD

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES ENTITY0=1-100
hh: ALPHARMSD RESIDUES=all TYPE=OPTIMAL R_0=0.1
```

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 i-th and the j-th 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.

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

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

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
```

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 continguous 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 interatomic 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 resemebles an anti paralel 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 anthing other than TYPE=DRMSD. For more details as to how to do this see WHOLEMOLECULES.

Description of components

By default this Action calculates the number of structural units that are within a certain distance of a idealised 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 referened 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 are customizable. In particular, by using the LABEL keyword in the description of you LESS_THAN function you can set name of the component that you are calculating

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 lowest of the quantitities 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 quantitities 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
VERBOSE	(default=off) write a more detailed output
SERIAL	(default=off) do the calculation in serial. Do not parallelize
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

Examples

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

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=beta.pdb
ab: ANTIBETARMSD RESIDUES=all STRANDS_CUTOFF=1
```

Here the same is done use RMSD instead of DRMSD

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES ENTITY0=1-100
hh: ANTIBETARMSD RESIDUES=all TYPE=OPTIMAL R_0=0.1  STRANDS_CUTOFF=1
```

5.2.6 CELL

This is part of the colvar module

Calculate the components of the simulation cell

Description of components

By default this Action calculates the following quantities. These quanties 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

Examples

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

```
BEGIN_PLUMED_FILE
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
```

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.

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 these customizable components the following quantities will always be output

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

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
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
cn: CONSTANT VALUE=1.0
dis: DISTANCE ATOMS=1
sss: SORT ARG=cn,dis
PRINT ARG=sss.1
```

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`.

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 quanties 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 contant 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...

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
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 SWITH), while R_0 is the contact distance. WEIGHT gives the relative weight of each contact to the final distance measure.

```
BEGIN_PLUMED_FILE
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
# Full 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
[snip]
ATOMS275=183,213 SWITCH275={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.355} WEIGHT275=0.003597
ATOMS276=183,234 SWITCH276={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.428} WEIGHT276=0.003597
ATOMS277=183,250 SWITCH277={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.3832} WEIGHT277=0.003597
ATOMS278=197,220 SWITCH278={Q R_0=0.01 BETA=50.0 LAMBDA=1.5 REF=0.3827} WEIGHT278=0.003597
LABEL=cmap
SUM
... CONTACTMAP

PRINT ARG=cmap FILE=colvar
```

(See also switchingfunction)

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 practise, s_{ij} is replaced with a switching function to make it differentiable. 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".

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 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
SWITCH	This keyword is used if you want to employ an alternative to the continuous swiching 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.

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 neighbour list is used to make this calculation faster, this neighbour list is updated every 100 steps.

```
BEGIN_PLUMED_FILE
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
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
# 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

# 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
```

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.

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 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

Examples

```
BEGIN_PLUMED_FILE
# 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
```

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.

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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

Examples

The following provides an example input for the dihcor action

```
BEGIN_PLUMED_FILE
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 correation 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 betwene 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 torsions 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
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
DIHCOR ...
ATOMS1=@phi-3,@psi-3
ATOMS2=@psi-3,@phi-4
ATOMS4=@phi-4,@psi-4
... DIHCOR
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 4th residue of the protein.

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, JCTC 13, 425 (2017). 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.

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 centroids.

Examples

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

```
BEGIN_PLUMED_FILE
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
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 atomlists 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
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
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
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.

5.2.13 DIPOLE

	This is part of the colvar module

Calculate the dipole moment for a group of atoms.

Warning

The atoms used for DIPOLE calculation should be from a whole molecule. In case the molecule is broken by the host MD code, please use WHOLEMOLECULES to reconstruct it before DIPOLE calculation.

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 quanties 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
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

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
d: DIPOLE GROUP=1-10
PRINT FILE=output STRIDE=5 ARG=5
```

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.

5.2.14 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 caculates the distance projected on the x, y or z axis between the position of some test particle and this surface of constant field density.

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 amongst 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 the label of the components customizable. 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 refererred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be refererred 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 lowest of the quantitities 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 quantitities 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 refererred 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 esimtation
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 absoluate 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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

Examples

In this example atoms 2-100 are assumed to be concentraed along some part of the z axis so that you an interface between a liquid/solid and the vapour. 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
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
```

5.2.15 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.

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 quanties 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

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
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
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
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
# 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).

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 neighbourlist 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.

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	The buffer to the intrinsic cutoff used when calculating pairwise interactions.
NL_STRIDE	The frequency with which the neighbourlist 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
TEMP_CORRECTION	(default=off) Correct free energy of solvation constants for temperatures different from 298.15 K

Examples

```
BEGIN_PLUMED_FILE
MOLINFO MOLTYPE=protein STRUCTURE=peptide.pdb
WHOLEMOLECULES ENTITY0=1-111

# This allows us to select only non-hydrogen atoms
protein-h: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein-H

# We extend the cutoff by 0.2 nm and update the neighbourlist every 10 steps
solv: EEFSOLV ATOMS=protein-h NL_STRIDE=10 NL_BUFFER=0.2

# Here we actually add our calculated energy back to the potential
bias: BIASVALUE ARG=solv

PRINT ARG=solv FILE=SOLV
```

5.2.17 ENERGY

This is part of the colvar module

Calculate the total energy of the simulation box.

Total energy can be biased with umbrella sampling [9] or with well tempered metadynamics [10].

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 Acceptance for replica exchange when ENERGY is biased is computed correctly only of all the replicas has the same potential energy function. This is for instance not true when using GROMACS with lambda replica exchange of with plumed-hrex branch.

Examples

The following input instructs plumed to print the energy of the system

```
BEGIN_PLUMED_FILE
ene: ENERGY
PRINT ARG=ene
```

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 $\theta \approx 60^{\circ}$.
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 weghting 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 [11]

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.

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.

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
MOLINFO STRUCTURE=reference.pdb
eRMSD1: ERMSD REFERENCE=reference.pdb ATOMS=@lcs-1,@lcs-2,@lcs-3,@lcs-4,@lcs-5,@lcs-6
```

5.2.19 FAKE

This is part of the colvar module

This is a fake colvar container used by cltools or various other actions and just support input and period definition

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 PERIODIC=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

Examples

```
BEGIN_PLUMED_FILE
FAKE ATOMS=1 PERIODIC=-3.14,3.14   LABEL=d2
```

5.2.20 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 [12]. 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 ammount by which a set of collective variables change. As such this implementation of the propertymap 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.

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 parallelize
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

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
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
```

5.2.21 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 [13]

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 PBCs 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.

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

Examples

The following input tells plumed to print the radius of gyration of the chain containing atoms 10 to 20.

```
BEGIN_PLUMED_FILE
GYRATION TYPE=RADIUS ATOMS=10-20 LABEL=rg
PRINT ARG=rg STRIDE=1 FILE=colvar
```

5.2.22 PARABETARMSD

	This is part of the secondarystructure module

Probe the parallel beta sheet content of your protein structure.

Two protein segments containing three continguous 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 interatomic 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 anthing other than TYPE=DRMSD. For more details as to how to do this see WHOLEMOLECULES.

Description of components

By default this Action calculates the number of structural units that are within a certain distance of a idealised 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 referened 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 are customizable. In particular, by using the LABEL keyword in the description of you LESS_THAN function you can set name of the component that you are calculating

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 lowest of the quantitities 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 quantitities 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, S↩ IMPLE 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
VERBOSE	(default=off) write a more detailed output
SERIAL	(default=off) do the calculation in serial. Do not parallelize
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

Examples

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

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=beta.pdb
pb: PARABETARMSD RESIDUES=all STRANDS_CUTOFF=1
```

Here the same is done use RMSD instead of DRMSD

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=helix.pdb
WHOLEMOLECULES ENTITY0=1-100
hh: PARABETARMSD RESIDUES=all TYPE=OPTIMAL R_0=0.1  STRANDS_CUTOFF=1
```

5.2.23 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).

Warning

The molecule used for PATHMSD calculation should be whole (both atoms used in alignment and in displacement calculation). In case it is broken by the host MD code, please use WHOLEMOLECULES to reconstruct it before PATHMSD calculation.

Description of components

By default this Action calculates the following quantities. These quanties 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
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
LOG_CLOSE	same as LOG-CLOSE
DEBUG_CLOSE	same as DEBUG-CLOSE

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
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 neighborlist parameter (optional but recommended for performance) and states that the neighbor list will be calculated every 4 timesteps and consider only the closest 8 member to the actual md snapshots.

In the REFERENCE PDB file the frames must be separated either using END or ENDMDL.

Note

The implementation of this collective variable and of PROPERTYMAP is shared, as well as most input options.

5.2.24 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 ammount 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 [14]. 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:

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

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.

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 quanties 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 parallelize
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...

Examples

In the example below the path is defined using RMSD distance from frames. The reference frames in the path are defined in the pdb file. In this frame each configuration in the path is separated by a line containing just the word END.

```
BEGIN_PLUMED_FILE
p1: PATH REFERENCE=file.pdb TYPE=OPTIMAL LAMBDA=500.0
PRINT ARG=p1.sss,p1.zzz STRIDE=1 FILE=colvar FMT=%8.4f
```

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 alegebraic expressions that are used by default.

```
BEGIN_PLUMED_FILE
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 formulae 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.

The following input instructs PLUMED to calculate the values of the path collective variables. The frames that make up this path are defined in the file all.pdb and all distances are measured using the OPTIMAL metric that is discussed in the manual page on RMSD.

```
BEGIN_PLUMED_FILE
p2: PATH REFERENCE=all.pdb LAMBDA=69087
PRINT ARG=p2.spath,p2.zpath STRIDE=1 FILE=colvar
```

If you wish to use collective variable values in the definition of your path you would use an input file with something like this:

```
BEGIN_PLUMED_FILE
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=3,4a
p2: PATH REFERENCE=mypath.pdb LAMBDA=2 TYPE=EUCLIDEAN
PRINT ARG=p2.spath,p2.zpath STRIDE=1 FILE=colvar
```

The corresponding pdb file containing the definitions of the frames in the path would then look like this:

```
DESCRIPTION: a defintiion of a PATH
REMARK TYPE=EUCLIDEAN
REMARK ARG=d1,d2
REMARK d1=1.0 d2=1.0
END
REMARK TYPE=EUCLIDEAN
REMARK ARG=d1,d2
REMARK d1=2.0 d2=2.0
END
```

For each frame in the path you must specify the arguments that should be used to calculate the distance between the instantaneous configuration of the system and the reference configurations together with the values that these arguments take in each of the reference configurations.

5.2.25 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 instaneous 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 explicitally 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.

Description of components

By default this Action calculates the following quantities. These quanties 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, e_i, that are contained in the rows of A. In other words this is $\sqrt{(r^2 - \sum_i [\mathbf{r}.\mathbf{e_i}]^2)}$ where r is the distance between the instantaneous position and the reference point.

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

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
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 refered 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:

```
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
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 explicitally. 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 DIRECTION. 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
```

5.2.26 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.

Description of components

By default this Action calculates the following quantities. These quanties 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

Examples

```
BEGIN_PLUMED_FILE
# align to a template
FIT_TO_TEMPLATE REFERENCE=ref.pdb
p: POSITION ATOM=3
PRINT ARG=p.x,p.y,p.z
```

5.2.27 PROPERTYMAP

This is part of the colvar module

Calculate generic property maps.

This Colvar calculates the property maps according to the work of Spiwok [12].

Basically it calculates

$$X = \frac{\sum_i X_i * \exp(-\lambda D_i(x))}{\sum_i \exp(-\lambda D_i(x))} \tag{5.1}$$

$$Y = \frac{\sum_i Y_i * \exp(-\lambda D_i(x))}{\sum_i \exp(-\lambda D_i(x))} \tag{5.2}$$

$$\dots \tag{5.3}$$

$$zzz = -\frac{1}{\lambda}\log(\sum_i \exp(-\lambda D_i(x))) \tag{5.4}$$

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 MSD after optimal alignment calculated on the pdb frames you input (see Kearsley).

Warning

The molecule used for PROPERTYMAP calculation should be whole (both atoms used in alignment and in displacement calculation). In case it is broken by the host MD code, please use WHOLEMOLECULES to reconstruct it before PROPERTYMAP calculation.

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 these customizable components 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
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
LOG_CLOSE	same as LOG-CLOSE
DEBUG_CLOSE	same as DEBUG-CLOSE

Examples

```
BEGIN_PLUMED_FILE
p3: PROPERTYMAP REFERENCE=../../trajectories/path_msd/allv.pdb PROPERTY=X,Y LAMBDA=69087 NEIGH_SIZE=8 NEIGH_ST
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 neighborlist parameter (optional but recommended for performance) and states that the neighbor list will be calculated every 4 timesteps 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.

5.2.28 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 [15] . 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 [15] . 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 [16] as also discussed in [17] . 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).

Bug The 6-membered ring implementation distributed with this version of PLUMED leads to qx and qy values that have an opposite sign with respect to those originally defined in [16]. The bug will be fixed in a later version but is here kept to preserve reproducibility.

Components of this action are:

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 quanties 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

Examples

This input tells plumed to print the puckering phase angle of the 3rd nucleotide of a RNA molecule on file COLVAR.

```
BEGIN_PLUMED_FILE
 MOLINFO STRUCTURE=rna.pdb MOLTYPE=rna
 PUCKERING ATOMS=@sugar-3 LABEL=puck
 PRINT ARG=puck.phs FILE=COLVAR
```

5.2.29 TEMPLATE

	This is part of the colvar module

This file provides a template for if you want to introduce a new CV.

The atoms involved can be specified using

ATOMS	the keyword with which you specify what atoms to use should be added like this. For more information on how to specify lists of atoms see Groups and Virtual Atoms

Compulsory keywords

TEMPLATE_COMPULSORY	all compulsory keywords should be added like this with a description here

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
NOPBC	(default=off) ignore the periodic boundary conditions when calculating distances
TEMPLATE_DEFAULT_OFF_FLAG	(default=off) flags that are by default not performed should be specified like this
TEMPLATE_DEFAULT_ON_FLAG	(default=on) flags that are by default performed should be specified like this
TEMPLATE_OPTIONAL	all optional keywords that have input should be added like a description here

Examples

```
BEGIN_PLUMED_FILE
# This should be a sample input.
t: TEMPLATE ATOMS=1,2
PRINT ARG=t STRIDE=100 FILE=COLVAR
```

(see also PRINT)

5.2.30 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.

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

Examples

This input tells plumed to print the torsional angle between atoms 1, 2, 3 and 4 on file COLVAR.

```
BEGIN_PLUMED_FILE
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 TOR↩ SION in combination with the MOLINFO command. This can be done by using the following syntax.

```
BEGIN_PLUMED_FILE
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 4th 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
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
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
```

5.2.31 VOLUME

	This is part of the colvar module

Calculate the volume of the simulation box.

Options

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

Examples

The following input tells plumed to print the volume of the system

```
BEGIN_PLUMED_FILE
vol: VOLUME
PRINT ARG=vol
```

5.3 Distances from reference configurations

One colvar that has been shown to be very sucessful 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 instantanous 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 practise 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 ammount by which a large number of collective variables change in moving from the reference to the instaneous configurations. One could even combine RMSD distances with the ammount 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	An alias to the MULTI-RMSD function.
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 (see [18] and [19]) for a number of provided eigenvectors and an average structure. 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)
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 someset 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

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

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
DRMSD REFERENCE=file.pdb LOWER_CUTOFF=0.1 UPPER_CUTOFF=0.8
```

The following tells plumed to calculate a DRMSD value for a pair of molecules.

```
BEGIN_PLUMED_FILE
DRMSD REFERENCE=file.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.

5.3.2 MULTI_RMSD

This is part of the colvar module

An alias to the MULTI-RMSD function.

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
SQUARED	(default=off) This should be setted if you want MSD instead of RMSD

Examples

Just replace MULTI-RMSD with MULTI_RMSD

```
BEGIN_PLUMED_FILE
MULTI_RMSD REFERENCE=file.pdb TYPE=MULTI-DRMSD
```

5.3.3 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.

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
SQUARED	(default=off) This should be setted if you want MSD instead of RMSD

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 Kearseley algorithm for each of the domains.

```
BEGIN_PLUMED_FILE
MULTI-RMSD REFERENCE=file.pdb TYPE=MULTI-OPTIMAL
```

The following tells plumed to calculate the RMSD distance btween 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
MULTI-RMSD REFERENCE=file.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
```

5.3.4 PCARMSD

	This is part of the colvar module

Calculate the PCA components (see [18] and [19]) for a number of provided eigenvectors and an average structure. 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)

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 quanties 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 MSD 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 setted if you want RMSD instead of MSD
SQUARED_ROOT	(default=off) Same as SQUARED-ROOT

Examples

```
BEGIN_PLUMED_FILE
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).

5.3.5 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 1st, 4th, 6th and 28th atoms in the MD codes input file then the indices of the corresponding refernece 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 centre 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 correclty 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 [20] algorithm. Again different sets of weights are used for the alignment (w') and for the displacement calcuations (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.

Warning

The molecule used for RMSD calculation should be whole (both atoms used in alignment and in displacement calculation). In case it is broken by the host MD code, please use WHOLEMOLECULES to reconstruct it before RMSD calculation.

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
SQUARED	(default=off) This should be setted if you want MSD instead of RMSD

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 Kearseley algorithm is used so this is done optimally.

```
BEGIN_PLUMED_FILE
RMSD REFERENCE=file.pdb TYPE=OPTIMAL

...
```

5.3.6 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 instaneous 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 containg the values of the CVs in the reference configuration. $\mathbf{\Sigma}$ is then an $N \times N$ matrix that is specified in the input.

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

Examples

The following input calculates the distance between a reference configuration and the instaneous 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
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=3,4
t1: TARGET REFERENCE=myref.pdb TYPE=EUCLIDEAN
PRINT ARG=t1 FILE=colvar
```

The contents of the file containing the reference structure (myref.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
```

5.4 Functions

When performing biased dynamics or analysing a trajectory you may wish to analyse/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
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	An alias to the MATHEVAL function.
ENSEMBLE	Calculates the replica averaging of a collective variable over multiple replicas.
FUNCPATHMSD	This function calculates path collective variables.
FUNCSUMHILLS	This function is intended to be called by the command line tool sum_hillsand it is meant to integrate a HILLS file or an HILLS file interpreted asa histogram i a variety of ways. Therefore it is not expected that you use thisduring your dynamics (it will crash!)
LOCALENSEMBLE	Calculates the average over multiple arguments.
MATHEVAL	Calculate a combination of variables using a matheval expression.
PIECEWISE	Compute a piecewise straight line through its arguments that passes througha 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 particular it calculates and store as components the sum of the squared deviations, the correlation, theslope and the intercept of a linear fit.

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

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.

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 coefficents 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 proceding 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 componets 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...

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
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
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
```

5.4.2 CUSTOM

	This is part of the function module

An alias to the MATHEVAL function.

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 proceding 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 componets 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.

Examples

Just replace MATHEVAL with CUSTOM.

```
BEGIN_PLUMED_FILE
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,n FILE=checkme
RESTRAINT ARG=d AT=0.5 KAPPA=10.0
```

(see also DISTANCE, PRINT, and RESTRAINT)

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.

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 proceding 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 componets 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)

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
dist: DISTANCE ATOMS=3,5
ens: ENSEMBLE ARG=dist
PRINT ARG=dist,ens.dist
```

5.4.4 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 MSD that are obtained by the joint use of MSD variable and SQUARED flag (see below).

Description of components

By default this Action calculates the following quantities. These quanties 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 proceding 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 componets 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

Examples

Here below is a case where you have defined three frames and you want to calculate the progress alng the path and the distance from it in p1

```
BEGIN_PLUMED_FILE
t1: RMSD REFERENCE=frame_1.dat TYPE=OPTIMAL SQUARED
t2: RMSD REFERENCE=frame_21.dat TYPE=OPTIMAL SQUARED
t3: RMSD REFERENCE=frame_42.dat 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
```

In this second example is shown how to define a PATH in the CONTACTMAP space:

```
BEGIN_PLUMED_FILE
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
```

5.4.5 FUNCSUMHILLS

This is part of the function module

This function is intended to be called by the command line tool sum_hills and it is meant to integrate a HILLS file or an HILLS file interpreted as a histogram i a variety of ways. Therefore it is 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

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
ISCLTOOL	(default=on) use via plumed commandline: 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 welltempered 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 proceding 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 componets 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 monodimensional 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

Examples

There are currently no examples for this keyword.

5.4.6 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.

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 proceding 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 componets 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...

Examples

The following input tells plumed to calculate the chemical shifts for four different proteins in the same simulation box then average them, calcualated the sum of the squared deviation with respect to the experiemntal 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.,
```

5.4.7 MATHEVAL

This is part of the function module

Calculate a combination of variables using a matheval expression.

This action computes an arbitrary function of one or more precomputed 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).

If you want a function that depends not only on collective variables but also on time you can use the TIME action.

Attention

The MATHEVAL object only works if one of these conditions is satisfied: (a) libmatheval is installed on the system and PLUMED has been linked to it or (b) the environment variable `PLUMED_USE_LEPTON` is set equal to `yes` at runtime (in this case, the internal lepton library will be used). Notice that in version v2.5 the internal lepton library will be used by default.

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 proceding 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 componets 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.

Examples

The following input tells plumed to perform a metadynamics using as a CV the difference between two distances.

```
BEGIN_PLUMED_FILE
dAB: DISTANCE ATOMS=10,12
dAC: DISTANCE ATOMS=10,15
diff: MATHEVAL 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 WIDTH=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 MATHEVAL 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, which does not require libmatheval to be installed on your system.

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
DISTANCE LABEL=d1 ATOMS=1,2 COMPONENTS
DISTANCE LABEL=d2 ATOMS=2,3 COMPONENTS
MATHEVAL ...
  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
... MATHEVAL
PRINT ARG=theta
```

(See also PRINT and DISTANCE).

Notice that the matheval library 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
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,n 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 MATHEVAL 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.

MATHEVAL 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
# 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: MATHEVAL 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: MATHEVAL 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 [21].

5.4.7.1 TIME

	This is part of the generic module

retrieve the time of the simulation to be used elsewere

Options

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

Examples

```
BEGIN_PLUMED_FILE
TIME            LABEL=t1
PRINT ARG=t1
```

5.4.8 PIECEWISE

	This is part of the function module

Compute a piecewise 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.

Description of components

By default this Action calculates the following quantities. These quanties 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 will be referenceable 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 piecewise 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 proceding 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 componets 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...

Examples

```
BEGIN_PLUMED_FILE
dist1: DISTANCE ATOMS=1,10
dist2: DISTANCE ATOMS=2,11

pw: PIECEWISE POINT0=1,10 POINT1=1,PI POINT2=3,10 ARG=dist1
ppww: PIECEWISE POINT0=1,10 POINT1=1,PI POINT2=3,10 ARG=dist1,dist2
PRINT ARG=pw,ppww.dist1_pfunc,ppww.dist2_pfunc
```

5.4.9 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.

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 proceding 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 componets 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...

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
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
```

5.4.10 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 store 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).

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 these customizable components the following quantities will always be output

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 proceding 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 componets 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

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
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.*
```

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 funciton 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 dupilcating 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 ofcoordination 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 calculatethe 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 ofdistances such as the minimum, the number less than a certain quantity and so on.
FCCUBIC	Measure how similar the environment around atoms is to that found in a FCC structure.
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 4th order Steinhardt parameters.
Q6	Calculate 6th order Steinhardt parameters.
SIMPLECUBIC	Calculate whether or not the coordination spheres of atoms are arranged as they would be in a simplecubic 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 ofvalues 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.You can then calculate functions of the distribution ofvalues such as the minimum, the number less than a certain quantity and so on.

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.You can then calculate functions of the distribution ofvalues such as the minimum, the number less than a certain quantity and so on.
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 ofvalues 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.You can then calculate functions of the distribution ofvalues such as the minimum, the number less than a certain quantity and so on.
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 ofvalues 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
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 quanties that are to be calculated from the atoms involved and any parameters you need to specify what function of these base quanties 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 multicolvarso 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 multicolvarso 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 multicolvarso that one can compute the mean and so on for only those multicolvars more than a tolerance.

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 collectivevariables 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 collectivevariables 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 collectivevariables 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 collectivevariables 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 collectivevariables 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 variablesfor 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 argumet 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 [22] 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 torsions 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 vectoron 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 vectoron 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 vectoron 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"
POLYMER_ANGLES	Calculate a function to investigate the relative orientations of polymer angles
SMAC	Calculate a variant on the SMAC collective variable discussed in [23]

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 useed to transform the colvar values calculated by a multicolvar using a histogrambead
MTRANSFORM_LESS	This action can be useed to transform the colvar values calculated by a multicolvar using a switchingfunction
MTRANSFORM_MORE	This action can be useed to transform the colvar values calculated by a multicolvar using one minus a switchingfunction

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
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 forcefield or to implement your forcefield.

5.5.3 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 [24].

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 amongst 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 the label of the components customizable. 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 refererred 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 conjuction 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 conjuction 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 parallelize
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...
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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

Examples

The following example instructs plumed to find the average of two angles and to print it to a file

```
BEGIN_PLUMED_FILE
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
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. A discretized-normalized histogram of the distribution is then output

```
BEGIN_PLUMED_FILE
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
```

5.5.4 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.

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 amongst 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 the label of the components customizable. 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 refererred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 swiching 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...

Examples

5.5.5 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.

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 parallelize
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

Examples

The following example instructs plumed to calculate the number of water molecules that are bridging betweeen atoms 1-10 and atoms 11-20 and to print the value to a file

```
BEGIN_PLUMED_FILE
BRIDGE BRIDGING_ATOMS=100-200 GROUPA=1-10 GROUPB=11-20 LABEL=w1
PRINT ARG=a1.mean FILE=colvar
```

5.5.6 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.

To make the calculation of coordination numbers differentiable 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 [25] 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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
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
```

5.5.7 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.

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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

Examples

The following example calculates the number of atoms in one half of the simulation box.

```
BEGIN_PLUMED_FILE
DENSITY SPECIES=1-100 LABEL=d
AROUND ARG=d XLOWER=0.0 XUPPER=0.5 LABEL=d1
PRINT ARG=d1.* FILE=colvar1 FMT=%8.4f
```

5.5.8 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
DISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
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
DISTANCES ATOMS1=3,5 ATOMS2=1,2 LABEL=d1 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lt0.1
```

(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
DISTANCES GROUP=1-3 MEAN LABEL=d1
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
DISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.gt0.1
```

(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
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 differentiability the minimum is calculated using 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 coordinatetion 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
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.}
... DISTANCES
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.

5.5.9 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 [26] and again in this article [27] 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
FCCUBIC SPECIES=1-64 SWITCH={RATIONAL D_0=3.0 R_0=1.5} MEAN LABEL=d
PRINT ARG=d.* FILE=colv
```

5.5.10 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 defin 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
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
```

5.5.11 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.

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 amongst 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 the label of the components customizable. 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 refererred 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. 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 parallelize
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...

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
MOLECULES MOL1=1,2 MOL2=3,4 MOL3=5,6 MOL4=7,8 MEAN LABEL=mm
PRINT ARG=mm.mean FILE=colvar
```

5.5.12 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 centre 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 parallelize
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...

Examples

5.5.13 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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 paramters that describe the arrangement of chlorine ions around the sodium atoms in NaCl. 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 paramter is calculated and output to a file called colvar

```
BEGIN_PLUMED_FILE
Q3 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN LABEL=q3
PRINT ARG=q3.mean FILE=colvar
```

5.5.14 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 4th order Steinhardt parameters.

The 4th 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 4th 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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 paramters that describe the arrangement of chlorine ions around the sodium atoms in NaCl. 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 paramter is calculated and output to a file called colvar

```
BEGIN_PLUMED_FILE
Q4 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN LABEL=q4
PRINT ARG=q4.mean FILE=colvar
```

5.5.15 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 6th order Steinhardt parameters.

The 6th 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 6th 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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 paramters that describe the arrangement of chlorine ions around the sodium atoms in NaCl. 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 paramter is calculated and output to a file called colvar

```
BEGIN_PLUMED_FILE
Q6 SPECIESA=1-64 SPECIESB=65-128 D_0=1.3 R_0=0.2 MEAN LABEL=q6
PRINT ARG=q6.mean FILE=colvar
```

5.5.16 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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}
```

5.5.17 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 first coordination shell around any atom, i is tetrahedrally ordered 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 fo 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

Examples

The following command calculates the average value of the tetrahedrality parameter for a set of 64 atoms all of the same type and outputs this quantity to a file called colvar.

```
BEGIN_PLUMED_FILE
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 tetrahedrality 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
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
```

5.5.18 TORSIONS

	This is part of the multicolvar module

Calculate whether or not a set of torsional angles are within a particular range.

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 amongst 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 the label of the components customizable. 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.

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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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...

Examples

The following provides an example of the input for the torsions command

```
BEGIN_PLUMED_FILE
TORSIONS ...
ATOMS1=168,170,172,188
ATOMS2=170,172,188,190
ATOMS3=188,190,192,230
LABEL=ab
... TORSIONS
PRINT ARG=ab.* FILE=colvar STRIDE=10
```

Writing out the atoms involved in all the torsions 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
MOLINFO MOLTYPE=protein STRUCTURE=myprotein.pdb
TORSIONS ...
ATOMS1=@phi-3
ATOMS2=@psi-3
ATOMS3=@phi-4
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 4th residue of the protein.

5.5.19 XANGLES

This is part of the multicolvar module

Calculate the angles between the vector connecting two atoms and the x axis.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
XANGLES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.20 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
XDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
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
XDISTANCES ATOMS1=3,5 ATOMS2=1,2 LABEL=d1 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lt0.1
```

(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
XDISTANCES GROUP=1-3 AVERAGE LABEL=d1
PRINT ARG=d1.average
```

(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
XDISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.gt0.1
```

(See also PRINT switchingfunction)

5.5.21 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
XYDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.22 XYTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the x axis from the positive y direction.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 torsions 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
XYTORSIONS ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.23 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
XZDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.24 XZTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the x axis from the positive z direction.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 torsions 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize

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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
XZTORSIONS ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.25 YANGLES

	This is part of the multicolvar module

Calculate the angles between the vector connecting two atoms and the y axis.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
YANGLES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.26 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
YDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
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
YDISTANCES ATOMS1=3,5 ATOMS2=1,2 LABEL=d1 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lt0.1
```

(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
YDISTANCES GROUP=1-3 AVERAGE LABEL=d1
PRINT ARG=d1.average
```

(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
YDISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.gt0.1
```

(See also PRINT switchingfunction)

5.5.27 YXTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the y axis from the positive x direction.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 torsions 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
YXTORSIONS ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.28 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
YZDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.29 YZTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the y axis from the positive z direction.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 torsions 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
YZTORSIONS ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.30 ZANGLES

This is part of the multicolvar module

Calculate the angles between the vector connecting two atoms and the z axis.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
ZANGLES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.31 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
ZDISTANCES ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
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
ZDISTANCES ATOMS1=3,5 ATOMS2=1,2 LABEL=d1 LESS_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.lt0.1
```

(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
ZDISTANCES GROUP=1-3 AVERAGE LABEL=d1
PRINT ARG=d1.average
```

(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
ZDISTANCES GROUPA=1 GROUPB=2,3 MORE_THAN={RATIONAL R_0=0.1}
PRINT ARG=d1.gt0.1
```

(See also PRINT switchingfunction)

5.5.32 ZXTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the z axis from the positive x direction.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 torsions 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
ZXTORSIONS ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.33 ZYTORSIONS

This is part of the multicolvar module

Calculate the torsional angle around the z axis from the positive y direction.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 torsions 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 incides 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 conjuction with GROUPB.
GROUPB	Calculate the distances between all the atoms in GROUPA and all the atoms in GROUPB. This must be used in conjuction 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 parallelize

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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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.

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
ZYTORSIONS ATOMS1=3,5 ATOMS2=1,2 MIN={BETA=0.1} LABEL=d1
PRINT ARG=d1.min
```

(See also PRINT).

5.5.34 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 practise 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 histograмbead 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 ammount 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 parallelize
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.

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 defeind above. The following provides information on the histograмbead that are available. When this keyword is present you no longer need the LOWER, U↩ PPER and SMEAR keywords.

Examples

The example shown below calculates the mean for those distances that are between 0 and 3 nm in length

```
BEGIN_PLUMED_FILE
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
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}
```

5.5.35 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 practise 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 parallelize
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.
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 swiching 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.

Examples

The example shown below calculates the mean for those distances that less than 1.5 nm in length

```
BEGIN_PLUMED_FILE
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
```

5.5.36 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 practise 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 parallelize
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.

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 swiching 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.

Examples

The example shown below calculates the mean for those distances that greater than 1.5 nm in length

```
BEGIN_PLUMED_FILE
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
1: 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}
```

5.5.37 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 histogrambead 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

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 amongst 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 the label of the components customizable. 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 refererred 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 refererred 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
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 parallelize
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_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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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, SU↩ M2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

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
COM ATOMS=1-100 LABEL=c1
COORDINATIONNUMBER SPECIES=1-100 R_0=1.0 LABEL=c
AROUND DATA=c ORIGIN=c1 XLOWER=-2.0 XUPPER=2.0 SIGMA=0.1 MEAN LABEL=s
```

5.5.38 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 refererence 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 refererred 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 parallelize
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, 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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...
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

Examples

The following commands tell plumed to calculate the number of atoms in an ion chanel in a protein. The extent of the chanel is calculated from the positions of atoms 1, 4, 5 and 11. The final value will be labeled cav.

```
BEGIN_PLUMED_FILE
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
d1: COORDINATIONNUMBER SPECIES=20-500
CAVITY DATA=d1 ATOMS=1,4,5,11 SIGMA=0.1 MEAN MORE_THAN={RATIONAL R_0=4} LABEL=cav
```

5.5.39 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

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 amongst 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 the label of the components customizable. 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 refererred 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 refererred 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
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 cyclinder 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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...
SIGMA	the width of the function to be used for kernel density estimation

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
c1: COORDINATIONNUMBER SPECIES=1-100 SWITCH={RATIONAL R_0=0.1}
d2: INCYLINDER ATOM=101 DATA=d1 DIRECTION=Z RADIUS={TANH R_0=1.5} SIGMA=0.1 LOWER=-0.1 UPPER=0.1 MEAN
PRINT ARG=d2.* FILE=colvar
```

5.5.40 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 refererred 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 esimtation
CONTOUR	a switching funciton 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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...

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.

```
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,rr.* FILE=colvar
```

5.5.41 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

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 amongst 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 the label of the components customizable. 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 refererred 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 refererred 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 sphereical 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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...

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
c1: COORDINATIONNUMBER SPECIES=1-100 SWITCH={RATIONAL R_0=0.1}
d2: INSPHERE ATOM=101 DATA=d1 RADIUS={TANH R_0=1.5} SIGMA=0.1 LOWER=-0.1 UPPER=0.1 MEAN
PRINT ARG=d2.* FILE=colvar
```

5.5.42 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 refererence 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 refererred 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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...
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

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
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
d1: COORDINATIONNUMBER SPECIES=20-500
CAVITY DATA=d1 ATOMS=1,4,5,11 SIGMA=0.1 MEAN MORE_THAN={RATIONAL R_0=4} LABEL=cav
```

5.5.43 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 [28].

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 graident. 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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

Examples

The input below calculates the gradient of the density of atoms in the manner described in [28] in order to detect whether or not atoms are distributed uniformly along the x-axis of the simulation cell.

```
BEGIN_PLUMED_FILE
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
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
```

5.5.44 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 torsions 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.

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 M↩ OLA one atom from MOLB 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 M↩ OLA one atom from MOLB 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 parallelize
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 swiching 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.

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 centeres 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 tt_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
m1: MOLECULES MOL1=1,2 MOL2=3,4 MOL3=5,6 MOL4=7,8
tt_p: INTERMOLECULARTORSIONS MOLS=m1 SWITCH={RATIONAL R_0=0.25 D_0=2.0 D_MAX=3.0}
htt_p: HISTOGRAM DATA=tt_p GRID_MIN=-pi GRID_MAX=pi BANDWIDTH=0.1 GRID_BIN=200 STRIDE=1
DUMPGRID GRID=htt_p FILE=myhist.out
```

5.5.45 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 4th 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 [22] 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-centred 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.

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 amongst 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 the label of the components customizable. 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 refererred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be refererred 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 refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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_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...
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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...

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
COORDINATIONNUMBER SPECIES=1-64 D_0=1.3 R_0=0.2 LABEL=d1
LOCAL_AVERAGE ARG=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 outputeed to a file. This calculates the quantities that were used in the paper by Lechner and Dellago [22]

```
BEGIN_PLUMED_FILE
Q4 SPECIES=1-64 SWITCH={RATIONAL D_0=1.3 R_0=0.2} LABEL=q4
LOCAL_AVERAGE ARG=q4 SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LABEL=la
PRINT ARG=la.* FILE=colvar
```

5.5.46 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 parameteters 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 asterix 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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
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
```

5.5.47 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 parameteters 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 4th order Steinhardt vectors calculated for atom i and atom j respectively and the asterix 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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
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
```

5.5.48 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 parameteters 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 6th order Steinhardt vectors calculated for atom i and atom j respectively and the asterix 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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
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=q4a,q4b SWITCH={RATIONAL D_0=1.3 R_0=0.2} MEAN LOWMEM LABEL=w4
PRINT ARG=w6.* FILE=colvar
```

5.5.49 MCOLV_COMBINE

This is part of the multicolvar module

Calculate linear combinations of multiple multicolvars

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 coeficients 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 parallelize
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...
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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 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.

Examples

5.5.50 MCOLV_PRODUCT

This is part of the multicolvar module

Calculate a product of multiple multicolvars

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 parallelize
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...
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

Examples

5.5.51 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.

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 parallelize
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 swiching 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.

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
DENSITY SPECIES=1-64 LABEL=d1
NLINKS ARG=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 neighbouring atoms in a system containg 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
Q6 SPECIES=1-64 SWITCH={RATIONAL D_0=1.3 R_0=0.2} LABEL=q6
NLINKS ARG=q6 SWITCH={RATIONAL D_0=1.3 R_0=0.2} LABEL=dd
PRINT ARG=dd FILE=colvar
```

5.5.52 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 normals of pairs of input vectors:

$$s = \frac{3\cos\theta - 1}{2}$$

This average of this quantity over all the vectors in the first coordination sphere around each of the PLANES specified is then calculated.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
```

5.5.53 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 discussed in [23]

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 moleclues 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities calculated by this action
mean	**MEAN**	the mean value. The output component can be refererred 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 cooordination numbers specifies how many of the atoms specifies using SPEC↩ IESB 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 parallelize
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 swiching 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_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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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
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
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
```

5.5.54 MTRANSFORM_BETWEEN

This is part of the multicolvar module

This action can be useed to transform the colvar values calculated by a multicolvar using a histograbead

In this action each colvar, s_i, calculated by multicolvar is transformed by a histograbead 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 histograbead. 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 histograbead. If one thus calculates a MEAN for thia action one computes:

$$\mu = \frac{\sum_{i=1}^{N} f_i}{N}$$

In other words, you are calculating the mean for the transformed colvar.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 ammount 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 parallelize
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.
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 defeind above. The following provides information on the histograbead that are available. When this keyword is present you no longer need the LOWER, UPPER and SMEAR keywords.

Examples

The following input gives an example of how a MTRANSFORM_BETWEEN action can be used to duplicate functionality that is elsehwere in PLUMED.

```
BEGIN_PLUMED_FILE
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
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

5.5.55 MTRANSFORM_LESS

This is part of the multicolvar module

This action can be useed to transform the colvar values calculated by a multicolvar using a switchingfunction

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 thia action one computes:

$$\mu = \frac{\sum_{i=1}^{N}(s_i)}{N}$$

In other words, you are calculating the mean for the transformed colvar.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 parallelize
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.
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 swiching 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.

Examples

The following input gives an example of how a MTRANSFORM_LESS action can be used to duplicate functionality that is elsehwere in PLUMED.

```
BEGIN_PLUMED_FILE
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
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

5.5.56 MTRANSFORM_MORE

This is part of the multicolvar module

This action can be useed to transform the colvar values calculated by a multicolvar using one minus a switchingfunction

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 histogrambead. 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.

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 amongst 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 the label of the components customizable. 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 refererred 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 lowest of the quantitities calculated by this action
lowest	**LOWEST**	the lowest of the quantitities 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 refererred 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 parallelize
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.
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 swiching 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.

Examples

The following input gives an example of how a MTRANSFORM_MORE action can be used to duplicate functionality that is elsehwere in PLUMED.

```
BEGIN_PLUMED_FILE
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
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

5.5.57 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.

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 quanties 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 refererred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be refererred 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 lowest of the quantitities 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 quantitities 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 refererred 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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

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
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
```

5.5.58 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.

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 quanties 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 refererred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be refererred 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 lowest of the quantitities 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 quantitities 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 refererred 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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

Examples

The following set of commands can be used to stop a cluster composed of 20 atoms subliming. The position of the centre 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 centre of mass of the cluster.

```
BEGIN_PLUMED_FILE
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
```

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 defininining 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 centre's 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.
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 connnectivity 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 analysed using a number of other algorithms as is detailed in [29].

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.

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 parallelize
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 swiching 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. 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...

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 centeres 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
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}
rr: ROWSUMS MATRIX=mat MORE_THAN={RATIONAL D_0=1.0 R_0=0.1}
PRINT ARG=rr.* FILE=colvar
```

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 analysed using a number of other algorithms as is detailed in [29].

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.

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 parallelize
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 swiching 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. You can use multiple instances of this keyword i.e. SWITCH1, SWITCH2, SWITCH3...

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
aa: 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
```

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 analysed using a number of other algorithms as is detailed in [29].

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 asymetric. 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.

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 amongst 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 the label of the components customizable. 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 specifiedas 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 parallelize
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, ASWITCH2, 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, SUM2, SUM3... The corresponding values are then referenced using *label*.sum-1, *label*.sum-2, *label*.sum-3...

Examples

The following input can be used to analyse 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 its hydrogens to the neighbouring 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
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.x
```

5.6.4 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 moleclues 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.

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 parallelize
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 swiching 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. 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, K↩ ERNEL2, KERNEL3...

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 connectes molecules is output to a colvar file

```
BEGIN_PLUMED_FILE
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=smacm
... SMAC_MATRIX

dfs1: DFSCLUSTERING MATRIX=smacm
cc2: CLUSTER_NATOMS CLUSTERS=dfs1 CLUSTER=1
PRINT ARG=smac.*,cc1.*,cc2 FILE=colvar
```

5.6.5 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

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 parallelize
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 swiching 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. 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, RA↩ DIUS3...
CYLINDER_SWITCH	a switching function on (r_ij . r_ik - 1)/r_ij You can use multiple instances of this keyword i.e. CYLINDER_SWITCH1, CYLINDER_SWITCH2, CYLINDE↩ R_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...

Examples

5.6.6 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 analysing 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 sucessfully in the forward flux sampling simulations described in this paper [30]

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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

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
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
```

5.6.7 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 calcualtes the adjacency matrix vessel we would like to analyse

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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
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
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
```

5.6.8 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 [29] these matrices provide a representation of a graph and can thus can be analysed 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.

Compulsory keywords

MATRIX	the action that calcualtes the adjacency matrix vessel we would like to analyse
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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

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 [29].

```
BEGIN_PLUMED_FILE
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
```

5.6.9 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.

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 amongst 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 the label of the components customizable. 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 lowest of the quantitities 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 quantitities 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 refererred 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 calcualtes the adjacency matrix vessel we would like to analyse

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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.

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
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
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
```

5.6.10 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.

Description of components

By default this Action calculates the following quantities. These quanties 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 labelled *label*.coord-1, the second smallest will be labelleled *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 refererred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be refererred 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 lowest of the quantitities 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 quantitities 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 refererred 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 calcualtes the adjacency matrix vessel we would like to analyse

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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

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 ?? so two atoms are adjacent if they are within a cutoff:

```
BEGIN_PLUMED_FILE
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 foa 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
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
```

5.6.11 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 analysing 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 paris 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 is not differentiable. As such it cannot be used as a collective variable in a biased simulation.

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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

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
# 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
```

5.6.12 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 [29].

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 amongst 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 the label of the components customizable. 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 parallelize
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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...
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...

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 componets in this graph that contain more than 27 atoms is then computed. As discussed in [29] this input was used to analyse the formation of a polycrystal of GeTe from amorphous GeTe.

```
BEGIN_PLUMED_FILE
q6: Q6 SPECIES=1-32768 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
```

5.6.13 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 analysing 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 is not differentiable. As such it cannot be used as a collective variable in a biased simulation.

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 parallelize
LOWMEM	(default=off) lower the memory requirements

TIMINGS	(default=off) output information on the timings of the various parts of the calculation

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
# 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
```

5.6.14 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 heterogenous nucleation of a solute from a solvent) you might be interested in how many atoms are present in the largest cluster [29].

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 amongst 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 the label of the components customizable. 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 refererred to elsewhere in the input file by using the label.vmean
vsum	**VSUM**	the norm of sum of vectors. The output component can be refererred 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 lowest of the quantitities 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 quantitities 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 refererred 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 parallelize
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...
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 a discretized histogram of the distribution of values. This shortcut allows you to calculates NBIN quantites 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.
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

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 [29].

```
BEGIN_PLUMED_FILE
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
```

5.6.15 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 connnectivity of the nodes in the graph in dot format.

Compulsory keywords

MATRIX	the action that calcualtes the adjacency matrix vessel we would like to analyse
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

Examples

5.6.16 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.

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 reconstuct clusters

Options

MAKE_WHOLE	(default=off) reconstruct the clusters and remove all periodic boundary conditions.

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
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
```

Chapter 6

Analysis

PLUMED can be used to analyse trajectories either on the fly during an MD run or via postprocessing a trajectory using driver. 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. This command can only be used when running an molecular dynamics trajectory - it cannot be used when analysing a trajectory using driver. As detailed in the documentation for COMMITTOR this command tells PLUMED (and the underlying MD code) to stop the calculation one some criteria is satisified.

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 enseble 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 behavoir 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 cannonical ensemble at the appropriate temperature.

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	Calculate weights for ensemble averages allow for the computing of ensemble averages at temperatures lower/higher than that used in your original simulation.

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 analysed 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_SURFACE	Find an isocontour by searching along either the x, y or z direction.
FIND_CONTOUR	Find an isocontour in a smooth function.
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 5th 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
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 analyse 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
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.

6.1 Dimensionality Reduction

The remainder of the analysis tools within PLUMED allow one to do some form of dimensionality reduction as detailed below.

CLASSICAL_MDS	Create a low-dimensional projection of a trajectory using the classical multidimension-alscaling algorithm.
PCA	Perform principal component analysis (PCA) using either the positions of the atoms a large number of collective variables as input.

As with the grids described previously the STRIDE keyword tells PLUMED how frequently to collect data from the trajectory. The RUN keyword then tells PLUMED how frequently to do the dimensionality reduction. As described above if RUN is not present and you are analysing trajectories using driver all the data in the traejctory (with the expection of the first frame) will be analysed.

6.2 COMMITTOR

	This is part of the analysis module

Does a committor analysis.

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 analysed

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 proceding 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 componets 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

Examples

The following input monitors two torsional angles during a simulation, defines two basins (A and B) as a function of the two torsions 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
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
```

6.3 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 ouput 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.

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

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
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
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 lentght 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
# 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
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
COM ATOMS=11-20 LABEL=c1
DUMPATOMS STRIDE=10 FILE=file.xtc ATOMS=1-10,c1 PRECISION=7
```

6.4 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.

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 proceding 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 componets 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. A↩ RG1, ARG2, ARG3...
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

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
DISTANCE ATOM=1,2 LABEL=distance
DISTANCE ATOM=1,2 LABEL=distanceN NUMERICAL_DERIVATIVES
DUMPDERIVATIVES ARG=distance,distanceN STRIDE=1 FILE=deriv
```

(See also DISTANCE)

6.5 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.

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 proceding 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 componets 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

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
DISTANCE ATOM=1,2 LABEL=distance
DUMPFORCES ARG=distance STRIDE=1 FILE=forces
```

6.6 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.

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. .gro extension is automatically detected

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
c1: COM ATOMS=1-10
c2: COM ATOMS=11-20
PRINT ARG=c1,c2 FILE=colvar 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
solute_ions: GROUP ATOMS=1-121,200-2012
DUMPATOMS FILE=traj.gro ATOMS=solute_ions STRIDE=100
DUMPMASSCHARGE FILE=mcfile ATOMS=solute_ions
```

Notice however that if you want to process the charges with the driver (e.g. reading traj.gro) you have to fix atom numbers first, e.g. with the script

```
awk 'BEGIN{c=0}{
  if(match($0,"#")) print ; else {print c,$2,$3; c++}
}' < mc > newmc
}'
```

then

```
plumed driver --mc newmc --plumed plumed.dat --ixyz traj.gro
```

6.7 DUMPMULTICOLVAR

This is part of the multicolvar module

Dump atom positions and multicolvar on a file.

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

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
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)

6.8 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).

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 proceding 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 componets 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. A↩ RG1, ARG2, ARG3...
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

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
x1: CENTER ATOMS=1-10
x2: CENTER ATOMS=11-20
d: DISTANCE ATOMS=x1,x2
DUMPPROJECTIONS ARG=d FILE=proj STRIDE=20
```

6.9 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.

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 proceding 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 componets 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

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
# 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.

6.9.1 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 arrise 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.

Compulsory keywords

STRIDE	the frequency with which all the open files should be flushed

Examples

A command like this in the input will instruct plumed to flush all the output files every 100 steps

```
BEGIN_PLUMED_FILE
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).

6.10 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.

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 proceding 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 componets 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

Examples

The following input instructs plumed dump all the snapshots where an atom is in touch with the solute.

```
BEGIN_PLUMED_FILE
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
```

6.11 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:

$$\langle P(s',t)\rangle = \frac{\sum_{t'}^{t} \delta(s(x)-s') \exp\left(+\frac{V(x,t')}{k_B T}\right)}{\sum_{t}^{\prime t} \exp\left(+\frac{V(x,t')}{k_B T}\right)}$$

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.

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

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
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
```

6.12 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 [31] when constructing a histogram of the configurations visited during a metadynamics simulation.

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

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
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 REWEIGHTING_NGRID=1

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
```

6.13 REWEIGHT_TEMP

This is part of the bias module

Calculate weights for ensemble averages allow for the computing of ensemble averages at temperatures lower/higher than that used in your original simulation.

We can use our knowledge of the Boltzmann distribution in the cannonical ensemble to reweight the data contained in trajectories. Using this procedure we can take trajectory at temperature T_1 and use it to extract probabilities at a different temperature, T_2, using:

$$P(s',t) = \frac{\sum_{t'}^{t} \delta(s(x) - s') \exp\left(+(\left[\frac{1}{T_1} - \frac{1}{T_2}\right] \frac{U(x,t')}{k_B}\right)}{\sum_{t'}^{t} \exp\left(+\left[\frac{1}{T_1} - \frac{1}{T_2}\right] \frac{U(x,t')}{k_B}\right)}$$

The weights calculated by this action are equal to $\exp\left(+(\left[\frac{1}{T_1} - \frac{1}{T_2}\right] \frac{U(x,t')}{k_B}\right)$ and take the effect the bias has on the system into account. 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.

Compulsory keywords

REWEIGHT_TEMP	reweight data from a trajectory at one temperature and output the probability distribution at a second temperature. This is not possible during postprocessing.

Options

TEMP	the system temperature. This is not required if your MD code passes this quantity to PLUMED

Examples

The following input can be used to postprocess a molecular dynamics trajectory calculated at a temperature of 500 K. The HISTOGRAM as a function of the distance between atoms 1 and 2 that would have been obtained if the simulation had been run at the lower temperature of 300 K is estimated using the data from the higher temperature trajectory and output to a file.

```
BEGIN_PLUMED_FILE
x: DISTANCE ATOMS=1,2
aa: REWEIGHT_TEMP TEMP=500 REWEIGHT_TEMP=300
hB: HISTOGRAM ARG=x GRID_MIN=0.0 GRID_MAX=3.0 GRID_BIN=100 BANDWIDTH=0.1 LOGWEIGHTS=aa
DUMPGRID GRID=hB FILE=histoB
```

6.14 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 evalulated 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 $a \leq s \leq b$ the ensemble average is evaluated using:

$$\langle s \rangle = a + \frac{b-a}{2\pi} \arctan \left[\frac{\sum_{t'=0}^{t} w(t') \sin\left(\frac{2\pi[s(t')-a]}{b-a}\right)}{\sum_{t'=0}^{t} w(t') \cos\left(\frac{2\pi[s(t')-a]}{b-a}\right)} \right]$$

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 proceding 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 componets 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...

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 formulae above can thus all be set equal to one.

```
BEGIN_PLUMED_FILE
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 aquired 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
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 canononical ensemble at a temperature of 300 K.

```
BEGIN_PLUMED_FILE
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
```

6.15 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 bandwith 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 evalulated 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 Belfast tutorial: Analyzing CVs and Lugano tutorial: Analyzing CVs.

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 caclulate 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 unormalized estimate of the histogram 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.

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 esimtation
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 parallelize
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 proceding 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 componets 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 dimsnional 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

Examples

The following input monitors two torsional angles during a simulation and outputs a continuos histogram as a function of them at the end of the simulation.

```
BEGIN_PLUMED_FILE
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
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
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
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
  GRID_WFILE=histo
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo STRIDE=100000
```

6.16 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.

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 esimtation
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 parallelize
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 specifes 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 specifes 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 specifes 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 specifes 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 specifes 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 specifes the upper bound for the region of the z-axis that for which you are calculating the density/average

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 ouptuts 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
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 analyse 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
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.

6.17 CONVERT_TO_FES

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

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 parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
CONCENTRATION	the concentration parameter for Von Mises-Fisher distributions
COMPONENT	if your input is a vector field use this to specifiy the component of the input vector field for which you wish to use
TEMP	the temperature at which you are operating

Examples

This is a typical example showing how CONVERT_TO_FES might be used when postprocessing 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
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
```

6.18 DUMPCUBE

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.

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.

Options

FMT	the format that should be used to output real numbers
COMPONENT	if your input is a vector field use this to specifiy the component of the input vector field for which you wish to output

Examples

The input below can be used to postprocess 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
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
```

6.19 DUMPGRID

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, whenver 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 beheath 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.

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.

Options

FMT	the format that should be used to output real numbers

Examples

The following input monitors two torsional angles during a simulation and outputs a continuos histogram as a function of them at the end of the simulation.

```
BEGIN_PLUMED_FILE
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
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
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
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
  GRID_WFILE=histo
  LABEL=hh
... HISTOGRAM

DUMPGRID GRID=hh FILE=histo STRIDE=100000
```

6.20 FIND_CONTOUR_SURFACE

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 calcualted 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 wher 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 countours 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.

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 parallelize
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 specifiy the component of the input vector field for which you wish to use

Examples

The input shown below was used to analyse 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 [32] 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
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
```

6.21 FIND_CONTOUR

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 calcualted 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.

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 parallelize
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 specifiy the component of the input vector field for which you wish to use
PRECISION	The number of digits in trajectory file

Examples

The input below allows you to calculate something akin to a Willard-Chandler dividing surface [32]. 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
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 SWITCH={SMAP R_0=0.5 A=8 B=8}
center: CENTER_OF_MULTICOLVAR DATA=tfcc

MULTICOLVARDENS ...
  DATA=tfcc ORIGIN=center DIR=xyz LABEL=dens
  NBINS=80,80,80 BANDWIDTH=1.0,1.0,1.0 STRIDE=25
  LABEL=dens STRIDE=1 CLEAR=1
... MULTICOLVARDENS

FIND_CONTOUR GRID=dens CONTOUR=0.5 FILE=mycontour.dat
```

6.22 FIND_SPHERICAL_CONTOUR

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 calcualted 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 wher 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-speciified 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 countours 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.

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 parallelize
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 specifiy the component of the input vector field for which you wish to use

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 vapour. 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 teh droplet from the surrounding gas. The value of the phase field on this isocontour is equal to 0.75.

```
BEGIN_PLUMED_FILE
# 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
# 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
FIND_SPHERICAL_CONTOUR GRID=dens CONTOUR=0.85 INNER_RADIUS=10.0 OUTER_RADIUS=40.0 FILE=mysurface.xyz UNITS=A F
```

6.23 FOURIER_TRANSFORM

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 $b = 1$.

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 parallelize
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 specifiy 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).

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
FOURIER_TRANSFORM STRIDE=1 GRID=density FT_TYPE=complex FOURIER_PARAMETERS=0,-1 FILE=fourier.dat
```

6.24 GRID_TO_XYZ

Output the function on the grid to an xyz file

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.
UNITS	(default=PLUMED) the units in which to print out the coordinates. PLUMED means internal PLU↩ MED units

Options

COMPONENT	if your input is a vector field use this to specifiy the component of the input vector field for which you wish to output
PRECISION	The number of digits in trajectory file

Examples

6.25 INTEGRATE_GRID

Calculate the total integral of the function on the input grid

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 parallelize
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 specifiy the component of the input vector field for which you wish to use

Examples

6.26 INTERPOLATE_GRID

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.

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 parallelize
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 specifiy 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)

Examples

The input below can be used to postprocess 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
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
```

6.27 CLASSICAL_MDS

	This is part of the analysis module

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 analyse simulations can be found in the tutorial Belfast tutorial: Adaptive variables II and in the following `short video.`

The atoms involved can be specified using

ATOMS	the atoms whose positions we are tracking for the purpose of analysing the data. 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
METRIC	(default=EUCLIDEAN) how are we measuring the distances between configurations
RUN	(default=0) the frequency with which to run the analysis algorithm. The default value of zero assumes you want to analyse the whole trajectory
LANDMARKS	(default=ALL) only use a subset of the data that was collected. For more information on the landmark selection algorithms that are available in plumed see landmarkselection.
NLOW_DIM	number of low-dimensional coordinates required
OUTPUT_FILE	file on which to output the final embedding coordinates
EMBEDDING_OFILE	(default=dont output) file on which to output the embedding in plumed input format

Options

SERIAL	(default=off) do the calculation in serial. Do not parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation

WRITE_CHECKPOINT	(default=off) write out a checkpoint so that the analysis can be restarted in a later run
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 proceding 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 componets 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 PL↩ UMED with the appropriate flag. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
FMT	the format that should be used in analysis output files
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

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
CLASSICAL_MDS ...
  ATOMS=1-256
  METRIC=OPTIMAL-FAST
  NLOW_DIM=2
  OUTPUT_FILE=rmsd-embed
... CLASSICAL_MDS
```

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.

6.27.1 Method of optimisation

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:

$$\mathbf{J} = \mathbf{I} - \frac{1}{M}\mathbf{1}\mathbf{1^T}$$

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:

$$
\begin{aligned}
-\frac{1}{2}\mathbf{J}\mathbf{D}^2(\mathbf{X})\mathbf{J} &= -\frac{1}{2}\mathbf{J}(\mathbf{c1^T} + \mathbf{1c^T} - 2\mathbf{XX^T})\mathbf{J} \\
&= -\frac{1}{2}\mathbf{Jc1^T J} - \frac{1}{2}\mathbf{J1c^T J} + \frac{1}{2}\mathbf{J}(2\mathbf{XX^T})\mathbf{J} \\
&= \mathbf{JXX^T J} = \mathbf{XX^T}
\end{aligned}
$$

The fist two terms in this expression disappear because $\mathbf{1^T J} = \mathbf{J1} = \mathbf{0}$, 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 $\mathbf{XX^T}$ adds up to zero, which is what allows us to write $\mathbf{JXX^T J} = \mathbf{XX^T}$.

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, $\mathbf{XX^T}$, 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 minimise the stress. If you use an interative 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.

6.28 PCA

This is part of the analysis module

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 analysed 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.

The atoms involved can be specified using

ATOMS	the atoms whose positions we are tracking for the purpose of analysing the data. 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
METRIC	(default=EUCLIDEAN) how are we measuring the distances between configurations

RUN	(default=0) the frequency with which to run the analysis algorithm. The default value of zero assumes you want to analyse the whole trajectory
NLOW_DIM	number of PCA coordinates required
OFILE	the file on which to output the eigenvectors

Options

SERIAL	(default=off) do the calculation in serial. Do not parallelize
LOWMEM	(default=off) lower the memory requirements
TIMINGS	(default=off) output information on the timings of the various parts of the calculation
WRITE_CHECKPOINT	(default=off) write out a checkpoint so that the analysis can be restarted in a later run
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 proceding 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 componets 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 PL↩ UMED with the appropriate flag. You can use multiple instances of this keyword i.e. ARG1, ARG2, ARG3...
FMT	the format that should be used in analysis output files
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

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
PCA METRIC=OPTIMAL ATOMS=1-22 STRIDE=1 NLOW_DIM=2 OFILE=pca-comp.pdb
```

The following input instructs PLUMED to perform a principal component analysis in which the covariance matrix is calculated from chnages in the six distances seen in the previous lines. Notice that here the TYPE=EUCLID↩ EAN keyword is used to indicate that no alighment 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 REWE↩ IGHT_BIAS keyword in this input tells PLUMED that rather that ascribing a weight of one to each of the frames when calculating averages and covariances 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
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

PCA ARG=d1,d2,d3,d4,d5,d6 METRIC=EUCLIDEAN STRIDE=5 RUN=1000 NLOW_DIM=2 REWEIGHT_BIAS OFILE=pca-comp.pdb
```

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 $f_i\left(x\right)$ satisfying the maximum entropy principle as proposed in Ref. [33] .
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:

DRR	(from Extended-System Adaptive Biasing Force module) Used to performed extended-system adaptive biasing force(eABF) [34] methodon one or more collective variables. This method is alsocalled dynamic reference restraining(DRR) [35] .
EDS	(from Experiment Directed Simulation module) Add a linear bias on a set of observables.
METAINFERENCE	(from PLUMED-ISDB module) Calculates the Metainference energy for a set of experimental data.
RESCALE	(from PLUMED-ISDB module) Rescales the value of an another action, being a Collective Variable or a Bias.
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 [36] [37] [38]. 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.

Description of components

By default this Action calculates the following quantities. These quanties 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 will be refereceable 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 proceding 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 componets 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)

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
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
```

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.

Description of components

By default this Action calculates the following quantities. These quanties 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 will be refereceable 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 proceding 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 componets 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...

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
DISTANCE ATOMS=3,5 LABEL=d1
DISTANCE ATOMS=3,6 LABEL=d2
BIASVALUE ARG=d1,d2 LABEL=b
PRINT ARG=d1,d2,b.d1,b.d2
```

Another thing one can do is asking one system to follow a circle in sin/cos according a time dependence

```
BEGIN_PLUMED_FILE
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.bias.vv1 STRIDE=1 FILE=colvar FMT=%8.4f
```

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 i-th 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 [39] . 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 [40] [41] . 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 postion 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.

Description of components

By default this Action calculates the following quantities. These quanties 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 will be refereceable 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 will be refereceable 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 proceding 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 componets 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)

Examples

The following input tells plumed to perform a metadynamics with an extended Lagrangian on two torsional angles.

```
BEGIN_PLUMED_FILE
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
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
```

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

Description of components

By default this Action calculates the following quantities. These quanties 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.

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 proceding 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 componets 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...

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
DISTANCE ATOMS=3,5 LABEL=d1
EXTERNAL ARG=d1 FILE=bias.dat LABEL=external
```

The header in the file bias.dat should read:

```
#! FIELDS d1 external.bias der_d1
#! SET min_d1 0.0
#! SET max_d1 1.0
#! SET nbins_d1 100
#! SET periodic_d1 false
```

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
TORSION ATOMS=4,5,6,7 LABEL=t1
TORSION ATOMS=6,7,8,9 LABEL=t2
EXTERNAL ARG=t1,t2 FILE=bias.dat LABEL=ext
```

The header in the file bias.dat for this calculation would read:

```
#! FIELDS t1 t2 ext.bias der_t1 der_t2
#! SET min_t1 -pi
#! SET max_t1 +pi
#! SET nbins_t1 100
#! SET periodic_t1 true
#! SET min_t2 -pi
#! SET max_t2 +pi
#! SET nbins_t2 100
#! SET periodic_t2 true
```

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 untill you get to the 100th 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.

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.

Description of components

By default this Action calculates the following quantities. These quanties 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 proceding 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 componets 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...

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
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).

7.6 MAXENT

This is part of the bias module

Add a linear biasing potential on one or more variables $f_i(x)$ satisfying the maximum entropy principle as proposed in Ref. [33] .

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 choosen 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.

Description of components

By default this Action calculates the following quantities. These quanties 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. EQAUL to restrain the variable at a given equilibrium valueINEQUAL< to restrain the variable to be smaller than a given valueINEQUAL> 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 comunicated 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 proceding 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 componets 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 priorLAPL↩ ACE: use a Laplace prior
TSTART	time in ps 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 erros 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 correspondece 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 (usefulf 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)

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
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
PRINT ARG=restraint.bias
... MAXENT
```

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
DISTANCE ATOMS=7,15 LABEL=d
MAXENT ARG=d TYPE=INEQUAL> AT=0.02 KAPPA=35000.0 TAU= LABEL=restraint
PRINT ARG=restraint.bias
```

(See also DISTANCE and PRINT).

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 [42].

$$V(\vec{s},t) = \sum_{k\tau<t} W(k\tau) \exp\left(-\sum_{i=1}^{d} \frac{(s_i - s_i^{(0)}(k\tau))^2}{2\sigma_i^2}\right).$$

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 Gaussians. To avoid this issue you can store the bias on a grid. This approach is similar to that proposed in [43] but has the advantage that the grid spacing is independent on the Gaussian width. Notice that 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.

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.

Another option that is available in plumed is well-tempered metadynamics [44]. In this varient of metadynamics the heights of the Gaussian hills are rescaled at each step so the bias is now given by:

$$V(s,t) = \sum_{t'=0,\tau_G,2\tau_G,\ldots}^{t'<t} W e^{-V(s(q(t'),t')/\Delta T} \exp\left(-\sum_{i=1}^{d} \frac{(s_i(q) - s_i(q(t'))^2}{2\sigma_i^2}\right),$$

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 [45] 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 (ADAP↩ TIVE=GEOM) or a time (ADAPTIVE=DIFF). Note that a specific integration technique for the deposited gaussians 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 [46]. If, for example, metadynamics is performed on a CV s and one is interested only to the free energy for s $>$ sw, the history dependent potential is still updated according to the above equations but the metadynamics force is set to zero for s $<$ sw. Notice that Gaussians are added also if s $<$ sw, as the tails of these Gaussians influence VG in the relevant region s $>$ sw. In this way, the force on the system in the region s $>$ sw comes from both metadynamics and the force field, in the region s $<$ sw 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 sw in a region where the free energy derivative is not large;
- If in the region outside the limit sw the system has a free energy minimum, the INTERVAL keyword should be used together with a UPPER_WALLS or LOWER_WALLS at sw.

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 [47] 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 using Eq. 12 in Eq. 3 in [3] and 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-ct) which can be used to obtain a reweighted histogram. The calculation of c(t) is enabled by using the keyword REWEIGHTING_NGRID where the grid used for the calculation is specified. This grid should have a size that is equal or larger than the grid given in GRID_BIN./ By default c(t) is updated every 50 Gaussian hills but this can be changed by using the REWEIGHTING_NHILLS keyword. This option can only be employed together with Well-Tempered Metadynamics and requires that a grid is used.

Additional material and examples can be also found in the tutorials:

- Belfast tutorial: Metadynamics
- Belfast tutorial: Replica exchange I
- Belfast tutorial: Replica exchange II and Multiple walkers

Notice that at variance with PLUMED 1.3 it is now straightforward to apply concurrent metadynamics as done e.g. in Ref. [48] . This indeed can be obtained by using the METAD action multiple times in the same input file.

Description of components

By default this Action calculates the following quantities. These quanties 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	accumulator for work

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
rbias	**REWEIGHTING_NGRID**	the instantaneous value of the bias normalized using the $c(t)$ reweighting factor [rbias=bias-c(t)].This component can be used to obtain a reweighted histogram.
rct	**REWEIGHTING_NGRID**	the reweighting factor $c(t)$.
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)

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
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
WALKERS_MPI	(default=off) Switch on MPI version of multiple walkers - not compatible with WALKERS_$*$ options other than WALKERS_DIR
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)
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 proceding 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 componets 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 biasfactor. Please note you must also specify temp
RECT	list of bias factors for all the replicas
DAMPFACTOR	damp hills with exp(-max(V)/(kbT∗DAMPFACTOR)
TTBIASFACTOR	use transition tempered metadynamics with this biasfactor. 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 (kb∗DeltaT∗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)
REWEIGHTING_NGRID	calculate the c(t) reweighting factor and use that to obtain the normalized bias [rbias=bias-c(t)].Here you should specify the number of grid points required in each dimension.The number of grid points should be equal or larger to the number of grid points given in GRID_BIN.This method is not compatible with metadynamics not on a grid.
REWEIGHTING_NHILLS	how many Gaussian hills should be deposited between calculating the c(t) reweighting factor.The default is to do this every 50 hills.
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
ADAPTIVE	use a geometric (=GEOM) or diffusion (=DIFF) based hills width scheme. Sigma is one number that has distance units or timestep dimensions
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	monodimensional lower and upper limits, outside the limits the system will not feel the biasing force.
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
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 TRANSITIONWELLx with x=0,1,2,...,n. 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, TRANSITIONWEL↩ L3...
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

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
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 Gaussians, with diffusion scheme where you use a Gaussian that should cover the space of 20 timesteps in collective variables. Note that in this case the histogram correction is needed when summing up hills.

```
BEGIN_PLUMED_FILE
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 Gaussians, 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
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 Gaussians you might want to limit how the hills width can change. You can use SIG↩ MA_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
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 [47] 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
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 REWEIGHTING_NGRID where the grid used for the calculation is set. The number of grid points given in REWEIGHTING_NGRID should be equal or larger than the number of grid points given in GRID_BIN.

```
BEGIN_PLUMED_FILE
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
 REWEIGHTING_NGRID=150,150
 REWEIGHTING_NHILLS=20
... METAD
```

Here we have asked that the calculation is performed every 20 hills by using REWEIGHTING_NHILLS keyword. If this keyword is not given the calculation will by default be performed every 50 hills. 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-c(t)] 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 analysed on the fly as in [49]. The flag ACCEL↩ ERATION 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.

You can also provide a target distribution using the keyword TARGET [50] [51] [52] The TARGET should be a grid containing a free-energy (i.e. the -kbT∗log of the desired target distribution). Gaussians 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 [52] This choice allows to avoid exceedingly large Gaussians 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 [53] Alternatively, if you use a BIASFACTOR yout 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
DISTANCE ATOMS=3,5 LABEL=d1
METAD ...
 LABEL=t1
 ARG=d1 SIGMA=0.05 TAU=200 DAMPFACTOR=100 PACE=250
 GRID_MIN=0 GRID_MAX=2 GRID_BIN=200
 TARGET=dist.dat
... METAD

PRINT ARG=d1,t1.bias STRIDE=100 FILE=COLVAR
```

The header in the file dist.dat for this calculation would read:

```
#! FIELDS d1 t1.target der_d1
#! SET min_d1 0
#! SET max_d1 2
#! SET nbins_d1  200
#! SET periodic_d1 false
```

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
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
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.

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 [54] and Jarzynski sampling [55].

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. Inbetween 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

Description of components

By default this Action calculates the following quantities. These quanties 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 will be refereceable 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 will be refereceable 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 will be refereceable 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 STEPx with x=0,1,2,...,n. Each value given represents the MD step at which the restraint parameters take the values KAPPAx and ATx. You can use multiple instances of this keyword i.e. STEP1, STEP2, STEP3...
AT	ATx is equal to the position of the restraint at time STEPx. For intermediate times this parameter is linearly interpolated. If no ATx is specified for STEPx then the values of AT are kept constant during the interval of time between STEPx-1 and STEPx. You can use multiple instances of this keyword i.e. AT1, AT2, AT3...
KAPPA	KAPPAx is equal to the value of the force constants at time STEPx. For intermediate times this parameter is linearly interpolated. If no KAPPAx is specified for STEPx then the values of KAPPAx are kept constant during the interval of time between STEPx-1 and STEPx. 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 proceding 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 componets 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...

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
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
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
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.

7.9 PBMETAD

This is part of the bias module

Used to performed Parallel Bias MetaDynamics.

This action activate Parallel Bias MetaDynamics (PBMetaD) [56], a version of MetaDynamics [42] 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:

$$W_i(k\tau)=W_0\frac{\exp\left(-\frac{V(s_i,k\tau)}{k_BT}\right)}{\sum_{i=1}^{N}\exp\left(-\frac{V(s_i,k\tau)}{k_BT}\right)}$$

where W_0 is the initial Gaussian height.

The PBMetaD bias potential is defined by:

$$V_{PB}(\vec{s},t)=-k_BT\log\sum_{i=1}^{N}\exp\left(-\frac{V(s_i,t)}{k_BT}\right).$$

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 [44]. In this variant of PBMetaD the heights of the Gaussian hills are rescaled 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 [45] 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 (ADAP↩ TIVE=GEOM) or a time (ADAPTIVE=DIFF). Note that a specific integration technique for the deposited gaussians 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 [46]. If, for example, metadynamics is performed on a CV s and one is interested only to the free energy for s $>$ sw, the history dependent potential is still updated according to the above equations but the metadynamics force is set to zero for s $<$ sw. Notice that Gaussians are added also if s $<$ sw, as the tails of these Gaussians influence VG in the relevant region s $>$ sw. In this way, the force on the system in the region s $>$ sw comes from both metadynamics and the force field, in the region s $<$ sw 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 sw in a region where the free energy derivative is not large;
- If in the region outside the limit sw the system has a free energy minimum, the INTERVAL keyword should be used together with a UPPER_WALLS or LOWER_WALLS at sw.

Multiple walkers [47] can also be used. See below the examples.

Description of components

By default this Action calculates the following quantities. These quanties 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 proceding 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 componets 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 biasfactor, 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 (kb∗DeltaT∗pace∗timestep)/tau
GRID_RFILES	read grid for the bias
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_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)
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	monodimensional lower limits, outside the limits the system will not feel the biasing force.
INTERVAL_MAX	monodimensional upper limits, outside the limits the system will not feel the biasing force.
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

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

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
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 biasfactor and initial Gaussian height.

```
BEGIN_PLUMED_FILE
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
```

The following input enables the MPI version of multiple-walkers.

```
BEGIN_PLUMED_FILE
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
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.

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 Belfast tutorial: Umbrella sampling

Description of components

By default this Action calculates the following quantities. These quanties 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 proceding 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 componets 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...

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
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
```

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.

Description of components

By default this Action calculates the following quantities. These quanties 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 proceding 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 componets 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...

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
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
```

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.

Options

NO	(default=off) switch off restart - can be used to override the behavior of the MD engine

Examples

Using the following input:

```
BEGIN_PLUMED_FILE
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
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
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
RESTART NO
d1: DISTANCE ATOMS=1,2
PRINT ARG=d1 FILE=out1
```

Chapter 8

Additional Modules

Here is collected the documentation for the additional modules contributed to PLUMED.

Module	Description	Authors	References
PLUMED-ISDB	Integrative Structural and Dynamical Biology with PLUMED	Max Bonomi and Carlo Camilloni	[57]
Experiment Directed Simulation	Methods for incorporating additional information about CVs into MD simulations by adaptively determined linear bias parameters	Glen Hocky, Andrew White	[25] [58]
Extended-System Adaptive Biasing Force	Methods for performing eABF or DRR method to calculate PMF along CVs	Haochuan Chen, Haohao Fu	[34] [59] [60]
Variationally Enhanced Sampling (VES code)	Module that implements enhanced sampling methods based on Variationally Enhanced Sampling	Omar Valsson	[61]

8.1 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
- Biases Documentation

Furthermore 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.

Additional tutorials focused on the ISDB module are included in the following and are meant as advanced tutorials.

- Tutorials

8.1.1 CVs Documentation

The following list contains descriptions of a number of the colvars that are currently implemented in the PLUMED-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 atomsor ambiguous NOE.
PCS	Calculates the Pseudocontact shift of a nucleus determined by the presence of a metal ion susceptible to anisotropic magnetization.
PRE	Calculates the Paramegnetic Resonance Enhancement intensity ratio between a spinlabel atom and a list of atoms .
RDC	Calculates the (Residual) Dipolar Coupling between two atoms.
SAXS	Calculates SAXS scattered intensity using the Debye equation.

8.1.1.1 CS2BACKBONE

This is part of the isdb module

Calculates the backbone chemical shifts for a protein.

The functional form is that of CamShift [62]. The chemical shifts of the selected nuclei/residues are saved as components. Reference experimental values can also be stored as components. The two sets of components can then be used to calculate either a scoring function as in [63] [64], using the keyword CAMSHIFT or to calculate ensemble averaged chemical shift as in [65] [66] (see ENSEMBLE, STATS and RESTRAINT). Finally they can also be used as input for METAINFERENCE, [67] . In the current implementation there is no need to pass the data to METAINFERENCE because CS2BACKBONE can internally enable Metainference using the keywork DOSCORE.

CamShift calculation is relatively heavy because it often uses a large number of atoms, in order to make it faster it is currently parallelised with OpenMP.

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.

In general the system for which chemical shifts are calculated must be completly included in ATOMS and a TEMPLATE pdb file for the same atoms should be provided as well in the folder DATADIR. The atoms are made automatically whole unless NOPBC is used, in particular if the system is made of by multiple chains it is usually better to use NOPBC and make the molecule whole WHOLEMOLECULES selecting an appropriate order.

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), all the six files should always be present. 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 go from 1 to N irrespectively of the numbers used in the pdb file. The first and last residue of each chain should be preceeded by a # character. Termini groups like ACE or NME should be removed from the PDB.

```
CAshifts.dat:
#1 0.0
2 55.5
3 58.4
.
.
#last 0.0
#last+1 (first) of second chain
.
#last of second chain
```

The default behaviour 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 backcalculated values.

A pdb file is needed to the generate a simple topology of the protein. For histidines 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.

One more standard file is also needed in the folder DATADIR: camshift.db. This file includes all the CamShift parameters and can be found in regtest/isdb/rt-cs2backbone/data/ .

All the above files must be in a single folder that must be specified with the keyword DATADIR.

Additional material and examples can be also found in the tutorial Belfast tutorial: NMR restraints

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 these customizable components the following quantities will always be output

Quantity	Description
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance
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
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt 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=1000) 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 to initialise ALMOST.
NEIGH_FREQ	(default=20) Period in step for neighbour list update.
NRES	Number of residues, corresponding to the number of chemical shifts.

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
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 experimetnal 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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)

Examples

In this first example the chemical shifts are used to calculate a scoring function to be used in NMR driven Metadynamics [64] :

```
BEGIN_PLUMED_FILE
whole: GROUP ATOMS=2612-2514:-1,961-1:-1,2466-962:-1,2513-2467:-1
WHOLEMOLECULES ENTITY0=whole
cs: CS2BACKBONE ATOMS=1-2612 NRES=176 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 [65] [66].

```
BEGIN_PLUMED_FILE
cs: CS2BACKBONE ATOMS=1-174 DATADIR=data/ NRES=13
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
cs: CS2BACKBONE ATOMS=1-174 DATADIR=data/ NRES=13 DOSCORE NDATA=24
csbias: BIASVALUE ARG=cs.score

PRINT ARG=(cs\.hn_.*),(cs\.nh_.*) FILE=CS.dat STRIDE=1000
PRINT ARG=cs.score FILE=BIAS STRIDE=100
```

8.1.1.2 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. [68] . 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 esemble of structures using the Metainference approach [67] .

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.
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

Description of components

By default this Action calculates the following quantities. These quanties 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	Bayesian score
scoreb	Beta Bayesian score

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
TEMP	temperature
NL_CUTOFF	The cutoff in overlap for the neighbor list
NL_STRIDE	The frequency with which we are updating the neighbor list
SIGMA_MEAN	starting value for the uncertainty in the mean estimate

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
ANALYSIS	(default=off) run in analysis mode

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:

```
#! 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_MEAN=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
```

8.1.1.3 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.

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

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
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
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

8.1.1.4 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 ADD↩ COUPLINGS flag and 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 .

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 these customizable components the following quantities will always be output

Quantity	Description
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance
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
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**ADDCOUPLINGS**	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, 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=1000) 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
ADDCOUPLINGS	(default=off) Set this flag if you 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps

MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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...

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 datapoints and compute the correlation and other measures and finally print the results.

```
BEGIN_PLUMED_FILE

MOLINFO MOLTYPE=protein STRUCTURE=peptide.pdb
WHOLEMOLECULES ENTITY0=1-111

JCOUPLING ...
    ADDCOUPLINGS
    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
```

8.1.1.5 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.

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 these customizable components the following quantities will always be output

Quantity	Description
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance
noe	the # NOE

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**ADDEXP**	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, GROUP↩ A3...
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, GROUP↩ B3...

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=1000) 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
ADDEXP	(default=off) Set to TRUE if you want to have fixed components with the experimental reference 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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...

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
NOE ...
GROUPA1=1,3 GROUPB1=2,2
GROUPA2=5 GROUPB2=7
GROUPA3=4,4,8,8 GROUPB3=15,16,15,16
DOSCORE
LABEL=noes
... NOE

PRINT ARG=noes.* FILE=colvar
```

8.1.1.6 PCS

This is part of the isdb module

Calculates the Pseudocontact 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 [69] . While in principle the averaging resulting from the tumbling should remove the pseudocontact shift, in presence of the NMR magnetic field the magnatically anisotropic molecule bound to system will break the rotational symmetry does resulting in measurable PCSs and RDCs.

PCSs 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 perfomed using PCSs, ENSEMBLE, STATS and RESTRAINT . Metainference simulations can be performed with this CV and METAINFERENCE .

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 these customizable components the following quantities will always be output

Quantity	Description
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance
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
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**SVD/ADDCOUPLINGS**	the experimental # RDC

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=1000) 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 backcalculate using Single Value Decomposition (need GSL at compilation time).
ADDCOUPLINGS	(default=off) Set to TRUE if you want to have fixed components with the experimetnal 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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 usefull for ef STATS). You can use multiple instances of this keyword i.e. COUPLING1, COUPLING2, COUPLING3...

Examples

In the following example five PCSs 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 PCSs are calculated using a Single Value Decomposition algorithm.

```
BEGIN_PLUMED_FILE
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
```

8.1.1.7 PRE

This is part of the isdb module

Calculates the Paramegnetic Resonance Enhancement intensity ratio between a spinlabel 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 GROUPA1, 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.

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 these customizable components the following quantities will always be output

Quantity	Description
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance
pre	the # PRE

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**ADDEXP**	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, GROUPA2, 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=1000) 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
ADDEXP	(default=off) Set to TRUE if you want to have fixed components with the experimetnal 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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...

Examples

In the following example five PRE intensities are calculated using the distance between the oxigen of the spin label and the backbone hydrogens. Omega is the NMR frequency, RTWO the R2 for the hydrogens, INEPT of 8 ms for the experiment and a TAUC of 1.21 ns

```
BEGIN_PLUMED_FILE
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
```

8.1.1.8 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 [70] . 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 perfomed 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 Belfast tutorial: NMR restraints

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 these customizable components the following quantities will always be output

Quantity	Description
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance
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
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**SVD/ADDCOUPLINGS**	the experimental # RDC

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=1000) 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 backcalculate using Single Value Decomposition (need GSL at compilation time).
ADDCOUPLINGS	(default=off) Set to TRUE if you want to have fixed components with the experimetnal 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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 usefull for ef STATS). You can use multiple instances of this keyword i.e. COUPLING1, COUPLING2, COUPLING3...

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 experimenta data.

```
BEGIN_PLUMED_FILE
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.*

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
```

8.1.1.9 SAXS

	This is part of the isdb module

Calculates SAXS scattered intensity using the Debye equation.

Intensities are calculated for a set of scattering lenght set using QVALUES numbered keywords, QVALUE cannot be 0. Structure factors can be either assigned using a polynomial expansion to any order using the PARAME↩ TERS keywords; automatically assigned to atoms using the ATOMISTIC flag reading a PDB file, a correction for the water density is automatically added; automatically assigned to Martini pseudoatoms usign the MARTINI flag. The calculated intensities can be scaled using the SCEXP keywords. This is applied by rescaling the structure factors. Experimental reference intensities can be added using the ADDEXP and EXPINT flag and keywords. METAINFERENCE can be activated using DOSCORE and the other relevant keywords.

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 these customizable components the following quantities will always be output

Quantity	Description
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance
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
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt the bias
scale	**SCALEDATA**	scale parameter
offset	**ADDOFFSET**	offset parameter
ftilde	**GENERIC**	ensemble average estimator
exp	**ADDEXP**	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,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=1000) write the status to a file every N steps, this can be used for restart/continuation
WATERDENS	(default=0.334) Density of the water to be used for the correction of atomistic structure factors.
SCEXP	(default=1.0) SCALING value of the experimental data. Usefull to simplify the comparison.

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
ATOMISTIC	(default=off) calculate SAXS for an atomistic model
MARTINI	(default=off) calculate SAXS for a Martini model
ADDEXP	(default=off) Set to TRUE if you 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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 lenghts in Angstrom are given as QVALUE1, QVALUE2, ... 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 structure factor for the i-th atom/bead. You can use multiple instances of this keyword i.e. PARAMETERS1, PARAMETERS2, PARA↩ METERS3...
EXPINT	Add an experimental value for each q value. You can use multiple instances of this keyword i.e. EXPINT1, EXPINT2, EXPINT3...

Examples

in the following example the saxs intensities for a martini model are calculated. structure factors are obtained from the pdb file indicated in the MOLINFO.

```
BEGIN_PLUMED_FILE
MOLINFO STRUCTURE=template.pdb

SAXS ...
LABEL=saxs
ATOMS=1-355
ADDEXP
SCEXP=3920000
MARTINI
QVALUE1=0.02 EXPINT1=1.0902
QVALUE2=0.05 EXPINT2=0.790632
QVALUE3=0.08 EXPINT3=0.453808
QVALUE4=0.11 EXPINT4=0.254737
QVALUE5=0.14 EXPINT5=0.154928
QVALUE6=0.17 EXPINT6=0.0921503
QVALUE7=0.2 EXPINT7=0.052633
QVALUE8=0.23 EXPINT8=0.0276557
QVALUE9=0.26 EXPINT9=0.0122775
QVALUE10=0.29 EXPINT10=0.00880634
QVALUE11=0.32 EXPINT11=0.0137301
QVALUE12=0.35 EXPINT12=0.0180036
QVALUE13=0.38 EXPINT13=0.0193374
QVALUE14=0.41 EXPINT14=0.0210131
QVALUE15=0.44 EXPINT15=0.0220506
... SAXS

PRINT ARG=(saxs\.q_.*),(saxs\.exp_.*) FILE=colvar STRIDE=1
```

8.1.2 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.

8.1.2.1 SELECT

	This is part of the isdb module

Selects an argument based on the value of a SELECTOR.

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 proceding 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 componets 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...

Examples

In this example we use a simulated-tempering like approach activated by the RESCALE action. For each value of the rescale 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
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
```

8.1.3 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.

METAINFERENCE	Calculates the Metainference energy for a set of experimental data.
RESCALE	Rescales the value of an another action, being a Collective Variable or a Bias.

8.1.3.1 METAINFERENCE

This is part of the isdb module

Calculates the Metainference energy for a set of experimental data.

Metainference [67] 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.

Metadynamic Metainference [71] 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 instantaneus 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 experiemntal data can chosen using the keywork 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_MEAN 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 optimisation as introduced in [72] (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_STRIDE 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.

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 these customizable components the following quantities will always be output

Quantity	Description
bias	the instantaneous value of the bias potential
sigma	uncertainty parameter
sigmaMean	uncertainty in the mean estimate
acceptSigma	MC acceptance

In addition the following quantities can be calculated by employing the keywords listed below

Quantity	Keyword	Description
acceptScale	**SCALEDATA**	MC acceptance
weight	**REWEIGHT**	weights of the weighted average
biasDer	**REWEIGHT**	derivatives wrt 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=1000) 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 proceding 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 componets 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
DSIGMA	maximum MC move of the uncertainty parameter
SIGMA_MEAN0	starting value for the uncertainty in the mean estimate
TEMP	the system temperature - this is only needed if code doesnt' pass the temperature to plumed
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
MC_CHUNKSIZE	MC chunksize
STATUS_FILE	write a file with all the data usefull 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)

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
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_MEAN=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
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 ...
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_MEAN=0.001
LABEL=spe
... METAINFERENCE
```

(See also RDC, PBMETAD).

8.1.3.2 RESCALE

This is part of the isdb module

Rescales 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 rescaled, 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 rescaled, 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.

Description of components

By default this Action calculates the following quantities. These quanties 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 proceding 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 componets 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 rescaled
MC_STEPS	number of MC steps
MC_STRIDE	MC stride
PACE	Pace for adding bias, in MC stride unit

Examples

In this example we use RESCALE to implement a simulated-tempering like approach. The total potential energy of the system is rescaled 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
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 rescaled, but just used in the calculation of the Metropolis acceptance probability when proposing a move in the rescaling parameter.

```
BEGIN_PLUMED_FILE
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
```

8.1.4 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.

Compulsory keywords

NAME	name of the SELECTOR
VALUE	set (initial) value of the SELECTOR

Examples

A typical example is the simulated-tempering like approach activated by RESCALE. In this example the total potential energy of the system is rescaled by a parameter defined on a grid of dimension NBIN in the range from 1 to MAX_RESCALE. The value of the rescale 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
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
```

8.1.5 Tutorials

The following are tutorials meant to learn how to use the different methods implemented in the ISDB module.

ISDB: setting up a Metadynamic Metainference simulation	This tutorial show an example on how to use PLUMED-ISDB to run Metadynamic Metainference

8.1.5.1 ISDB: setting up a Metadynamic Metainference simulation

8.1.5.1.1 Aims

The aim of this tutorial is to introduce the users to the ISDB module and in particular to Metadynamic Metainfenrence [67] [71] ensemble determination. We will reproduce the setup of the simulation for a simple system [72] . For a general overview of the problem of ensembles determination please read [73] .

8.1.5.1.2 Objectives

Once this tutorial is completed students will be able to:

- Setup their own PLUMED-ISDB simulation.

8.1.5.1.3 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 metadynamic metainference simulation

This tutorial has been tested on a pre-release version of version 2.4.

8.1.5.1.4 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. Furthemore, 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 employng 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.

8.1.5.1.5 Run a reference simulation

The system we use is the EGAAWAASS peptide used in ref. [72] . 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 potentianl 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
# this is optional and tell to VIM that this is a PLUMED file
# vim: ft=plumed
# see comments just below this input file
MOLINFO MOLTYPE=protein STRUCTURE=egaawaass.pdb
WHOLEMOLECULES ENTITY0=1-111

# EEF1SB Implicit solvation
protein-h: GROUP NDX_FILE=index.ndx NDX_GROUP=Protein-H
solv: EEFSOLV ATOMS=protein-h NOPBC NL_STRIDE=10 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 suboptimal 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 usefull to setup a Metadynamics calculation. In particular we will use PBMETAD because it is then a natural choice for Metadynamic Metainference later. The following input file is meant to be appended to the former.

```
BEGIN_PLUMED_FILE
# 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=1000
    SIGMA_MIN=0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.06,0.001
```

```
    SIGMA_MAX=0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.6,0.2
    ADAPTIVE=DIFF
    HEIGHT=0.5
    PACE=200
    BIASFACTOR=40
    GRID_MIN=-pi,-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,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 parallelise 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
```

8.1.5.1.6 Metadynamic Metainference

The former simulations should provide a converged (check for this) ensemble for the peptide. As shown in [72] the agreement with the multiple avaible NMR experimental data is not perfect. In order to generate an ensemble compatible with most of the available experimetnal 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 splitted 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. In this latter case we use a different noise model, i.e. NOISE=MOUTLIERS because the forward model for chemical shifts can result in systematic errors for some of them.

The following input file is meant to be appended to the formers.

```
BEGIN_PLUMED_FILE
# EXPERIMENTAL DATA SECTION

# RDCs (Grzesiek et al.)
# xGAAWAASS
RDC ...
    ADDCOUPLINGS
    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 ...
    ADDCOUPLINGS
    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

# xGxAWxASx
JCOUPLING ...
    ADDCOUPLINGS
    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 ...
    ADDCOUPLINGS
    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 ...
    ADDCOUPLINGS
    TYPE=CCG
    NOPBC
    ATOMS1=67,47,49,52 COUPLING1=1.59
    LABEL=jccg
... JCOUPLING

# xxxxWxxxx
JCOUPLING ...
    ADDCOUPLINGS
    TYPE=NCG
    NOPBC
    ATOMS1=47,49,52,53 COUPLING1=1.21
    LABEL=jncg
... JCOUPLING

# Chemical shifts
cs: CS2BACKBONE ATOMS=1-111 NRES=9 DATA=data TEMPLATE=egaawaass.pdb NOPBC

# metainference entries

#RDCS
METAINFERENCE ...
    ARG=(nh\.rdc_.*),pb.bias
    PARARG=(nh\.exp_.*)
    REWEIGHT
    NOISETYPE=MGAUSS
    OPTSIGMAMEAN=SEM AVERAGING=200
    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 AVERAGING=200
    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

#JC
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 AVERAGING=200
    SIGMA0=5.0 SIGMA_MIN=0.0001 SIGMA_MAX=15.0 DSIGMA=0.1
    WRITE_STRIDE=10000
    LABEL=byj
... METAINFERENCE

#CS
METAINFERENCE ...
    ARG=(cs\.ca_.*),(cs\.cb_.*),pb.bias
    PARARG=(cs\.expca.*),(cs\.expcb.*)
    REWEIGHT
    NOISETYPE=MOUTLIERS
    OPTSIGMAMEAN=SEM AVERAGING=200
    SIGMA0=5.0 SIGMA_MIN=0.0001 SIGMA_MAX=15.0 DSIGMA=0.1
    WRITE_STRIDE=10000
    LABEL=bycs
... METAINFERENCE

# 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=bycs.*      STRIDE=200 FILE=BAYES.CS

# the following are usefull for the analysis on-the-fly of the quality of the agreement with the experimentl d
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 [71] .

```
mpiexec -np 14 gmx_mpi mdrun -s topolnew -multi 14 -plumed plumed-eef1-pbmetad-m_m.dat -table table.xvg -table
```

8.2 Experiment Directed Simulation

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 [25] as well as additional minimization algorithms for Coarse-Grained Directed Simulation [58].

Notice that a similar method is available as MAXENT, although with different features and using a different optimization algorithm.

Installation

This module is not installed by default. Add '--enable-modules=eds' to your './configure' command when building PLUMED to enable these features.

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`.

Module Contents

- Biases Documentation

8.2.1 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.

8.2.1.1 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 prefactors adaptively using the scheme of White and Voth [25] in order to match target observable values for a set of CVs. 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.

Description of components

By default this Action calculates the following quantities. These quanties 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
_coupling	For each named CV biased, there will be a corresponding output CV_coupling storing the current linear bias prefactor.

Compulsory keywords

RANGE	(default=3.0) The largest magnitude of the force constant which one expects (in kBT) for each CV based
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.

Options

NUMERICAL_DERIVATIVES	(default=off) calculate the derivatives for these quantities numerically
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 proceding 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 componets 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 values
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 experimental 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.
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)

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
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=50000 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
#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
eds: 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
#add the option to write to a restart file
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 PERIOD=50000 TEMP=1.0 OUT_RESTART=restart.dat
```

Read in a previous restart file. Adding RESTART flag makes output append

```
BEGIN_PLUMED_FILE
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 PERIOD=50000 TEMP=1.0 IN_RESTART=restart.dat RESTART
```

Read in a previous restart file and freeze the bias at the final level from the previous simulation

```
BEGIN_PLUMED_FILE
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 TEMP=1.0 IN_RESTART=restart.dat FREEZE
```

Read in a previous restart file and freeze the bias at the mean from the previous simulation

```
BEGIN_PLUMED_FILE
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 TEMP=1.0 IN_RESTART=restart.dat FREEZE MEAN
```

Read in a previous restart file and continue the bias, but use the mean from the previous run as the starting point

```
BEGIN_PLUMED_FILE
eds: EDS ARG=dist,dist2 CENTER=2.0,1.0 PERIOD=50000 TEMP=1.0 IN_RESTART=restart.dat MEAN
```

8.3 Extended-System Adaptive Biasing Force

Overview

This module contains the eABF/DRR method to do free energy calculation or enhance sampling along CVs.

Installation

This module is not installed by default. Add '--enable-modules=drr --enable-boost_serialization' to your './configure' command when building PLUMED to enable these features.

Usage

Please read drr_tool and DRR for more information.

Module Contents

- Biases Documentation
- Command Line Tools

8.3.1 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) [34] methodon one or more collective variables. This method is alsocalled dynamic reference restraining(DRR) [35] .

8.3.1.1 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) [34] method on one or more collective variables. This method is also called dynamic reference restraining(DRR) [35] .

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 jusk 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) [59] 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:

$$A'(\xi^*) = \frac{\sum_\lambda N(\xi^*, \lambda)\left[\frac{\xi^* - \langle\xi\rangle_\lambda}{\beta\sigma^2_\lambda} - k(\xi^* - \lambda)\right]}{\sum_\lambda N(\xi^*, \lambda)}$$

The code performing UI(colvar_UIestimator.h) is contributed by Haohao Fu [60] . 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. To get PMF, the abf_integrate(https://github.com/Colvars/colvars/tree/master/colvartools) is useful.

Description of components

By default this Action calculates the following quantities. These quanties 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 will be refereceable 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 will be refereceable 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.

Compulsory keywords

TAU	(default=0.5) specifies relaxation time on each of variables are, similar to extendedTimeConstant in Colvars
FRICTION	(default=8.0) add a friction to the variable, similar to extendedLangevinDamping 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)
FULLSAMPLES	(default=500) number of samples in a bin prior to application of the ABF
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
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 proceding 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 componets 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 kbt/(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)
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)

Examples

The following input tells plumed to perform a eABF/DRR simulation on two torsional angles.

```
BEGIN_PLUMED_FILE
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
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
```

8.3.2 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

8.3.2.1 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

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

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

8.4 Variationally Enhanced Sampling (VES code)

The VES code is a module for PLUMED that implements enhanced sampling methods based on *Variationally Enhanced Sampling* (VES) [61]. 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 needes 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

8.4.1 Biases

The following list contains the biases available in the VES code.

VES_LINEAR_EXPANSION	Linear basis set expansion bias.

8.4.1.1 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 sufficent.

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 specfiy 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 postprocessing 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 coefficents. In this case the coefficents should be read in from the coefficent file given in the COEFFS keyword.

Bias Cutoff

It is possible to impose a cutoff on the bias potential using the procedure introduced in [74] 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.

Description of components

By default this Action calculates the following quantities. These quanties 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.
COEFFS	read in the coefficents 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 proceding 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 componets 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...

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 coefficents 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 coefficents iterations (the FES_OUTPUT keyword). Parameters that are very specfic to the problem at hand, like the order of the basis functions, the interval on which the basis functions are defined, and the stepsize used in the optimizer, are left unfilled.

```
BEGIN_PLUMED_FILE
bf1: BF_LEGENDRE ORDER=__ MINIMUM=__ MAXIMUM=__

VES_LINEAR_EXPANSION ...
 ARG=d1
 BASIS_FUNCTIONS=bf1
 TEMP=__
 GRID_BINS=200
 LABEL=b1
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
 BIAS=b1
 STRIDE=1000
 LABEL=o1
 STEPSIZE=__
 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
bf1: BF_FOURIER  ORDER=__ MINIMUM=__ MAXIMUM=__
bf2: BF_LEGENDRE ORDER=__ MINIMUM=__ MAXIMUM=__

td_wt: TD_WELLTEMPERED BIASFACTOR=10.0

VES_LINEAR_EXPANSION ...
 ARG=cv1,cv2
 BASIS_FUNCTIONS=bf1,bf2
 TEMP=__
 GRID_BINS=100
 LABEL=b1
 TARGET_DISTRIBUTION=td_wt
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
 BIAS=b1
 STRIDE=1000
 LABEL=o1
 STEPSIZE=__
 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
bf1: BF_LEGENDRE ORDER=__ MINIMUM=__ MAXIMUM=__
bf2: BF_LEGENDRE ORDER=__ MINIMUM=__ MAXIMUM=__

VES_LINEAR_EXPANSION ...
 ARG=cv1,cv2
 BASIS_FUNCTIONS=bf1,bf2
 TEMP=__
 GRID_BINS=100
 LABEL=b1
 BIAS_CUTOFF=20.0
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
 BIAS=b1
 STRIDE=1000
 LABEL=o1
 STEPSIZE=__
 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 coefficents 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
bf1: BF_LEGENDRE ORDER=__ MINIMUM=__ MAXIMUM=__
bf2: BF_LEGENDRE ORDER=__ MINIMUM=__ MAXIMUM=__

VES_LINEAR_EXPANSION ...
 ARG=cv1,cv2
 BASIS_FUNCTIONS=bf1,bf2
 TEMP=__
 GRID_BINS=100
 LABEL=b1
 BIAS_CUTOFF=20.0
 COEFFS=coeffs_final.data
... VES_LINEAR_EXPANSION
```

8.4.2 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_CUSTOM	Basis functions given by arbitrary mathematical expressions.
BF_FOURIER	Fourier basis functions.
BF_LEGENDRE	Legendre polynomials basis functions.
BF_POWERS	Polynomial power basis functions.
BF_SINE	Fourier sine basis functions.

8.4.2.1 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 $\frac{1}{\sqrt{1-x^2}}$

$$\int_{-1}^{1} dx\,T_n(x)\,T_m(x)\,\frac{1}{\sqrt{1-x^2}}=\{\,0\;n\neq m\pi n=m=0\pi/2n=m\neq 0$$

For further mathematical properties of the Chebyshev polynomials see for example the Wikipedia page.

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.

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
bfC: BF_CHEBYSHEV MINIMUM=0.0 MAXIMUM=10.0 ORDER=20
```

8.4.2.2 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

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.

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
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.

8.4.2.3 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 $[a, b]$

$$\int_a^b dx\, f_n(x)\, f_m(x) = \{\, 0\ n \neq m(b-a)n = m = 0(b-a)/2n = m \neq 0.$$

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.

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
BF_COSINE MINIMUM=-pi MAXIMUM=+pi ORDER=10 LABEL=bf1
```

8.4.2.4 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.

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.

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
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
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
```

8.4.2.5 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 $[a,b]$

$$\int_a^b dx\, f_n(x)\, f_m(x) = \{\, 0\ n\neq m(b-a)n=m=0(b-a)/2n=m\neq 0.$$

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.

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
BF_FOURIER MINIMUM=-pi MAXIMUM=+pi ORDER=10 LABEL=bf_fourier
```

8.4.2.6 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.

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.

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
bf_leg: BF_LEGENDRE MINIMUM=-4.0 MAXIMUM=8.0 ORDER=20
```

Examples

8.4.2.7 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 $f_0(x) = 1$
$f_1(x) = x$
$f_2(x) = x^2$
$\vdots$
$f_n(x) = x^n$
$\vdots$
$f_N(x) = x^N$

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.

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.

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
BF_POWERS MINIMUM=-2.0 MAXIMUM=2.0 ORDER=5 LABEL=bf_pow
```

8.4.2.8 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 $[a, b]$

$$\int_a^b dx\, f_n(x)\, f_m(x) = \{\, 0\ n \neq m(b-a)n = m = 0(b-a)/2n = m \neq 0.$$

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.

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
BF_SINE MINIMUM=-pi MAXIMUM=+pi ORDER=10 LABEL=bfS
```

Examples

8.4.3 Target Distributions

The following list contains the target distributions available in the VES code.

TD_CHISQUARED	Chi-squared distribution (static).
TD_CHI	Chi distribution (static).
TD_CUSTOM	Target distribution given by an arbitrary mathematical expression (static or dynamic).
TD_EXPONENTIALLY_MODIFIED_GAUSSIAN	Target distribution given by a sum of exponentially modified Gaussian distributions (static).
TD_EXPONENTIAL	Exponential distribution (static).
TD_GAUSSIAN	Target distribution given by a sum of Gaussians (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_PRODUCT_COMBINATION	Target distribution given by product combination of distributions (static or dynamic).
TD_PRODUCT_DISTRIBUTION	Target distribution given by a separable productof 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).

8.4.3.1 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 postive 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.

Compulsory keywords

MINIMUM	The minimum of the chi-squared distribution.
SIGMA	The σ parameter of the chi-squared distribution given as a postive number.
KAPPA	The k parameter of the chi-squared distribution given as postive 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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

Chi-squared distribution with $a = -10.0$, $\sigma = 2.0$, and $k = 2$

```
BEGIN_PLUMED_FILE
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
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
```

8.4.3.2 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 postive 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.

Compulsory keywords

MINIMUM	The minimum of the chi distribution.
SIGMA	The σ parameter of the chi distribution given as a postive number.
KAPPA	The k parameter of the chi distribution given as postive 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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

Chi distribution with $a = 10.0$, $\sigma = 2.0$, and $k = 2$

```
BEGIN_PLUMED_FILE
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
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
```

8.4.3.3 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 *kBT* 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 normalizable. The code will perform checks to make sure that this is indeed the case.

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 *kBT* 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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

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
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
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 $F(\mathbf{s})$. For example, the following input is identical to using TD_WELLTEMPERED with BIASFACTOR=10.

```
BEGIN_PLUMED_FILE
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 *kBT* variable. The following syntax will give the exact same results as the syntax above

```
BEGIN_PLUMED_FILE
TD_CUSTOM ...
 FUNCTION=exp(-(1.0/(kBT*10.0))*FE)}
 LABEL=td
... TD_CUSTOM
```

8.4.3.4 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.

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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

An exponentially modified Gaussian distribution in one-dimension

```
BEGIN_PLUMED_FILE
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
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
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
```

8.4.3.5 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.

Compulsory keywords

MINIMUM	The minimum of the exponential distribution.
LAMBDA	The λ parameter of the exponential distribution given as postive 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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

Exponential distribution with $a = 10.0$ and $\lambda = 0.5$

```
BEGIN_PLUMED_FILE
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
td_uni: TD_UNIFORM

td_exp: TD_EXPONENTIAL  MINIMUM=-10.0  LAMBDA=0.5

td_pd: TD_PRODUCT_DISTRIBUTION DISTRIBUTIONS=td_uni,td_exp
```

8.4.3.6 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 Gaussians (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 Gaussians` 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 Gaussians as these distributions properly account for the periodicity of the CVs.

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 Gaussians is considered as un-correlated (i.e. value of 0.0). You can use multiple instances of this keyword i.e. CORRELATION1, CORRELATION2, 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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

One single Gaussians in one-dimension.

```
BEGIN_PLUMED_FILE
td: TD_GAUSSIAN CENTER1=-1.5 SIGMA1=0.8
```

Sum of three Gaussians in two-dimensions with equal weights as no weights are given.

```
BEGIN_PLUMED_FILE
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 Gaussians 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
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 Gaussians where there is correlation of $\rho_2 = 0.75$ between the two arguments for the second Gaussian.

```
BEGIN_PLUMED_FILE
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
```

8.4.3.7 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.

Compulsory keywords

LOCATION	The μ parameter of the generalized extreme value distribution.
SCALE	The σ parameter for the generalized extreme value distribution given as a postive number.
SHAPE	The ξ 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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

Generalized extreme value distribution with $\mu = 0.0$, $\sigma = 2.0$, and $\xi = 0.0$ (Gumbel distribution)

```
BEGIN_PLUMED_FILE
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
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
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
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
```

8.4.3.8 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 gaamma 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.

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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

A generalized normal distribution in one-dimensional

```
BEGIN_PLUMED_FILE
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
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
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
```

8.4.3.9 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.

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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

Generally you only need to provide the the filename of the external grid file.

```
BEGIN_PLUMED_FILE
td: TD_GRID FILE=input-grid.data
```

8.4.3.10 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.

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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

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
td_uni: TD_UNIFORM

td_gauss: TD_GAUSSIAN CENTER1=-2.0 SIGMA1=0.5

td_comb: TD_LINEAR_COMBINATION DISTRIBUTIONS=td_uniform,td_gaussian
```

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
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 Gaussians 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 Gaussians. 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
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
```

8.4.3.11 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 normalizable. 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.

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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

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
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
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
```

8.4.3.12 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.

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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

In the following example we employ a uniform distribution for argument 1 and a Gaussian distribution for argument 2.

```
BEGIN_PLUMED_FILE
td_uni: TD_UNIFORM

td_gauss: TD_GAUSSIAN CENTER=-2.0 SIGMA=0.5

td_pd: TD_PRODUCT_DISTRIBUTION DISTRIBUTIONS=td_uni,td_gauss
```

Note that order of the labels is important, using DISTRIBUTIONS=td_gauss,td_uni 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.

8.4.3.13 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 $[a_k, b_k]$

$$p_k(s_k) = \{ \ 1(b_k - a_k) \text{if } a_k \leq s_k \leq b_k \ 0 \text{otherwise}$$

The overall distribution is then given as

$$p(\mathbf{s}) = \prod_k^d p_k(s_k) = \{ \prod_k^d \frac{1}{(b_k - a_k)} \text{if } a_k \leq s_k \leq b_k \text{ for all } k0 \text{otherwise}$$

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.

Options

MINIMA	The minima of the intervals where the target distribution is taken as uniform. You should give one value for each argument.
MAXIMA	The maxima 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 maxima 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.

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 all of the following examples equivalent

```
BEGIN_PLUMED_FILE
td: TD_UNIFORM
```

```
BEGIN_PLUMED_FILE
TD_UNIFORM ...
 MINIMA=0.0,0,2
 MAXIMA=10.0,1.0
 LABEL=td
 ... TD_UNIFORM
```

```
BEGIN_PLUMED_FILE
td: TD_UNIFORM  MAXIMA=10.0,1.0


BEGIN_PLUMED_FILE
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
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
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
TD_UNIFORM ...
 MAXIMA=+7.0
 SIGMA_MAXIMA=0.0
 LABEL=td
... TD_UNIFORM
```

or

```
BEGIN_PLUMED_FILE
td: TD_UNIFORM MAXIMA=+7.0
```

8.4.3.14 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.

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/ γ) where γ is the well tempered factor given here. If this option is active the distribution will be automatically normalized.

Examples

Sum of two Von Mises distribution in one dimension that have equal weights as no weights are given.

```
BEGIN_PLUMED_FILE
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
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
```

8.4.3.15 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 [44] 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 correponds to a biased ensemble where, as compared to the unbiased one, the probability peaks have been broaden and the fluctations of the CVs are enhanced. The value of the bias factor γ determines by how much the fluctations 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 [75] 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.

Compulsory keywords

BIASFACTOR	The bias factor used for the well-tempered distribution.

Examples

Employ a well-tempered target distribution with a bias factor of 10

```
BEGIN_PLUMED_FILE
td_welltemp: TD_WELLTEMPERED BIASFACTOR=10
```

8.4.4 Optimizers

The following list contains the optimizers available in the VES code.

OPT_AVERAGED_SGD	Averaged stochastic gradient decent with fixed step size.
OPT_DUMMY	Dummy optimizer for debugging.

8.4.4.1 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.

Algorithim

This optimizer updates the coefficients according to the averaged stochastic gradient decent algorithm described in ref [76]. 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 employes a dynamic target distribution that needes to be iteratively updated (e.g. TD_WELLTEMPERED) [75], 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 coefficent 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 FESs and biases is controlled by the FES_OUTPUT and the BIAS_OUTPUT keywords. It is also possible to output one-dimensional projections of the FESs 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_O↩ UTPUT 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 sufficent 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 coefficents and/or deactive 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 coefficent file. It is possible to generate a template mask file by using the OUTPUT_MASK_FILE keyword.

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 these customizable components the following quantities will always be output

Quantity	Keyword	Description
gradrms	**MONITOR_INSTANTANEOUS_GRADIENT**	the root mean square value of the coefficent 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 coefficent 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 coefficent 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 coefficent 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 identfy 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 coefficents
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 coefficent iterations. If no value is given are the averages only written at the begining 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 coefficent iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficent 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 coefficent 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 coefficents and/or deactive 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 coefficent 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 coefficent 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 coefficent iterations.
EXP_DECAYING_AVER	calculate the averaged coefficients using exponentially decaying averaging using the decaying constant given here in the number of iterations

Examples

In the following input we emloy an averaged stochastic gradient decent with a fixed step size of 1.0 and update the coefficent every 1000 MD steps (e.g. every 2 ps if the MD time step is 0.02 ps). The coefficent are outputted to the coeffs.data every 50 iterations while the FES and bias is outputted to files every 500 iterations (e.g. every 1000 ps).

```
BEGIN_PLUMED_FILE
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=coeffs.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
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=coeffs.data
  COEFFS_OUTPUT=50
  FES_OUTPUT=500
  FES_PROJ_OUTPUT=500
  BIAS_OUTPUT=500
  TARGETDIST_STRIDE=500
  TARGETDIST_OUTPUT=2000
... OPT_AVERAGED_SGD
```

8.4.4.2 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.

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 these customizable components the following quantities will always be output

Quantity	Keyword	Description
gradrms	**MONITOR_INSTANTANEOUS_GRADIENT**	the root mean square value of the coefficent 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 coefficent 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 coefficent 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 coefficent 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 identfy 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 coefficents
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 coefficent iterations. If no value is given are the averages only written at the begining 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 coefficent iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficent 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 coefficent 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

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
```

```
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
```

8.4.5 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 FESs for VES biases from previously obtained coefficients.
VES_OUTPUT_TARGET_DISTRIBUTION	Output target distribution to file.

8.4.5.1 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 programing 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.

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.
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...

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
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 conf.gro is needed to trick the code to exit correctly.

```
plumed driver --plumed plumed.dat --igro conf.gro
```

8.4.5.2 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 FESs for VES biases from previously obtained coefficients.

This action can be used to output to file biases and FESs for VES biases from previously obtained coefficients. It should be used through the driver and can only be used in postprocessing. 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).

Compulsory keywords

BIAS	the label of the VES bias for to output the FESs 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 coefficent iterations.
FES_OUTPUT	how often the FES(s) should be written out to file. Note that the value is given in terms of coefficent 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 coefficent iterations.

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
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
```

This input should be run through the driver by using a command similar to the following one where the trajectory/configuration file conf.gro is needed to correctly define the CVs

```
plumed driver --plumed plumed.dat --igro conf.gro
```

8.4.5.3 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.

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	specfiy 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

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
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 conf.gro is needed to trick the code to exit correctly.

```
plumed driver --plumed plumed.dat --igro conf.gro
```

8.4.6 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.

8.4.6.1 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.

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 seperate value for each replica.
friction	(default=10.) The friction of the Langevin thermostat. For multiple replica you can give a seperate 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 seperate 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 seperate 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 coefficent file for the potential. For multiple replica you can give a seperate value for each replica.
output_coeffs	(default=potential-coeffs.out.data) Filename of the output coefficent file for the potential.
output_coeffs_fmt	(default=%30.16e) Format of the output coefficent 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 coefficents with. For multiple replica you can give a seperate value for each replica.
template_coeffs_file	only generate a template coefficent file with the filename given and exit.

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) postion of the particle as CVs by using the POSITION action as shown in the following PLUMED input

```
BEGIN_PLUMED_FILE
p: POSITION ATOM=1
ene: ENERGY
PRINT ARG=p.x,p.y,ene FILE=colvar.data FMT=%8.4f
```

8.4.7 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.

8.4.7.1 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, Annu. Rev. Phys. Chem. 2016

Variationally Enhanced Sampling:

Variational Approach to Enhanced Sampling and Free Energy Calculations, Phys. Rev. Lett. 2014

Variationally Optimized Free-Energy Flooding for Rate Calculation, Phys. Rev. Lett. 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

8.4.7.2 MARVEL-VES tutorial (Lugano Feb 2017): Metadynamics

8.4.7.2.1 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

8.4.7.2.2 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.

8.4.7.2.3 Instructions

8.4.7.2.3.1 The system

We consider the association/dissociation of NaCl in acqueous 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 [42] and also in reference [77] . We will use the potential developed in ref. [78] for NaCl and TIP3P water with parameters corrected to be used with long-range Coulomb solvers [79]. The system contains 1 Na, 1 Cl, and 106 water molecules (total 320 atoms).

8.4.7.2.3.2 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
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
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 $\sim 5k_{\mathrm{B}}T$ 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 bounday 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.

8.4.7.2.3.3 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 $k_{\mathrm{B}}T$. 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 $k_{\mathrm{B}}T$. 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 $k_{\mathrm{B}}T$. The input is:

```
BEGIN_PLUMED_FILE
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
 REWEIGHTING_NGRID=300
... METAD
```

Here the REWEIGHTING_NGRID 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
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 gaussians 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 gaussians. Normally a bin of size $\sigma/5$ (with σ the gaussian width) is small enough.

8.4.7.2.3.4 Assess convergence

One way to identify if a WT-MetaD 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 gaussians. 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 gaussians, 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 behaviour:

$$V(\mathbf{s},t) = -\left(1-\frac{1}{\gamma}\right)F(\mathbf{s}) + c(t).$$

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.

8.4.7.2.3.5 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 quasistationary 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 quasistationary 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 input script for Plumed is:

```
BEGIN_PLUMED_FILE
# 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.

8.4.7.2.3.6 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.

8.4.7.2.3.7 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
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
 REWEIGHTING_NGRID=400,400
... 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
```

8.4.7.2.4 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.

8.4.7.3 MARVEL-VES tutorial (Lugano Feb 2017): VES 1

8.4.7.3.1 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.

8.4.7.3.2 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.

8.4.7.3.3 Summary of theory

Variationally enhanced sampling [61] 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 [76]. 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.

8.4.7.3.4 Instructions

8.4.7.3.4.1 The system

We will consider the same system employed for the metadynamics tutorial.

8.4.7.3.4.2 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
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
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
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.

8.4.7.3.4.3 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:

$$p(s) = \{ \ 1C \text{ if } s < s_0 \frac{1}{C} e^{-\frac{(s-s_0)^2}{2\sigma^2}} \text{ if } s > s_0$$

where $s_0 = 0.6$ nm and $\sigma = 0.05$. To define this $p(s)$ in Plumed the input is:

```
BEGIN_PLUMED_FILE
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
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
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 [80] . 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?

8.4.7.3.4.4 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 Plumed's input 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.

8.4.7.3.4.5 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
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 [81].

8.4.7.3.4.6 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.

8.4.7.3.5 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.

8.4.7.4 MARVEL-VES tutorial (Lugano Feb 2017): VES 2

8.4.7.4.1 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.

8.4.7.4.2 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.

8.4.7.4.3 Summary of theory

One of the most useful target distribution is the well-tempered one. The well-tempered target distribution is [75] :

$$p(\mathbf{s}) = \frac{e^{-(\beta/\gamma)F(\mathbf{s})}}{\int d\mathbf{s}\, e^{-(\beta/\gamma)F(\mathbf{s})}}$$

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 unphysical 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 [75], 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 $p(\mathbf{s})$ [75].

8.4.7.4.4 Instructions

8.4.7.4.4.1 The system

We will consider the same system employed in previous tutorials.

8.4.7.4.4.2 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:

$$p_{\mathrm{barrier}}(s) = \{\ 1C \text{ if } s < s_0 \frac{1}{C} e^{-\frac{(s-s_0)^2}{2\sigma^2}} \text{ if } s > s_0$$

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
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's input:

```
BEGIN_PLUMED_FILE
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.

8.4.7.4.4.3 Example 2: Constructing a 2 dimensional bias

In this example we will construct a 2 dimensional bias on the distance Na-Cl and the coodrination 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
# 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
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?

8.4.7.4.5 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.

8.4.7.5 MARVEL-VES tutorial (Lugano Feb 2017): Kinetics

8.4.7.5.1 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 rescale the accelerated MD time in a post-processing procedure.

8.4.7.5.2 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.

8.4.7.5.3 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

8.4.7.5.4 Requirements

- python with numpy, scipy, and statsmodels
- LAMMPS compiled with MANYBODY package
- VMD for visualization

8.4.7.5.5 Instructions

8.4.7.5.5.1 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 neccessary 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
# 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
# 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
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
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
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.

8.4.7.5.5.2 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.

8.4.7.5.5.3 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
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
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
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.

8.4.7.5.5.4 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 rescale 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?

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.

config	inquire plumed about how it was configure
driver-float	Equivalent to driver, but using single precision reals.
driver	driver is a tool that allows one to to use plumed to post-process an existing trajectory.
gentemplate	gentemplate is a tool that you can use to construct template inputs for the variousactions
info	This tool allows you to obtain information about your plumed version
kt	Print out the value of $k_B T$ 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
pesmd	Pesmd allows one to do (biased) Langevin dynamics on a two-dimensional potential energy surface.
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 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 xdrfile enabled
> plumed config has xdrfile
Check if plumed as xdrfile AND zlib enabled
> plumed config has xdrfile zlib
Check if plumed as module colvar active
> plumed config module colvar
```

9.2 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.

The input trajectory is specified using one of the following

--ixyz	the trajectory in xyz format
--igro	the trajectory in gro 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
--dump-full-virial	(default=off) with –dump-forces, it dumps the 9 components of the virial
--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 kBT, 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 orthorombic, 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

Examples

```
plumed driver-float --plumed plumed.dat --ixyz trajectory.xyz
```

See also examples in driver

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 .

The input trajectory is specified using one of the following

--ixyz	the trajectory in xyz format
--igro	the trajectory in gro 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

--dump-full-virial	(default=off) with –dump-forces, it dumps the 9 components of the virial
--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 kBT, 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 orthorombic, 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

Examples

The following command tells plumed to postprocess 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
# 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 postprocess 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 natively 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.

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 labelleled 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 quanties 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.

Compulsory keywords

STRIDE	(default=1) the frequency with which the file should be read.
EVERY	(default=1) only read every ith 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 postprocessing that does not involve outputting forces
UPDATE_FROM	Only update this action from this time
UPDATE_UNTIL	Only update this action until this time

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
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
```

9.4 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.

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

Examples

The following generates template input for the action DISTANCE.

```
plumed gentemplate --action DISTANCE
```

9.5 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

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)

Examples

The following command returns the root directory for your plumed distribution.

```
plumed info --root
```

9.6 kt

This is part of the cltools module

Print out the value of $k_B T$ at a particular temperature

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

Examples

The following command will tell you the value of $k_B T$ when T is equal to 300 K in eV

```
plumed kt --temp 300 --units eV
```

9.7 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 plumed's html manual. If you need to use this tool outside those scripts the input is specified using the following command line arguments.

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

Examples

The following generates the html manual for the action DISTANCE.

```
plumed manual --action DISTANCE
```

9.8 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'
```

9.9 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'
```

9.10 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.

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.
```

9.11 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
  -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.
```

9.12 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 ofen 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.

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 path reparameterization algorithm
--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

Examples

The example below shows how you can take a set of unequally spaced frames from a pdb file named inpath.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 outpath.pdb.

```
plumed pathtools --path inpath.pdb --metric EUCLIDEAN --out outpath.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 cvs 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 reparameterise 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
```

9.13 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:

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=on) 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

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.

9.14 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 spcified 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:

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
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
maxneighbours	(default=10000) The maximum number of neighbours an atom can have

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

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

```
nputfile 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
```

9.15 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

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 welltempered 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 (usefull 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 outputfile for sumhills
--outhisto	specify the outputfile 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

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 gaussians), plumed will understand it from the HILLS file.

now sum_hills tool accepts als 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 welltempered 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 gaussians, 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..

9.16 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. Usefull to insert the pdf in a LaTex file as ima
--fs              Specify the fontsize of the pdf output.
--colors          Specify the color palette. Allowed values are: default/ac
```

Chapter 10

Miscelaneous

- Comments
- Continuation lines
- Using VIM syntax file
- Including other files
- Loading shared libraries
- Debugging the code
- Changing exchange patterns in replica exchange
- List of modules
- Special replica syntax
- Parsing constants
- Frequently used tools

10.1 Comments

If you are an organised 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
# This is the distance between two atoms:
DISTANCE ATOM=1,2 LABEL=d1
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
d: DISTANCE ATOMS=1,10
PRINT ARG=d FILE=COLVAR STRIDE=10
ENDPLUMED
commands here are ignored
PRINT ARG=d FILE=COLVAR STRIDE=1
```

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
DISTANCES ATOMS1=1,300 ATOMS2=1,400 ATOMS3=1,500 LABEL=dist
```

(see DISTANCES)

equivalent to this:

```
BEGIN_PLUMED_FILE
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
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

...
```

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
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
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 impatients:

- 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 multiline 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 outpt 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 environemnt 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 preinstalled 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 hardcoded 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
# 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 `metad:` or `LABEL=metad`) 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
m: METAD ...
  ARG=d
  HEIGHT=1.0
  SIGMA=0.5
  PACE=100
...
```

will be folded to

```
+--  6 lines: m: METAD ...-----------------------------------------------------
```

and

```
BEGIN_PLUMED_FILE
METAD LABEL=m ...
  ARG=d
  HEIGHT=1.0
  SIGMA=0.5
  PACE=100
...
```

will be folded to

```
+--  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><C↩TRL+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

```
BEGIN_PLUMED_FILE
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 7th 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 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
INCLUDE FILE=filename
```

So, for example, a single "plumed.dat" file:

```
BEGIN_PLUMED_FILE
DISTANCE ATOMS=0,1 LABEL=dist
RESTRAINT ARG=dist
```

could be split up into two files as shown below:

```
BEGIN_PLUMED_FILE
DISTANCE ATOMS=0,1 LABEL=dist
INCLUDE FILE=toBeIncluded.dat
```

plus a "toBeIncluded.dat" file

```
BEGIN_PLUMED_FILE
RESTRAINT ARG=dist
```

However, when you do this it is important to recognise 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
```

10.4.1 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.

Compulsory keywords

FILE	file to be included

Examples

This input:

```
BEGIN_PLUMED_FILE
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
INCLUDE FILE=pippo.dat
d: DISTANCE ATOMS=c1,c2
PRINT ARG=d
```

where the content of file pippo.dat is

```
BEGIN_PLUMED_FILE
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
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 huge file containing group definitions such as

```
BEGIN_PLUMED_FILE
groupa: GROUP ...
  ATOMS={
    10
    50
    60
## imagine a long list here
    70
    80
```

```
    120
  }
...
groupb: GROUP ...
  ATOMS={
    11
    51
    61
## imagine a long list here
    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
# this is common.dat
t: TORSION ATOMS=1,2,3,4
```

Then `plumed.0.dat`:

```
BEGIN_PLUMED_FILE
# this is plumed.0.dat
INCLUDE FILE=common.dat
RESTRAINT ARG=t AT=1.0 KAPPA=10
```

And `plumed.1.dat`:

```
BEGIN_PLUMED_FILE
# 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 `plumed.dat`:

```
BEGIN_PLUMED_FILE
# this is plumed.dat
t: TORSION ATOMS=1,2,3,4
INCLUDE FILE=other.dat
```

Then `other.0.dat`:

```
BEGIN_PLUMED_FILE
# this is other.0.dat
RESTRAINT ARG=t AT=1.0 KAPPA=10
```

And `other.1.dat`:

```
BEGIN_PLUMED_FILE
# this is other.1.dat
RESTRAINT ARG=t AT=1.2 KAPPA=10
```

10.5 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 independely - we won't be offended), then you can create your own dynamic library. To use this in conjuction with PLUMED you can then load it at runtime by using the LOAD keyword as shown below:

```
BEGIN_PLUMED_FILE
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.

10.5.1 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.

Compulsory keywords

FILE	file to be loaded

Examples

If you have a shared object named extensions.so and want to use the functionalities implemented in it within PL↩ UMED you can load it with the following syntax

```
BEGIN_PLUMED_FILE
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 occurence 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
# 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
# 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.

10.6 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.

10.6.1 DEBUG

	This is part of the generic module

Set some debug options.

Can be used while debugging or optimizing plumed.

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 entirity of the simulation
DETAILED_TIMERS	(default=off) switch on detailed timers
FILE	the name of the file on which to output these quantities

Examples

```
BEGIN_PLUMED_FILE
# 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
```

10.7 Changing exchange patterns in replica exchange

Using the RANDOM_EXCHANGES keyword it is possible to make exchanges betweem randomly chosen replicas. This is useful e.g. for bias exchange metadynamics [82].

10.7.1 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 [82].

Options

SEED	seed for random exchanges

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
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
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
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.

10.8 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 PLUMED's more complicated features. 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 minimalistic PLUMED which can be used as a library but 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
bias	on
cltools	on
colvar	on
crystallization	off
drr	off
eds	off
function	on
generic	on
isdb	on
manyrestraints	off
mapping	on
molfile	on
multicolvar	on
secondarystructure	on
setup	on
vatom	on
ves	off

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.

10.9 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
# 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 [82] simulations:

```
BEGIN_PLUMED_FILE
# 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
# 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
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
...
```

In short, whenever there are keywords that should vary across replicas, you should set them usign 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.

10.10 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
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
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).

10.11 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
landmarkselection	INTERNAL	This is currently a filler page.
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/Outpt

10.11.1 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 distribution (pdf) 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 funciton that must integrate to one that is often called a kernel function and w is a smearing parameter. From a pdf 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 descretized 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:

$$w_j(s) = \int_{a+\frac{j-1}{n}(b-a)}^{a+\frac{j}{n}(b-a)} \sum_i K\left(\frac{s - s_i}{w}\right)$$

10.11.2 kernelfunctions

Functions that are used to construct histograms Functions that are used to construct histograms

Constructing histograms is something you learnt 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 elipse in an n dimensional space which is given by:

$$\begin{aligned} V &= |\Sigma^{-1}| \frac{\pi^{\frac{n}{2}}}{\left(\frac{n}{2}\right)!} \qquad \text{for even} \quad n \\ V &= |\Sigma^{-1}| \frac{2^{\frac{n+1}{2}} \pi^{\frac{n-1}{2}}}{n!!} \end{aligned}$$

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 trucated gaussian is scaled so that its integral over the grid is equal to one when it is normalised. 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.

10.11.3 landmarkselection

This is part of the analysis module

This is currently a filler page. This is currently a filler page.

Just use LANDMARKS=ALL. More complex versions will appear in later versions.

10.11.4 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 parsable real number 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 alignement 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.

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`).

10.11.5 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 d_0. For $r \leq d_0 \quad s(r) = 1.0$ 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}	
MATHEVAL	$s(r) = FUNC$	{MATHEVAL FU↩ NC=1/(1+x^6) R_0= r_0 D_0= d_0}	

Attention

Similarly to the MATHEVAL function, the MATHEVAL switching function only works if one of these two conditions is satisfied: (a) libmatheval is installed on the system and PLUMED has been linked to it or (b) the environment variable `PLUMED_USE_LEPTON` is set equal to `yes` at runtime (in this case, the internal lepton library will be used). Notice that in version v2.5 the internal lepton library will be used by default. Also notice that using MATHEVAL is much slower than using e.g. RATIONAL. Thus, the MATHEVAL switching function is useful to perform quick tests on switching functions with arbitrary form before proceeding to their implementation in C++.

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.

10.11.6 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
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 unescaped dot is a wildcard which is matched by any character, So as an example

```
BEGIN_PLUMED_FILE
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
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
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 parethesis `()` 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
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.

10.11.7 Files

We tried to design PLUMED in such a manner that input/output is done consistently irrespectively of the file type. Most of the files written or read by PLUMED thus follow the very same conventions discussed below.

10.11.7.1 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.

10.11.7.2 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.

10.11.7.3 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
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 multireplica context an input such as

```
BEGIN_PLUMED_FILE
d: DISTANCE ATOMS=1,2
PRINT ARG=d FILE=COLVAR.gz
METAD ARG=d FILE=test.HILLS SIGMA=0.1 HEIGHT=0.1
```

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

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.
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

Lugano tutorial: Analyzing CVs	This tutorial explains how to use PLUMED to analyze CVs
Lugano tutorial: Path CVs	This tutorial explains how to use various kinds of path collective variables
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.

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

11.1 Trieste tutorial: Analyzing trajectories using PLUMED

11.1.1 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.

11.1.2 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 analyse 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.

11.1.3 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.

11.1.4 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 Lugano 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.

11.1.5 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.

11.1.6 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
# 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 autocompleting 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 LABEL=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.

11.1.6.1 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 oxygens. For COORDINATION, set reference distance `R_0` to 2.5 A (be careful with units!!). Try to be smart in selecting the water oxygens without listing all of them explicitly.
- Same as before but against water hydrogen. Also in this case you should be smart to select water hydrogens. 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
# 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 feedbacks 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
# 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 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.

11.1.6.2 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
# 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
```

11.1.7 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 surpising 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 PBCs 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 PBCs within PLUMED. If you know alternative tools that can reconstruct PBCs, it is a good idea to also use them and compare the result with PLUMED.

11.1.7.1 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
# 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*.

11.1.7.2 Exercise 2b: Mistakes with WHOLEMOLECULES

Prepare a PLUMED input file that makes all the water molecules whole. Use the following template

```
BEGIN_PLUMED_FILE
# 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=O↩PTIMAL`, the whole system is translated and rotated. The whole system here means all atoms plus the vectors defining the periodic box.

11.1.7.3 Exercise 2c: Mastering FIT_TO_TEMPLATE

Check how the periodic box rotates when using FIT_TO_TEMPLATE. Use the following template

```
BEGIN_PLUMED_FILE
# 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 ro the `cell_after.*` variables of the last example. This would allow you to force your molecule to a specific orientation.

11.1.7.4 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.

11.2 Trieste tutorial: Averaging, histograms and block analysis

11.2.1 Introduction

The aim of this tutorial is for you to understand how we analyse 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.

11.2.2 Objectives

Once this tutorial is completed students will:

- Be able to explain the role played by the central limit theorem in many of the analyses 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 analyses 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.

11.2.3 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 ammount 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 quantiative 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 Boltzmann's 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 Thoerem. 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. $\langle Y \rangle$ 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 probabiilty 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

11.2.4 Instructions

11.2.4.1 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 analyse using PLUMED. The following short python script will generate 10000 (pseudo) random variables between 0 and 1 from a uniform disribution 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 > mydata
```

This will generate a file called mydata that contains 10001 uniform random variables. 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 'mydata' 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 quantieis by calculating $\frac{S_n}{n}$ from the points we generated and exploting the central limit theorem. We can do this calculation by writing a PLUMED input file that reads:

```
BEGIN_PLUMED_FILE
data: READ FILE=mydata 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 flucutations, $(\delta X)^2$, 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.

11.2.4.2 Calculating a histogram

We can use what we have learnt 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/mathNET/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
data: READ FILE=mydata VALUES=rand
hh: HISTOGRAM ARG=data STRIDE=1 GRID_MIN=0 GRID_MAX=1.0 GRID_BIN=20 KERNEL=DISCRETE
DUMPGRID GRID=hh FILE=myhist.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 'myhist.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?

11.2.4.3 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 realisation should always be at the forefront of our minds whenever we analyse 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/mathNET/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
data: READ FILE=mydata 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 intruction 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 exptracted 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
data: READ FILE=mydata 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=myhist.dat STRIDE=1000
```

Notice that the input here has the same structure as the input for the AVERAGE. Once again we have a CLEAR=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.myhist.dat = analysis of data in first 1000 frames
- analysis.1.myhist.dat = analysis of data in second 1000 frames
- analysis.2.myhist.dat = analysis of data in third 1000 frames
- analysis.3.myhist.dat = analysis of data in fourth 1000 frames
- analysis.4.myhist.dat = analysis of data in fifth 1000 frames
- analysis.5.myhist.dat = analysis of data in sixth 1000 frames
- analysis.6.myhist.dat = analysis of data in seventh 1000 frames
- analysis.7.myhist.dat = analysis of data in eigth 1000 frames
- analysis.8.myhist.dat = analysis of data in ninth 1000 frames
- myhist.dat = analysis of data in tenth 1000 frames

We can plot all of these histograms using gnuplot and the command:

```
gnuplot> p 'analysis.0.myhist.dat' u 1:2 w l, 'analysis.1.myhist.dat' u 1:2 w l, 'analysis.2.myhist.dat' u 1:2
```

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.

11.2.4.4 Problem II: Dealing with rare events and simulation biases

As is discussed in many of the tutorials that are provided here one of PLUMED's primary purposes 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 > mynormal
```

Use what you have learnt 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 doenst 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 analysed the data generated from the Gaussin 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
UNITS NATURAL  #This ensures that Boltzmann's constant is one
data: READ FILE=mynormal 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
UNITS NATURAL #This ensures that Boltzmann's constant is one
data: READ FILE=mynormal 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=myhist.dat
```

Plot the histogram that you obtain from this calculation using the command:

```
gnuplot> p 'myhist.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?

11.2.4.5 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 exericse. 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 esimtate 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 calcuate 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 > mycorr
```

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 wil begin by analysing the data we generated by sampling from the uniform distribution using the following PLUMED input:

```
BEGIN_PLUMED_FILE
data: READ FILE=mydata 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 analysed 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 eror 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 > myaverages.dat
```

This will output a single file called myaverages.dat that can be plotted using gnuplot and the command:

```
gnuplot> p 'myaverages.dat' u 1:2:3 w e, 'myaverages.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 wathcing 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.

11.2.4.6 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 > mcdata
```

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. 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
UNITS NATURAL #This ensures that Boltzmann's constant is one
data: READ FILE=mcdata 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=myhist.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 = "myhist.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 )
# Calcualte 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

11.2.5 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 artefacts 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 analysing the data from your own simulations.

11.3 Trieste tutorial: Using restraints

11.3.1 Aims

The aim of this tutorial is to introduce the users to the use of constant biases in PLUMED.

11.3.2 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 unbiasing of a restrained simulation
- Add an external potential in the form of an analytical or numerical function

11.3.3 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.

11.3.4 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 refered 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.

11.3.4.1 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 challanging to characterise with a standard MD simulation and we will see how we can use an interative 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 subfolders if you want to run the same simulation with multiple choices for the parameters

11.3.5 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 analyse 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
# 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=myhist.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)

11.3.6 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 carefull about the statistics: in the simulation of exercise 1 you were postprocessing 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 myhist.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
# 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 analysing 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 easely possible to recover the unbias distribution at least in the regions of the conformational space that have been througly 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 analyse any property. Furthermore plumed can also automatically use the bias to reweight the accumulated histogram.

```
BEGIN_PLUMED_FILE
# 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 carefull again about the difference in the way statistics is accumulated on-the-fly or for post processising. 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 hardcoded.

```
> 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.

11.3.7 Exercise 3: Apply a quadratic restraint on the same collective variable

Do you expect a different behaviour? This time we can write the plumed input file in such a way to compare directly the biased and unbiased histograms.

```
BEGIN_PLUMED_FILE
# 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=myhist.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 hardcoded.

```
> python3 do_block_histo.py > histo-biased.dat
> python3 do_block_histo.py > histo-reweighted.dat
```

11.3.8 Exercise 4: Apply an upper wall on the distance.

In the above cases we have always applied weak biases. Sometimes biases are usefull 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
# 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__
# calcualte 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 througly enough it is not possible to estimate the histogram in that region even using reweighting (reweighting is not magic!).

11.3.9 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 relavent 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 continuos function centered at the value of the accumulated point and added accumulated on the discrete represattion of the histogram, see Kernel density estimation .

```
BEGIN_PLUMED_FILE
# 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
# vim:ft=plumed
d: DISTANCE ATOMS=1,4
EXTERNAL ARG=d FILE=__FILL__ LABEL=ff
# accumulate the biased histogram
__FILL__
#dumpit
__FILL__
# calcualte 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.

11.3.10 Exercise 6: Preliminary run with Alanine dipeptide

Alanine dipeptide is characterised 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
# 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 defininining complex analytical functions of collective variables, and the bias BIASVALUE that allows using any continuos 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, becasue 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:

11.3.11 Exercise 7: First biased run with Alanine dipeptide

```
BEGIN_PLUMED_FILE
# 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.

11.3.12 Exercise 8: Second biased run with Alanine dipeptide

```
BEGIN_PLUMED_FILE
# 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
# 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.

11.4 Trieste tutorial: Metadynamics simulations with PLUMED

11.4.1 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.

11.4.2 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

11.4.3 Resources

TheTARBALL 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 exercizes, so don't delete it!

11.4.4 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) [42]. This potential is built as a sum of Gaussians 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, Gaussians 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 [44], 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 "biasfactor" 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 biasfactor 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 [83] [84] [85].

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 .

11.4.5 Exercise 1: my first metadynamics calculation

11.4.5.1 Exercise 1a: setup and run

In this excercise 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
# 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 kJoule/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 kJoule/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 Gaussians 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 biasfactor. 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 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.

11.4.5.2 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 Gaussians 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 Gaussians 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 Gaussians 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

11.4.6 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.

11.4.7 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 "unbiasing" 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 [45].

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
# 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 quantites 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 "unbiasing 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 analysi 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 Kjoule/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?

11.4.8 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.

11.5 Trieste tutorial: Running and analyzing multi-replica simulations.

11.5.1 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.

11.5.2 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.

11.5.3 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.

11.5.4 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.

11.5.5 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
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 ony way to do multi-replica simulatons 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.

11.5.6 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
# 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 [82] simulations:

```
BEGIN_PLUMED_FILE
# 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
# 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
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
...
```

In short, whenever there are keywords that should vary across replicas, you should set them usign 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.

11.5.7 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
# 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.

11.5.8 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 [45]. Notice that also here it would be possible to use [3] instead.

```
BEGIN_PLUMED_FILE
# 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 kbT (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, irrespectively of how different were the employed bias potentials. This includes bias-exchange metadynamics [82] when using well-tempered metadynamics [44]. Notice that for non-well-tempered bias exchange metadynamics one can use a very similar approach based on binning that is described in [86]. In addition, the approach illustrated here can be used direcly in bias-exchange metadynamics simulations with additional restraints which are different in different replicas (as in [87]) as well as other flavors of biased replica exchange simulations (e.g. [88] [48]). With some modification to take into account the force field energy it can be used to analyze parallel tempering simulations [89] and simulated tempering simulations [90] (notice that they can be implemented with PLUMED + GROMACS using [91]).

11.5.9 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 correclty, 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.

11.5.10 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.

11.5.11 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.

11.6 Trieste tutorial: Real-life applications with complex CVs

11.6.1 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.

11.6.2 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

11.6.3 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.

11.6.4 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.

11.6.5 Exercise 1: analysis of the BARD1 complex simulation

The BARD1 complex is a heterodimer 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 aminoacids:

```
BEGIN_PLUMED_FILE
rmsd: RMSD REFERENCE=bard1_rmsd.pdb TYPE=OPTIMAL
```

while this one can be used to calculate the DRMSD between chains:

```
BEGIN_PLUMED_FILE
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
# 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
# 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__
```

11.6.6 Exercise 2: analysis of the cmyc-urea simulation

Cmyc is a small disordered peptide made of 11 aminoacid. 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 aminoacids 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
# 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 aminoacid 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.

11.6.7 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
#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
```

11.6.8 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

11.7 Belfast tutorial: Analyzing CVs

11.7.1 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 analyse a trajectory in terms of probabilty distributions and free energy.

11.7.2 Learning Outcomes

Once this tutorial is completed students will:

- Know how to write a simple plumed input file
- Know how to analyse a trajectory using plumed

11.7.3 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 conjuction with the MOLINFO command

11.7.4 Instructions

PLUMED2 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 the plumed available functionalities, other can be used to tell plumed to do something: from analysing 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 analyse a trajectory.

11.7.4.1 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.

11.7.4.2 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 analyse 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 analyse 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 PLUMED2 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 behaviour in this trajectory of both the terminal carbons as well as of the centre 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 analyse 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 rationalise its behaviour.

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.

11.7.4.3 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 PLUMED2 this kind of collective variables fall under the name of MULTICOLVAR (cf. MultiColvar.) Here we are going to analyse 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 infrastracture of multicolvar has been used to develop many PLUMED2 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.

11.7.4.4 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 analyse the above probabilty 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. PLUMED2 includes the possibility of histogramming 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 normalised function, for example a normalised 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 analyse the trajectory. Furthemore try to play with the HISTOGRAM parameters to see the effect of using KERNEL in analysing data.

11.8 Belfast tutorial: Adaptive variables I

11.8.1 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.

11.8.2 Resources

Here is the tarball with the files referenced in the following.

11.8.3 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 plasicity 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 change or desolvation contribution which 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, coordinations and dihedrals (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 desolvation pattern.
Other typical examples are chemical reactions. Nucleophiloic attacks typically happen with a role from the solvent (see some nice paper from Nobel-prize winner Arieh Warshel) and sizeable geometric distortions of the neighboring groups.

11.8.4 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 methylphosphate) 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 is more resemblant to the reactant 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 exactely 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 exactely $S(X) = j$.

In order to provide a value which is continuous, the parameter λ should be correclty 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 gies 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 coordinations (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 Ang^2. PATHMSD produces two arguments that can be printed or used in other ActionWithArguments. 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 $|X - X_i|$ 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

11.8.5 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 stochastically back and forth (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 analyse 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 sinec 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 gullwing 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 configurational 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?

11.8.6 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 mimick 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 mimick also the torsion with sub-Angstrom accuracy.

11.8.7 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 suddedly 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.

11.8.8 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 whith 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 .

11.8.9 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 woward 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 follwing:

```
# 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 confortable 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 intuituve 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 torsions, are periodic and their codomain 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 highlt resemblant and representative of what hapens 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).

11.8.10 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 reindexed from 1. This is also the case when you select a subensemble 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 PDBs 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.

11.8.11 Fast forward: metadynamics on the path

This section is actually set a bit foward 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 higly 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 flucutates 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 [45] 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 heigh in the welltempered 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 compenstates 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 actually resemblant of 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?

11.9 Belfast tutorial: Adaptive variables II

11.9.1 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 analysing 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.

11.9.2 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

11.9.3 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 conjuction with the MOLINFO command

11.9.4 Instructions

11.9.4.1 Visualising 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 trajetories mystified!

11.9.4.2 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.

11.9.4.3 Finding collective variables

Right so lets come up with some CVs to analyse 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 Wikkepedia 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 visualise 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?

11.9.4.4 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 bussiness 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 transtion 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 visualise 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?

11.9.5 Extensions

As discussed in the previous section this approach to trajectory analysis works by calcalating 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 defintion 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

11.9.6 Further Reading

There is a growing community of people using these ideas to analyse 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

11.10 Belfast tutorial: Umbrella sampling

11.10.1 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.

11.10.2 Summary of theory

11.10.2.1 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.

11.10.2.2 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 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.

11.10.2.3 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:

$$P_i(s) = \frac{P(s)e^{-\frac{V_i(s)}{k_BT}}}{\int ds' P(s')e^{-\frac{V_i(s')}{k_BT}}} = \frac{P(s)e^{-\frac{V_i(s)}{k_BT}}}{Z_i}$$

where

$$Z_i = \sum_q e^{-(U(q)+V_i(q))}$$

The likelyhood 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 likelyhood 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.

11.10.3 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.

11.10.4 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).

11.10.5 Instructions

11.10.5.1 The model system

We here use a a model system alanine dipeptide with CHARM27 all atom force field already seen in the previous section.

11.10.5.2 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 [80] . 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 Hooke's law:

$$U(s) = \frac{k}{2}(x - x_0)^2$$

.

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.

11.10.5.3 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 preprocessing, 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 $\exp(\frac{V}{k_B T}$ 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 $k_B T$ in kj/mol units.

Repeat this calculation for different values of AT. Does the result depend on AT?

11.10.5.4 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 kbT 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"
```

11.10.5.5 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.

11.10.6 Comments

11.10.6.1 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.

11.10.7 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

11.11 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 of draggings and can be used also to sample multiple events within a single simulation. Here I will explain the concepts of it and show some examples

11.11.1 Resources

Here is the tarball with the files referenced in the following .

11.11.2 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 freqeuncy 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 cvs 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 ATX, STEPX, KAPPAX where X is a incremental starting from 0 that assign the center and the harness of the spring at step STEPX. 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 in two different STEPs 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 sigmoids 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 follwing 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.

$$\begin{aligned}
W &= \int_0^{t_s} dt \, \frac{\partial H_\lambda(t)}{\partial t} \\
&= \int_0^{t_s} dt \, \frac{\partial H_\lambda(t)}{\partial \lambda}\frac{\partial \lambda(t)}{\partial t} + \int_0^{t_s} dt \, \frac{\partial H_\lambda(t)}{\partial k}\frac{\partial k}{\partial t} \\
&= \int_0^{t_s} -k(t)(s(X)-\lambda(t))\frac{\partial \lambda(t)}{\partial t}dt + \int_0^{t_s} \frac{(s(X)-\lambda(t))^2}{2}\frac{\partial k}{\partial t}dt \\
&= \int_0^{t_s} -k(t)(s(X)-\lambda(t))v dt + \int_0^{t_s} \frac{(s(X)-\lambda(t))^2}{2}\frac{\partial k}{\partial t}dt \\
&\simeq \sum_i -k(t_i)(s(X(t_i))-\lambda(t_i))\Delta\lambda(t_i) + \sum_i \frac{(s(X(t_i))-\lambda(t_i))^2}{2}\Delta k(t_i)
\end{aligned}$$

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.

11.11.3 Moving on a more complex path

Very often it is useful to use this movingrestraint 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 movingrestraint, 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 bewtween the arguments! This is a very common fault in writing the inputs.

By plotting the instataneous 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

11.11.4 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) being exponentiated. 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?

11.11.5 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 torsions 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 torsions are very fast to compute. Note that here the work always increase with time and never gets lower which is somewhat surprising if you tink 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.

11.12 Belfast tutorial: Metadynamics

11.12.1 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.

11.12.2 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) [42]. This potential is built as a sum of Gaussians 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, Gaussians 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 [44], 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 "biasfactor" 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 biasfactor 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 [83] [84] [85].

11.12.3 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

11.12.4 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, dihedrals 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, dihedrals phi and psi
- Exercise_5: run a well-tempered metadynamics simulation with 1 CV, dihedral psi

11.12.5 Instructions

11.12.5.1 The model system

Here we use as model system alanine dipeptide in vacuum with AMBER99SB-ILDN all-atom force field.

11.12.5.2 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 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 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 kJoule/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 Gaussians 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 biasfactor. 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 Gaussians 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.

11.12.5.3 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 Gaussians from the HILLS file and append the new information to both COLVAR and HILLS files.

11.12.5.4 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 Gaussians 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 Gaussians 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 Gaussians 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 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.

11.12.5.5 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 biasfactor and temperature of the simulation. For the first example, we will try a biasfactor 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 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 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 biasfactor 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 Gaussians 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 biasfactor and see the effect on the simulation. If we choose a biasfactor 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 biasfactor 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 biasfactor 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 biasfactor is sufficiently high for the free-energy barriers of the system under study to be crossed.

11.12.5.6 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 biasfactor 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 serie 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) Gaussians 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.

11.13 Belfast tutorial: Replica exchange I

11.13.1 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).

11.13.2 Summary of theory

In Replica Exchange Methods [89] (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

$$\Delta_{i,j}^{PT} = \left(\frac{1}{k_B T_i} - \frac{1}{k_B T_j}\right)(U(R_i) - U(R_j)),$$

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 [92]. 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 i-th and j-th 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) [10]. In this ensemble, the average energy remains close to the canonical value but its fluctuations are enhanced in a tunable way, thus improving sampling.
In the so-called PTMetaD-WTE scheme [93], each replica diffusion in temperature space is enhanced by the increased energy fluctuations at all temperatures.

11.13.3 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

11.13.4 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 subdirectory where topology and configuration files for Gromacs are stored.

11.13.5 Instructions

11.13.5.1 The model system

Here we use as model systems alanine dipeptide in vacuum and water with AMBER99SB-ILDN all-atom force field.

11.13.5.2 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 processess in parallel, using the topology files topol0.tpr and topol1.tpr stored in the TOPO subdirectory. 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.

11.13.5.3 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.

11.13.5.4 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.

11.13.5.5 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 serie 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 PLUM↩ ED 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 serie 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 [93]). 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
```

11.14 Belfast tutorial: Replica exchange II and Multiple walkers

11.14.1 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 analyse 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.

11.14.1.1 Learning Outcomes

Once this tutorial is completed students will:

- Know how to run a Bias-Exchange simulation using PLUMED and GROMACS
- Know how to analyse the results of BEMETA with the help of METAGUI
- Know how to run a Multiple Walker simulation

11.14.2 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

11.14.3 Instructions

11.14.3.1 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 COL↩ VARs are compatible with METAGUI
2. if you want to print additional information, like the METAD bias it is possibile 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.

11.14.3.2 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 incresing 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 weigheted 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:

11.14.3.3 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 multidensional free energy surfaces in $>$=N dimensions. A simple way to perform this analysis is to use METAGUI. METAGUI performs the following operations:

1. Clusterizes 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 approch 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 additionaly analysed). They MUST be named cv1 ... cvN.
2. the COLVAR.# files must be synchronised with the trajectories, this means that for each frame in the trajectory at time t there must be a line in each colvar at time t and viceversa. The best option is usually to analyse the trajectories a posteriori using plumed driver.

3. a keyword #! ACTIVE NBIASEDCV BIASEDCV LABEL is needed, where NBIASEDCV is the number of biased cv in that replica (not overall), BIASEDCV is the index of the biased cv 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 usufull 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! Synchronisation 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 visualise 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 visualised 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 sidechains 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 bottons 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 usefull:

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.

11.14.3.4 Multiple Walker Metadynamics

Multiple Walker metadynamics is the simplest way to parallelise a MetaD 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 usefull because it provides a parallelisation that does not need a parallel code. In this case the walkers read with a given frequency the gaussians deposited by the others and add them to their own METAD.

11.14.4 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. Biarnés, 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).

11.15 Belfast tutorial: NMR restraints

11.15.1 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.

11.15.1.1 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

11.15.2 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

11.15.3 Instructions

11.15.3.1 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/pseudocontact 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
```

11.15.3.2 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 withouth 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 backcalculated 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
```

11.15.4 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).

11.16 Belfast tutorial: Steinhardt Parameters

11.16.1 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 crystalisation 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.

11.16.1.1 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.

11.16.2 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

11.16.3 Instructions

11.16.3.1 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 posisitions 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.) Visualise 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?

11.16.3.2 Coordination Numbers

At the end of the previous section you were able to make very sophisticated judgements about whether or not the arrangment 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 simpled simulations described at the end of the previous section. Is the average coordination number good at differentiaing between solid and liquid configurations? Given your knowledge of physics/chemistry is this result surprising?

11.16.3.3 Steinhard 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 learnt 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 recognise 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?

11.16.3.4 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, unambiously 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 unambigously 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 unambigously 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?

11.16.3.5 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 ordred 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 arround a central atom. This quantity can be calcualted using plumed. If you have time try to use the manual to work out how.

11.16.4 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

11.17 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.

11.17.1 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 .

11.17.2 Exercise 1. Metadynamics

11.17.2.1 Resources

The tarball for this project contains all the files needed to run this exercise.

11.17.2.2 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) [42]. This potential is built as a sum of Gaussians 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, Gaussians 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 [44], 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 "biasfactor" 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 biasfactor 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 [83] [84] [85].

11.17.2.3 Setup, run, and analyse 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 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 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 kJoule/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 Gaussians 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 biasfactor 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 Gaussians 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.

11.17.2.4 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 Gaussians 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 Gaussians 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 Gaussians 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.

11.17.3 Exercise 2. Bias-Exchange Metadynamics

11.17.3.1 Resources

The `tarball` for this project contains all the files needed to run this exercise.

11.17.3.2 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 Hamiltionan 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.

11.17.3.3 Setup, run, and analyse 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 BE-WTMetaD 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 BE-MetaD, 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 usefull a posteriori, they all gain from the good ones and nothing is lost.

11.17.3.4 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 Gaussians 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.

11.17.4 Exercise 3. Replica-Average Metadynamics

11.17.4.1 Resources

The `tarball` for this project contains all the files needed to run this exercise.

11.17.4.2 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 [94] [95] [96] [97] .

11.17.4.3 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
```

11.17.4.4 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 cvs 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 preselected 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 monodimensional free energies, this will tipically 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.

11.18 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.

11.18.1 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 subdirectories, one per exercise.

11.18.2 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 .

11.18.3 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.

11.18.3.1 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 timesteps 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 `topolB.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.

11.18.3.2 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 snapshop every 1000 timesteps.

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 serie of the gyration radius.

11.18.4 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.

11.18.4.1 Metadynamics

11.18.4.1.1 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) [42]. This potential is built as a sum of Gaussians 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, Gaussians 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 [44], 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 "biasfactor" 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 biasfactor 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 [83] [84] [85].

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.

11.18.4.1.2 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 kJoule/mol, biasfactor 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 kJoule/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 Gaussians 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 biasfactor. 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 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 Gaussians 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 Gaussians 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 Gaussians 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 Gaussians 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.

11.18.4.1.3 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 kJoule/mol, biasfactor 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.

11.18.4.2 Restraints

11.18.4.2.1 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.

11.18.4.2.2 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 COMP↩ ONENTS keyword). Printing the bias allows one to know how much a given snapshop 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.

11.18.4.3 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.

11.18.4.4 Using multiple restraints with replica exchange

11.18.4.4.1 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 likelyhood 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 likelyhood 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.

11.18.4.4.2 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.

11.18.4.4.3 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 continous 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 continous 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.

11.19 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 [91]. 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.

11.19.1 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 [90].

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.

11.19.2 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
```

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 (hardcode 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.
- 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.

11.20 Julich tutorial: Developing CVs in plumed

11.20.1 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 visualised 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.

11.20.2 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` .

11.20.3 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:
```

11.20.4 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 PLUMED's nicest features 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 specialised 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 inheritted. This process is what we will try and take you through in this short tutorial.

11.20.5 Instructions

11.20.5.1 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 indespensible as well.

11.20.5.2 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 repsectively. 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.

11.20.6 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 workflow 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 realise 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.

11.21 Lugano tutorial: Analyzing CVs

11.21.1 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 analyse a trajectory by calculating ensemble averages, histograms and free energy surfaces.

11.21.2 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.

11.21.3 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 tuturial will use this trajectory.
- template.pdb : a single frame from the trajectory that can be used in conjuction 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

11.21.4 Instructions

PLUMED2 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 PLUMED2 to perform a variety of different analyses on the fly and to bias the sampling in the molecular dynamics engine. PLUMED2 can also, however, be used as a standalone code for analysing trajectories. If you are using the code in this way you can, once PLUMED2 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 PLUMED2 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 PLUMED2 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 PLUMED2 to analyse 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 PLUMED2 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.

11.21.4.1 PLUMED2's 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.

11.21.4.2 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 analyse 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 analyse 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 PLUMED2's 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 favourite 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.

11.21.4.3 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 PLUMED2 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.

11.21.4.4 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 analysing 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 analysing 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 behaviour 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 summarise what we have learnt 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.

11.21.4.5 Calculating torsions

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 libraray that was used to gather all the various implementations people had written for different collective variables (CVs) that people had used to "analyse" trajectories over the years (analyse 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 analyse 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 analyse the trajectory you downloaded at the start of this exercise by using the commands descibred 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 torsions of a protein you often wish to calculate particular, named torsions (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 6th residue of the protein and the ψ angle in the 8th residue of the protein. You can do so using the following input:

```
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 analyse the trajectory you downloaded at the start of this exercise by using the commands descibred 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.

11.21.4.6 An exercise with the radius of gyration

Lets now see if you can use everything you have learnt 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.

11.21.4.7 Coordination numbers

In the previous sections we have learnt 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 analyse 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 analyse 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?

11.21.4.8 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 analyse 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 infrastracture of multicolvar has been used to develop many PLUMED2 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.

```
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.

11.21.4.9 Understanding the need for ensemble averages

In the previous sections we have learnt 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 analyse 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 analysing 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

Regardlessly, 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 consisitent and certainly much more consisitent than the set of instantaneous CV values.

11.21.4.10 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?

11.21.4.11 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
hh: HISTOGRAM ARG=d1 BANDWIDTH=0.05 GRID_MIN=0 GRID_MAX=4.0 GRID_BIN=200 STRIDE=10
DUMPGRID GRID=hh 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 hh 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 hh 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 learnt in this tutorial in order to analyse the protein trajectory that was downloaded at the start of the exercise.

11.21.4.12 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 claculate perform this analysis is shown below:

```
# 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 analysed
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 visualise 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:

```
# 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 analysed
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.

11.21.5 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 analyse the protein trajectory that you have worked on in this tutorial.

11.22 Lugano tutorial: Path CVs

11.22.1 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 constrast 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 visualise what we are doing. Hopefully, however, you will be able to use what you learn from this tutorial to generalise these ideas so that you can use PATH and PCAVARS in higher-dimensional spaces.

11.22.2 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.

11.22.3 Resources

The tarball for this project contains the following files:

- transformation.pdb : a trajectory that shows the transition between the C7ax and C7eq conformers of alanine dipeptide.
- pca-reference.pdb : a file that gives the start and end points of the vector that connects the C7ax and C7eq conformers. This file contains the positions of the atoms in these two structures.
- PCA-isocommittor : a directory containing the files required to run isocommitor simulations that monitor the values of ϕ, ψ and the PCA coordinate.
- PATH-isocommittor : a directory containing the files required to run isocommitor simulations that monitor the values of the PATH collective variable $S(X)$.
- 2CV-isocommittor : a directory containing the files required to run isocommitor simulations that monitor the values of the PATH collective variables $S(X)$ and $Z(X)$

11.22.4 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 overstudied molecule!). It is well known that you can understand the interconversion 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.

11.22.4.1 PCA coordinates

Consider the free energy surface shown in figure lugano-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 lugano-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 C7eq 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 C7ax configuration from a thermal fluctuations. Unsurprinsgly then it has been shown that metadynamics simulations that use just the ϕ angle as a collective variable can effectively drive the system from the C7ax configuration to the C7eq 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 C7ax configuration to the C7eq 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 C7ax configuration to the C7eq 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 C7ax and C7eq 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 C7eq configuration are (ϕ_1, ψ_1) and the ramachandran angles in the C7ax 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 C7eq state to the C7ax 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 trignometry 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
pca: PCAVARS REFERENCE=angle-pca-reference.pdb TYPE=EUCLIDEAN
PRINT ARG=t1,t2,pca.* FILE=colvar
```

We cannot, however, do this in practise (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.

11.22.4.2 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 C7eq and C7ax 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 C7eq configuration and $\mathbf{x}^{(2)}$ for the C7ax configuration and if the instanenous 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
pca: PCAVARS REFERENCE=pca-reference.pdb TYPE=OPTIMAL
PRINT ARG=t1,t2,pca.* FILE=colvar
```

Use this input to analyse 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 C7ax configuration shown in figure lugano-2-transition-fig to the C7eq configuration shown in figure lugano-2-transition-fig comes from the translation of all the atoms. To put this another way if I had, in figure lugano-2-transition-fig, shown two images of the C7ax 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 lugano-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 practise 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.

11.22.4.3 The isocommitor 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 isocommitor. The essential notion behind this techinique is that there will be a saddle point between the two states of interest (the C7ax and C7eq 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 isocomittor 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 isocomittors 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 simualtions. 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 C7eq and C7ax configurations of alanine dipeptide. In addition, this command also generates some scripts that allow us to visualise how the ϕ, ψ and PCA coordinates that we introduced in the previous sections change duing 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 C7eq basin from the trajectories that moved to the C7ax 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 C7eq from the trajectories that moved to C7ax. It is abundantly clear, however, that the number of trajectories that moved to C7eq is not equal to the number of trajectories that moved to C7ax. 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 C7eq state to the C7ax 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.

11.22.4.4 Path collective variables

Consider the black path that connects the C7ax and C7eq 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 isocomittor analsysis 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. Furhthermore, 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 C7ax from those that moved to C7eq. 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.

11.22.4.5 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 C7eq and C7ax. 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 C7eq and C7ax that we used in the previous sections when talking about PCAVARS. Each of these configurations lies along the path connecting C7eq and C7ax 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 $|X - X_i|$ is the distance between the instaneous coordinate of the system, X, in the high-dimensional space and X_i is the coordinate of the ith waymark 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:

11.22.4.6 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 isocomittor. 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-isocommitor. 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. Furhthermore, 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 lugano-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 waymarkers. 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 C7eq or C7ax 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 traejctory 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 waymarkers to define the PATH. Another flexibility comes, however, when you recognise that you can also change the way in which the distance, $|X - X_i|$, is calculated in the two formulas above. For example this distance can be calculated using the RMSD distance or it can be calculated by mesauring 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.

11.22.4.7 Optimising 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 waymarkers 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 ehnanced 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 set of spacings between all 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 $|X_j - X_i|$ 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 gullwing shape in lugano-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 neighbouring 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:

$$\lambda = \frac{2.3(N-1)}{\sum_{i=1}^{N-1} |X_i - X_{i+1}|}$$

11.22.5 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 analyse 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 isocommitor surfaces and the transition state ensemble you should read up on the transition path sampling method.

11.23 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.

11.23.1 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
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 [48] .

11.23.2 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
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.

11.23.3 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
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.

11.23.4 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 the personalization of 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.

11.23.5 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 syntaxes 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 WALKE↩ RS_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

11.24 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 analsys scripts are provided in this TARBALL .

Users are expected to write PLUMED input files based on the instructions below.

11.24.1 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 .

11.24.2 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
# 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.

11.24.2.1 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 timesteps 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.

11.24.2.1.1 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
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.

11.24.2.2 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
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 snapshop every 1000 timesteps.

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
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.

11.24.2.3 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 containes 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 trajecories 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 preceeding 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
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
MOLINFO STRUCTURE=../TOPO/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.

11.24.2.4 Other analysis tools

PLUMED also allows you to make some analyis on the collective variables you are calculating. For example, you can compute a histogram with an input like this one

```
BEGIN_PLUMED_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
PRINT ARG=phi,psi,gyr FILE=analyze
HISTOGRAM ...
  ARG=gyr
  USE_ALL_DATA
  KERNEL=discrete
  GRID_MIN=0
  GRID_MAX=1
  GRID_BIN=50
  GRID_WFILE=histogram
... 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 Analsys (PLUMED). This graphical interface can also be used to quickly build PLUMED input files based on template lines.

11.24.3 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.

11.24.3.1 Metadynamics

11.24.3.1.1 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) [42]. This potential is built as a sum of Gaussians 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, Gaussians 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 [44], 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 "biasfactor" 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 biasfactor 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 [83] [84] [85].

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.

11.24.3.1.2 Exercise 1

Now run a metadynamics simulation with the following input

```
BEGIN_PLUMED_FILE
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 biasfactor 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 biasfactor is $\gamma \approx \frac{\Delta G}{2 k_B T}$.

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.

11.24.3.1.3 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
# 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 kJoule/mol, biasfactor 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 kJoule/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 Gaussians 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 biasfactor. 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 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 Gaussians 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 Gaussians 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 Gaussians 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 Gaussians 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.

11.24.3.1.4 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
# 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 kJoule/mol, biasfactor 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.

11.24.3.2 Restraints

11.24.3.2.1 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.

11.24.3.2.2 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 $k_B T$ 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 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

```
BEGIN_PLUMED_FILE
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 snapshop 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.

11.24.3.3 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
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 AT0=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
```

11.24.3.4 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.

11.24.3.5 Using multiple restraints with replica exchange

11.24.3.5.1 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 likelyhood 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 likelyhood 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.

11.24.3.5.2 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.

11.24.3.5.3 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 continous 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 continous 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.

PLUMED makes use of MPI and OpenMP to parallelise some of its functions, try to always compile it with these features enabled. Furthermore, newer compilers with proper optimisation flags can provide a drammatic boost to performances.

PLUMED collects atoms from an external code and sends back forces, so it is key to minimise the effect of P↩ LUMED 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 comunication 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.

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 varing the cut-off for short-range interactions GROMACS can optimise 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 optimised 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

12.2 Metadynamics

Metadynamics can be sped up significantly using grids, which are activated setting the GRID_MIN and GRID_MAX keywords of METAD. 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

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
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
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 [98]. See also EFFECTIVE_ENERGY_DRIFT.

12.3.1 EFFECTIVE_ENERGY_DRIFT

	This is part of the generic module

Print the effective energy drift described in Ref [98]

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

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
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
d: DISTANCE ATOMS=10,20
RESTRAINT ARG=d KAPPA=100000 AT=0.6
EFFECTIVE_ENERGY_DRIFT PRINT_STRIDE=100 FILE=eff
```

12.4 Multicolvar

Whenever you have a multicolvar action such as:

```
BEGIN_PLUMED_FILE
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 collosal 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 lage 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 Neighbour Lists

Collective variables that can be speed up making us of neighbour lists:

- COORDINATION
- DHENERGY
- PATHMSD

By tuning the cut-off for the neighbour 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.

Warning

Notice that the calculation of the neighbour list is not not parallelized for COORDINATION and DHENERGY. As a consequence, if you run with many processors and/or OpenMP threads, the neighbor list might even make the calculation slower.

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_NUM_THREADS to the number of threads you wish to use with PLUMED. By default (if PLUMED_NUM_THREADS is unset) openmp will be disabled at runtime. E.g., to run with gromacs you should do:

```
export PLUMED_NUM_THREADS=8
mdrun -plumed
```

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 roundoff 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 opemMP (currently, COORDINATION and DHENERGY).
- You might want to tune also the environmental variable PLUMED_CACHELINE_SIZE, by default 512, to set the size of cachelines 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 particulary demanding if you want to calculate them for all the residues of a protein. This is particularty 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. 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 somenthing else)

To have an accurate timing of your input you can use the DEBUG DETAILED_TIMERS.

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.
ANGLES	MCOLVAR	Calculate functions of the distribution of angles .
ANGLE	COLVAR	Calculate an angle.
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_CUSTOM	VES_BASISF	Basis functions given by arbitrary mathematical expressions.
BF_FOURIER	VES_BASISF	Fourier 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.
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
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_OF_MULTICOLVAR	VATOM	Calculate a a weighted average position based on the value of some multicolvar.
CENTER	VATOM	Calculate the center for a group of atoms, with arbitrary weights.
CLASSICAL_MDS	DIMRED	Create a low-dimensional projection of a trajectory using the classical multidimensionalscaling 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.
COLUMNSUMS	MATRIXF	Sum the columns of a contact matrix
COMBINE	FUNCTION	Calculate a polynomial combination of a set of other variables.
COMMITTOR	PRINTANALYSIS	Does a committor analysis.
COM	VATOM	Calculate the center of mass for a group of atoms.
config	TOOLS	inquire plumed about how it was configure
CONSTANT	COLVAR	Return one or more constant quantitieswith 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 $F(x) = -k_B T \ln H(x)$.
COORDINATIONNUMBER	MCOLVAR	Calculate the coordination numbers of atoms so that you can then calculate functions of the distribution ofcoordination numbers such as the minimum, the number less than a certain quantity and so on.
COORDINATION	COLVAR	Calculate coordination numbers.
CS2BACKBONE	ISDB_COLVAR	Calculates the backbone chemical shifts for a protein.
CUSTOM	FUNCTION	An alias to the MATHEVAL function.
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 calculatethe 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 GROUPB.
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_FROM_CONTOUR	COLVAR	Calculate the perpendicular distance from a Willard-Chandler dividing surface.
DISTANCES	MCOLVAR	Calculate the distances between one or many pairs of atoms. You can then calculate functions of the distribution ofdistances such as the minimum, the number less than a certain quantity and so on.
DISTANCE	COLVAR	Calculate the distance between a pair of atoms.
driver-float	TOOLS	Equivalent to driver, but using single precision reals.
driver	TOOLS	driver is a tool that allows one to to use plumed to post-process an existing trajectory.
DRMSD	DCOLVAR	Calculate the distance RMSD with respect to a reference structure.
DRR	EABFMOD_BIAS	Used to performed extended-system adaptive biasing force(eABF) [34] methodon one or more collective variables. This method is alsocalled dynamic reference restraining(DRR) [35] .

drr_tool	EABFMOD_TOOLS	- Extract .grad and .count files from the binary output .drrstate - Merge windows
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 connnectivity 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).
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 described in Ref [98]
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 energy of the simulation box.
ENSEMBLE	FUNCTION	Calculates the replica averaging of a collective variable over multiple replicas.
ERMSD	COLVAR	Calculate eRMSD with respect to a reference structure.
EXTENDED_LAGRANGIAN	BIAS	Add extended Lagrangian.
EXTERNAL	BIAS	Calculate a restraint that is defined on a grid that is read during start up
FAKE	COLVAR	This is a fake colvar container used by cltools or various other actionsand just support input and period definition
FCCUBIC	MCOLVAR	Measure how similar the environment around atoms is to that found in a FCC structure.
FIND_CONTOUR_SURFACE	GRIDANALYSIS	Find an isocontour by searching along either the x, y or z direction.
FIND_CONTOUR	GRIDANALYSIS	Find an isocontour in a smooth function.
FIND_SPHERICAL_CONTOUR	GRIDANALYSIS	Find an isocontour in a three dimensional grid by searching over a Fibonacci sphere.
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:
FUNCPATHMSD	FUNCTION	This function calculates path collective variables.
FUNCSUMHILLS	FUNCTION	This function is intended to be called by the command line tool sum_hillsand it is meant to integrate a HILLS file or an HILLS file interpreted asa histogram i a variety of ways. Therefore it is not expected that you use thisduring your dynamics (it will crash!)
gentemplate	TOOLS	gentemplate is a tool that you can use to construct template inputs for the various-actions
GHOST	VATOM	Calculate the absolute position of a ghost atom with fixed coordinatesin the local reference frame formed by three atoms.The computed ghost atom is stored as a virtual atom that can be accessed inan atom list through the the label for the GHOST action that creates it.
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 labelin 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.
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.
info	TOOLS	This tool allows you to obtain information about your plumed version
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 torsions 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.
kt	TOOLS	Print out the value of k_BT at a particular temperature
LOAD	GENERIC	Loads a library, possibly defining new actions.
LOCAL_AVERAGE	MCOLVARF	Calculate averages over spherical regions centered on atoms
LOCALENSEMBLE	FUNCTION	Calculates the average over multiple arguments.
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 vectoron 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 vectoron 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 vectoron the atoms in the first coordination sphere.
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
manual	TOOLS	manual is a tool that you can use to construct the manual page fora particular action
MATHEVAL	FUNCTION	Calculate a combination of variables using a matheval expression.

MAXENT	BIAS	Add a linear biasing potential on one or more variables $f_i(x)$ satisfying the maximum entropy principle as proposed in Ref. [33] .
MCOLV_COMBINE	MCOLVARF	Calculate linear combinations of multiple multicolvars
MCOLV_PRODUCT	MCOLVARF	Calculate a product of multiple multicolvars
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 multicolvarso 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 multicolvarso 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 multicolvarso that one can compute the mean and so on for only those multicolvars more than a tolerance.
mklib	TOOLS	compile a .cpp file into a shared library
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 useed to transform the colvar values calculated by a multicolvar using a histogrambead
MTRANSFORM_LESS	MTRANSFORMS	This action can be useed to transform the colvar values calculated by a multicolvar using a switchingfunction
MTRANSFORM_MORE	MTRANSFORMS	This action can be useed to transform the colvar values calculated by a multicolvar using one minus a switchingfunction
MULTICOLVARDENS	GRIDCALC	Evaluate the average value of a multicolvar on a grid.
MULTI_RMSD	DCOLVAR	An alias to the MULTI-RMSD function.
MULTI-RMSD	DCOLVAR	Calculate the RMSD distance moved by a number of separated domains from their positions in a reference structure.
newcv	TOOLS	create a new collective variable from a template
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 atomsor ambiguous NOE.
OPT_AVERAGED_SGD	VES_OPTIMIZER	Averaged stochastic gradient decent with fixed step size.
OPT_DUMMY	VES_OPTIMIZER	Dummy optimizer for debugging.
OUTPUT_CLUSTER	CONCOMP	Output the indices of the atoms in one of the clusters identified by a clustering object
PARABETARMSD	COLVAR	Probe the parallel beta sheet content of your protein structure.
partial_tempering	TOOLS	scale parameters in a gromacs topology to implement solute or partial tempering
patch	TOOLS	patch an MD engine
PATHMSD	COLVAR	This Colvar calculates path collective variables.
PATH	COLVAR	Path collective variables with a more flexible framework for the distance metric being used.
pathtools	TOOLS	pathtools can be used to construct paths from pdb data
PBMETAD	BIAS	Used to performed Parallel Bias Meta↩ Dynamics.
PCARMSD	DCOLVAR	Calculate the PCA components (see [18] and [19]) for a number of provided eigenvectors and an average structure. 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)
PCA	DIMRED	Perform principal component analysis (P↩ CA) using either the positions of the atoms a large number of collective variables as input.
PCAVARS	COLVAR	Projection on principal component eigenvectors or other high dimensional linear subspace
PCS	ISDB_COLVAR	Calculates the Pseudocontact shift of a nucleus determined by the presence of a metal ion susceptible to anisotropic magnetization.
pesmd	TOOLS	Pesmd allows one to do (biased) Langevin dynamics on a two-dimensional potential energy surface.
PIECEWISE	FUNCTION	Compute a piecewise straight line through its arguments that passes througha set of ordered control points.
PLANES	MCOLVAR	Calculate the plane perpendicular to two vectors in order to represent the orientation of a planar molecule.
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 Paramegnetic Resonance Enhancement intensity ratio between a spinlabel atom and a list of atoms .
PRINT	PRINTANALYSIS	Print quantities to a file.
PROPERTYMAP	COLVAR	Calculate generic property maps.
PUCKERING	COLVAR	Calculate sugar pseudorotation coordinates.
Q3	MCOLVAR	Calculate 3rd order Steinhardt parameters.
Q4	MCOLVAR	Calculate 4th order Steinhardt parameters.
Q6	MCOLVAR	Calculate 6th 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.
RESCALE	ISDB_BIAS	Rescales the value of an another action, being a Collective Variable or a Bias.
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	REWEIGHTING	Calculate weights for ensemble averages allow for the computing of ensemble averages at temperatures lower/higher than that used in your original simulation.
RMSD	DCOLVAR	Calculate the RMSD with respect to a reference structure.
ROWSUMS	MATRIXF	Sum the rows of a adjacency matrix.
SAXS	ISDB_COLVAR	Calculates SAXS scattered intensity using the Debye equation.
SELECTOR	ISDB_GENERIC	Defines a variable (of the type double) inside the PLUMED code that can be used and modified by other actions.
SELECT	ISDB_FUNCTION	Selects an argument based on the value of a SELECTOR.
SIMPLECUBIC	MCOLVAR	Calculate whether or not the coordination spheres of atoms are arranged as they would be in a simplecubic structure.
simplemd	TOOLS	simplemd allows one to do molecular dynamics on systems of Lennard-Jones atoms.
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.
SMAC	MCOLVARF	Calculate a variant on the SMAC collective variable discussed in [23]

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.In particular it calculates and store as components the sum of the squared deviations, the correlation, theslope and the intercept of a linear fit.
sum_hills	TOOLS	sum_hills is a tool that allows one to to use plumed to post-process an existing hills/colvar file
TARGET	DCOLVAR	This function measures the pythagorean distance from a particular structure measured in the space defined by someset of collective variables.
TD_CHISQUARED	VES_TARGETDIST	Chi-squared distribution (static).
TD_CHI	VES_TARGETDIST	Chi distribution (static).
TD_CUSTOM	VES_TARGETDIST	Target distribution given by an arbitrary mathematical expression (static or dynamic).
TD_EXPONENTIALLY_MODIFIED_GAUSSIAN	VES_TARGETDIST	Target distribution given by a sum of exponentially modified Gaussian distributions (static).
TD_EXPONENTIAL	VES_TARGETDIST	Exponential distribution (static).
TD_GAUSSIAN	VES_TARGETDIST	Target distribution given by a sum of Gaussians (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_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 productof 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.
TETRAHEDRALPORE	VOLUMES	This quantity can be used to calculate functions of the distribution of collective variablesfor the atoms lie that lie in a box defined by the positions of four atoms at the corners of a tetrahedron.

TETRAHEDRAL	MCOLVAR	Calculate the degree to which the environment about ions has a tetrahedral order.
TIME	GENERIC	retrieve the time of the simulation to be used elsewere
TOPOLOGY_MATRIX	MATRIX	Adjacency matrix in which two atoms are adjacent if they are connected topologically
TORSIONS	MCOLVAR	Calculate whether or not a set of torsional angles are within a particular range.
TORSION	COLVAR	Calculate a torsional angle.
UNITS	GENERIC	This command sets the internal units for the code. A new unit can be set by either-specifying how to convert from the plumed default unit into that new unit or by us-ingthe shortcuts described below. This directive MUST appear at the BEGINNING of theplumed.dat file. The same units must be used througout the plumed.dat file.
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_LINEAR_EXPANSION	VES_BIAS	Linear basis set expansion bias.
ves_md_linearexpansion	VES_TOOLS	Simple MD code for dynamics on a potential energy surface given by a linear basis set expansion.
VES_OUTPUT_BASISFUNCTIONS	VES_UTILS	Output basis functions to file.
VES_OUTPUT_FES	VES_UTILS	Tool to output biases and FESs for VES biases from previously obtained coefficients.
VES_OUTPUT_TARGET_DISTRIBUTION	VES_UTILS	Output target distribution to file.
vim2html	TOOLS	convert plumed input file to colored html using vim syntax
VOLUME	COLVAR	Calculate the volume of the simulation box.
WHOLEMOLECULES	GENERIC	This action is used to rebuild molecules that can become split by the periodicboundary 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 ofvalues 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.You can then calculate functions of the distribution ofvalues such as the minimum, the number less than a certain quantity and so on.

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.You can then calculate functions of the distribution ofvalues such as the minimum, the number less than a certain quantity and so on.
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 ofvalues 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.You can then calculate functions of the distribution ofvalues such as the minimum, the number less than a certain quantity and so on.
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 ofvalues 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.

Chapter 14

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

Acceptance for replica exchange when ENERGY is biased is computed correctly only of all the replicas has the same potential energy function. This is for instance not true when using GROMACS with lambda replica exchange of with plumed-hrex branch.

Page lammps-6Apr13

The provided patch does not pass the virial correctly to PLUMED (see https://github.com/plumed/plumed2/issues/377). If you need to perform constant pressure simulations you should download a LAMMPS version newer than November 2018, where the mentioned bug has been fixed, that natively supports PLUMED 2.4.

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 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 explicitally 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.

Page PUCKERING

The 6-membered ring implementation distributed with this version of PLUMED leads to qx and qy values that have an opposite sign with respect to those originally defined in [16]. The bug will be fixed in a later version but is here kept to preserve reproducibility.

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, 396, 400, 517, 618, 626

[4] Grisell Díaz Leines and Bernd Ensing. Path finding on high-dimensional free energy landscapes. *Phys. Rev. Lett.*, 109:020601, Feb 2012. 26, 71

[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. 71, 106, 107, 134

[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. 74, 78, 103

[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. 84

[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. 86

[9] 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. 97

[10] M. Bonomi and M. Parrinello. Enhanced sampling in the well-tempered ensemble. *Phys. Rev. Lett.*, 104:190601, 2010. 97, 671

[11] 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. 98

[12] Vojtech Spiwok and Blanka Králová. Metadynamics in the conformational space nonlinearly dimensionally reduced by Isomap. *Journal of Chemical Physics*, 135(22):224504, 2011. 100, 114

[13] 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. 102

[14] Grisell Díaz Leines and Bernd Ensing. Path finding on high-dimensional free energy landscapes. *Phys. Rev. Lett.*, 109:020601, 2012. 108

[15] 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. 116

[16] D t Cremer and JA Pople. General definition of ring puckering coordinates. *Journal of the American Chemical Society*, 97(6):1354–1358, 1975. 116, 779

[17] 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. 116

[18] 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. 120, 125, 774

[19] 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. 120, 125, 774

[20] S. K. Kearsley. On the orthogonal transformation used for structural comparison. *Acta Cryst. A*, 45:208–210, 1989. 127

[21] 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. 140

[22] 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. 147, 264, 267

[23] 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. 148, 289, 775

[24] 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. 149

[25] 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. 155, 413, 453, 454

[26] 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. 163

[27] Bingqing Cheng, Gareth A. Tribello, and Michele Ceriotti. Solid-liquid interfacial free energy out of equilibrium. *Phys. Rev. B*, 92:180102, 2015. 163

[28] Federico Giberti, Gareth A. Tribello, and Michele Parrinello. Transient polymorphism in nacl. *Journal of Chemical Theory and Computation*, 9(2526-2530):null, 2013. 261, 262

[29] 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. 307, 309, 310, 319, 320, 328, 330, 331, 334

[30] 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. 315

[31] 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. 354

[32] Adam P. Willard and David Chandler. Instantaneous liquid interfaces. *The Journal of Physical Chemistry B*, 114(5):1954–1958, 2010. 368, 370

[33] 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. 383, 392, 773

[34] 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. 383, 413, 457, 769

[35] 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. 383, 457, 769

[36] 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. 384

[37] 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. 384

[38] 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. 384

[39] 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. 387

[40] 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. 387

[41] 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. 387

[42] A. Laio and M. Parrinello. Escaping free energy minima. *Proc. Natl. Acad. Sci. USA*, 99:12562–12566, 2002. 395, 405, 513, 614, 664, 691, 702, 748

[43] V. Babin, C. Roland, and C. Sagui. Adaptively biased molecular dynamics for free energy calculations. *J. Chem. Phys.*, 128:134101, 2008. 395

[44] 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. 395, 405, 498, 614, 627, 664, 691, 702, 749

[45] D Branduardi, G Bussi, and M PARRINELLO. Metadynamics with adaptive Gaussians. *J. Chem. Theory Comput.*, 8(7):2247–2254, 2012. 395, 406, 618, 626, 646

[46] 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. 396, 406

[47] 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. 396, 399, 406

[48] 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. 396, 627, 740

[49] Pratyush Tiwary and Michele Parrinello. From metadynamics to dynamics. *Phys. Rev. Lett.*, 111:230602, 2013. 400

[50] 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. 400

[51] 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. 400

[52] Alejandro Gil-Ley and Giovanni Bussi. Empirical corrections to the amber rna force field with target metadynamics, 2016. 400

[53] James F Dama, Michele Parrinello, and Gregory A Voth. Well-tempered metadynamics converges asymptotically. *Phys. Rev. Lett.*, 112(24):240602, 2014. 400

[54] H. Grubmüller, B. A. Heymann, and P. Tavan. *Science*, 271:997–999, 1996. 401

[55] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, 78:2690–2693, 1997. 401

[56] 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. 405

[57] Massimiliano Bonomi and Carlo Camilloni. Integrative structural and dynamical biology with PLUMED-ISDB. *Bioinformatics*, 33:3999–4000, 2017. 413

[58] 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. 413, 453

[59] 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. 413, 457

[60] 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. 413, 458

[61] Omar Valsson and Michele Parrinello. Variational approach to enhanced sampling and free energy calculations. *Phys. Rev. Lett.*, 113(9):090601, 2014. 413, 461, 520

[62] 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. 414

[63] 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. 414

[64] 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. 414, 417

[65] 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. 414, 418

[66] 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. 414, 418

[67] Massimiliano Bonomi, Carlo Camilloni, Andrea Cavalli, and Michele Vendruscolo. Metainference: A Bayesian inference method for heterogeneous systems. *Science Advances*, 2(1):e1501177, 2016. 414, 418, 441, 448

[68] 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. 418

[69] 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. 427

[70] 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. 433

[71] Massimiliano Bonomi, Carlo Camilloni, and Michele Vendruscolo. Metadynamic metainference: Enhanced sampling of the metainference ensemble using metadynamics. *Sci. Rep.*, 6:31232, 2016. 441, 448, 453

[72] 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. 442, 448, 449, 450

[73] Massimiliano Bonomi, Gabriella T Heller, Carlo Camilloni, and Michele Vendruscolo. Principles of protein structural ensemble determination. *Curr. Opin. Struct. Biol.*, 42:106–116, 2017. 448

[74] James McCarty, Omar Valsson, Pratyush Tiwary, and Michele Parrinello. Variationally optimized free-energy flooding for rate calculation. *Phys. Rev. Lett.*, 115(7):070601, 2015. 463

[75] Omar Valsson and Michele Parrinello. Well-Tempered Variational Approach to Enhanced Sampling. *J. Chem. Theory Comput.*, 11(5):1996–2002, 2015. 498, 499, 526

[76] 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. 499, 520

[77] 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. 513

[78] 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. 513

[79] 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. 513

[80] 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. 524, 654

[81] 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. 525

[82] Stefano Piana and Alessandro Laio. A bias-exchange approach to protein folding. *J. Phys. Chem. B*, 111(17):4553–9, 2007. 568, 570, 623, 627

[83] 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. 615, 664, 691, 703, 749

[84] Alessandro Barducci, Massimiliano Bonomi, and Michele Parrinello. Metadynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(5):826–843, 2011. 615, 664, 691, 703, 749

[85] Ludovico Sutto, Simone Marsili, and Francesco Luigi Gervasio. New advances in metadynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(5):771–779, 2012. 615, 664, 691, 703, 749

[86] 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. 627

[87] Richard A Cunha and Giovanni Bussi. Unraveling mg2+–rna binding with atomistic molecular dynamics. *RNA*, 23(5):628–638, 2017. 627

[88] 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. 627

[89] Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314(1–2):141–151, November 1999. 627, 671

[90] 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. 627, 713

[91] Giovanni Bussi. Hamiltonian replica-exchange in gromacs: a flexible implementation. *Mol. Phys.*, 2013. DOI: 10.1080/00268976.2013.824126. 627, 712

[92] 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. 671

[93] 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. 671, 677

[94] 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. 697

[95] Carlo Camilloni, Andrea Cavalli, and Michele Vendruscolo. Replica-Averaged Metadynamics. *J. Chem. Theory Comput.*, 9(12):5610–5617, 2013. 697

[96] 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. 697

[97] 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. 697

[98] 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. 763, 770