

PLUMED User's Guide

*A portable plugin for free-energy calculations
with molecular dynamics*

Version 1.2 – May 2010

Contents

1	Introduction	4
1.1	What is PLUMED?	4
1.2	Supported codes	5
1.3	Features	5
1.4	New in version 1.2	6
1.5	Restrictions	7
1.6	The PLUMED package	7
1.7	Online resources	8
1.8	Credits	8
1.9	Citing PLUMED	9
1.10	License	9
2	Installation	10
2.1	Compiling PLUMED	10
2.1.1	Compiling the ACEMD plugin with PLUMED	13
2.2	Testing the installation	14
2.3	Back to the original code	15
3	Running free-energy simulations	16
3.1	How to activate PLUMED	16
3.2	The input file	18
3.3	A note on units	19
3.4	Metadynamics	20
3.4.1	Typical output	20
3.4.2	Bias potential	21
3.4.3	Well-tempered metadynamics	22
3.4.4	Restarting a metadynamics run	22
3.4.5	Using GRID	23

3.4.6	Multiple walkers	27
3.4.7	Monitoring a collective variable without biasing it	28
3.4.8	Inversion condition	29
3.5	Running in parallel	31
3.6	Replica exchange methods	32
3.6.1	Parallel tempering metadynamics	32
3.6.2	Bias exchange simulations	33
3.7	Umbrella sampling	33
3.8	Steered MD	34
3.8.1	Steerplan	35
3.9	Adiabatic Bias MD	37
3.10	External potentials	38
3.10.1	Walls	38
3.10.2	Tabulated potentials	38
3.11	Commitment analysis	41
4	Collective variables	42
4.1	Absolute position	44
4.2	Distance	45
4.3	Minimum distance	46
4.4	Angles	47
4.5	Torsion	48
4.6	Coordination number	48
4.7	Hydrogen bonds	49
4.8	Interfacial water	51
4.9	Radius of gyration	51
4.10	Dipole	53
4.11	Dihedral correlation	53
4.12	Alpha-beta similarity	54
4.13	Alpharmsd	54
4.14	Antibetarmsd	56
4.15	Parabetarmsd	57
4.16	Torsional RMSD	58
4.17	Electrostatic potential	58
4.18	Puckering coordinates	59
4.19	Path collective variables	60
4.19.1	Root mean square deviation	62
4.19.2	Distance root mean square deviation	63

4.19.3	Contact map distances	64
4.19.4	Using path variables as RMSD, DRMSD and CMAP and the TARGETED statement	67
4.20	Energy	68
4.21	Helix loops	69
4.22	A note on periodic boundary conditions	69
5	Postprocessing	71
5.1	Estimating the free energy after a metadynamics run	71
5.1.1	Installation instructions	71
5.1.2	Usage	71
5.2	Evaluating collective variables on MD trajectories	74
5.2.1	Installation instructions	74
5.2.2	Usage	74
5.3	Processing COLVAR files	76
5.4	PLUMED as a standalone program	77
5.4.1	Installation instructions	77
5.4.2	Usage	78

Chapter 1

Introduction

1.1 What is PLUMED?

PLUMED is a plugin for free-energy calculations in molecular systems. It works with some of the most popular classical molecular dynamics (MD) codes, such as GROMACS [1], NAMD [2], DL_POLY [3], LAMMPS [4] and the SANDER module in AMBER [5]. It also works with the very fast CUDA/GPU MD code ACEMD [6] and, more recently, it has been extended to work with *ab initio* MD codes, such as Quantum-ESPRESSO [7].

Free-energy calculations can be performed as a function of many order parameters, with a particular focus on biological problems, and using state-of-the-art methods such as metadynamics [8], umbrella sampling [9, 10, 11] and Jarzynski-equation based steered MD [12, 13].

Here is a brief outline of this guide:

- In this chapter we give an overview of the features and restrictions of the current release of PLUMED.
- In the second chapter we describe the procedure for installing the plugin and testing the correct installation.
- The third chapter explains how PLUMED can be used to perform free-energy calculations, setting up the input file and analyzing the output.
- The fourth chapter contains a list of collective variables (CVs) which are implemented and allow a wide variety of physical and chemical problems to be addressed.

- The fifth chapter is dedicated to postprocessing. It explains how to reconstruct the free-energy profile from the output of a metadynamics run and how to extract the CV values from MD trajectories.

1.2 Supported codes

PLUMED works as an add-on to some of the most popular MD codes: NAMD, GROMACS, SANDER, DL_POLY, Quantum-ESPRESSO, ACEMD and LAMMPS. These codes are not distributed with the PLUMED package, but they must be obtained separately.

NAMD 2.6/2.7b1/2.7b2

<http://www.ks.uiuc.edu/Research/namd/>

GROMACS 3.3/4.0

<http://www.gromacs.org/>

AMBER 9/10

<http://ambermd.org/>

DL_POLY 2.16/2.19/2.20

http://www.cse.scitech.ac.uk/ccg/software/DL_POLY/

Quantum-ESPRESSO 4.1.2

<http://www.quantum-espresso.org/>

ACEMD 1.1

<http://multiscalelab.org/acemd>

LAMMPS 15-Jan-2010

<http://lammps.sandia.gov>

Note that at the moment of this release of PLUMED, only those specific versions of the listed codes have been tested.

1.3 Features

PLUMED can perform several different types of calculation:

- Metadynamics with a large variety of CVs;

- Well-tempered metadynamics [14];
- Multiple walkers metadynamics [15];
- Combined parallel tempering and metadynamics [16];
- Bias-exchange metadynamics [17];
- Steered MD;
- Umbrella sampling;
- Commitment analysis.

1.4 New in version 1.2

PLUMED version 1.2 presents several new features, including new collective variables and support to new codes. Among these:

- Compatible with the parallel implementation of AMBER (sander module);
- Adiabatic Biased Molecular Dynamics [18];
- Redesigned Multiple Walkers, more robust on slow filesystems;
- Inversion condition to treat CV boundaries;
- External fixed potential acting on CVs read from file;
- Read and write metadynamics bias from/to file;
- New standalone utility to run PLUMED as an external tool;
- Steerplan for complex steered MD simulations;
- New format for COLVAR file, including tags, and a new utility to parse it;
- Optimization of code for mean-square-displacement;
- Option to add a constant force on CVs;

- New collective variables: Potential energy, similarity to ideal alpha helix and beta sheet structure, number of alpha-helix loops, distance from and projection onto a generic axis, difference between two distances;
- Added support to LAMMPS, Quantum-ESPRESSO 4.1.2, NAMD 2.7b2 and GROMACS 4.0.7.

A full list including also the bug fixed in the current release can be found in the CHANGES file distributed with the package.

1.5 Restrictions

The current release of PLUMED has a few restrictions:

- Parallel tempering plus metadynamics and bias-exchange metadynamics are available only in the GROMACS version;
- The patched version of GROMACS cannot perform hamiltonian (lambda) replica exchange;
- The `ENERGY` collective variable is available only for GROMACS4, AMBER and DL-POLY and cannot be used with multiple time step algorithm;
- Only orthorhombic cells are supported in ACEMD and DRIVER.
- Only orthorhombic and truncated octahedron cells are supported in AMBER.

1.6 The PLUMED package

The plugin package has the following directory structure:

- `common_files`. The directory containing all the basic routines.
- `tests`. A variety of examples of different CVs and free-energy methods provided with topology and input files for the supported codes. These examples, combined with a script adapted from CP2K [19], work also as a regtest for the plugin.

- **patches.** A collection of patches to interface PLUMED with different codes.
- **utilities.** A few small utilities:
 - **sum_hills** is a post-processing program which reads the **HILLS** file produced by the plugin in a metadynamics simulation and returns the free energy by summing the Gaussians that have been deposited.
 - **driver** is a tool that calculates the value of selected CVs along a MD trajectory. It requires a PDB file, a trajectory in DCD format and a file with the same syntax of the PLUMED input file.
 - **standalone** is a program to run PLUMED as a standalone code.
 - **plumedat.sh** is a shell script to extract information from PLUMED output files.
- **manual.** This manual.
- **ACEMD.** It contains the files needed to compile the plugin for ACEMD.

1.7 Online resources

You can find more information on the web:

<http://merlino.mi.infn.it/plumed>

For any questions, please subscribe to our mailing list:

plumed-users@googlegroups.com

1.8 Credits

PLUMED has been developed by Massimiliano Bonomi, Davide Branduardi, Giovanni Bussi, Carlo Camilloni, Davide Provati, Paolo Raiteri, Davide Donadio, Fabrizio Marinelli, Fabio Pietrucci, Francesco Luigi Gervasio and others. However, this work would not have been possible without the joint effort of many people. Among these, we would like to thank (in alphabetical

order): Alessandro Barducci, Anna Berteotti, Rosa Bulo, Matteo Ceccarelli, Michele Ceriotti, Paolo Elvati, Antonio Fortea-Rodriguez, Alessandro Laio, Matteo Masetti, Fawzi Mohamed, Ferenc Molnar, Gabriele Petraglio, Jim Pfaendtner and Federica Trudu. Francesco Marini is kindly acknowledged for his technical support and Joost VandeVondele for permission to use his regtest script.

Some **PLUMED** users have also contributed to the implementation of new features and the debugging of old bugs. Among these, we would like to thank: Toni Giorgino, Marcello Sega, Emmanuel Autieri, Gareth Tribello, Andrea Coletta, Ludovico Sutto, Layla Martin-Samos, Walter Rocchia and Alessio Lodola.

1.9 Citing PLUMED

You may wish to cite the following reference if you have utilized **PLUMED** in your work:

M. Bonomi, D. Branduardi, G. Bussi, C. Camilloni, D. Provasi, P. Raiteri, D. Donadio, F. Marinelli, F. Pietrucci, R.A. Broglia and M. Parrinello.

PLUMED: a portable plugin for free-energy calculations with molecular dynamics, *Comp. Phys. Comm.* 2009 vol. 180 (10) pp. 1961-1972.

1.10 License

PLUMED is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. **PLUMED** is distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of **MERCHANTABILITY** or **FITNESS FOR A PARTICULAR PURPOSE**. See the GNU Lesser General Public License for more detail. You should have received a copy of the GNU Lesser General Public License along with **PLUMED**. If not, see <http://www.gnu.org/licenses/>.

Chapter 2

Installation

The plugin installation requires the molecular dynamics code to be recompiled after the calls to the plugin routines have been inserted at the appropriate places in the original program. For a list of the supported molecular dynamics codes see Sec. 1.2.

All the basic plugin routines are contained in the `common_files` folder of the PLUMED distribution package. The insertions are automatically performed by a series of scripts provided in the `patches` directory.

In the following we will briefly describe the procedure for applying these patches to the different MD codes supported by PLUMED. We will refer to the root directory of the distribution package as `PLUMED_root`.

2.1 Compiling PLUMED

A similar procedure can be followed for all the supported codes except ACEMD. When code-specific procedures are needed, we state it explicitly. A different procedure is required for ACEMD (see below).

Different patches are available for different code versions. The name of the suitable patch is `plumedpatch.CODENAME.CODEVERSION.sh`. If the patch corresponding to the exact `CODEVERSION` that you are using is not available, choose the closest match. In the following, we shall indicate with `plumedpatch.sh` the proper patch script.

- Extract the source code from its archive and then move into its root directory.

- Configure the code as usual (not necessary for DL_POLY)
- NAMD and LAMMPS only: In the `plumedpatch` script, modify the `myarch` variable to match your architecture.
- Set the environmental variable `plumedir` to point to `PLUMED_root`.
- Copy or link the `plumedpatch.sh` file from the `patches` folder to the current directory.
- Execute the script: `"./plumedpatch.sh -patch"`.
- DL_POLY-only: copy the proper Makefile from `build/` to `srcmod/` (or `src/` in older versions)
- Compile the code as usual.

Further details can be found in the `patches/README` file.

Example.

This is the procedure for compiling the serial version of AMBER 9 with PLUMED using g95 in the Bourne shell.

```
tar xzf AMBER9.tgz
cd amber9/
export plumedir="PLUMED_root"
cp $plumedir/patches/plumedpatch_sander_9.sh .
cd src/
./configure g95
cd ..
./plumedpatch_sander_9.sh -patch
make
```

Example.

This is the procedure for compiling the serial version of GROMACS, using the GNU compilers.

```
tar xzf gromacs-4.0.5.tar.gz
cd gromacs-4.0.5
export plumedir="PLUMED_root"
cp $plumedir/patches/plumedpatch_gromacs_4.0.4.sh .
CC=gcc CXX=g++ ./configure
./plumedpatch_gromacs_4.0.4.sh -patch
make
make install
```

Example.

This is the procedure for compiling NAMD on an Intel Mac using the GNU g++ compiler and the FFTW.

```
tar xzf NAMD_2.6_Source.tar.gz
cd NAMD_2.6_Source
export plumedir="PLUMED_root"
cp $plumedir/patches/plumedpatch_namd.2.6.sh .
./config fftw MacOSX-i686-g++
```

Edit ./plumedpatch_namd.2.6.sh by setting the myarch variable to MacOSX-i686-g++.

```
./plumedpatch_namd.2.6.sh -patch
cd MacOSX-i686-g++
make
```

Example.

This is a sample procedure for compiling the scalar version of DL_POLY_2.20 with the gfortran compiler in the Bourne Shell environment.

```
tar xzf dl_poly_2.20.tar.gz
cd dl_poly_2.20
export plumedir=PLUMED_root
cp $plumedir/patches/plumedpatch_dlpoly_2.19.sh .
./plumedpatch_dlpoly_2.19.sh -patch
cp build/MakeSEQ srcmod/Makefile
cd srcmod
make gfortran
```

Example.

This is a sample procedure for compiling the openmpi version of LAMMPS with the mpic++ compiler in the Bourne Shell environment. Please, note that the LAMMPS tarball comes with the download date but unless major changes are done in the host code, the patching procedure will stay unchanged and the latest available patching file has to be used.

```
tar xvf lammps-15Jan10.tar
cd lammps-15Jan10/
```

Edit ./src/MAKE/Makefile.openmpi to suit your machine.

```
export plumedir=PLUMED_root
cp $plumedir/patches/plumedpatch_lammps_15-01-2010.sh ./
```

Edit ./plumedpatch_lammps_15-01-2010.sh to set myarch to openmpi.

```
./plumedpatch_lammps_15-01-2010.sh -patch
cd src
make openmpi
```

2.1.1 Compiling the ACEMD plugin with PLUMED

PLUMED works with ACEMD 1.1. As the source code for ACEMD is not available, PLUMED will work as a plug-in. For this reason the compilation will be slightly different from that of the other codes. Moreover, given the unavailability of source code, this version of PLUMED will **not be supported** by the PLUMED development team.

- Extract the source code from its archive.
- Set the environmental variable `plumedir` to point to `PLUMED_root`.
- Copy or link the `plumedpatch_acemd.sh` file from the `patches` folder to the `PLUMED_root/ACEMD` directory.
- Execute the script: `./plumedpatch_acemd.sh -patch`.
- Use `make` to compile the plugin.

Example.

This is a sample procedure for compiling the ACEMD plug-in in the Bourne Shell environment.

```
export plumedir=PLUMED_root
cd $plumedir ACEMD
cp patches/plumedpatch_acemd.sh .
./plumedpatch_acemd.sh -patch
make
```

Once the plugin.so is compiled, copy it where you will run ACEMD. Add the following lines to the ACEMD input file:

```
pluginload testplug ./plugin.so
pluginarg testplug input META_INP
pluginarg testplug boxx xx
pluginarg testplug boxy yy
pluginarg testplug boxz zz
pluginfreq 1
```

where `META_INP` is the PLUMED input file and `xx,yy,zz` are the box dimensions. At this time the support of the ACEMD plugin will be provided by the ACEMD developers. Please note that only orthorhombic cells are available in ACEMD 1.1.

2.2 Testing the installation

Once the installation process has been completed successfully, the user is encouraged to test the chosen MD package for any problems. The `tests` directory contains regtest scripts for the different MD codes. These also serve as a regularity test in case the user implements his own modifications and as a basic illustration of the capabilities of the plugin. The user should edit the test script, setting up the path where the test suite is found and giving the location of the executable.

For GROMACS users only. Please note that:

- The tests for GROMACS are designed for and should be executed with the double-precision version of the code;
- Biasxmd, ptmetad, dd and pd are designed for the parallel version of GROMACS. The user should specify in the test script the location of the parallel executable and the version (3 or 4) of GROMACS used. These tests will fail if the parallel version of GROMACS has not been compiled.

Example.

In the case of NAMD, the regtest script is called `do_regtest_namd.sh`. Here, the user should modify the `dir_base` and the `namd_prefix`:

```
dir_base=/Programs/md_meta/tests/namd
namd_prefix=/chicco/bin/namd_plugin/namd2.
```

Regtest scripts for the other programs require an identical procedure.

The script executes a list of tests and then compares the results with the outcome of previously run simulations. Please note that when the script is run for the first time it produces the reference. Finally, the script produces a summary of the results. The test result can be one of the following:

- 'OK' if the results match those of a previous run precisely. The execution time is also given.
- 'NEW' if they have not been executed previously. The reference result is generated automatically in this run. Tests can also be 'NEW' if they have been reset, *i.e.* been newly added to the `TEST_FILES_RESET` files.

- 'RUNTIME FAILURE' if they stopped unexpectedly (e.g. core dump, or stop).
- 'WRONG RESULT' if they produce a result that deviates from an old reference.

The last two options generally mean that a bug has been introduced, which requires investigation. Since regtesting only yields information relative to a previously known result, it is most useful to do a regtest before and after you make changes.

2.3 Back to the original code

At any time the user may want to "unpatch" the MD code and revert back to the original distribution. To do so, the user should go to the directory where the PLUMED patch has been copied and type `./plumedpatch -revert`.

Chapter 3

Running free-energy simulations

In this chapter we describe how to activate PLUMED and how to create the correct PLUMED input file for a specific type of free energy calculation. The typical output of these calculations is also explained in detail.

3.1 How to activate PLUMED

PLUMED input is contained in one single file, named `plumed.dat` by default, which defines the CVs, the type of run to be performed and the parameters for the bias potential generation.

- The users of NAMD, SANDER and DL_POLY can instruct the code to parse the PLUMED input file by setting the PLUMED variable to `on` (or `1` for SANDER) in the MD input file. It is also possible to change the default name for the PLUMED input file by setting the `plumedfile` variable in the MD input;
- GROMACS users should activate PLUMED on the command line by specifying the flag `-plumed` followed by the input file name. The extension of such a file must be `.dat`;
- In Quantum-ESPRESSO, to activate PLUMED just add `use_plumed = .true.` to the `control` namelist in the input. The name of the PLUMED input file is currently hardcoded as `plumed.dat`;

- In LAMMPS, PLUMED is activated using the fix `plumed`. The input file is specified by `plumedfile`, the output file by `outfile`.

Example.

A typical SANDER input file for a metadynamics calculation:

```
METADYNAMICS TEST
&cntrl
imin=0, irest=0, ntx=1, ig=71278,
nstlim=1001, dt=0.0002,
ntc=1, ntf=1,
ntt=3, gamma_ln=5,
tempi=300.0, temp0=300.0,
ntpr=200, ntwx=0,
ntb=0, igb=0,
cut=999.,
plumed=1 , plumedfile='plumed.dat'
/
```

Example.

The NAMD and DL_POLY input file should contain, in addition to the usual keywords defining the run, the following lines;

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

Example.

To perform a free-energy calculation with GROMACS, PLUMED must be activated on the command line:

```
mdrun -plumed plumed.dat ...
```

Example.

The control card for a typical Quantum-ESPRESSO (pw.x) input file for a metadynamics calculation:

```
&control
title = 'ch3cl',
calculation = 'md',
restart_mode = 'from_scratch',
nstep = 3,
dt = 20,
pseudo_dir = '../pseudo/',
prefix = 'ch3cl',
use_plumed = .true.,
/
```

Example.

The LAMMPS input file should contain, in addition to the usual keywords defining the run, the following line;

```
fix 3 all plumed plumedfile plumed.dat outfile metaout.dat
```

3.2 The input file

In the following sections we describe the syntax used in the PLUMED input file.

The commands contained in this file can be divided in two groups according to the functionality required. A first group of commands defines the type of simulation (metadynamics, steering and umbrella sampling, replica exchange methods) and the parameters needed for the chosen algorithm. These are described in this chapter. A second part defines the degrees of freedom on which the algorithms operate, the so-called collective variables or CVs. The details of such commands are described in the next chapter.

As a general rule, each setting is defined by a principal keyword that must be placed at the beginning of the line, in upper case. Additional input pertaining to the setting can be specified on the same line, using additional keywords that can be added in any order. A line can be continued to the next one by adding a backslash or an ampersand as a last character in the line. Three kinds of keyword may exist: the *directives*, which must be placed at the beginning of a line and define the argument of the line, the *keywords*, which specify the attributes of the different fields in the line and *flags*, which simply turn a given option on or off.

Example.

The following is an example of a complete PLUMED input file. In this case the input defines a well-tempered metadynamics run with two CVs; the first is the distance between two atoms, and the second a dihedral angle.

```
# general options
HILLS HEIGHT 0.1 W_STRIDE 100
WELLTEMPERED SIMTEMP 300 BIASFACTOR 10

# print each 50 time units and add a time offset of 20.0 to COLVAR
PRINT W_STRIDE 50 T_OFFSET 20.0

# definition of CVs

NOTE distance between hydrogens
DISTANCE LIST 13 46 SIGMA 0.35

NOTE torsional angle
TORSION LIST 1 4 65 344 SIGMA 0.1

# wall on the CV
UWALL CV 1 LIMIT 15.0 KAPPA 100.0 EXP 4.0 EPS 1.0 OFF 0.0

ENDMETA
```

The `ENDMETA` directive is the last line read from the PLUMED input file and any line added after this keyword will be ignored.

The symbol `#` allows the user to comment any line in the input file. The directive `NOTE` allows the user to place comments which are copied to the PLUMED log file.

The `PRINT` allows to dump a file called `COLVAR` (see more in 3.4) which is written each `W_STRIDE` timesteps and (optional) a time offset can be specified through the `T_OFFSET` keyword. To append the colvar values to an existing `COLVAR` file, the flag `APPEND` can be used.

3.3 A note on units

The values in the PLUMED input file are read in the internal units for the MD engine. For DL_POLY the energy in input can be in different user-specified units; to be consistent, the values in the PLUMED input file must be in the same units specified in the `FIELD` file.

3.4 Metadynamics

The metadynamics algorithm applies additional forces to a standard molecular dynamics simulation. In this case, the **PLUMED** input file must contain the definition of at least one CV (see chapter 4 for the required syntax), and the **HILLS** keyword which defines the details of the bias potential (see section 3.4.2). In this case the biasing potential will be calculated and applied to the microscopic degrees of freedom during the run. Before discussing the additional optional commands (section 3.4.2), we give a brief overview of the file produced in a typical metadynamics run.

3.4.1 Typical output

Standard metadynamics will produce, in addition to the usual output files generated by the MD engine, a file called **COLVAR** that contains the following data:

- The first column contains the time step;
- The following d columns contain the values of the CV(s);
- Two additional columns contain the value $V(s, t)$ of the bias potential at the given point and time, and the potential due to the walls (if defined).

Since **PLUMED** 1.2, a more flexible format for **COLVAR** has been introduced. Here, the first line of the **COLVAR** file is a header. The line begins with **#**, so as to be ignored by plotting programs such as **gnuplot** and **xmgrace**. A small script **plumedat.sh** to interpret this file is provided in the utilities.

Another file, called **HILLS**, contains the information of the biasing potential needed to estimate the free-energy, and to restart metadynamics runs. The bias potential is given by:

$$V(\vec{s}, t) = \sum_{k\tau < t} W(k\tau) \exp \left(- \sum_{i=1}^d \frac{(s_i - s_i^{(0)}(k\tau))^2}{2\sigma_i^2} \right). \quad (3.1)$$

Specifically,

- The first column contains the time step at which the contribution to the bias potential was added, τ , 2τ , etc.;

- The following d columns contain the values $\vec{s}^{(0)}(t)$ specifying the position of the centroid of the Gaussian;
- The next d columns contain the values $\vec{\sigma}$ specifying the Gaussian width along the different directions in the CV space;
- The last column but one contains the value W of the Gaussian height;
- The last column contains the bias factor defined in well-tempered metadynamics.

Please refer to section 5.1 for a description of how the potential in equation 3.1 can be computed from the HILLS file.

It is important to note that a metadynamics run should typically start from an equilibrated system. The equilibration protocol can be applied without resorting to the PLUMED input file. However, in order to gain important information concerning the behavior of the run and the tuning of the parameters of the metadynamics biasing potential, it is useful to monitor the behavior of the CVs during the equilibration run. To this aim, just skip the HILLS directive: in this way, only the COLVAR file will be generated.

3.4.2 Bias potential

The HILLS directive is used to define the details of the biasing potential. It must be followed by the definition of the weight factor W , preceded by the keyword HEIGHT (see the note on units in section 3.3). Also the frequency at which the Gaussians are deposited and written into the HILLS file must be set using W_STRIDE followed by the number of MD steps between two successive depositions. The Gaussian width must be set in the proper CV line using the keyword SIGMA.

Notice that the HILLS directive is not compatible with the COMMITMENT directive.

Example.

The following line switches on metadynamics with Gaussian height of 0.1 (in energy unit of the MD code) and deposition stride of 1000 MD steps.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
```

3.4.3 Well-tempered metadynamics

The `WELLTEMPERED` directive activates well-tempered metadynamics [14] which, by rescaling the Gaussian weight factor, guarantees the theoretical convergence of metadynamics. In the well-tempered algorithm, the rate at which the bias potential is added is decreased during the simulation proportionally to $e^{-V(s,t)/\Delta T}$, where ΔT is a characteristic energy:

$$V(s, t) = \sum_{t'=0, \tau_G, 2\tau_G, \dots}^{t' < t} W e^{-V(s(q(t'), t')/\Delta T} \exp \left(- \sum_{i=1}^d \frac{(s_i(q) - s_i(q(t'))^2}{2\sigma_i^2} \right), \quad (3.2)$$

where $W = \tau_G \omega$ is the height of a single Gaussian.

For a given temperature of the system T (specified with the keyword `SIMTEMP`), the CVs are sampled at a fictitious higher temperature $T + \Delta T$ determined by the bias factor $(T + \Delta T)/T$. The user must specify this bias factor using the keyword `BIASFACTOR`.

Example.

The following commands define a well-tempered metadynamics run, in which the CVs are sampled at the higher temperature of 3000 K. The initial Gaussian height is 0.1 (in energy unit of the MD code) and the deposition stride is 1000 MD steps.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
WELLTEMPERED SIMTEMP 300 BIASFACTOR 10
```

It should be noted that, in the case of well-tempered metadynamics, in the output printed on the `HILLS` file the Gaussian height is rescaled using the bias factor. This is done in order to directly obtain the free energy (and not the bias), when summing all the Gaussians deposited during the run.

3.4.4 Restarting a metadynamics run

In order to restart a metadynamics run, the flag `RESTART` must be added on the line of the directive `HILLS`. It allows a metadynamics simulation to be restarted after an interruption or after a run has finished. The `HILLS` files will be read at the beginning of the simulation and the bias potential applied to the dynamics. Note that the presence of the `RESTART` flag only affects the metadynamics part of the simulation, and thus the usual procedure for restarting a MD run must be followed. This depends on the particular MD engine used and can be found in the relative documentation.

Example.

The following is an example of input file for restarting a metadynamics simulation.

```
HILLS RESTART HEIGHT 0.1 W_STRIDE 1000
```

In case of well-tempered metadynamics, the Gaussians height is rescaled in input according to the bias factor. This is done assuming that the sum of the Gaussians stored in the `HILLS` file is an estimate of the (negative) free energy landscape. Since the estimate of the free energy is in principle independent from the choice of the bias factor, it is correct to restart a well-tempered simulation with a different bias factor or even restart from a non-well-tempered simulation.

3.4.5 Using GRID

Normally the additional forces of metadynamics are calculated every MD step by summing the contribution coming from the Gaussians deposited up to this point, following Eq. 3.2. As the simulation goes on, the computational time spent in the evaluation of these forces becomes larger and larger and eventually comparable with the time needed to calculate the contribution of the force field itself. This effect is particularly visible when the system simulated is small or when using a simplified coarse grained potential.

A possible solution is to store an array containing the current value of the bias potential (and of the derivatives with respect to the CVs) on a grid. In this way the computational cost of metadynamics becomes constant all over the simulation and corresponds to the cost of evaluating a single Gaussian function on the whole grid with a frequency given by the stride between subsequent hills. This approach is similar to that proposed in Ref. [20], but has the advantage that the grid spacing is independent on the Gaussian width.

This operation can be demanding if the number of collective variable and/or the number of grid bins is high. However, the cost of adding the Gaussian on the grid can be substantially reduced taking into account that this function is almost zero outside a characteristic range determined by the Gaussian sigma. In `PLUMED` this interval is calculated once (or at every modification of sigma) and used to build a smaller sub-grid on which the potential is updated.

In order to use the grid, the directive `GRID` must be added together with

the keyword **CV** to specify the collective variable, **MIN** and **MAX** to fix the CV interval, **NBIN** for the number of bins and the flag **PBC** if the CV is periodic.

Example.

In this example we run metadynamics using only one CV, a distance between two atoms, and we put the bias potential on a grid of 200 bins in the interval between 0.0 and 10.0.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
PRINT W_STRIDE 50
DISTANCE LIST 13 46 SIGMA 0.35
GRID CV 1 MIN 0.0 MAX 10.0 NBIN 200
ENDMETA
```

Special labels can be used in the definition of the interval with **MIN** and **MAX**, such as **-pi**, **+pi**, **+2pi**, **-2pi**, **pi**, **2pi**. These labels may be particularly useful with the CVs **ANGLE** or **TORSION**.

Example.

In this example we run metadynamics using a dihedral angle and putting the bias potential on a periodic grid of 200 bins in the interval between -pi and pi.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
PRINT W_STRIDE 50
TORSION LIST 13 15 17 19 SIGMA 0.35
GRID CV 1 MIN -pi MAX +pi NBIN 200 PBC
ENDMETA
```

As in standard metadynamics, a **HILLS** file containing the list of Gaussians deposited is produced. This file is needed for restarting a metadynamics simulation also when using **GRID**.

The bias potential in a generic point is calculated by a polynomial interpolation which has the proper values of the function and of its derivatives in 2^d points, where d is the number of collective variables. The forces are then obtained as the analytical derivatives of the bias. You can switch off the use of splines by using the directive **NOSPLINE**. In this case, the bias and forces are simply taken at a close grid point (this requires a much denser grid).

Please also note that:

- **GRID** must be activated (or switched off) on ALL the CVs;

- GRID can be used together with multiple walkers metadynamics, bias-exchange and parallel tempering metadynamics;
- For a correct calculation of the potential and forces, the bin size must be smaller than half the Gaussian sigma. If a larger size is used, the code will stop.
- If the simulation goes out of the grid, the code will stop. Please increase MIN or MAX and restart metadynamics.

Writing a GRID to file

The directive `WRITE_GRID` allows to save on a file the grid on which the bias potential is stored and the relative forces. The keyword `W_STRIDE` can be used to specify the writing stride and `FILENAME` the name of the file. Since saving the entire grid on file may take some time, a reasonable writing stride should be used.

Example.

The following command controls a metadynamics calculation using as CV the distance between two atoms. The bias potential is stored on a grid and saved to the file `bias.dat` every 100000 steps.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
DISTANCE LIST 10 12 SIGMA 3.0
GRID CV 1 MIN 0.0 MAX 10.0 NBIN 10
WRITE_GRID FILENAME bias.dat W_STRIDE 100000
```

The file on which the grid is saved has a specific format. A header contains information about the presence of force data, the number and type of CVs, the grid dimension and boundaries and whether the CVs are periodic or not. The rest of the file contains for each grid point the value of the bias potential and forces (i.e. minus the derivative of the potential with respect to the CVs).

Example.

The header states that force data are on file, only one CV is present (a distance, see Tab. 3.1 for a legend). The grid is made of 10 bins ranging from 0 to 10 in unit of the CV. This CV is not periodic.

```

#! FORCE 1
#! NVAR 1
#! TYPE 1
#! BIN 10
#! MIN 0.000000
#! MAX 10.000000
#! PBC 0
0.000000 0.998838 -0.016081
1.000000 0.960195 0.091229
2.000000 0.825978 0.170252
3.000000 0.635806 0.201699
4.000000 0.437950 0.187594
5.000000 0.269942 0.145621
6.000000 0.148888 0.096861
7.000000 0.073484 0.055971
8.000000 0.032454 0.028326
9.000000 0.012826 0.012620
10.000000 0.004536 0.004967

```

This file can be used to plot the free energy resulting from a metadynamics simulation instead of summing the Gaussians stored on the `HILLS` file with the post-processing code `sum_hills`. When using well-tempered metadynamics, please remember that the bias potential does not compensate exactly the underlying free energy [14]. In this case, the potential written on file must be rescaled accordingly.

Reading a GRID from file

The bias potential stored on file can be used to restart a metadynamics simulation by specifying the directive `READ_GRID` and the file name with `FILENAME`.

Example.

The following command controls a metadynamics calculation using as CV the distance between two atoms. The initial bias potential is read from the file `bias.dat`.

```

HILLS HEIGHT 0.1 W_STRIDE 1000
DISTANCE LIST 10 12 SIGMA 3.0
GRID CV 1 MIN 0.0 MAX 10.0 NBIN 10
READ_GRID FILENAME bias.dat

```

Please also note that:

- The CV number and type in the header must be consistent with what declared in the `PLUMED` input file;
- The number of bins and the boundaries in the header can be different from what declared in the `PLUMED` input file. In this case the parameters of the `PLUMED` input file will be used and the grid present on file will be interpolated to fit the new dimensions;
- If force data are not present on file, they will be calculated from the bias potential using finite differences;
- `READ_GRID` is not compatible with the `MULTIPLE_WALKERS` directive.

Restarting a metadynamics run from a grid written on file is fully compatible with the standard restart by reading a `HILLS` file created in a previous run.

Example.

The following command controls a metadynamics calculation using as CV the distance between two atoms. The initial bias potential is read from the file `bias.dat`. To this initial bias, the Gaussians deposited on the `HILLS` file are added.

```
HILLS RESTART HEIGHT 0.1 W_STRIDE 1000
DISTANCE LIST 10 12 SIGMA 3.0
GRID CV 1 MIN 0.0 MAX 10.0 NBIN 10
READ_GRID FILENAME bias.dat
```

3.4.6 Multiple walkers

The `MULTIPLE_WALKERS` directive sets the multiple walkers [15] running mode. All the processes must have the same CVs, in the same order. Each process will write its own bias potential in the directory specified by the `HILLS_DIR` keyword and in a file named `HILLS.0`, `HILLS.1`, etc etc. Multiple processes must be launched independently and must point to the same directory `HILLS_DIR`, so that each one will contribute to the total bias potential.

The keyword `NWALKERS` sets the total number of walkers. This can be set to a value greater than the actual number of processes running and it can be increased during the run. `ID` determines a unique id of the walker, starting from 0. The `R_STRIDE` keyword sets the stride (in time steps) at which each

individual bias potential is updated by reading all the HILLS files contained in HILLS_DIR.

Example.

The following command controls a multiple walkers calculation using a maximum of 10 walkers. Gaussians are added every 1000 steps and updated every 5000 steps.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
MULTIPLE_WALKERS HILLS_DIR /scratch/HILLS R_STRIDE 5000 NWALKERS 10 ID 0
```

3.4.7 Monitoring a collective variable without biasing it

The directive `NOHILLS` can be used to monitor a CV during a metadynamics run without applying a bias on it. The keyword `CV` must be specified to select the CV to be monitored. This directive must be used for metadynamics simulations with Bias-Exchange.

Example.

In this example we run metadynamics using only one CV, a distance between two atoms. During the simulation we monitor the behavior of another CV, the dihedral defined by a set of four atoms, without putting a bias on it.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
PRINT W_STRIDE 50
DISTANCE LIST 13 46 SIGMA 0.35
TORSION LIST 1 4 65 344 SIGMA 0.1
NOHILLS CV 2
ENDMETA
```

As an alternative you may also avoid the `SIGMA` value in the CV. This will be understood as a directive that no hills must be put on this CV. The previous example therefore becomes:

Example.

```
HILLS HEIGHT 0.1 W_STRIDE 1000
PRINT W_STRIDE 50
DISTANCE LIST 13 46 SIGMA 0.35
TORSION LIST 1 4 65 344
ENDMETA
```

where the `NOHILLS` keyword for CV 2 has been omitted as implicitly included by the missing `SIGMA` parameter.

3.4.8 Inversion condition

The `INVERT` keyword activates the inversion condition near predefined boundaries `LIMIT1`, `LIMIT2`[21]. This is useful to avoid the onset of systematic errors in the free energy reconstruction that occur at the boundaries of the CVs space[21] for intrinsically limited CVs (like e.g. `ALPHABETA`, see chapter 4) or artificially limited CVs (using for example an external potential, see section 3.10). To simplify the notation, we here assume the CVs space is one-dimensional with the boundary at $s = 0$. The inversion condition ensure that near $s = 0$ the bias potential satisfy the following relation:

$$V(-s, t) \approx 2V(0, t) - V(s, t) \quad (3.3)$$

This property ensures that, at stationary conditions, the history-dependent potential is approximately linear close to the boundary, but it does not impose the value of its derivative, that is iteratively determined by the thermodynamic bias. In practice this is achieved by adding extra Gaussians out of the boundaries according to the following roles:

- An interval centered in 0 is chosen, whose width χ_1 is of the order of σ .
- If $s < \chi_1$ another Gaussian centered in $-s$ and with the same width and height is added.
- If $s > \chi_1$ another Gaussian centered in $-s$ and with the same width is added. In this case, the height of the extra Gaussians depends on V and is given by:

$$w = [2V(0, t) - V(-s, t) - V(s, t)] y(s), \quad (3.4)$$

where $y(s) = 1 / [1 + (s/(\chi_2\chi_1))^{10}]$ and $\chi_2 > \chi_1$

The second factor in Eq. (3.4) is approximately one for $|s| < \chi_2\chi_1$ and goes to zero for $|s| > \chi_2\chi_1$. This ensures that V goes smoothly to zero out of the boundaries. χ_2 defines the width of the inversion interval and it

is regulated by the keyword **INVERSION** (σ units). χ_1 is regulated by the keyword **REFLECTION** (σ units). The keyword **MAXHEIGHT** defines the largest $|w|$ in gaussian height units and it regulates the speed of variation of V out of the boundaries. The **INVERT** keyword can be used in presence of an external potential that limits the CVs space exploration (see section 3.10) like in the example below;

Example.

The following input file contains the inversion condition in presence of soft walls:

```
HILLS HEIGHT 0.1 W_STRIDE 500
COORD LIST <g1> <g2> NN 8 MM 13 D_0 0 R_0 0.25 SIGMA 0.15
g1->
26 27
g1<-
g2->
29 30
g2<-
INVERT CV 1 REFLECTION 1.6 INVERSION 6 MAXHEIGHT 4 LIMIT1 0.2 LIMIT2 2.8
UWALL CV 1 LIMIT 2.8 KAPPA 3000.0
LWALL CV 1 LIMIT 0.2 KAPPA 100000.0
ENDMETA
```

or if an intrinsically limited CV is used (see chapter 4):

Example.

The following input file contains the inversion condition in presence of an instinsically limited CV, such as ALPHABETA:

```
HILLS HEIGHT 0.1 W_STRIDE 250
ALPHABETA NDIH 1 SIGMA 0.05
5 12 14 23 -3.14159
INVERT CV 1 REFLECTION 1.6 INVERSION 6 MAXHEIGHT 4 LIMIT1 0.0 LIMIT2 1.0
ENDMETA
```

It is worth to note that **LIMIT1** and **LIMIT2** must not be in unphysical regions of the CVs space, i.e. they must be visited during the simulation. The present implementation does not remove systematics errors for multidimensional metadynamics in regions of the CVs space that are near crossing boundaries (e.g. if **LIMIT1**=0 for CV 1 and **LIMIT1**=0 for CV 2 there still can be systematic error near (0,0)). The values for the **REFLECTION**,

INVERSION and MAXHEIGHT keywords chosen in the examples are also their default values. They were chosen in other to minimize the free energy error at the boundaries for several different test cases. Further information can be found in ref.[21].

3.5 Running in parallel

Simulations of large systems can often accelerated by using parallel machines. The behavior of PLUMED in parallel simulations is dependent on the host code:

- NAMD: the present version of PLUMED is fully compatible with NAMD for parallel simulations. However, since PLUMED runs on the first processor only, the computational effort required to evaluate the collective variables and their derivatives, as well as the history dependent potential in metadynamics simulations, should be kept lower than $1/N_p$ of the total computational effort, where N_p is the number of processors. Thus, pay attention to heavy variables and metadynamics simulations with many hills.
- GROMACS, DL_POLY, AMBER and LAMMPS: the present version of PLUMED is fully compatible with these codes for parallel simulations. Consider the following notes:
 - The coordinates of the particles involved in collective variables are replicated over all nodes at every step. If you don't want to slow down your simulation, minimize the number of involved atoms.
 - The computational effort required to evaluate the collective variables and their derivatives is replicated on all processors, thus is effectively not scaling. Pay attention to heavy variables (such as coordination numbers with long lists).
 - The computational effort required to evaluate the history dependent potential in metadynamics simulations is spread over processors, thus should scale linearly.
 - With GROMACS and domain decomposition, pay attention to periodic boundary conditions (see Section. 4.22).
 - With DL_POLY, use the patched version of MakePAR to compile PLUMED.

3.6 Replica exchange methods

When combined with GROMACS (both version 3.3 and 4.0), PLUMED can perform replica-exchange simulations coupled with metadynamics in two different ways: parallel tempering metadynamics (PTMetaD) [16, 22] and bias-exchange metadynamics (BE-META) [17].

3.6.1 Parallel tempering metadynamics

Parallel tempering metadynamics (currently implemented only for the GROMACS engine) is selected using the PTMETAD directive.

To run parallel tempering simulations with metadynamics, one has to follow the standard GROMACS procedure for parallel tempering (see GROMACS manual). A binary topology `.tpr` file must be prepared for each replica, while only one plugin input file is required.

Example.

The following input file defines a parallel-tempering metadynamics:

```
# switching on metadynamics and Gaussian parameters
HILLS HEIGHT 0.1 W_STRIDE 500
# switching on parallel tempering
PTMETAD
# instruction for CVs printout
PRINT W_STRIDE 50
# the CV: radius of gyration
RGYR LIST <CA> SIGMA 0.1
CA->
20 22 26 30 32
CA<-
# end of the input
ENDMETA
```

The PTMETAD directive switches on parallel tempering. All replicas have the same CVs, in this particular case the radius of gyration defined by the group of atoms `<CA>`.

The Gaussian height set by the keyword `HEIGHT` is automatically rescaled with temperature, following $W_i = W_0 \frac{T_i}{T_0}$, where i is the index of a replica and T_i its temperature.

Similarly, the simulation temperature needed to use the well-tempered algorithm (which, for non-parallel simulations is set with the `SIMTEMP` keyword) is here taken directly from the GROMACS input at each replica. As

a result, the value of ΔT for the well-tempered algorithm is rescaled across the replicas.

The plugin will produce one `COLVAR` file and one `HILLS` file for each replica.

3.6.2 Bias exchange simulations

Bias exchange simulations must be run using the `BIASXMD` directive. Only the GROMACS engine currently supports this feature.

The procedure for running bias-exchange simulations is similar to the one described earlier for parallel tempering. However, one plugin input and one binary topology file must be provided for each replica. These files must be named with the replica index (e.g., `META_INP0`, `META_INP1`, ...) and must contain all the CVs, in the same order, and the directive `BIASXMD`. The first replica (`META_INP0`) must have the `NOHILLS` directive for *all* the CVs; the other replicas must switch off all the variables they do not use with a list of keywords `NOHILLS` (each CV must be excluded separately by using `NOHILLS` multiple times). Also in this case, the plugin will produce one `COLVAR` file and one `HILLS` file for each replica.

3.7 Umbrella sampling

The directive `UMBRELLA` allows umbrella sampling calculations to be performed on the CV specified by the parameter keyword `CV`. The position s_0 of the umbrella restraint is determined by the keyword `AT`, and the spring constant k - in internal unit of the main code - by the keyword `KAPPA`. Optionally, also a constant force m can be specified by the keyword `SLOPE`. This turns on a potential of the following functional form:

$$V_{\text{umb}}(s) = \frac{1}{2}k(s - s_0)^2 + m(s - s_0). \quad (3.5)$$

Example.

The following input file defines an umbrella potential acting on the first CV and centered in $s_0 = -1.0$.

```
TORSION LIST 13 15 17 1
UMBRELLA CV 1 KAPPA 200 AT -1.0
PRINT W_STRIDE 100
ENDMETA
```

In the case of umbrella sampling runs, the value of the CVs is printed on the COLVAR file with a stride fixed by the directive `PRINT` and the keyword `W_STRIDE`. The file contains the time, the CV values, the metadynamics potential, the harmonic potential of umbrella sampling, the CV on which the restraint acts and the position of the restraint. The final calculation of the free energy as a function of this CV can be done using the weighted histogram analysis method, choosing one of the many possible implementations, for instance the wham code by Alan Grossfield [23]. The directive `UMBRELLA` can be used multiple times in the case of multidimensional umbrella sampling calculations (one directive for each CV).

The keyword `RESTART` can be used when restarting an umbrella sampling calculation to append the value of the CVs on the COLVAR file.

3.8 Steered MD

PLUMED can be used to drag a system to a target value in CV space using an harmonic potential moving at constant speed. If the process is reversible, *i.e.* for velocities tending to zero, the work done in the dragging corresponds to the free energy. In the case of finite velocity it is possible to obtain an estimate of the free-energy from the work distribution using Jarzynski [12] or Crooks [13] relations.

The directive `STEER` activates the steering on the collective variable specified by the keyword `CV`. The target value is determined by the keyword `TO`, the velocity, in the same unit as the corresponding CV every 1000 steps, by `VEL` and the spring constant by `KAPPA`. An additional keyword `FROM` can be used to specify the starting point of the dragging, otherwise the CVs values at the first step are taken as the starting point. The functional form of the dragging potential is the same as the one of formula 3.5, with the reference position s_0 moving at the specified velocity.

The keyword `RESTART` can be used when restarting a steered MD calculation to append the value of the CVs on the COLVAR file.

Example.

The following input file defines a steered MD on the angle CV to a target value of 3.0 rad.

```
ANGLE LIST 13 15 17
STEER CV 1 TO 3.0 VEL 0.5 KAPPA 500.0
PRINT W_STRIDE 100
ENDMETA
```

3.8.1 Steerplan

PLUMED can be used to drag a system on a pathway that is composed of successive steering runs on chosen degrees of freedom in a planned fashion. While the directive **STEER** is rather easy and intuitive, **STEERPLAN** is more flexible and it allows to avoid a lot of scripting whenever you plan to simulate complex transitions by means of out-of-equilibrium runs.

The directive **STEERPLAN** activates this option and reads the name of a file that contains the plan as an input.

Example.

The following input file contains many collective variables and a steerplan directive.

```
PRINT W_STRIDE 5
#
# these CVs are put here just to show that you may use more variable than the
# ones defined in the steerplan
#
TARGETED TYPE RMSD FRAMESET restrained.pdb
TARGETED TYPE RMSD FRAMESET waters.pdb
TARGETED TYPE RMSD FRAMESET ref.H.pdb
#
# difference of distances for proton transfers
#
DISTANCE LIST 53 703 DIFFDIST 703 702
DISTANCE LIST 702 913 DIFFDIST 913 912
#
# steer plan read from file myplan
#
STEERPLAN myplan
ENDMETA
```

The steerplan file **myplan** contains the control sequence of the steerplan and it is composed as follows. The first column contains the time (in timeunits of the program) at which a particular action is planned. The following columns are composed in blocks of five. Each block contains: the **CV** keyword, the index of the CV to be used, the spring constant at a given time, the value of the center of the spring potential and a keyword among **CENTRAL** (the potential is a normal parabolic shape), **POSITIVE** (the potential with parabolic shape is applied only when the CV value is higher than the center of the spring) and **NEGATIVE** (the potential with parabolic shape is applied only when the CV value is lower than the center of the spring). A Block

of four columns without the latter keyword is also accepted. In this case CENTRAL is assumed. Wildcards (*) are also accepted for the position. Their meaning is much clearer in the following example. Comments are allowed (#) .

Example.

An example of steerplan file.

```
# Bring CV 4 from wherever it is (*) to -0.5 at 2 ps by increasing
# gradually the spring constant from 0 to 800. CV 5 does the same and goes to -0.64.
0.00 CV 4 000.0 * POSITIVE CV 5 000.0 * POSITIVE
2000.00 CV 4 800.0 -0.5 POSITIVE CV 5 800.0 -0.64 POSITIVE
# Now in 400 fs keep CV 4 at its value while releasing the other
# slowly to 0 spring constant.
2400.00 CV 4 800.0 -0.5 POSITIVE CV 5 0.0 -0.64 POSITIVE
# Now drag back only CV 4 to 1.4 with a value that acts only
# on the negative part. The potential on CV 5 is off.
8400.00 CV 4 800.0 1.4 NEGATIVE CV 5 0.0 -0.64 POSITIVE
```

Please note that the wildcards have particular meaning:

Example.

Here the dragging force starts from whatever value the system assumes at 0 fs and drags it to -0.5 at 2 ps. The spring constant is linearly increasing from 0.0 to 800.0. When the starting point is a wildcard this takes the current value.

```
0.00 CV 4 000.0 * POSITIVE
2000.00 CV 4 800.0 -0.5 POSITIVE
```

Here the center of the harmonic potential follows the system at each step from 0 to 2 ps. When the position of the ending point is a wildcard then the potential will follow the coordinates (this means that the potential is not applied as the force is zero everywhere).

```
0.00 CV 4 800.0 * POSITIVE
2000.00 CV 4 800.0 * POSITIVE
```

This case is considered identical as the one before

```
0.00 CV 4 800.0 0.5 POSITIVE
2000.00 CV 4 800.0 * POSITIVE
```

3.9 Adiabatic Bias MD

PLUMED 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[18].

The directive **ABMD** activates the biasing on the collective variable specified by the keyword **CV**. The target value is determined by the keyword **TO**, the spring constant by **KAPPA**. The biasing potential is implemented as

$$V(\rho(t)) = \begin{cases} \frac{\alpha}{2} (\rho(t) - \rho_m(t))^2, & \rho(t) > \rho_m(t) \\ 0, & \rho(t) \leq \rho_m(t), \end{cases} \quad (3.6)$$

where

$$\rho(t) = (CV(t) - TO)^2 \quad (3.7)$$

and

$$\rho_m(t) = \min_{0 \leq \tau \leq t} \rho(\tau). \quad (3.8)$$

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 at moving 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.

The keyword **RESTART** can be used when restarting an adiabatic bias MD calculation to append the value of the CVs on the **COLVAR** file and to set the best value previously reached.

Example.

The following input file defines an adiabatic biased MD on the angle CV to a target value of 3.0 rad.

```
ANGLE LIST 13 15 17
ABMD CV 1 TO 3.0 KAPPA 500.0
PRINT W_STRIDE 100
ENDMETA
```

3.10 External potentials

3.10.1 Walls

The **UWALL** and **LWALL** keywords define a wall for the value of the CV s 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 **UWALL**) or lower (in the case of **LWALL**) than a certain limit **LIMIT** minus an offset **OFF**.

The functional form of this potential is the following:

$$V_{wall}(s) = \text{KAPPA} \left(\frac{s - \text{LIMIT} + \text{OFF}}{\text{EPS}} \right)^{\text{EXP}}, \quad (3.9)$$

where **KAPPA** is an energy constant in internal unit of the code, **EPS** a rescaling factor and **EXP** the exponent determining the power law.

By default: **EXP** = 4, **EPS** = 1.0, **OFF** = 0.

Example.

To run a well-tempered metadynamics simulation using as CV the distance between one atom and the center of mass of a group of atoms and limiting its value below 15 Å, you have to use the following input file:

```
HILLS HEIGHT 0.1 W_STRIDE 100
WELLTEMPERED SIMTEMP 300 BIASFACTOR 10
PRINT W_STRIDE 50
DISTANCE LIST 13 <g1> SIGMA 0.35
g1->
17 20 22 30
g1<-
UWALL CV 1 LIMIT 15.0 KAPPA 100.0
ENDMETA
```

3.10.2 Tabulated potentials

An external potential of generic form can be added to any collective variables using the directive **EXTERNAL**. The user must specify the total number of CVs on which the potential acts with the keyword **NCV** and the variables with **CV**. The external potential must be provided in a tabulated form in the file specified by **FILENAME**. The format used for the external potential is the same as the one described in 3.4.5 for the case of metadynamics potential on a **GRID**.

Example.

The following input file controls a simulation with an external potential acting on two collective variables. The tabulated potential is provided in the file `external.dat`.

```
PRINT W_STRIDE 100
DISTANCE LIST 13 20
TORSION LIST 5 8 10 12
EXTERNAL NCV 2 CV 1 2 FILENAME external.dat
```

Every collective variables implemented in **PLUMED** has a unique ID. This number must be used to define the type of CV in the header of the external potential file and must match the CV activated in the **PLUMED** input file. In Tab. 3.1 we provide a legend.

Example.

The header of the external potential file of the previous example looks like this:

```
#! FORCE 1
#! NVAR 2
#! TYPE 1 5
#! BIN 100 100
#! MIN 0.0 -3.14159
#! MAX 10.0 3.14159
#! PBC 0 1
```

If force data are not present on file, they will be calculated from the tabulated potential using finite differences.

ID	CV
1	Distance
2	Minimum distance
3	Coordination number
4	Angle
5	Torsion
6	Alpha-beta similarity
7	Hydrogen bonds
8	Dipole
11	Radius of gyration
14	Torsional RMSD
16	Dihedral correlation
20	Interfacial water
30	Path collective variable S
31	Path collective variable Z
32	Absolute position
33	Electrostatic potential
34	Puckering coordinates
35	Energy
36	Helix loops
37	Alpha helix rmsd
38	Antiparallel beta rmsd
39	Parallel beta rmsd

Table 3.1: ID of the collective variables implemented in PLUMED.

3.11 Commitment analysis

The `COMMITMENT` directive is used to run commitment analysis. The keyword `NCV` determines the total number of CVs for the analysis, while `CV` must be used to specify the variable id. Following this line, `NCV` lines must be provided, each of which containing the upper and lower limits of the \mathcal{A} and \mathcal{B} basins for the i -th variable.

Example.

The following line defines a commitment analysis on the first two collective variables s_1 and s_2 , while the third is only monitored. The commitment basins are defined as $\mathcal{A} = \{(s_1, s_2) | s_1 \in (0, 1), s_2 \in (-1, 1)\}$, and $\mathcal{B} = \{(s_1, s_2) | s_1 \in (1, 2), s_2 \in (-1, 1)\}$.

```
COMMITMENT NCV 2 CV 1 2
0.0 1.0 1.0 2.0
-1.0 1.0 -1.0 1.0

DISTANCE LIST 12 30
DISTANCE LIST 20 24
TORSION LIST 14 16 20 22
ENDMETA
```

Chapter 4

Collective variables

PLUMED contains implementations of a large number of CVs so one can properly describe the processes involved in a wide variety of interesting problems. In the following chapter we describe all the CVs, which are implemented in PLUMED together with their analytic first derivative.

In general to instruct PLUMED to use a particular CV a line starting with the keyword indicating the CV type must be included in the input file. This keyword should then be followed by the various pieces of CV-specific information required to calculate the variable along with the keywords that tell PLUMED how this particular CV is to be employed. (N.B. unless stated explicitly these pieces of data can be specified in any order)

For metadynamics the lines defining the variables on which the user desires there to be a biasing potential should contain the **SIGMA** keyword. This keyword should then be followed by the width of the Gaussian hills (in the units of the CV) on that particular CV. This keyword serves two functions; namely, it instructs PLUMED to use metadynamics and tells it the widths of the hills. Obviously the **SIGMA** keyword is not only required if you are running metadynamics and is not required with other methods.

Specifying lists of atoms

Most collective variables require the user to specify one or several groups of atoms in their definition. Whenever a set of atom groups is required, the **LIST** keyword must be used. This keyword is then followed a set of tags which specify the number and order of the groups of atoms to be employed. The atoms involved in each of the groups invoked must be specified somewhere in

the input file. These group specifications must then start with the the name of the group followed by the `->` sign and finish with the same name followed by the `<-` sign. Between these two delimiters the indices of the atoms which comprise the group must then be listed, separated by spaces or line feeds.

Example.

The following syntax instructs PLUMED to use the distance between the center of mass of atoms 6 and 10 and the center of mass of atoms 8, 15 and 21 as a CV:

```
DISTANCE LIST <g1> <g2> SIGMA 1.0
g1->
6 10
g1<-

g2->
8 15 21
g2<-
```

Inside the group definition blocks, one can either specify the atom numbers explicitly, or one can use the `LOOP` keyword to define a regular sequence of indices with a given starting number, end number and stride.

Example.

The following two commands are equivalent definitions of the group `g1`:

```
g1->
10 12 14 16 18 20
g1<-

g1->
LOOP 10 20 2
g1<-
```

For the special, and rather common, case of a group composed of a single atom the user can specify the number of the atom of interest rather than the corresponding `<g>` tag.

Example.

The following two commands are equivalent ways instruct PLUMED to use the distance between atom 5 and group <g1> as a CV:

```
DISTANCE LIST <g1> <g2> SIGMA 1.0
g1->
10 12 14 16 18 20
g1<-

g2->
5
g2<-

DISTANCE LIST <g1> 5 SIGMA 1.0
g1->
10 12 14 16 18 20
g1<-
```

4.1 Absolute position

The **POSITION** keyword instructs PLUMED to use the absolute position of an atom or a group of atoms, specified by using the **LIST** keyword, as a CV. This CV accepts several options that allow the user to restrict the bias to a given direction, e.g. *z*, to bias the position of the particle as projected on a selected line segment or, in analogy with the path CV, to bias the atoms' distance from a line segment. The keyword **DIR** accepts as input *X*, *Y* or *Z* and limits the restraint to the chosen direction.

Example.

The following line instructs PLUMED to use the *y* coordinate of atom 13 as a CV.

```
POSITION LIST 13 SIGMA 0.35 DIR Y
```

The keyword **LINE_POS** instructs PLUMED to use the projection of the atoms position on a line as a CV, while the keyword **LINE_DIST** instructs PLUMED to use the distance from the line as a CV. In both these cases the line is defined by stating its start and end points. The line can then either be in constrained to be in the *XY*, *XZ* or *YZ* planes (keywords (*XY*, *XZ* or *YZ* respectively) or it can have an arbitrary orientation in space (keyword *XYZ*). Obviously if the line is constrained to the *XY*, *XZ* or *YZ* planes then the start and end points are two dimensional vectors whereas if it has an arbitrary orientation these vectors have three components.

Example.

The following lines instruct PLUMED to use the projection and distance of the coordinates of atom 13 on a generic line segment defined by the start and end points (0,0,0) and (2,3,4) respectively as the two collective coordinates.

```
POSITION LIST 13 SIGMA 0.35 LINE_POS XYZ 0. 0. 0. 2. 3. 4.
```

```
POSITION LIST 13 SIGMA 0.35 LINE_DIST XYZ 0. 0. 0. 2. 3. 4.
```

4.2 Distance

The **DISTANCE** keyword instructs PLUMED to use the distance between the center of mass of two groups of atoms as a CV. Two groups must be defined using the **LIST** keyword and the syntax described in section 4.

Example.

The following lines instruct PLUMED to use the distance between atom number 13 and the center of mass of the four atoms in list <g1> as a CV.

```
DISTANCE LIST 13 <g1> SIGMA 0.35
g1->
17 20 22 30
g1<-
```

The optional flag **NOPBC** can be used to calculate the distance without applying periodic boundary conditions. This should be done only if all the atoms in the groups are part of the same molecule. See also Sec. 4.22.

The keyword **DIR** can be used to calculate the component of this distance along the cartesian axes (X, Y or Z) and or the component in the planes (XY, XZ or YZ).

Example.

The following line instruct PLUMED to use the X-component of the distance between atom number 20 and 25 as a CV.

```
DISTANCE LIST 20 25 DIR X SIGMA 0.35
```

Whenever one wants to use the difference between two distances one may use the keyword **DIFFDIST**. This may turn to be useful in bond breaking/bond formation.

Example.

The following line instruct PLUMED to use the difference between the distance between 20 and 25 and the distance between 30 and 31 as CV.

```
DISTANCE LIST 20 25 DIFFDIST 30 31
```

Groups are also accepted as input instead of two atoms.

Other two useful variants are the following: the distance of a point from an axis and the projection of the point on the axis. The first is introduced by the additional keyword `POINT_FROM_AXIS` followed by the atom or the group defining the point respect to which the distance has to be calculated. The axis is defined through the standard two groups appearing in the definition of the distance collective variable.

Example.

The following line instruct PLUMED to use the distance between one atom, 26 and the axis defined by the two atoms 20 and 25 as CV.

```
DISTANCE LIST 20 25 POINT_FROM_AXIS 26
```

In a similar way it is possible to calculate the projection of this point on an axis to be used as a CV by using the keyword `PROJ_ON_AXIS`.

Example.

The following line instruct PLUMED to use the projection of the coordinate of atom 26 on the axis defined by the two atoms 20 and 25 as CV.

```
DISTANCE LIST 20 25 PROJ_ON_AXIS 26
```

Similarly to all the other variables, these two keywords may accept groups instead of atom indexes.

4.3 Minimum distance

The `MINDIST` keyword instructs PLUMED to use the minimum distance between two groups of atoms as a CV. To ensure differentiability, this quantity is implemented as:

$$s = \frac{\beta}{\log \sum_{ij} \exp(\beta/||r_{ij}||)},$$

where by default $\beta = 500$. The value of β can however be tuned if needed by using the optional keyword **BETA**. Much like the distance variable when calculating the minimum distance one must define two groups using the **LIST** keyword and the syntax described in section 4.

Example.

The following lines instruct PLUMED to use the minimum distance between atom number 13 and the set of atoms in list <g1> as a CV.

```
MINDIST LIST 13 <g1> SIGMA 0.35 BETA 500.
g1->
17 20 22 30
g1<-
```

The optional flag **NOPBC** can be used to calculate the distance without applying periodic boundary conditions. This should be only be done if all the atoms in the groups are part of the same molecule. See also Sec. 4.22.

4.4 Angles

The **ANGLE** keyword instructs PLUMED to use the angle defined by the centers of mass of three groups of atoms as a CV. The compulsory **LIST** keyword must be followed by three properly defined groups (see Section 4).

Example.

The following lines instruct PLUMED to use the angle defined by atom number 102 and the centers of mass of the atoms in groups g1 and g2 as a CV.

```
ANGLE LIST <g1> <g2> 102 SIGMA 0.05
g1->
13 15
g1<-

g2->
LOOP 1000 3000 3
g2<-
```


4.5 Torsion

The `TORSION` keyword instructs `PLUMED` to use a dihedral angle as the CV. This angle can either be defined by four atoms or, more generally, by the positions of the centers of mass of four groups of atoms. The compulsory `LIST` keyword must be followed by four properly defined groups (see Section 4).

Example.

The following lines instruct `PLUMED` to use the torsion angle about the centers of mass of the four groups `<g1>`, `<g2>`, `<g3>`, `<g4>` as a CV.

```
TORSION LIST <g1> <g2> <g3> <g4> SIGMA 0.35
```

It is also possible to use the sine or cosine of the torsional angle as a collective coordinate by including the `SIN` or `COS` keywords respectively.

Example.

The following line instructs `PLUMED` to use the cosine of the torsion angle about the centers of mass of the four groups `<g1>`, `<g2>`, `<g3>`, `<g4>` as a CV.

```
TORSION LIST <g1> <g2> <g3> <g4> COS SIGMA 0.35
```

4.6 Coordination number

The `COORD` keyword instructs `PLUMED` to use the total number of contacts between the atoms in group \mathcal{G}_1 and those in group \mathcal{G}_2 - the coordination number between these two groups. To ensure differentiability, this is implemented as the sum:

$$s = \sum_{i \in \mathcal{G}_1} \sum_{j \in \mathcal{G}_2} s_{ij},$$

where this sum is extended to all pairs of atoms with $i \in \mathcal{G}_1$ and $j \in \mathcal{G}_2$. The individual contributions s_{ij} are defined using a switching function, which, in the present case, is given by:

$$s_{ij} = \begin{cases} 1 & \text{for } r_{ij} \leq 0 \\ \frac{1 - (\frac{r_{ij}}{r_0})^n}{1 - (\frac{r_{ij}}{r_0})^m} & \text{for } r_{ij} > 0 \end{cases}$$

where $r_{ij} = |r_i - r_j| - d_0$. The user must supply the r_0 , d_0 , n and m parameters, using the additional keywords `R_0`, `D_0`, `NN` and `MM` respectively and thus has a great deal of control over the definition of the switching function. In general a good first guess for these parameters can be achieved by looking at the pair distribution function and setting d_0 equal to the position of the first peak in the pair distribution function, r_0 as the full width at half maximum of the peak and n and m to force $s_{ij} \simeq 0$ at the first minimum of the pair distribution function. However, oftentimes different choices for these parameters will lead to better results because of certain specific properties of the system of interest. An optional keyword `PAIR` treats the atoms in a pairwise fashion so that instead of simply counting the number of bonds between two groups of atoms one can define which precise bonds between the two groups should be monitored. In this case the groups, `<g1>` and `<g2>` must have the same number of atoms as the switching functions are on the distances between the i th atom of group `<g1>` and the i th atom of group `<g2>`.

Example.

The following lines instruct PLUMED to use the coordination of the atoms in group `g1` – 13 and 15 – with the atoms in group `solvent` as the CV.

```
COORD LIST <g1> <solvent> NN 6 MM 12 D_0 2.5 R_0 0.5 SIGMA 0.35
g1->
13 15
g1<-

solvent->
LOOP 1000 3000 3
solvent<-
```

The optional flag `NOPBC` can be used to calculate the distance without applying periodic boundary conditions. This should only be done when all the atoms are part of the same molecule. See also Sec. 4.22.

4.7 Hydrogen bonds

`HBONDS` is the keyword for a variable that counts the number of intra-molecular hydrogen bonds between a group of hydrogen bond donors and a group of hydrogen bond acceptors. This is defined as:

$$s = \sum_{ij} \frac{1 - \left(\frac{d_{ij}}{r_0}\right)^n}{1 - \left(\frac{d_{ij}}{r_0}\right)^m},$$

where $i \in \mathcal{D}$ is the group of donors and $j \in \mathcal{A}$ are the acceptors.

The two groups must be defined using the compulsory **LIST** keyword followed by two groups (see Section 4). **PLUMED** then assumes that there is only one donor/acceptor per residue and, that within the list, the donor/acceptor atoms on neighboring residues are consecutive. The values of r_0 , n and m can be specified using the **R_0**, **NN** and **MM** keywords. If no value is given for r_0 , n and m the default values of $r_0 = 2.5$, $n = 6$ and $m = 12$ are assumed.

The **TYPE** keyword selects which residues to include in the count:

- With **TYPE 0**, all donor-acceptor pairs are included;
- If **TYPE 1** is specified only those donor-acceptor pairs separated by an odd number of residues greater than 4 are counted. This allows one to monitor parallel β -sheet formations.
- If **TYPE 2** is specified only those donor-acceptor pairs that are separated by exactly 4 residues are included. This allows the formation of α -helical conformations (α type) to be monitored;
- If **TYPE 3** is specified, only those donor-acceptor pairs that are separated by an even number of residues greater than 4 are counted. This allows anti-parallel β -sheet formations (β -even type) to be monitored.
- If **TYPE 4** is specified, only the first donor and the first acceptor and so on are counted. This allows to monitor a set of native hydrogen bonds.

Example.

The following lines instruct **PLUMED** to use the count of the total number of hydrogen bonds between the pairs in groups **H** and **O** as a CV. The default switching function with $r_0 = 2.5$, $n = 6$ and $m = 12$ is implied.

```
HBONDS LIST <H> <O> TYPE 0 SIGMA 0.1
H->
6 10
H<-

O->
8 12
O<-
```

In the following example, a modified switching function is employed.

```
HBONDS LIST <H> <O> TYPE 0 SIGMA 0.1 NN 8 MM 20 R_0 2.5
```

The optional flag `NOPBC` can be used to calculate the distance without applying periodic boundary conditions. This should be done only when the atoms are part of the same molecule. See also Sec. 4.22.

4.8 Interfacial water

`WATERBRIDGE` is the keyword for a CV that counts the number of interfacial contacts. This variable does this by calculating the number of atoms from group \mathcal{G}_0 that are simultaneously in contact with atoms from both groups \mathcal{G}_1 and \mathcal{G}_2 . A typical application of this CV is to count the number of water molecules at the interface of two surfaces. This is calculated using:

$$s_{\text{WatBr}} = \sum_i^{n_0} \left(\sum_j^{n_1} \frac{1 - \left(\frac{|\mathbf{r}_i - \mathbf{r}_j|}{r_0}\right)^n}{1 - \left(\frac{|\mathbf{r}_i - \mathbf{r}_j|}{r_0}\right)^m} \right) \left(\sum_j^{n_2} \frac{1 - \left(\frac{|\mathbf{r}_i - \mathbf{r}_j|}{r_0}\right)^n}{1 - \left(\frac{|\mathbf{r}_i - \mathbf{r}_j|}{r_0}\right)^m} \right).$$

The syntax of the command requires the user to specify three groups of atoms after the keyword `LIST` starting with the \mathcal{G}_1 and \mathcal{G}_2 groups and closing with the \mathcal{G}_0 group. (see Section 4). The parameters of the switching function are then defined using the usual `NN`, `MM` and `R_0` keywords.

Example.

The following command instructs `PLUMED` to use the number of atoms in the `solvent` group that are simultaneously in contact with either atom 6 or 10 and either atom 8, 15 or 21 as a CV.

```
WATERBRIDGE LIST <type1> <type2> <solvent> NN 8 MM 12 R_0 4.0 SIGMA 0.1
type1->
6 10
type1<-

type2->
8 15 21
type2<-

solvent->
LOOP 100 1000 3
solvent<-
```

4.9 Radius of gyration

One can employ the radius of gyration of a group of atoms defined with the compulsory additional keyword `LIST` by using the `RGYR` directive. The `LIST`

keyword (see Section 4) must be followed by only one properly defined group. This CV is calculated using :

$$s_{\text{Gyr}} = \left(\frac{\sum_i^n |r_i - r_{\text{COM}}|^2}{\sum_i^n m_i} \right)^{1/2},$$

where the sums are over the n atoms in group \mathcal{G} and the center of mass is defined using:

$$r_{\text{COM}} = \frac{\sum_i^n r_i m_i}{\sum_i^n m_i}.$$

Example.

The following lines instruct PLUMED to use the radius of gyration of the group <g1> as a CV.

```
RGYR LIST <g1> SIGMA 0.35
```

As an alternative to the radius of gyration one might be interested in using the trace of the inertia tensor as a CV. It can be shown that this quantity is equivalent to the quantity calculated using:

$$s_{\text{Tr}[I]} = 2(s_{\text{gyration}})^2 \sum_{i \in G} m_i.$$

One can also use this quantity as a CV in PLUMED by using the INERTIA directive, which has the same syntax as the gyration radius.

Example.

The following line instructs PLUMED to use the radius of gyration of the group <g1> as a CV.

```
INERTIA LIST <g1> SIGMA 0.35
```

N.B. The radius of gyration is calculated without applying periodic boundary conditions so the atoms in group <g1> should all be part of the same molecule. See also Sec. 4.22.

4.10 Dipole

DIPOLE instructs PLUMED to use the electrical dipole generated by a group of atoms as a CV:

$$s_{dipole} = \left| \sum_i^n \mathbf{r}_i q_i \right|.$$

Only the LIST keyword followed by one properly defined group is required to define this CV (see Section 4).

Example.

The following lines instruct PLUMED to use the dipole of the group <g1> as a CV.

```
DIPOLE LIST <g1> SIGMA 0.35
```

4.11 Dihedral correlation

DIHCOR is the keyword for a CV that measures the similarity between adjacent dihedral angles:

$$s_{DC} = \sum_{i=2}^{N_D} \frac{1}{2} (1 + \cos(\phi_i - \phi_{i-1})).$$

The syntax for this CV requires the user to specify the number of dihedrals N_D and, in the subsequent N_D lines, the indices of the four atoms defining each dihedral ϕ_i .

Example.

The following lines instruct PLUMED to use the dihedral correlation for the 3 dihedrals listed as a CV.

```
DIHCOR NDIH 3 SIGMA 0.1
168 170 172 188
170 172 188 190
172 188 190 197
```

4.12 Alpha-beta similarity

ALPHABETA is a keyword for a CV that measures the similarity of dihedral angles to a reference value (see also 4.16). It is calculated using:

$$s_{\alpha\beta} = \frac{1}{2} \sum_{i=1}^{N_D} \left(1 + \cos \left(\phi_i - \phi_i^{\text{Ref}} \right) \right).$$

The syntax for this CV requires the user to specify the number of dihedrals N_D and, in the subsequent N_D lines, the indices of the four atoms defining each dihedral followed by the value of the dihedral in the reference conformation ϕ_i^{Ref} .

Example.

The following lines instruct PLUMED to use the Alpha-beta similarity of three dihedrals as a CV.

```
ALPHABETA NDIH 3 SIGMA 0.1
168 170 172 188 3.14
170 172 188 190 .56
188 190 192 230 3.14
```

4.13 Alpharmsd

ALPHARMSD is the keyword for a CV that counts the number of 6-residue segments in the protein chain that resemble an ideal alpha helix (i.e. the average experimental structure). This CV is calculated using:

$$s_{\alpha \text{rmsd}} = \sum_{\alpha} n \left[\text{RMSD} \left(\{\mathbf{R}_i\}_{i \in \Omega_{\alpha}}, \{\mathbf{R}^0\} \right) \right],$$

where n is the coordination (switching) function defined in Sec. 4.6. The sum over α in the above runs over all the 6-residue segments in the protein chain, where each of the residues is defined based on the positions of the backbone atoms N, CA, C, O and CB, Ω_{α} . The distance used in the switching function is then the root mean square difference between the distance matrix of the atoms in the set Ω_{α} with the corresponding distances between these atoms in the ideal alpha helix $\{\mathbf{R}^0\}$:

$$\text{RMSD} \left(\{\mathbf{R}_i\}_{i \in \Omega_{\alpha}}, \{\mathbf{R}^0\} \right) = \sqrt{\frac{1}{N_{\text{pairs}}} \sum_{i,j \in \Omega_{\alpha}} \left(d_{ij} - d_{ij}^0 \right)^2}.$$

See Ref.[24] for more details (although note that in PLUMED the RMSD between distance matrices is used rather than the cartesian RMSD. However, these two measures are essentially equivalent).

To use this CV the syntax requires the user to specify the coordination function parameters `R_0`, `D_0`, `NN` and `MM` (see Sec. 4.6) (we suggest values of `R_0`=0.8 Angstrom, `NN`=8, `MM`=12, `D_0`=0), and a conversion factor `ANGSTROM_SCALE` which converts the ideal alpha positions from Angstrom to the length units of the MD code (e.g. in Gromacs units are nanometers therefore `ANGSTROM_SCALE` is 0.1). In addition a list of list of atom indices for the N, CA, C, O and CB (in this order) for each residue must be provided for all the consecutive residues (in ascending order) which form the chain. For those residues, such as glycine, which do not have a CB the atom index for the corresponding hydrogen should be used.

Important note: for those MD codes which, like Gromacs4, do not keep the protein whole but instead split it by the PBC, `ALPHARMSD` requires the additional option `NOPBC`, together with the `ALIGN_ATOMS` command for all the atoms employed in the `ALPHARMSD`.

Example.

The following lines define an Alpharmsd CV for all 16 consecutive residues of a protein in the MD code Gromacs4 (which uses nanometers as the length unit).

```
ALPHARMSD LIST <ncacocb> SIGMA 0.5 R_0 0.08 NN 8 MM 12 ANGSTROM_SCALE 0.1 NOPBC
ncacocb->
1 5 23 24 7
25 27 42 43 29
44 54 56 57 51
58 68 70 71 65
72 74 77 78 76
79 81 101 102 83
103 105 116 117 107
118 120 138 139 122
140 142 162 163 144
164 166 179 180 168
181 183 190 191 185
192 194 214 215 196
216 218 225 226 220
227 229 236 237 231
238 240 243 244 242
245 247 267 268 249
ncacocb->
```


4.14 Antibetarmsd

ANTIBETARMSD counts the number of pairs of 3-residue segments in the protein chain which are similar to the ideal antiparallel beta (i.e. the average experimental structure). See Ref.[24] for details (but remember that here the RMSD among distance matrices is used instead of the cartesian RMSD as the two are basically equivalent). The definition, input parameters and syntax for this CV are the same as for the ALPHARMSD and again we suggest values of R_0=0.8 Angstrom, NN=8, MM=12, D_0=0 for the parameters. The only difference in the implementation is the additional option STRANDS_CUTOFF which allows the user to specify a threshold distance beyond which pairs of 3-residue segments are considered far. This option considerably speeds up the computation (often 1 nm is good choice for this quantity).

Important note: for those MD codes which, like Gromacs4, do not keep the protein whole but instead split it by the PBC, ANTIBETARMSD requires the additional option NOPBC, together with the ALIGN_ATOMS command for all the atoms employed in the ANTIBETARMSD.

Example.

The following lines define an Antibetarmsd CV for all 16 consecutive residues of a protein in the MD code Gromacs4 (which uses nanometers as the length unit).

```
ANTIBETARMSD LIST <ncacocb> SIGMA 0.5 R_0 0.08 NN 8 MM 12 ANGSTROM_SCALE 0.1
STRANDS_CUTOFF 1. NOPBC
ncacocb->
1 5 23 24 7
25 27 42 43 29
44 54 56 57 51
58 68 70 71 65
72 74 77 78 76
79 81 101 102 83
103 105 116 117 107
118 120 138 139 122
140 142 162 163 144
164 166 179 180 168
181 183 190 191 185
192 194 214 215 196
216 218 225 226 220
227 229 236 237 231
238 240 243 244 242
245 247 267 268 249
ncacocb->
```

4.15 Parabetarmsd

PARABETARMSD counts the number of pairs of 3-residue segments in the protein chain which are similar to the ideal parallel beta (i.e. the average experimental structure). See Ref.[24] for details (but remember that here the RMSD among distance matrices is used instead of the cartesian RMSD as the two are basically equivalent). The definition, input parameters, and syntax are the same as for theALPHARMSD and again we suggest values of R_0=0.8 Angstrom, NN=8, MM=12, D_0=0 for the parameters. The only difference in the implementation is the additional option STRANDS_CUTOFF which allows the user to specify a threshold distance beyond which pairs of 3-residue segments are considered far. This option considerably speeds up the computation (often 1 nm is good choice for this quantity).

Important note: for those MD codes which, like Gromacs4, do not keep the protein whole but instead split it by the PBC, PARABETARMSD requires the additional option NOPBC, together with the ALIGN_ATOMS command for all the atoms employed in the PARABETARMSD.

Example.

The following lines define a Parabetarmsd CV for all 16 consecutive residues of a protein in the MD code Gromacs4 (which uses nanometers as the length unit).

```
PARABETARMSD LIST <ncacocb> SIGMA 0.5 R_0 0.08 NN 8 MM 12 ANGSTROM_SCALE 0.1
STRANDS_CUTOFF 1. NOPBC
ncacocb->
1 5 23 24 7
25 27 42 43 29
44 54 56 57 51
58 68 70 71 65
72 74 77 78 76
79 81 101 102 83
103 105 116 117 107
118 120 138 139 122
140 142 162 163 144
164 166 179 180 168
181 183 190 191 185
192 194 214 215 196
216 218 225 226 220
227 229 236 237 231
238 240 243 244 242
245 247 267 268 249
ncacocb->
```

4.16 Torsional RMSD

RMSDTOR is the keyword for a CV whose value is given by the root mean square deviation of a set of dihedral angles from a reference configuration. This is calculated using:

$$s_{\text{TR}} = \sqrt{\frac{\sum_i^{N_D} (\theta_i - \theta_i^{\text{Ref}})^2}{N_D}}.$$

The syntax for this CV, like that for ALPHABETA, requires the user to specify the number of dihedrals N_D and, in the subsequent N_D lines, the indices of the four atoms defining each dihedral followed by the value of the dihedral in the reference conformation θ_i^{Ref} .

Example.

The following lines instruct PLUMED to use the torsional root mean square deviation of two dihedrals as a CV.

```
RMSDTOR NDIH 2 SIGMA 0.1
168 170 172 188 0.4
178 180 182 188 1.4
```

4.17 Electrostatic potential

Using the ELSTPOT keyword one can instruct PLUMED to use the electrostatic potential exerted by a group of atoms on the center of mass of a second group of atoms (or single atom) as a CV. This CV is calculated using:

$$s_{\text{ELST}} = \sum_i^{N_A} \frac{q_i}{|\mathbf{r}_i - \mathbf{r}_{\text{COM}}|} * f(|\mathbf{r}_i - \mathbf{r}_{\text{COM}}|, R_0, CUT)$$

where

$$\mathbf{r}_{\text{COM}} = \frac{\sum_i^{N_B} \mathbf{r}_i m_i}{\sum_i^{N_B} m_i}.$$

Here the sum in the first equation above is over the N_A atoms in the group whose charges exert the electric potential, while in the second equation the sum is over the N_B atoms in the group that defines the point at which this

potential is felt. $f(x)$ is a smoothing function defined as:

$$f(x, R_0, CUT) = \begin{cases} 1.0 & x < R_0 \\ \cos\left(\frac{\pi x}{2(CUT-R_0)}\right) & R_0 \leq x \leq CUT \\ 0 & x > CUT \end{cases}$$

where R_0 is the onset and CUT is a cutoff distance.

Example.

The following lines instruct PLUMED to use the electrostatic potential exerted by the atoms in group2 on the center of mass of the atoms in group1 as a CV:

```
ELSTPOT LIST <group1> <group2> R_0 4.0 CUT 12.0 SIGMA 0.01
group1->
LOOP 1 8 1
group1<-
group2->
LOOP 9 16 1
group2<-
```

4.18 Puckering coordinates

PUCKERING refers to the set of collective coordinates for 6-membered rings in polar coordinates [25]. Given the coordinates z_j , that represent the displacements of the j -th atom from the mean ring plane, three variables Q, θ, ϕ can be obtained starting from the general definition for 6-membered rings

$$q_2 \cos \phi_2 = \sqrt{\frac{1}{3}} \sum_{j=1}^6 z_j \cos \left[\frac{2\pi}{3}(j-1) \right]$$

$$q_2 \sin \phi_2 = -\sqrt{\frac{1}{3}} \sum_{j=1}^6 z_j \sin \left[\frac{2\pi}{3}(j-1) \right]$$

$$q_3 = \sqrt{\frac{1}{6}} \sum_{j=1}^6 (-1)^{j-1} z_j$$

as

$$Q = \sqrt{q_2^2 + q_3^2} \geq 0$$

$$\theta = \arctan(q_2/q_3) \in [0, \pi]$$

$$\phi = \phi_2 \in [0, 2\pi)$$

Each one can be selected as a TYPE of PUCKERING. The CV accept the LIST and SIGMA keywords. As of now LIST accept only 6 atoms. The atoms in the list have to be enumerated following the chemical sequence (although the numbering scheme in the topology does not have to be sequential). In order to fulfill IUPAC convention for sugar hexopyranose rings the first atom in the list has to be the ring oxygen, followed by the anomeric carbon. Q is in general a fast degree of freedom.

Example.

The following lines define a PUCKERING CV:

```
PUCKERING LIST <group1> TYPE PHI SIGMA 0.1
group1->
1 2 3 4 5 6
group1<-
```

4.19 Path collective variables

One can instruct PLUMED to use path collective variables [26] using the S_PATH and Z_PATH keywords. In this scheme one defines a path as a set of N reference conformations that define the path in configuration space \mathcal{X} from some initial state to some final state. The s variable (defined with S_PATH) then measures the position *along* the path, and is defined as:

$$s = \mathcal{Z}^{-1} \sum_{i=1}^N i e^{-\lambda d(X_i, X(t))},$$

where $X(t)$ is the configuration of the system at any given time, $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbf{R}_0^+$ is a metric on \mathcal{X} , and $\mathcal{Z} = \sum_{i=1}^N e^{-\lambda d(X_i, X(t))}$ is a normalization factor in which the prefactor, λ , should be chosen so as to have $\lambda d(X_i, X_{i\pm 1}) \simeq 2.3$ on average.

By contrast the z variable (defined with Z_PATH) measures the position *off* the path, and is defined as:

$$z = -\lambda^{-1} \log \mathcal{Z}.$$

For both S_PATH and Z_PATH the following keywords must be used:

- **TYPE**, which defines the metric used to calculate distances in configuration space. The following sections provide more details on this but, suffice to say, currently one can use either **RMSD** (root mean square deviation see section 4.19.1), **DRMSD** (distance root mean squared deviation see section 4.19.2) or **CMA**P (the distance between contact matrices see section 4.19.3) [27].
- **NFRAMES** which sets the number of reference structures, N , that are used in the definition of the path.
- **LAMBDA**, the prefactor λ in the exponential term of the equation that defines both s and z .
- Optionally, you can also use **NEIGHLIST** which defines a neighbor list on the closest frames to speed up the calculation followed by the number of steps in which the list must be calculated and the number of elements it must contain (e.g. **NEIGHLIST 50 10** means that each 50 steps the neighbour list must be calculated and it will contain the 10 closest elements to the current molecular dynamic snapshots. All the other are discarded up to the next neighbor list calculation.).

Other keywords are specific to the type of path variable being defined (*i.e.* root mean square displacement in Cartesian coordinates, RMSD in distances or contact map distance).

N.B. before using this feature please ensure that the parameters in `metadyn.h` (common files directory) are set properly. In particular look for the section containing:

```
// path dimensions
#define MAXATOMS_PATH 230
#define NMAX_PATH 8
#define MAXFRAMES_PATH 22
#define MAXATOMS_RMSD 230
#define MAXCHARS_PATH 40
// cmap
#define MAXDIM_CMAP 3800
#define MAXNUM_GROUP 10
#define MAXATOM_GROUP 30
```

and ensure that `MAXATOMS_PATH` is greater than or equal to the number of atoms per frame involved in your path and `MAXFRAMES_PATH` is greater than or equal to the number of frames you are using to define your path. In addition `NMAX_PATH` maximum number of path variables that you can use simultaneously. If you change any of these values you must subsequently recompile all the MD codes in which you have implemented `PLUMED` in order for your changes to take effect.

4.19.1 Root mean square deviation

As already briefly mentioned, when using the path collective coordinates `S_PATH` and `Z_PATH`, the command `RMSD` after the `TYPE` keyword instructs `PLUMED` to the the root mean square deviation of a subset of the atoms in the system (the *displacement* set B) calculated after the system has been aligned to another subset of the atoms (the *alignment* set A) as the metric used in the definition of the path. This quantity is calculated using

$$d(X_j, X_i) = \sum_{a=1}^{N_B} (X_a^{(j)} - M_{ij} X_a^{(i)})^2,$$

where M_{ij} is the roto-translation matrix calculated using the Kearsley [28] algorithm.

To use path CVs the user must specify the coordinates of the atoms in each of the reference frames of the path in a set of supplementary input files. The user should specify the basename of these files (`jbasenamei`) after the keyword `FRAMESET` and be aware that `PLUMED` then will expect to find N files named `jbasenamei.i.pdb`, which contain the coordinates of the atoms in both the *displacement* and *alignment* sets for the i th frame in the path. The particular set/s each atom is involved in is specified using the the last two numerical fields of the frameset file. Values of 1.0 and 0.0 indicate that the atom is to be used in the *alignment* set only, while values of 0.0 and 1.0 indicates that the atom is to be used in the *displacement* set only. Values of 1.0 and 1.0 indicate that the atom is to be used in both the *alignment* and *displacement* sets.

Version 1.1 and higher allow these alignment and displacement indicators to be non-integer. This allows users to perform a weighted alignment in cases where the alignment of one region of system is considered more important and a weighted displacement evaluation in cases where the displacement of a

particular atom/s is though to be of particular importance. Be aware however that this feature can produce strange numbers that are not trivial to interpret and is thus perhaps best left to experienced users.

N.B. PLUMED contains a hard coded limit on the number of atoms that can be used in the alignment so, like in the previous section, users should ensure that the value of this hard limit (`MAXATOMS_RMSD` in `metadyn.h`) is sufficiently large for their needs prior to compilation.

The unit of distance in the PDB files is Ångstrom. Engines like GROMACS, whose internal units are different, will perform appropriate conversions automatically.

Clearly, the atom indices in these PDB files must be the same as the indices of the atoms they refer to in the system topology. It is therefore likely that the atom indices in the frameset files may be non-consecutive.

Additional keywords are supported: `NO_ROT` and `NO_CENTER`. These two keywords prevent the rotation and the center of mass alignment respectively whenever one uses `RMSD`.

Example.

The following command defines a path using RMSD metrics. 2 frames are used to define the path, which has $\lambda = 9.0$.

```
S_PATH TYPE RMSD FRAMESET frame_ NFRAMES 2 LAMBDA 9.0 SIGMA 0.1
Z_PATH TYPE RMSD FRAMESET frame_ NFRAMES 2 LAMBDA 9.0 SIGMA 0.1
```

Two PDB files must be provided: `frame.1.pdb` and `frame.2.pdb`. The last two columns of these files specify which atoms are to be used for alignment and which are to be used to calculate the RMSD.

```
ATOM 1 C ALA 2 -0.186 -1.490 -0.181 1.00 0.00
ATOM 2 O ALA 2 -0.926 -2.447 -0.497 1.00 1.00
ATOM 15 N ALA 2 0.756 0.780 -0.955 1.00 0.00
ATOM 17 CA ALA 2 0.634 -0.653 -1.283 1.00 1.00
ATOM 19 CB ALA 2 2.063 -1.233 -1.286 1.00 1.00
END
```

4.19.2 Distance root mean square deviation

Instead of using the RMSD in the definition of the metric for the path in the `S_PATH` and `Z_PATH` CVs the command `DRMS` after the `TYPE` keyword allows one to use a metric based on the root mean square deviation of the distances

between a subset of the atoms in the system:

$$d(X_j, X_i) = \sum_a^{N_A} \sum_b^{N_A} (r_{ab}^{(j)} - r_{ab}^{(i)})^2,$$

where $r_{ab}^{(j)}$ is the distance of atoms a and b in the j -th reference frame. For this metric the coordinates of the atoms in the reference frames of the path are specified using the keyword **FRAMESET** along with a set of pdb files containing the atom coordinates. This is the same way that they are specified when the root mean square deviation metric is used (see section 4.19.1).

4.19.3 Contact map distances

The final choice one can employ to define the metric for the path in the **S_PATH** and **Z_PATH** CVs is to use the command **CMAP** after the **TYPE** keyword. This sets the metric for the path to be the distance between the contact matrices for a given subset of the atoms in the system:

$$d(X_j, X_i) = \|D_{ab}^{(j)} - D_{ab}^{(i)}\|.$$

Given two sets of atoms $a, b \in \mathcal{J}$, this contact matrix D_{ab} is calculated using:

$$D_{ab}(X) = \theta(c_{ab} - r_{ab}) w_{ab} \frac{(1 - (r_{ab}/r_{ab}^{(0)})_{ab}^n)}{(1 - (r_{ab}/r_{ab}^{(0)})_{ab}^m)},$$

where $\theta(x)$ is a step function which vanishes if $x < 0$.

As with other variables, the parameters $r_{ab}^{(0)}$, n_{ab} and m_{ab} allow for a great freedom in the definition of the switching function. What is more the parameter c_{ab} allows one to set the values of the switching function to zero at large separations. Finally, if the formation of particular contacts is deemed to be of great importance the weights w_{ab} can be used to change their relative importance.

Like the other metrics for the path collective variables to use path collective variables with a contact map metric information must be provided in supplementary files about the frames that make up the path. The syntax requires the user to specify a file name for the indices of the atoms and the parameters defining the calculation of the contact matrix (after the keyword **INDEX**) and another filename for the values of the reference matrices $D_{ab}^{(i)}$,

after the keyword **MAP**.

The *index* file, specified after the **INDEX** keyword, must contain one line for each of the elements in the contact matrix D_{ab} , which specified how that particular contact should be calculated. Each of these lines should begin with the **CONTACT** keyword followed by a numerical label for the contact. The next two fields are the indices a, b of the two atoms that make up the contact, which are followed by the values of $r_{ab}^{(0)}$, n_{ab} , m_{ab} , c_{ab} and w_{ab} in the switching function.

The *values* file, specified after the keyword **MAP**, contains the values of the reference matrices $D_{ab}^{(i)}$ used in the definition of the path. Each line in this file corresponds to one element in the contact matrix. These lines are formatted with the numerical label for the contact as the first field. This is followed by the indices of the two atoms that make up the contact a, b and finally the value of this particular switching function in the reference frame. Unlike the RMSD and DRMSD metrics all the reference frames are placed in a single file with each reference frame separated by the **END** keyword.

The optional flag **NOPBC** can be used to calculate the distance without applying periodic boundary conditions. This should be done only if all the atoms in the groups are part of the same molecule. See also Sec. 4.22.

Example.

The following command defines a path using the contact map metric. 2 frames are used to define the path with λ set equal to 0.1.

```
S.PATH TYPE CMAP NFRAMES 2 INDEX fr.ndx MAP fr.mps LAMBDA 0.1 SIGMA 1. NOPBC
```

The *fr.ndx* file should contain the details on how each of the switching functions in the contact matrix should be calculated:

```
CONTACT 1 1 2 3.0 6 10 100.0 1
CONTACT 2 1 15 3.0 6 10 100.0 1
CONTACT 3 17 2 3.0 6 10 100.0 1
CONTACT 4 15 19 3.0 6 10 100.0 1
```

The *fr.mps* file contains the values of the reference matrices:

```
1 1 2 0.99491645
2 1 15 0.76586085
3 17 2 0.79183088
4 15 19 0.81924184
END
1 1 2 0.99369661
2 1 15 0.76748693
3 17 2 0.76454272
4 15 19 0.72917217
END
```

One can also use contact matrices that involve contacts between the centers of mass of groups of atoms. However, in this case, a further additional file containing the definition of the groups must be provided. The name of this file is specified in the **PLUMED** input file, using the **GROUP** keyword. Then within the specified file each group is defined on a single line starting with the **GROUP** keyword. This keyword is followed by a numerical label for the group, the number of atoms in the group and then a list of the indices of the various atoms that make up the group. To instruct **PLUMED** to use these group contacts rather than atomic contacts one must use lines starting with the keyword **GROUP** in the index file. The remainder of the format of the lines in the index file and the format of the corresponding lines in the map file are identical to the lines used to specify atomic contacts. However, the indices that would have specified the indices of the atoms involved in the atomic contact must be replaced with the indices from the group file of the two groups that make up the contact.

Example.

The following command defines a path using the contact matrix metrics. 2 frames are used to define the path, and a value of $\lambda = 0.1$ is set.

```
S_PATH TYPE CMAP NFRAMES 2 INDEX fr.ndx MAP fr.mps GROUP fr.grp LAMBDA 0.1 SIGMA 1.
```

The file `fr.grp` defines three groups of atoms:

```
GROUP 1 4 23 43 56 457
GROUP 2 5 76 47 97 322 695
GROUP 3 4 17 15 19 2
```

The `fr.ndx` file contains the parameters of the contacts between groups:

```
GROUP 1 1 2 3.0 6 10 100.0 1
GROUP 2 1 3 3.0 6 10 100.0 1
GROUP 3 2 3 3.0 6 10 100.0 1
```

Finally, the `fr.mps` has the values of the contact matrix in the reference positions:

```
1 1 2 0.9949
2 1 3 0.7658
3 2 3 0.7918
END
1 1 2 0.9936
2 1 3 0.6674
3 2 3 0.8645
END
```

It is possible to have maps in which there are both atomic and group

contacts. However, be aware that, in the index files in which the switching functions are specified, the definitions of the group contacts **MUST** follow the definitions of the atomic **CONTACT** functions.

Example.

An example of an index file containing both atom-atom contacts and group contacts:

```
CONTACT 1 123 545 7.0 6 10 100.0 0.50
CONTACT 2 224 244 8.5 6 10 100.0 0.50
GROUP 3 1 2 3.0 6 10 100.0 1.00
```

4.19.4 Using path variables as RMSD, DRMSD and CMAP and the TARGETED statement

If one defines a z path collective variable with a single frame it is clear from the definition

$$z = -\lambda^{-1} \log \mathcal{Z} = -\lambda^{-1} \log \sum_f e^{-\lambda d_f}$$

that this is equivalent to using to the squared distance of the current configuration from a reference structure in the chosen metric. This CV can thus be used in simulations which employ the standard RMSD, distance RMSD or CMAP distance from a single frame as a collective coordinate. However, in this case we recommend use of the alias **TARGETED** instead, which defines the exact same CV but with a far simpler syntax.

Example.

The following command instructs PLUMED to do steered MD towards a target frame using RMSD metrics.

```
PRINT W_STRIDE 10
TARGETED TYPE RMSD FRAMESET ref_frame.pdb
STEER CV 1 TO 3.0 VEL 0.5 KAPPA 500.0
ENDMETA
```

A single PDB file must be provided: `ref_frame.pdb`, in which the last two columns specify which of the atoms are to be used for alignment and which are to be used to calculate the RMSD/DRMSD (see section 4.19.1).

```
ATOM 1 C ALA 2 -0.186 -1.490 -0.181 1.00 0.00
ATOM 2 O ALA 2 -0.926 -2.447 -0.497 1.00 1.00
ATOM 15 N ALA 2 0.756 0.780 -0.955 1.00 0.00
ATOM 17 CA ALA 2 0.634 -0.653 -1.283 1.00 1.00
ATOM 19 CB ALA 2 2.063 -1.233 -1.286 1.00 1.00
END
```

In the case of CMAP the input for PLUMED is

```
PRINT W_STRIDE 10
TARGETED TYPE CMAP INDEX CMAPINDEX MAP CMAPVALUES
STEER CV 1 TO 3.0 VEL 0.5 KAPPA 500.0
ENDMETA
```

where the CMAPVALUES file contains only one map (for details on the format of the CMAPINDEX and CMAPVALUES file see section 4.19.3).

4.20 Energy

The `ENERGY` keyword instructs PLUMED to use the total potential energy of the system [29, 30, 31, 32] as a CV. Currently this CV is available only in GROMACS4, AMBER and DL_POLY.

Example.

The following lines instruct PLUMED to use the potential energy of the system as a CV.

```
ENERGY SIGMA 100.0
```

4.21 Helix loops

The **HELIX** keyword instructs **PLUMED** to use the number of α -helix loops as a CV. A helix loop is formed when the pair of dihedral angles (Φ, Ψ) for three consecutive residues along the chain all adopt a particular pair of reference values $(\bar{\Phi}, \bar{\Psi})$. Typically in an alpha helix $\bar{\Phi}$ and $\bar{\Psi}$ have values of -1.200 and -0.785 respectively. Having specified reference values for the dihedrals the total number of loops is calculated using:

$$s = \sum_{i=2}^{N-1} \prod_{j=i-1}^{i+1} \frac{1}{4} \left[\cos(\Phi_j - \bar{\Phi}_i) + 1 \right] \left[\cos(\Psi_j - \bar{\Psi}_i) + 1 \right], \quad (4.1)$$

where N is the total number of residues. This CV requires the user to specify the number of loops with the **NLOOP** keyword. Then for each loop the user should provide three sets of four atoms that define the dihedrals $\Phi_{i-1}, \Phi_i, \Phi_{i+i}$ and a reference value for the dihedral $\bar{\Phi}_i$ along with three sets of four atoms that define the dihedrals $\Psi_{i-1}, \Psi_i, \Psi_{i+i}$ and a reference value the dihedral $\bar{\Psi}_i$.

Example.

The following lines instruct **PLUMED** to use the number of α -helix loops as a CV.

```
HELIX NLOOP 2 SIGMA 0.1
10 12 14 20 20 22 24 35 35 37 39 49 -1.200 12 14 20 22 22 24 35 37 37 39 49 51 -0.785
20 22 24 35 35 37 39 49 49 51 53 59 -1.200 22 24 35 37 37 39 49 51 51 53 59 61 -0.785
```

4.22 A note on periodic boundary conditions

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. However, there are some exceptions; namely:

- Average coordinate of a group of atoms;
- **RGYR**;
- **DISTANCE**, **MINDIST**, **COORD**, **HBONDS**, and path collective variables **S_PATH** and **Z_PATH** with contact map metrics (**TYPE CMAP**), if the **NOPBC** flag is used;

- Path Collective Variables `S_PATH` and `Z_PATH` with RMSD (`TYPE RMSD`);
- `ALPHARMSD`, `ANTIBETARMSD`, `PARABETARMSD`.

In all these cases, it is essential that the atoms involved in the definition of the CV are all part of a single, unbreakable object such as a molecule. Furthermore, it is essential that in the coordinates passed to `PLUMED` the molecules are kept intact. We are aware of at least two cases where this condition is not satisfied; namely, when using domain decomposition or the option for periodic molecules within the host code `GROMACS4`.

In these cases one must use the additional directive `ALIGN_ATOMS`, which takes the `LIST` keyword. By using this command the user can define an ordered group of atoms, which are to be aligned such that the distance between adjacent atoms in the list is minimized. In the majority of cases, the atoms in this list should be in the same order as they appear in the `pdb` file as usually these files are arranged in way that reflects how close together atoms are in the molecular topology. As for the number of atoms that must be specified in this list it is often sufficient to just specify those atoms involved in the CVs. However, in cases where the atoms involved are separated by a large distances along a chain alignment of intermediate atoms will also be required.

Example.

Input for running a metadynamics simulation using the end-to-end distance in a protein as a CV with `GROMACS4` and domain decomposition:

```
PRINT W_STRIDE 10
HILLS HEIGHT 2.0 W_STRIDE 10
DISTANCE LIST 9 238 SIGMA 0.35 NOPBC
ALIGN_ATOMS LIST <C-alpha>
C-alpha->
9 16 31 55 69 90 102 114 124 138 160 174 194 208 224 238
C-alpha<-
ENDMETA
```

Chapter 5

Postprocessing

5.1 Estimating the free energy after a metadynamics run

The program `sum_hills.f90` is a tool for summing up the Gaussians laid during the metadynamics trajectory and obtaining the free energy surface.

5.1.1 Installation instructions

As `sum_hills.f90` is a simple fortran 90 program, the installation is straightforward so long as you have a fortran compiler available on your machine. As an example, with the gnu g95 compiler one would compile `sum_hills.f90` using the following command:

```
g95 -O3 sum_hills.f90 serial.f90 -o sum_hills.x
```

For post processing of large HILLS files we recommend that, if you have a multicore machine available, you use the parallel version, which is compiled thus:

```
mpif90 -O3 sum_hills.f90 parallel.f90 -o sum_hills_mpi.x
```

5.1.2 Usage

The `sum_hills` program takes its input parameters from the command line. If run without options, this brief summary of options is printed out.

Example.

```
USAGE: sum_hills.x -file HILLS -out fes.dat -ndim 3 -ndw 1 2 -kt 0.6 -ngrid 100 100 100
```

```

-ndim 3          number of collective variables NCV
-ndw 1 ...       CVs for the free-energy surface
-ngrid 50 ...    mesh dimension.  DEFAULT :: 100
-dp ...         size of the mesh of the output free energy
-fix 1.1 ...     optional definition of the FES domain
-stride 10       how often the FES is written
-cutoff_e 1.e-6  the hills are cut off at 1.e-6
-cutoff_s 6.25   the hills are cut off at 6.25 std dev from the center
-2pi x          [0; 2 $\pi$ ] periodicity on the x CV,
                if -fix is not used 2pi is used
- $\pi$  x          [- $\pi$ ;  $\pi$ ] periodicity on the x CV,
                if -fix is not used 2pi is used
-kt 0.6         kT in the energy units
-grad           apply periodicity using degrees
-bias <biasfact> writing output the bias for a well tempered mtd
-file HILLS     input file
-out fes.dat    output file
-hills nhills   number of Gaussians that are read

```

The program works in the following way.

Using the `-file` and `-out` flags one tells the program the names of the input file containing the Gaussian hills file and the name of the file in which the free-energy will be outputted. In the absence of any instruction `sum_hills` assumes these files are to be called `HILLS` and `fes.dat`.

The number of CVs in the `HILLS` file is specified using `-ndim` whilst the number of CVs the output free energy surface is to be plotted as a function of is controlled by `-ndw` followed by the list of CVs in the desired order. Note that after being read the CV are reordered in the code according to `ndw`. As a direct result EVERY output uses this new order. If the number of CVs requested using `-ndw` is less than the number of CVs in in `HILLS` file the CVs not specified for output will be integrated out with the Boltzmann weight $K_b T$ specified by `-kt` (kT must be given in the energy units that were used by in the code in which the simulation was performed)

The position, width and height of the Gaussians are read from the file specified by the `-file` option (`HILLS` is the default) and the free-energy surface is printed out on a grid in gnuplot format with a blank line added after each block of data. Gnuplot is very handy for quick visualization of 3D data (e.g. the FES as a function of 2 CVs).

For efficiency, the Gaussians are truncated at a certain distance from their center before being placed on the grid. `-cutoff_s` and `-cutoff_e` allow one

to tune this cutoff distance. With `-cutoff_s` the value read specifies this truncation distance in terms of the number of standard deviations from the center. For consistency this should be set equal to the DP2CUTOFF used in the metadynamics simulation (the default value for this parameter is 6.25 both in PLUMED and in sum_hills). Alternatively, the user may specify this cutoff distance as the distance at which the energy of the Gaussian hill falls to less than some critical value specified using the `-cutoff_e` flag. The energy specified after this flag must be in the units used during the metadynamics simulation. N.B. if the cutoff used in post-processing is different to that used during the simulation the calculated free energy differs from the bias which was actually applied during the metadynamics simulation.

The size of the output grid can be controlled either with `-ngrid` followed by the number of grid points in each CV direction or by `-dp` followed by the size of the voxel in each CV direction. If neither of these options are specified, the grid size is assumed to be 100 in each direction and the voxel size is calculated such that all the input Gaussians fit into the grid. The `-fix` option allow one to fix the boundaries of the output free energy and after this flag. two real numbers are expected for each CV.

`-stride` allows one to print out the evolution of the free-energy as a function of time, *i.e.* as a function of the index of the Gaussian. `-stride` expects an integer which specifies how many Gaussians are added to the calculated free energy surface between print outs. The progressive free energy is printed in files named `fes.dat.XXX` where XXX is an increasing counter. The final free-energy is printed in `fes.dat`. This is useful for creating movies of the how the free-energy wells fill as a function of time.

`-hills` is used to integrate only a part of the Gaussian files. It expects an integer that specifies the maximum number of Gaussians to be read from input.

When periodic CVs like angles and dihedrals are used as CVs, periodicity options must be specified. The flags `-pi` and `-2pi`, which are followed by the CV index, specify that the CV is periodic between $[-\pi;\pi]$ or $[0;2\pi]$ respectively. The `-grad` flag specifies that the CV values are being given degrees rather than radians.

5.2 Evaluating collective variables on MD trajectories

The program `driver` can be used to evaluate the value of any of the CVs implemented in PLUMED for all the structures in a given trajectory file.

5.2.1 Installation instructions

Before compiling `driver`, the files contained in the `common_files` directory must be linked to the `utilities/driver` directory.

To compile using g95/gcc, type:

```
make g95
```

To compile with gfortran/gcc, type:

```
make gfortran
```

To compile with gfortran/gcc on a 64 bit machine, type:

```
make gfortan_64
```

To compile with ifort/icc Intel compilers, type:

```
make intel
```

To use other compilers and/or compiler flags you will need to modify the Makefile.

5.2.2 Usage

This program takes its input parameters from the command line. If run without options, this brief summary of options is printed out.

Example.

Invoking driver without arguments prints a list of the available options:

```

USAGE :
driver -pdb PDB_FILE -dcd DCD_FILE -plumed PLUMED_FILE -ncv
(-interval min1 max1 min2 max2 -out OUT_FILE -nopbc -cell CELLX CELLY CELLZ)

-pdb      pdb connected to dcd file
-dcd      trajectory file
-plumed   PLUMED-like input file
-ncv      number of collective variables
-out      pdb output filename for clustering format (optional)
-interval extract frames with CV in this interval (optional)
-nopbc    don't apply pbc (optional)
-cell     provide fixed box dimension in Angstrom
          for orthorhombic PBC (optional)

```

The user must provide a structure file in PDB format, within which the atoms' masses and charges are inserted in the occupancy field and in the B-factor field respectively. This data must be provided as it is required for the calculation of certain collective variables, such as the center of masses or dipoles.

The trajectory file must be a CHARMM format DCD file (also NAMD 2.1 and later). To convert other trajectory files into this format, the user can download the utility CatDCD from:

<http://www.ks.uiuc.edu/Development/MDTools/catdcd/>.

Within the input file driver the user can specify a printing stride (in number of DCD frames) using the keyword `W_STRIDE` and the list CVs he/she wishes to calculate. This file has the same syntax as the `PLUMED` input file. The CVs value will be printed on the `COLVAR` file. For a complete list of the CVs driver can calculate, please see section 4.

Example.

If you wish to evaluate the distance between atom 13 and 17 for every frame in your trajectory file, the input file for driver would be as follows:

```

PRINT W_STRIDE 1
DISTANCE LIST 13 17
ENDMETA

```

By default, **driver** uses orthorhombic periodic boundary conditions and looks for cell information in the DCD file. If these are not present, you must

either provide the fixed dimensions of the box in Angstrom (only orthorhombic PBCs are allowed) using the keyword `-cell CELLX CELLY CELLZ` or switch off the pbc using the `-nopbc` flag.

Driver can also be used to extract frames in a specific window of the CVs space and write these extracted frames to a PDB file. To use this functionality use the option `-out output.pdb` and specify the interval with `-interval CVmin CVmax`.

Example.

To extract those frames in which the distance between two specified atoms is between 10.0 and 12.0 Å while the angle defined by three specified atoms is between 2.0 and 2.3 rad, the user should prepare the following input file (named `plumed.dat`):

```
PRINT W_STRIDE 1
DISTANCE LIST 13 17
ANGLE LIST 19 20 22
ENDMETA
```

and then invoke `driver` using the following command:

```
./driver -pdb dialanine.pdb -dcd dialanine.dcd -plumed plumed.dat -ncv 2 -out
window.pdb -interval 10.0 12.0 2.0 3.0
```

The output file `window.pdb` is written in a format appropriate for use with the GROMACS tool `g_cluster`, which can be used to perform a cluster analysis on your set of frames.

5.3 Processing COLVAR files

Since PLUMED 1.2, a more flexible format has been adopted for COLVAR files. These files can be parsed with the simple `plumedat.sh` tool. If run it without options, this brief summary of options is printed out.

Example.

Invoking `plumedat.sh` without arguments prints a list of the available options:

```
syntax:
plumedat.sh f1 f2 ... < file
or
plumedat.sh -l < file
file is the name of the COLVAR file
f1, f2, ... are the names of the required fields
if a required field is not available in the COLVAR file, "NA" is written in the output
with -l, the available choices are listed
example:
plumedat.sh time temp < COLVAR
prints a two-column file, with the time in the first column and the temperature in the
second column
```

5.4 PLUMED as a standalone program

PLUMED can be run as a standalone program so that it can be easily integrated in a script. This is particularly useful whenever the time between one PLUMED call and the other is large (e.g. in ab initio programs) and one aims at minimizing the time needed for implementation. `plumed_standalone` consists in a simple program, very much like the **driver** tool, that reads the configuration and prints out the forces to be added and the other various output files typical of PLUMED.

5.4.1 Installation instructions

Before compiling `plumed_standalone`, the files contained in the `common_files` directory must be linked to the `utilities/standalone` directory.

To compile using `g95/gcc`, type:

```
make gnu
```

To compile with `ifort/icc` Intel compilers, type:

```
make intel
```

To use other compilers and/or compiler flags you will need to modify the Makefile. In this way you may obtain an executable which is `plumed_standalone`.

5.4.2 Usage

PLUMED standalone expects a standard PLUMED input, a configuration file for your system (much in the format of xyz) and gives a screen output. The configuration file has the following format:

Example.

Example of `plumed_standalone` input configuration

```
TIME 0.100000 100
AMPLI 1.000000
BOLTZ 0.001987
BOX 1000.000000 1000.000000 1000.000000
-13.523670417 0.140088464 4.263822219 131.207999 0.000000
-9.821878867 -0.695171589 5.042585958 101.119998 0.000000
-7.193416272 -2.392065561 2.830069705 163.190996 0.000000
-3.844793108 -3.737129154 4.056782255 129.197997 0.000000
:
:
```

The configuration file must have the keyword **TIME** followed by the step length (in the units of the program) and **AMPLI** which is the conversion factor between Å and the actual length unit in the configuration file. This is needed when a RMSD calculation with respect to a pdb file in Å must be performed. In this case, if the configuration is in Bohr, the **AMPLI** factor must be set equal to 1.889725989. **BOLTZ** is the Boltzmann factor in the chosen units and **BOX** followed by three numbers specifies the dimensions of the orthorombic simulation box.

The following lines specify the configuration. The first three columns are the x, y and z coordinates, the fourth is the mass in units of the program (in the case above a coarse grained program takes the mass of all the single residues) and in the last column you may put the charge (in the code above it was not needed). PLUMED invocation may be done with command line:

Example.

Example of invocation of PLUMED standalone

```
./plumed_standalone -coord x.xyz -plumed plumed_input.cfg
```

`plumed_standalone` returns back a file named `plumed_forces.dat` which contains the additional energy and forces from the bias potential calculated by the PLUMED module. Eventually, these must be summed to the ones of the original program.

Example.

Example of `plumed_standalone` output file. In the first line is the additional bias potential calculated by PLUMED, followed by the x, y and z components of the force for each atom.

```
2.35680
3.335998704981455 -3.684076849212145 -16.098665997131253
-0.9064447402311846 2.913567594563722 -1.0502785066801152
-0.7948355287665761 0.9861425018822194 0.6558434539267759
-1.9437110159112139 -0.6087819098852943 2.4119663558915296
:
:
```


Bibliography

- [1] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation, *J. Chem. Theory Comput.* 4 (3) (2008) 435–447.
- [2] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, K. Schulten, Scalable molecular dynamics with NAMD, *J. Comput. Chem.* 26 (16) (2005) 1781–802.
- [3] W. Smith, C. Yong, P. Rodger, *Mol. Simulat.* 28 (2002) 385–471.
- [4] S. J. Plimpton, Fast parallel algorithms for short-range molecular dynamics, *J. Comp. Phys.* 117 (1995) 1–19.
- [5] D. A. Case, T. A. Darden, T. E. C. III, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, K. M. Merz, D. A. Pearlman, M. Crowley, R. C. Walker, W. Zhang, B. Wang, S. Hayik, A. Roitberg, G. Seabra, K. F. Wong, F. Paesani, X. Wu, S. Brozell, V. Tsui, H. Gohlke, L. Yang, C. Tan, J. Mongan, V. Hornak, G. Cui, P. Beroza, D. H. Mathews, C. Schafmeister, W. S. Ross, P. Kollman, AMBER 9, University of California, San Francisco.
- [6] M. Harvey, G. Giupponi, G. D. Fabritiis, ACEMD: Accelerated molecular dynamics simulations in the microseconds timescale, *J. Chem. Theory and Comput.* 5 (2009) 1632.
- [7] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. D. Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto,

- C. Sbraccia, S. Scandolo, G. Sclauszero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials, *J. Phys.: Condens. Matter* 21 (2009) 395502.
- [8] A. Laio, M. Parrinello, Escaping free energy minima, *Proc. Natl. Acad. Sci. USA* 99 (2002) 12562–12566.
 - [9] G. Torrie, J. Valleau, Nonphysical sampling distributions in monte carlo free energy estimation: Umbrella sampling, *J. Comput. Phys.* 23 (1977) 187–199.
 - [10] S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, P. A. Kollman, Multidimensional free-energy calculations using the weighted histogram analysis method, *J. Comput. Chem.* 16 (1995) 1339–1350.
 - [11] B. Roux, The calculation of the potential of mean force using computer-simulations, *Comput. Phys. Comm.* 91 (1995) 275–282.
 - [12] C. Jarzynski, Nonequilibrium equality for free energy differences, *Phys. Rev. Lett.* 78 (1997) 2690–2693.
 - [13] G. Crooks, Nonequilibrium measurements of free energy differences for microscopically reversible markovian systems, *J. Stat. Phys.* 90 (5-6) (1998) 1481–1487.
 - [14] A. Barducci, G. Bussi, M. Parrinello, Well-tempered metadynamics: A smoothly converging and tunable free-energy method, *Phys. Rev. Lett.* 100 (2) (2008) 020603.
 - [15] P. Raiteri, A. Laio, F. Gervasio, C. Micheletti, M. Parrinello, Efficient reconstruction of complex free energy landscapes by multiple walkers metadynamics, *J. Phys. Chem. B* 110 (2006) 3533–3539.
 - [16] G. Bussi, F. L. Gervasio, A. Laio, M. Parrinello, Free-energy landscape for beta hairpin folding from combined parallel tempering and metadynamics, *J. Am. Chem. Soc.* 128 (41) (2006) 13435–41.
 - [17] S. Piana, A. Laio, A bias-exchange approach to protein folding, *J. Phys. Chem. B* 111 (17) (2007) 4553–9.

- [18] M. Marchi, P. Ballone, Adiabatic bias molecular dynamics: A method to navigate the conformational space of complex molecular systems, *J. Chem. Phys.* 110 (8) (1999) 3697–3702.
- [19] J. VandeVondele, M. Krack, F. Mohamed, M. Parrinello, Quickstep: Fast and accurate density functional calculations using a mixed gaussian and plane waves . . . , *Comp. Phys. Comm.* 167 (2005) 103–128.
- [20] V. Babin, C. Roland, C. Sagui, Adaptively biased molecular dynamics for free energy calculations, *J. Chem. Phys.* 128 (2008) 134101.
- [21] Y. Crespo, F. Marinelli, F. Pietrucci, A. Laio, Metadynamics convergence law in a multidimensional system, *Phys. Rev. E* 81 (2010) 055701(R).
- [22] C. Camilloni, D. Provasi, G. Tian, R. A. Broglia, Exploring the protein G helix free-energy surface by solute tempering metadynamics, *Proteins* 71 (2008) 1647.
- [23] A. Grossfield, Wham.<http://membrane.urmc.rochester.edu/Software/WHAM/WHAM.html>.
- [24] F. Pietrucci, 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) (2009) 2197–2201.
- [25] M. Sega, E. Autieri, F. Pederiva, On the calculation of puckering free energy surfaces, *J. Chem. Phys.* 130 (22) (2009) 225102.
- [26] D. Branduardi, F. L. Gervasio, M. Parrinello, From A to B in free energy space, *J. Chem. Phys.* 126 (5) (2007) 054103.
- [27] M. Bonomi, D. Branduardi, F. Gervasio, M. Parrinello, The unfolded ensemble and folding mechanism of the C-terminal GB1 β hairpin, *J. Am. Chem. Soc.* 130 (42) (2008) 13938–13944.
- [28] S. K. Kearsley, On the orthogonal transformation used for structural comparison, *Acta Cryst. A* 45 (1989) 208–210.
- [29] H. Li, D. Min, Y. Liu, W. Yang, Essential energy space random walk via energy space metadynamics method to accelerate molecular dynamics simulations, *J. Chem. Phys.* 127 (9) (2007) 094101.

- [30] C. Michel, A. Laio, A. Milet, Tracing the entropy along a reactive pathway: The energy as a generalized reaction coordinate, *J. Chem. Theory Comput.* 5 (9) (2009) 2193–2196.
- [31] D. Donadio, P. Raiteri, M. Parrinello, Topological defects and bulk melting of hexagonal ice, *J. Phys. Chem. B* 109 (12) (2005) 5421–5424.
- [32] M. Bonomi, M. Parrinello, Enhanced sampling in the well-tempered ensemble, *Phys. Rev. Lett.* 104 (2010) 190601.

Index

- Absolute position, *see* Collective variables, Absolute position
- Adiabatic bias molecular dynamics, 37
- Alpha-beta similarity, *see* Collective variables, Alpha-beta similarity
- Alpharmsd, *see* Collective variables, Alpharmsd
- Angle, *see* Collective variables, Angle
- Antibetarmsd, *see* Collective variables, Antibetarmsd
- Atom lists, 42
- Bias exchange simulations, 33
- Collective variables
 - Absolute position, 44
 - Alpha-beta similarity, 54
 - Alpharmsd, 54
 - Angle, 47
 - Antibetarmsd, 56
 - Contact Matrix distance, 67
 - Coordination number, 48
 - DRMSD, 67
 - Difference of distances, 45
 - Dihedral correlation, 53
 - Dipole, 53
 - Distance from an axis, 46
 - Distance, 45
 - Electrostatic potential, 58
 - Hydrogen bonds, 49
 - Interfacial water, 51
 - Minimal distance, 46
 - Parabetarmsd, 57
 - Path variables, 60
 - Projection on an axis, 46
 - Puckering coordinates, 59
 - RMSD, 67
 - Radius of Gyration, 51
 - Targeted, 67
 - Torsional Root mean square deviation, 58
 - Torsion, 48
- Contact Matrix distance, *see* Collective variables, Contact Matrix distance
- Coordination number, *see* Collective variables, Coordination number
- Difference of distances, *see* Collective variables, Difference of distances
- Dihedral correlation, *see* Collective variables, Dihedral correlation
- Dipole, *see* Collective variables, Dipole
- Distance, *see* Collective variables, Distance
- Distance from an axis, *see* Collective variables, Distance from an axis
- DRMSD, *see* Collective variables, DRMSD

Electrostatic potential, *see* Collective variables, Electrostatic potential

Grid for metadynamics, 23

Hydrogen bonds, *see* Collective variables, Hydrogen bonds

Interfacial water, *see* Collective variables, Interfacial water

Keywords

ABMD, 37
ALIGN_ATOMS, 70
ALPHABETA, 54, 58
ALPHARMSD, 54, 70
ANGLE, 47
ANTIBETARMSD, 56, 70
BETA, 47
BIASXMD, 33
COMMITMENT, 41
COORD, 69
COS, 48
DIHCOR, 53
DIPOLE, 53
DIR, 45
DISTANCE, 45, 69
ELSTPOT, 58
ENDMETA, 19
ENERGY, 7, 68
EXTERNAL, 38
FILENAME, 25, 26
GRID, 23, 24
HBONDS, 49, 69
HELIX, 69
HILLS, 20, 21
INERTIA, 52
INVERT, 29, 30

LIST, 42, 70
LWALL, 38
MINDIST, 46, 69
NLOOP, 69
NOHILLS, 28, 29, 33
NOPBC, 45, 47, 49, 51, 65, 69
NOSPLINE, 24
NOTE, 19
NO_CENTER, 63
NO_ROT, 63
PARABETARMSD, 57, 70
POINT_FROM_AXIS, 46
POSITION, 44
PRINT, 19
PROJ_ON_AXIS, 46
PTMETAD, 32
PUCKERING, 59
READ_GRID, 26
RGYR, 51, 69
RMSDTOR, 58
SIGMA, 29, 42
SIMTEMP, 32
SIN, 48
STEERPLAN, 35
STEER, 34, 35
S_PATH, 60, 62–64, 69, 70
TARGETED, 67
TORSION, 48
UMBRELLA, 33
UWALL, 38
WATERBRIDGE, 51
WRITE_GRID, 25
W_STRIDE, 25
Z_PATH, 60, 62–64, 69, 70
#, 19

Minimal distance, *see* Collective variables, Minimal distance

Output files, Metadynamics, 20
 Parabetarmsd, *see* Collective variables,
 Parabetarmsd
 Parallel tempering metadynamics, 32
 Path variables, *see* Collective variables,
 Path variables
 Periodic boundary conditions, 69
 plumedat.sh, 20, 76
 Projection on an axis, *see* Collective
 variables, Projection on an axis
 Puckering coordinates, *see* Collective
 variables, Puckering coordinates

 Radius of Gyration, *see* Collective vari-
 ables, Radius of Gyration
 RMSD, *see* Collective variables, RMSD

 Standalone, 77
 Steered MD, 34
 Steerplan, 35

 Tabulated potentials, 38
 Targeted, *see* Collective variables, Tar-
 geted
 Torsion, *see* Collective variables, Tor-
 sion
 Torsional Root mean square deviation,
 see Collective variables, Tor-
 sional Root mean square de-
 viation

 Umbrella sampling, 33
 Units, 19

 Walls, 38