PLUMED User's Guide

A portable plugin for free-energy calculations with molecular dynamics

Version 1.1 – July 2009

Contents

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] and the SANDER module in AMBER [4]. It also works with the very fast CUDA/GPU MD code ACEMD (http://multiscalelab.org/acemd).

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 [5], umbrella sampling [6, 7, 8] and Jarzynski-equation based steered MD [9, 10].

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

The current release of PLUMED is compatible with GROMACS version 3.3.3 and 4.0 (up to release 4.0.5), AMBER version 9 and 10, NAMD version 2.6 and 2.7b1, DL_POLY versions 2.16, 2.19 and 2.20, ACEMD version 1.1 (supported by ACEMD developers).
These codes are not distributed with the PLUMED package, but they must be obtained separately. For further information, please refer to chapter 2.

1.3 Features

PLUMED can perform several different types of calculation:

- Metadynamics with a large variety of CVs;
- Well-tempered metadynamics [11];
- Multiple walkers metadynamics [12];
- Combined parallel tempering and metadynamics [13];
- Bias-exchange metadynamics [14];
- Steered MD;
- Umbrella sampling;
- Committor analysis.

1.4 New in version 1.1

PLUMED version 1.1 presents several new features, including new collective variables and support to new codes. Among these:

- `PLUMED` is now compatible with the domain-decomposition parallelization of GROMACS4.
- The bias can be interpolated using cubic splines on a grid (accelerates long metadynamics simulations with many hills).
- Sum over hills is parallelized on GROMACS and DL_POLY (accelerates long metadynamics simulations with many hills).
- More flexibility in the application of PBC to collective variable (useful for intramolecular coordinates).
- New directive ALIGN_ATOMS to work around codes which break molecules (such as GROMACS4 with some setup).
- Redesigned parallel tempering-metadynamics code, less memory required.
- Redesigned parser, with more error checking, plus a few features such as line continuation and NOTE keyword.
- Redesigned patching system, with more error checking.
- Output files are now opened once at the beginning of the simulation (more efficient, especially in replica exchange simulations where a huge number of files is opened in the same directory).
- Possibility of switching off metadynamics on a single variable by omitting the SIGMA flag.
- Driver: support to NPT runs with variable cell dimensions; number of CVs is now unlimited; pdb output name has a default when -interval is specified.
- Sum_hills has been parallelized.
- New collective variables: Electrostatic potential felt by a atom or group of atoms, Puckering variables (thanks to Marcello Sega).
- Added support to ACEMD 1.1, NAMD 2.7b1, AMBER 10, DL_POLY 2.20 and GROMACS 4.0.5.

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 4.0 cannot perform hamiltonian (lambda) replica exchange;
- It works only with the scalar version of AMBER.
- Only orthorhombic cells are supported in AMBER, ACEMD and DRIVER.

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 GROMACS, NAMD, AMBER (sander module) and DL_POLY. These examples, combined with a script adapted from CP2K [15], work also as a regtest for the plugin.
- `patches`. A collection of patches to interface PLUMED with different codes.
- `utilities`. Two small utilities written in Fortran: `sum_hills` and `driver`. The former 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. The latter 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.
- `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 Provasi, 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 in the course of the last seven years. Among these, we should 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.

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 (in press), arXiv:0902.0874.

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

PLUMED works as an add-on to some of the most popular MD codes: SANDER, GROMACS, NAMD, DL_POLY. These codes are not included in the plugin package, but can be downloaded from the internet:

NAMD 2.6/2.7b1
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/

ACEMD 1.1
http://multiscalelab.org/acemd

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

PLUMED works only with the serial build of AMBER 9.0 and 10.0. Different patches are available for the two different versions: plumedpatch_sander_9.sh and plumedpatch_sander_10.sh. The patching procedure is the same in both cases, thus in the following we shall indicate with plumedpatch_sander.sh any of these two patches.

- Extract the source code from its archive and then move into the AMBER root directory.
- Set the environmental variable plumedir to point to PLUMED_root.
- Copy or link the plumedpatch_sander.sh file from the patches folder to the current directory.
- Use the usual configuration to produce the Makefile in the src directory.
- Execute the script: "./plumedpatch_sander.sh -patch ".
- Proceed to compile the code following the AMBER instructions.

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

```
tar zxf 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
```

2.2 Compiling GROMACS with PLUMED

PLUMED works with GROMACS 3.3 and 4.0. Version 3.3 has been tested with the 3.3.2 and the 3.3.3 releases, version 4.0 up to release 4.0.5. Different patches are available for the two different versions: `plumedpatch_gromacs_3.3.3.sh` and `plumedpatch_gromacs_4.0.4.sh`. The patching procedure is the same in both cases, thus in the following we shall indicate with `plumedpatch_gromacs.sh` any of these two patches.

- Extract the source code from its archive and then move into the GROMACS `root` directory.
- Set the environmental variable `plumedir` to point to `PLUMED_root`.
- Copy or link the `plumedpatch_gromacs.sh` file from the `patches` folder to the current directory.
- Use the usual configuration to produce the Makefile.
- Execute the script: "`./plumedpatch_gromacs.sh -patch`".
- Proceed to compile and install the code, following the GROMACS instructions.

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

```
tar zxf 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
```

2.3 Compiling NAMD with PLUMED

PLUMED works with NAMD 2.6 and 2.7b1. Different patches are available for the two different versions: `plumedpatch_namd_2.6.sh` and `plumedpatch_namd_2.7.sh`. The patching procedure is the same in both cases, thus in the following we shall indicate with `plumedpatch_namd.sh` any of these two patches.

- Extract the source code from its archive and then move into the NAMD `root` directory.
- Set the environmental variable `plumedir` to point to `PLUMED_root`.
- Copy or link the `plumedpatch_namd.sh` file from the `patches` folder to the current directory.
- Proceed to the usual configuration.
- In the patch script, modify the `myarch` variable to match your architecture.
- Execute the script: "`./plumedpatch_namd.sh -patch`".
- Proceed to compile the code following the NAMD instructions.

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

```
tar zxf 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
```

2.4 Compiling DL_POLY with PLUMED

PLUMED works with DL_POLY 2.16, 2.19 and 2.20. As the code was subjected to a major rewriting, two different patches are available, one for version 2.16 (plumedpatch_dlpoly_2.16.sh) and one for versions 2.19 and 2.20 (plumedpatch_dlpoly_2.19.sh) The patching procedure is the same in all cases, thus in the following we shall indicate with plumedpatch_dl_poly.sh any of these patches.

- Extract the source code from its archive and then move into the DL_POLY root directory.
- Set the environmental variable plumedir to point to PLUMED_root.
- Copy or link the plumedpatch_dlpoly.sh file from the patches folder to the current directory.
- Execute the script: "./plumedpatch_dlpoly.sh -patch".
- Select the appropriate Makefile from the build directory and compile DL_POLY in the usual way.

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 zxf 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
```

2.5 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 slighly 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.

2.6 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.

2.7 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`.

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 ...
```

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 W_STRIDE 50

# 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.

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 syantax), 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).

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:

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

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 [11] 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:

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

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 principles 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 at 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. [16], 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 actually 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.

3.4.6 Multiple walkers

The MULTIPLE_WALKERS directive sets the multiple walkers [12] running mode. The process will write its biasing potential (the HILLS file) in the directory specified by the HILLS_DIR keyword. Multiple processes can be launched independently pointing to the same directory, so that each one will contribute to the biasing potential. All the processes must have the same CVs, in the same order. Note that the HILLS directive of each process must contain the R_STRIDE keyword specifying the number of steps before each individual bias potential is brought up to date reading the bias in the HILLS file.

Example.
The following command allows a new Gaussian to be added to the bias potential every 1000 steps, and the updated potential to be read every 5000 steps.

```
HILLS HEIGHT 0.1 W_STRIDE 1000 R_STRIDE 5000
MULTIPLE_WALKERS HILLS_DIR /scratch/HILLS
```

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 has been omitted as this was implicitly included in the missing `SIGMA` parameter in the right CV.

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:

- AMBER: the present version of PLUMED is only compatible with the serial build of AMBER.
- 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 and DL_POLY: the present version of PLUMED is fully compatible with GROMACS and DL_POLY 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. 4.18).
  - 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) [13, 17] and bias-exchange metadynamics (BE-META) [14].

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 GRO-

MACS 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_INPO) 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. This turns on a potential of the following functional form:

$$V_{\mathrm{umb}}(s) = \frac{1}{2}k(s - s_0)^2. \tag{3.3}$$

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 [18]. The directive UMBRELLA can be used multiple times in the case of multidimensional umbrella sampling calculations (one directive for each CV).

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 [9] or Crooks [10] 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 3.3, with the reference position s_0 moving at the specified velocity.

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
```

3.9 Commitment analysis

The COMMITMENT directive is used to run commitment analysis. An additional keyword, NCV, must specify the dimension of the phase space, *i.e.* the number of CVs involved in the definition of the commitment basins. Following this line, NCV lines must be found, each of which contains the upper and lower edges of the $\mathcal{A}$ and $\mathcal{B}$ basins for the i-th variable.

Example.
The following line defines a commitment analysis run, in which the $\mathcal{A} = \{(s_1, s_2)|s_1 \in (0, 1), s_2 \in (-1, 1)\}$, and $\mathcal{B} = \{(s_1, s_2)|s_1 \in (1, 2), s_2 \in (-1, 1)\}$.

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

Chapter 4

Collective variables

The implementation of many CVs is required to deal with the different problems of interest and to give a proper description of the processes involved. In the following we describe all the possibilities that the current package offers, implemented with their own analytic first derivative.

To specify a CV, a line starting with the keyword indicating the CV type must be included in the input file. After this keyword, the information needed to define the variable can be added in any order.
In the case of metadynamics, all the variables on which the biasing potential is acting share the compulsory `SIGMA` keyword that defines the Gaussian width in CV units. This keyword is not needed when performing other types of free energy calculations.

Specifying lists of atoms

Most variables need one or several sets of atoms for their definition. Whenever a set of atom groups needs to be defined, the `LIST` keyword must be specified. After this keyword a set of tags specifies the number and order of the groups. The atoms contained in each group must be defined in the following lines as follows: for each group, a block starting with the name of the group followed by the `->` sign and terminating with the same name followed by the `<-` sign is expected. In the lines between the two delimiters, the atoms comprising the group must be indicated, separated by spaces or line feeds.

Example.
The following syntax specifies a CV defined as the distance between the center of mass of atoms 6 and 10 and the center of mass of atoms 8, 15 and 21:

```
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, or 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 group g1:

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

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

The special and rather common case of a group composed of a single atom can be specified by indicating the number of that atom instead of the corresponding <g> tag.

Example.
The following two commands are equivalent ways to define a CV as the distance between atom 5 and group <g1>:

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

Example.
The following lines define the CV to be the y coordinate of atom 13.

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

The keywords LINE_POS and LINE_DIST define the CV to be the position of the atoms as projected on a line and as the distance from the line respectively, the line being defined by its starting and final points. These keywords accept as input XY, XZ, YZ and XYZ and define whether the atom position should be projected on a Cartesian plane (XY, XZ or YZ) or if the line has a generic orientation in space (XYZ).

Example.
The following lines define the CVs to be the projection of the coordinates of atom 13 on a generic segment defined by its starting (0,0,0) and final point (2,3,4), and by its distance from the same segment.

```
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 defines a CV representing the distance between the center of mass of two groups of atoms. Two groups must be defined using the LIST keyword, following the syntax described in section 4.

Example.
The following lines define the CV to be the distance between atom number 13 and the center of mass of the four atoms in list <g1>.

```
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 when atoms are part of the same molecule. See also Sec. 4.18.

4.3 Minimal distance

The MINDIST keyword represents the minimal distance between two groups of atoms. To ensure differentiability, it is implemented as:

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

where $\beta = 5$. Like the distance variable, also the minimal distance needs two groups to be defined using the LIST keyword, following the syntax described in section 4.

Example.
The following lines define the CV to be the minimal distance between atom number 13 and the set of atoms in list <g1>.

```
MINDIST 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 when atoms are part of the same molecule. See also Sec. 4.18.

4.4 Angles

The `ANGLE` keyword defines the angle between the center of masses of three groups of atoms. The compulsory `LIST` keyword must be followed by three properly defined groups (see Section 4).

Example.
The following lines define the CV to be angle formed by the center of mass of the atoms in groups `g1` and `g2` and atom number 102.

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

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

4.5 Torsion

With the `TORSION` keyword, the CV represents the dihedral angle defined by four atoms or, more generally, the torsion angle among the center of masses of four groups of atoms. The compulsory `LIST` keyword must be followed by four properly defined groups (see Section 4).

Example.
The following lines define the CV to be the torsion angle defined by the center of mass of the four groups <g1>, <g2>, <g3>, <g4>.

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

4.6 Coordination number

The `COORD` keyword defines a CV representing the coordination number between two groups of atoms, namely the total number of contacts between

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

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

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`. Such user-supplied parameters allow large flexibility in 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 between the first and the second group of atoms. A good starting point can indeed be to take r_0 as the position of the first peak in the pair distribution function, d_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, depending on the system properties, different choices may lead to better results. An optional keyword `PAIR` allows the atoms to be considered pairwise, thus counting how many of the pairs defined by the lists in $\mathcal{G}_1$ and in $\mathcal{G}_2$, taken in the given order, are bonded. Clearly, in this case, `<g1>` and `<g2>` must have the same number of atoms.

Example.
The following lines define the CV to be the coordination of the atoms in group `g1` – 13 and 15 – with the atoms in group `solvent`.

```
COORD LIST <g1> <solvent> NN 6 MM 12 D_0 0.2 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 be done only when atoms are part of the same molecule. See also Sec. 4.18.

4.7 Hydrogen bonds

The HBONDS defines a variable counting the number of intra-molecular hydrogen bonds between two groups of atoms. This is defined as:

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

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

These groups must be defined using the compulsory LIST keyword followed by two groups (see Section 4). 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, default values are assumed: $r_0 = 2.5$, $n = 6$ and $m = 12$.

The TYPE keyword selects the residues included in the count:

- With TYPE 0, all donor-acceptor pairs are included in the count;
- If TYPE 1 is specified, only the donor-acceptor pairs that are distant an odd number of places, larger than 4 along the chain are counted; this allows parallel β-sheet formations (β-odd type) to be monitored;
- If TYPE 2 is specified only the donor-acceptor pairs that are distant exactly 4 along the chain are included; this allows the formation of α-helical conformations (α type) to be monitored;
- If TYPE 3 is specified, only the donor-acceptor pairs that are distant an even number of places, larger than 4 along the chain are counted; this allows anti-parallel β-sheet formations (β-even type) to be monitored.

Example.
The following lines define the CV to be the count of the total number of hydrogen bonds between the pairs in groups H and O. The default switching function with $r_0 = 2.5$, $n = 6$ and $m = 12$ is used.

```
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 used.

```
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 atoms are part of the same molecule. See also Sec. 4.18.

4.8 Interfacial water

The WATERBRIDGE keyword defines a CV counting the interfacial contacts. This variable is intended to calculate the number of atoms of a certain group $\mathcal{G}_0$ that are in contact with atoms of both groups $\mathcal{G}_1$ and $\mathcal{G}_2$ at the same time (typically the number of waters at the interface of two surfaces). This is calculated as:

$$s_{\mathrm{WatBr}} = \sum_i^{n0} \left(\sum_j^{n1} \frac{1-(\frac{|\mathbf{r}_i-\mathbf{r}_j|}{r_0})^n}{1-(\frac{|\mathbf{r}_i-\mathbf{r}_j|}{r_0})^m} \right) \left(\sum_j^{n2} \frac{1-(\frac{|\mathbf{r}_i-\mathbf{r}_j|}{r_0})^n}{1-(\frac{|\mathbf{r}_i-\mathbf{r}_j|}{r_0})^m} \right).$$

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

Example.
The following command defines a CV that counts the atoms in the solvent group that are simultaneously in contact with atoms 6 or 10 and 8,15 or 21.

```
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

The RGYR directive defines the use of the radius of gyration of a group of atoms defined with the compulsory additional keyword LIST. Only one properly

defined group must follow the `LIST` keyword (see Section 4). This CV can be calculated as :

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

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

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

Example.
The following lines define the CV to be the radius of gyration of the group <g1>.

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

Alternatively one might be interested in the trace of the inertia tensor, which can be shown to be equivalent to:

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

This second option can be activated with the `INERTIA` directive, followed by the same syntax as the gyration radius.

Example.
The following lines define the CV to be the radius of gyration of the group <g1>.

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

Since the radius of gyration is calculated without applying the periodic boundary conditions, the involved atoms should be part of the same molecule. See also Sec. 4.18.

4.10 Dipole

`DIPOLE` defines a CV calculated as the electrical dipole generated by a group of atoms:

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

Only the `LIST` keyword is needed and only one properly defined group must follow the `LIST` keyword (see Section 4).

Example.
The following lines define the CV to be the dipole of the group `<g1>`.

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

4.11 Dihedral correlation

`DIHCOR` defines the dihedral correlation, which is a similarity measure for adjacent dihedral angles:

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

The number N_D of dihedrals must be specified after the `NDIH` keyword. The next N_D must each contain the four numbers identifying the atoms that define each dihedral ϕ_i.

Example.
The following lines define the CV to be the dihedral correlation of the 3 dihedrals listed.

```
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 measure of similarity of dihedral angles with respect to a reference value (see also 4.13). It is calculated as:

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

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

Example.
The following lines define a CV as the Alpha-beta similarity of three dihedrals.

```
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 Torsional RMSD

`RMSDTOR` represents the root of mean square deviation of dihedral angles with respect to a reference configuration, calculated as:

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

The number N_D of dihedrals involved must be defined with the `NDIH` keyword; in the following N_D lines, the indices of the four atoms defining each dihedral must be listed, followed by the value in the reference conformation θ_i^{Ref}.

Example.
The following lines define a CV as the torsional root mean square deviation of two dihedrals.

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

4.14 Electrostatic potential

`ELSTPOT` is the electrostatic potential felt by one atom or by the center of mass of a group of atoms and created by another group of atoms. It is

calculated as:

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

where

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

N_B is the group of atoms that defines the point in which the electrostatic potential is felt. N_A defines the group of atoms whose charges create the electrostatic potential. $f(x)$ 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, CUT a cutoff distance.

Example.
The following lines define as CV the electrostatic potential felt by the center of mass of group1 and produced by the charges of the atoms belonging to group2:

```
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<-
```

4.15 Puckering coordinates

`PUCKERING` refers to the set of collective coordinates for 6-membered rings in polar coordinates [19]. 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

$$\begin{aligned} Q &= \sqrt{q_2^2 + q_3^2} \quad &\geq 0 \\ \theta &= \arctan\left(q_2/q_3\right) \quad &\in [0, \pi] \\ \phi &= \phi_2 \quad &\in [0, 2\pi) \end{aligned}$$

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<-
```

4.16 Path collective variables

Path collective variables [20] can be defined using the pair of `S_PATH` and `Z_PATH` keywords. The s variable (defined with `S_PATH`) 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, X_i is an ordered set of N reference conformations that define the path in the configuration space $\mathcal{X}$, $d : \mathcal{X} \times \mathcal{X} \to \mathbf{R}_0^+$ is a metric on $\mathcal{X}$, and $\mathcal{Z} = \sum_{i=1}^{N} e^{-\lambda d(X_i, X(t))}$

is a normalization factor. The prefactor λ should be chosen so as to have on average $\lambda d(X_i, X_{i\pm1}) \simeq 2.3$.

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

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

For either keyword the following keywords must be defined:

- `TYPE` defines the metric used to calculate distances in the configuration space; this can either be `RMSD` or `DRMSD`, or `CMAP` [21], which uses the distance between contact matrices (see below).
- `NFRAMES` sets the number N of reference structures that define the path.
- `LAMBDA`, the common prefactor λ in the exponential term in the equation that defines both s and z.
- Optionally, `NEIGHLIST` defines a neighbor list on the closest frames.

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

4.16.1 Root mean square deviation

The word `RMSD` after the `TYPE` keyword sets the metric for the path to be the root mean square deviation of a given subset of the system atoms (the *displacement* set B), after alignment over another subset (the *alignment* set A):

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

where M_{ij} is the roto-translation matrix calculated using the Kearsley [22] algorithm.

The user should specify the basename of the files containing the coordinates of the reference frames in the PDB format after the keyword `FRAMESET`. N files, whose name is obtained by appending to the basename a consecutive integer and the extension `.pdb`, are expected.
The atoms involved in the *alignment* and in the *displacement* sets must be specified using respectively the two last numerical fields of the frameset files

(1.00 for used, 0.00 for not used).
The position unit in the PDBs are in Ångstrom. Engines like GROMACS, whose internal units are different, will perform the proper conversion.
Clearly, the index number in the frameset files must match the one in the topology for the system under consideration.

Example.
The following command defines a path using RMSD metrics. 2 frames are used to define the path, and a value of $\lambda = 9.0$ is set.

```
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. In the last two columns of these files, the atoms used for alignment and for calculating the RMSD must be specified.

```
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
```

4.16.2 Distance root mean square deviation

The word `DRMS` after the `TYPE` keyword sets the metric for the path to be the root mean square deviation of the distances of a given subset of the system atoms:

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

where $r_{ab}^{(j)}$ is the distance of atoms a and b in the j-th reference frame. As in the case of root mean square deviation (see section 4.16.1), the frameset must be specified using the keyword `FRAMESET`, and the PDB format must be used to define the reference structures.

4.16.3 Contact map distances

The word `CMAP` after the `TYPE` keyword sets the metric for the path to be the distance of the contact matrices of a given subset of the system atoms:

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

Given a set of atoms $a, b \in \mathcal{J}$, the contact matrix D_{ab} is calculated as:

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

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

As in other variables, the parameters $r_{ab}^{(0)}$, n_{ab} and m_{ab} allow great freedom in the definition of the switching function. The parameter c_{ab} allows the definition of the contact to be cut off above a given threshold. Moreover, a weight w_{ab} can be used to chainge the relative importance of the different contacts in the definition of the CV.

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 the values of the pairs of indices a, b that define a contact, and the corresponding values of $r_{ab}^{(0)}$, n_{ab}, m_{ab}, c_{ab} and w_{ab}. Each contact is on a separate line, starting with the `CONTACT` keyword.
The *values* file, specified after the keyword `MAP`, contains the values of the reference matrices $D_{ab}^{(i)}$. Each matrix must be separated by the `END` keyword, and consists of lines with the contact number, the values a and b of the involved atoms, and the value of the contact map.

Example.
The following command defines a path using the contact map 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 LAMBDA 0.1 SIGMA 1.
```

The `fr.ndx` file should contain the indices of the atoms that define the contact matrix:

```
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
```

Additionally, one could define the matrix to include contacts between the centers of mass of groups of atoms. In this case, an additional file containing the definition of the groups must be specified in the PLUMED input file, with the GROUP keyword. This file must contain the definitions of the groups, as follows. Each group is defined on one line starting with the GROUP keyword; the first number labels the group, the second number specifies how many atoms are in the group, and the following numbers are the indices of the atoms forming the group under consideration.

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
```

The index file can contain both atom and group contacts, but in this case the group contacts must follow the `CONTACT` ones.

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
```

4.16.4 Using path variables as RMSD, DRMSD and CMAP

The definition of the z path variable,

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

makes it clear that, for $f = 1$, it coincides with the distance between a configuration and the reference structure, measured in the chosen metric (squared). Therefore, this CV should be used to reproduce the standard RMSD, distance RMSD or CMAP distance.

4.17 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:

$$V_{wall}(s) = \texttt{KAPPA}\left(\frac{s - \texttt{LIMIT} + \texttt{OFF}}{\texttt{EPS}}\right)^{\texttt{EXP}}, \tag{4.1}$$

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 $15\,\mathring{A}$, 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
```

4.18 A note on periodic boundary conditions

Most collective variables are designed so as to be applied taking into account the periodic boundary conditions as they are defined by the host code. However, there are some exceptions, including:

- Average coordinate of a group of atoms.
- RGYR.
- DISTANCE, MINDIST, COORD, HBONDS, if the NOPBC flag is used.

In all these cases, it is crucial that the involved atoms are chosen as part of a single, unbreakable object such as a molecule. Furthermore, it is necessary that the coordinates which are passed to PLUMED are keeping the molecules intact. We are aware of at least two cases where this condition is not satisfied, i.e. for the host code GROMACS4, when using domain decomposition or when using the option for periodic molecules.

In these cases it is necessary to take advantage of the additional directive ALIGN_ATOMS. Within this directive, the keyword LIST allows to define an ordered group of atoms which are aligned in such a way that the distance of atom with subsequent indexes in the sequence is the minimal one. Usually these atoms should be be listed in the same order as they appear in the pdb file, which takes into account how close they are in the molecule topology. Sometimes it is sufficient to align the atoms involved in the CVs. However, if the atoms involved in the CVs are far away from each other, it is necessary to also align intermediate atoms.

Example.
To run a metadynamics simulation using as a CV the end-to-end distance in a protein, using GROMACS4 with 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` allows for summing up the Gaussians laid along the metadynamics trajectory and obtaining the free energy surface.

5.1.1 Installation instructions

Being a simple fortran 90 program, the installation is straightforward with any compiler available on your machine. To give an example, with the gnu g95 compiler:

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

If you want to process a big HILLS file and you have a multicore machine available, it could be convenient to compile the parallel version

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

5.1.2 Usage

The program takes the 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;2π] periodicity on the x CV,
                     if -fix is not used 2pi is used
 -pi x               [−π;π] 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 HILLLS        input file
 -out fes.dat        output file
 -hills nhills       number of Gaussians that are read
```

The program works in the following way.

`-file` and `-out` define the input Gaussian file and output free-energy file respectively. HILLS and fes.dat are the default values.

The number of CVs in the HILLS file is specified with `-ndim` whilst the number and order of the CV is controlled by `-ndw` followed by the list of CV in the desired order. Note that after being read the CV are reordered according to ndw into the code, hence EVERY output refers to this new order. If the number of CV in output is different from the on in input the missing CV are integrated out with the Boltzmann weight K_bT specified by `-kt` (the energy units of the simulations are assumed).

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 as a grid in gnuplot format with a blank line added after each block of data. This is quite handy for visualizing quickly 3D data (e.g. the FES as a function of 2 CVs).

For efficiency purposes, the Gaussians are truncated at a certain distance from the center before being placed on the grid. `-cutoff_s` and `-cutoff_e` allow for tuning this cutoff distance. With `-cutoff_s` the value read is the number of standard deviations from the center at which the Gaussian is truncated and it should be equal to DP2CUTOFF as used in the metadynamics run (the default value is 6.25 in both cases). Alternatively the user can spec-

ify the energy at which the Gaussian is truncated by using `-cutoff_e`; here the energy units are the same as in the simulation. Note that if the cutoff is different in the run and in this post-processing 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 (as per input) or by `-dp` followed by the size of the voxel in each CV direction. If none 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 allows for fixing the boundaries of the output free energy, where two real numbers are expected for each CV.

`-stride` allows for printing out the evolution of the free-energy as a function of time, *i.e.* of the index of the Gaussian read in input. `-stride` expects an integer which specifies how many Gaussians are added amid each print out. 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 to create a movie of the filling of the free-energy well.

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

In the case of angles and dihedrals, periodicity options can be specified, namely `-pi` and `-2pi` followed by the CV index state that the CV is periodic between [-pi;pi] or [0;2pi] respectively, and `-grad` states that the CV values are in degrees and not in radians.

5.2 Evaluating collective variables on MD trajectories

The program `driver` evaluates the value of all the CVs implemented in `PLUMED` on 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. Then, to compile using g95, simply type:

`make gnu`

or, to compile with Intel ifort and icc compiler, type:

`make intel`.

In order to use different compilers and/or compiler flags you need to modify the Makefile.

5.2.2 Usage

The program takes the 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. In this file, the atoms' masses and charges should be inserted respectively in the occupancy field and in the B-factor field. These data are needed for the calculation of particular collective variables, such as center of masses or dipoles.

The trajectory file must be in CHARMM format DCD file (also NAMD 2.1 and later). To convert a trajectory into this format, the user can download the utility `CatDCD` from:
`http://www.ks.uiuc.edu/Development/MDTools/catdcd/`.

The input file has the same syntax as the `PLUMED` input file. Here, the user can specify a printing stride (in number of DCD frames) using the keyword

W_STRIDE and the list of desired CVs. The CVs value will be printed on the COLVAR file. For a complete list of the possible CVs, please see section 4.

Example.
If you wish to evaluate the distance between atom 13 and 17 along your trajectory file on every frame, the input looks 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 provide the fixed dimensions of the box in Angstrom (only orthorhombic) using the keyword `-cell CELLX CELLY CELLZ` or switch off the pbc with `-nopbc`.
To extract frames in a specific window of the CVs space and write them on a PDB file, please use the option `-out output.pdb` and specify the interval with `-interval CVmin CVmax`.

Example.
To extract the frames in which the distance between two atoms is between 10.0 and 12.0 Å and in which the angle defined by three 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 invoke `driver` with this syntax:

```
./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 the format needed by the GROMACS tool `g_cluster` to perform a cluster analysis on your set of frames.

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] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. Caldwell, P. Kollman, A second generation force field for the simulation of proteins, nucleic acids and organic molecules, J. Am. Chem. Soc. 117 (1995) 5179–5197.

[5] A. Laio, M. Parrinello, Escaping free energy minima, Proc. Natl. Acad. Sci. USA 20 (2002) 12562–12566.

[6] G. Torrie, J. Valleau, Nonphysical sampling distributions in monte carlo free energy estimation: Umbrella sampling, J. Comput. Phys. 23 (1977) 187–199.

[7] 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.

[8] B. Roux, The calculation of the potential of mean force using computer-simulations, Comput. Phys. Comm. 91 (1995) 275–282.

[9] C. Jarzynski, Nonequilibrium equality for free energy differences, Phys. Rev. Lett. 78 (1997) 2690–2693.

[10] G. Crooks, Nonequilibrium measurements of free energy differences for microscopically reversible markovian systems, J. Stat. Phys. 90 (5-6) (1998) 1481–1487.

[11] A. Barducci, G. Bussi, M. Parrinello, Well-tempered metadynamics: A smoothly converging and tunable free-energy method, Phys. Rev. Lett. 100 (2) (2008) 020603.

[12] 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.

[13] 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.

[14] S. Piana, A. Laio, A bias-exchange approach to protein folding, J. Phys. Chem. B 111 (17) (2007) 4553–9.

[15] 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.

[16] V. Babin, C. Roland, C. Sagui, Adaptively biased molecular dynamics for free energy calculations, J. Chem. Phys. 128 (2008) 134101.

[17] C. Camilloni, D. Provasi, G. Tiana, R. A. Broglia, Exploring the protein g helix free-energy surface by solute tempering metadynamics, Proteins 71 (2008) 1647.

[18] A. Grossfield, Wham.`http://membrane.urmc.rochester.edu/Software/WHAM/WHAM.html`.

[19] M. Sega, E. Autieri, F. Pederiva, On the calculation of puckering free energy surfaces.

[20] D. Branduardi, F. L. Gervasio, M. Parrinello, From a to b in free energy space, J. Chem. Phys. 126 (5) (2007) 054103.

[21] 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.

[22] S. K. Kearsley, On the orthogonal transformation used for structural comparison, Acta Cryst. A 45 (1989) 208–210.

Index