PLUMED User's Guide

A portable plugin for free-energy calculations with molecular dynamics

Version 1.3.0 – Nov 2011

Contents

Chapter 1

Introduction

1.1 What is PLUMED?

PLUMED[1] 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 [2], NAMD [3], DL_POLY [4] , LAMMPS [5] and the SANDER module in AMBER [6]. It also works with the very fast CUDA/GPU MD code ACEMD [7] and, more recently, it has been extended to work with *ab initio* MD codes, such as Quantum-ESPRESSO [8] and CPMD.

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 [9], umbrella sampling [10, 11, 12] and Jarzynski-equation based steered MD [13, 14].

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, LAMMPS and CPMD. These codes are not distributed with the PLUMED package, but they must be obtained separately.

NAMD 2.6/2.7/2.8
http://www.ks.uiuc.edu/Research/namd/

GROMACS 4.0/4.5.x
http://www.gromacs.org/

AMBER 10/11
http://ambermd.org/

DL_POLY 2.20
http://www.cse.scitech.ac.uk/ccg/software/DL_POLY/

Quantum-ESPRESSO 4.3.2
http://www.quantum-espresso.org/

ACEMD 1.2
http://multiscalelab.org/acemd

LAMMPS 27-Oct-2011
http://lammps.sandia.gov

CPMD 3.15.1
http://www.cpmd.org

Note that at the moment of this release of PLUMED, only those specific versions of the listed codes have been tested. Moreover, some other codes are available through the Python interface provided by Rosa Bulo.

Amsterdam Density Functional (ADF)
http://www.scm.com/

and all the codes supported by

Atomic Simulation Environment (ASE)
`https://wiki.fysik.dtu.dk/ase/`

Additionally there exist some porting to

FHI-AIMS
`https://aimsclub.fhi-berlin.mpg.de/`

which is not directly mantained by the `PLUMED` developer team. Additional information have to be retrieved by the developers of AIMS themselves.

1.3 Features

`PLUMED` can perform several different types of calculation:

- Metadynamics with a large variety of CVs [9];
- Well-tempered metadynamics [15];
- Multiple walkers metadynamics [16];
- Combined parallel tempering and metadynamics [17];
- Bias-exchange metadynamics [18];
- Reconaissance metadynamics [19];
- Steered MD;
- Umbrella sampling;
- Adiabatic biased molecular dynamics [20];
- Commitment analysis.

1.4 New in version 1.3

PLUMED version 1.3 presents several new features, including new collective variables and support to new codes. Among these:

- Reconaissance Metadynamics [19];
- Driven adiabatic free energy dynamics (d-AFED, contributed by Michel Cuendet)
- Definition of CVs as Polynomial combination of CVs and Function on CVs;
- New INTERVAL keyword to limit a region to be sampled using Metadynamics (contributed by Alessandro Laio)
- Better scalability with Gromacs;
- New collective variables: Projection on PCA eigenvectors (contributed by Ludovico Sutto), SPRINT topological variable and gyration tensor based CVs;
- New tool: reweight well-tempered metadynamics simulations [21];
- Added support to ACEMD 1.2, Lammps (27-oct-2011), Quantum Espresso 4.3.2, CPMD 3.15.1, NAMD 2.8, Gromacs 4.5.x and Amber11;
- Python interface (contributed by Rosa Bulo);
- New tool to perform bias-exchange simulations via the linux shell with any MD code;
- Bugs have been fixed and the manual has been improved thanks to the feedbacks of many users.

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 (however a tool allows bias-exchange simulations via linux shell with any MD engine);
- The patched version of GROMACS cannot perform hamiltonian (lambda) replica exchange;
- The ENERGY collective variable is available only for GROMACS, 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;
- Only Car-Parrinello and Born-Oppenheimer simulations are supported in CPMD;
- Only NVT simulations are supported because the contribution to the virial is not calculated.

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 [22], 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.
  - `reweight` is a tool to calculate the unbiased probability distribution of variables other than the CVs in well-tempered metadynamics simulations
  - `python_interface` a set of tools that enables to use `PLUMED` with Python (hey, a Python which is plumed with feathers is a really weird animal!).
  - `bias-exchange` a set of tools that allow to use `PLUMED` to perform bias-exchange simulations via the linux shell using any MD engine.

- `manual`. This manual.
- `ACEMD`. It contains the files needed to compile the plugin for ACEMD.
- `tutorial`. It contains the resources of the `PLUMED` tutorial 2010 (`http://sites.google.com/site/plumedtutorial2010/`)
- `recon_src`. It contains the source files specific for the Reconaissance Metadynamics.

1.7 Online resources

You can find more information on the web:

`http://www.plumed-code.org`

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. 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, Alessio Lodola, Luigi Capoferri, Michel Cuendet, Jiri Vymetal, Fahimeh Baftizadeh, and Katsumasa Kamiya.

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 explicitely. A different proceedure 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)
- CPMD only: you may need to change variables `cxx, cxxflag, extraflag` in the plumedpatch script depending on your computer architecture: see the examples below.
- 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 10 with PLUMED using g95 in the Bourne shell.

```
tar zxf AMBER10.tgz
cd amber10/
export plumedir="PLUMED_root"
cp $plumedir/patches/plumedpatch_sander_10.sh .
cd src/
./configure g95
cd ..
./plumedpatch_sander_10.sh -patch

make
```

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
```

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
```

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_2.20.sh .
./plumedpatch_dlpoly_2.20.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
```

Example.
This is a sample procedure for compiling CPMD-3.15.1 under different computer architectures.

```
tar zxvf cpmd-v3_15_1.tgz
cd CPMD
```

Generate a valid Makefile for your machine using the script `./mk_config.sh`, e.g.:
`./mk_config.sh PC-GFORTRAN > Makefile`

```
export plumedir="PLUMED_root"
cp $plumedir/patches/plumedpatch_cpmd-3.15.1.sh .
```

If necessary, modify the variables `cxx` (a valid C++ compiler), `cxxflag` (the corresponding flag to avoid exception handling), and `extraflag` (see below) in the script `plumedpatch_cpmd-3.15.1.sh`, e.g. for GNU compilers:

```
cxx="mpiCC"
cxxflag="-fno-exceptions"
```

for IBM machines like AIX-power7:

```
cxx="xlc_r -c"
cxxflag="-qnoeh"
```

for IBM BlueGene:

```
cxx="bgxlc++_r -c -O"
cxxflag="-qnoeh"
```

for PGI compilers:

```
cxx="pgCC"
cxxflag="--no_exceptions"
```

To successfully compile on some IBM architectures you may also need to set one or both of the following extra flags:

```
extraflag="-DPLUMED_CPMD_NOUNDERSCORE -DPLUMED_AIXXLF"
```

Finally, patch and compile:

```
./plumedpatch_cpmd-3.15.1.sh -patch
make
```

2.1.1 Compiling the ACEMD plugin with PLUMED

PLUMED works with ACEMD 1.2. 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`.

- Use `make` to compile the plugin inside the `plumedir/ACEMD` folder.

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

```
export plumedir=PLUMED_root
cd $plumedir ACEMD
make
```

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

```
pluginload testplug ./plumed.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 1.2.

2.2 Including reconnaissance metadynamics

The reconnaissance metadynamics routines (see section 3.13 included in `PLUMED` make use of the C++ standard library and lapack. On some machines these libraries are not installed. Hence, `PLUMED` has been written so that by default no attempt to compile reconnaissance metadynamics is made. Consequentially, if you wish to perform reconnaissance metadynamics you must patch `PLUMED` slightly differently.

To include reconnaissance metadynamics in your patched MD code is simply a matter of, during the patching procedure, telling `PLUMED` the location of the C standard library, the location of lapack and the location of

a c++ compiler. This is done by making some small modifications to the plumedpatch file. If this file is opened using a text editor you will notice at least one of the following two lines:

```
RECON_LIBS=
RECON_CPP=
```

If at least one of these lines is not present then reconnaissance metadynamics is not implemented for the particular code you are patching. The first of these two lines tells PLUMED the locations of the lapack and c standard libraries. If the space after the equal sign is left blank PLUMED patches in its default fashion and no attempt is made to compile the reconnaissance metadynamics routines. By contrast if there is any character after the equals sign PLUMED will endeavor to compile the reconnaissance metadynamics routines. For a correct compilation the RECON_LIBS}variable should indicate the locations of the lapack (i.e. -llapack) and c standard libraries (i.e. -lstdc++). The RECON_CPP= variable should be set equal to a c++ compiler (e.g. g++).

Example.
To include the reconnaissance metadynamics routines in the PLUMED compilation the use must change the values of RECON_LIBS and RECON_CPP in the plumedpatch file:

RECON_LIBS="-llapack -lstdc++"
RECON_CPP="g++"

There are some differences between the procedures for patching PLUMED+ reconnaissance metadynamics in the different MD codes between the different codes. This is because some of the codes already have lapack included or are compiled with a c++ compiler. These differences are described in table 2.1:

2.3 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

Code	Reconnaissance implemented	C++ compiler required	Lapack required	lstc++ required
NAMD	yes	no	yes	yes
GROMACS	yes	yes	no	yes
AMBER	no	-	-	-
DL_POLY	yes	yes	yes	yes
Q-ESPRESSO	yes	yes	no	yes
ACEMD	no	-	-	-
LAMMPS	no	-	-	-
CPMD	yes	yes	no	yes

Table 2.1: Details on compiling `PLUMED` compilation with the various codes

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 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.

2.4 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`.

2.5 The Python interface to `PLUMED`

Starting from version 1.3 `PLUMED` has also a Python interface contributed by Rosa Bulo. It is possible to find it in the `utilities/python_interface` directory. Inside it you might find a `README` file and a `Makefile` . After copying (or linking) all the `*.c` and `*.h` files from the `common_files` directory you can immediately do:
`make gnu`
if you want to compile with gnu compilers. Intel compilers are also available. This creates a file that is `libplumed.so` that will be loaded runtime by Python. You need to add the directory where this file resides into the `LD_LIBRARY_PATH` so that Python will be able to find it and load it runtime. The extension `amber` is only reflecting the fact that the interface is built by

exploiting the simple AMBER interface. Therefore the limitiations regarding AMBER in the functionalities apply also here.

For MacOSX users the compiler flags should be changed accordingly. For gnu compilers it requires to change the linking flags from
`-shared -Wl,-soname,libplumed.so`
to
`-bundle -flat_namespace -undefined suppress`.

Next, in the directory
`utilities/python_interface/pylib`
you can find the file `plumedamberlib.py` which is the actual Python module that will deal of loading `libplumed.so` and create the suitable Python interface. This provides the `init_metadyn` and `cv_calculation` methods that one can practically use through Python.

A practical use of this shown by the Python script
`/utilities/python_interface/test/run_water_amber.py`
where a water molecule is simply displaced and the collective coordinates are updated at each step. Notably, this interface is embodied in the Python tools provided with the Amsterdam Density Functional code (also called ADF, see `http://www.scm.com/`).

Additionally Rosa provided an interface with the powerful `Atomic Simulation Environment` (see `https://wiki.fysik.dtu.dk/ase/`). This enable `PLUMED` to be interfaced with plenty of DFT and ab-initio codes. Some examples are present in the directory `utilities/python_interface/test` and are `ASEmd_plumed_emt_water.py` and `ASEnvt_plumed_emt_water.py` .

Many thanks to Rosa for the contribution!

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`;
- Quantum-ESPRESSO users should activate PLUMED on the command line by specifying the flag `-plumed`. The name of the PLUMED input file is 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`;
- In CPMD, `PLUMED` is activated on the command line by specifying the flag `-plumed` after the CPMD input file and the pseudopotential directory. The `PLUMED` input file name is `plumed.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 ...
```

Example.
To perform a free-energy calculation with Quantum-ESPRESSO, `PLUMED` must be activated on the command line:

```
pw.x -plumed ...
```

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
```

Example.
To perform a free-energy calculation with LAMMPS the file should contain, in addition to the usual keywords defining the run, the following command:

```
fix ID all plumed plumedfile plumed.inp outfile plumed.out
```

where ID is the user assigned fix name.

Example.
To perform a free-energy calculation with CPMD, `PLUMED` must be activated on the command line:

```
cpmd.x input .  -plumed
```

where `input` is the CPMD input file, "." is the current directory (if pseudopotentials are located elsewhere, substitute with the right location), and the `PLUMED` input file is called `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 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 [9, 23, 24]. 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:

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

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 [15] 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 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. [25], 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 [15]. 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 [16] 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.

Avoiding systematic errors at the boundaries

Metadynamics is often used to reconstruct the free energy as a function of intrinsically limited CVs (like e.g. COORD and ALPHABETA, see chapter 4) or articially limited CVs (using for example an external potential, see section 3.10). In this case the finite-width Gaussians used in metadynamics can induce systematics errors at the boundaries. These errors become more and more severe with time, and are due to the fact that a sum of a finite number of Gaussians can not reproduce a free-energy profile with large or infinite derivative. However, PLUMED implements two different procedures aimed at avoiding the onset of these systematic errors, activated respectively by the keyword INTERVAL and by the keyword INVERT.

3.4.8 Defining an interval

With the keyword INTERVAL one changes the metadynamics algorithm setting the bias force equal to zero beyond a boundary [26]. If, for example, metadynamics is performed on a CV s and one is interested only to the free energy for $s > s_w$, the history dependent potential is still updated according

to Eq. 3.1, but the metadynamics force is set to zero for $s < s_w$:

$$\frac{\partial V_G(s,t)}{\partial s} = 0 \quad \forall s < s_w$$

Notice that Gaussians are added also if $s < s_w$, as the tails of these Gaussians influence V_G in the relevant region $s > s_w$. In this way, the force on the system in the region $s > s_w$ comes from both metadynamics and the force field, in the region $s < s_w$ only from the latter. This approach allows obtaining a history-dependent bias potential V_G that fluctuates around a stable estimator, equal to the negative of the free energy far enough from the boundaries [26].

- In order to obtain a parallel growing V_G, one should place the interval limit s_w in a region where the free energy derivative is not large. For example, if one uses the keyword with a coordination number CV (see Section 4.6), that is intrinsically limited to $s > 0$, the correct choice for s_w is not 0, but a number of the order of the width of the Gaussians (σ in Eq. 3.1).

- This remedy has the advantage of being robust and parameter-free, but works only for one-dimensional biases.

- If in the region $s < s_w$ the system has a free energy minimum, the `INTERVAL` keyword should be used together with a soft wall at s_w (keyword `LWALL`, see section 3.10), namely a time-indipendent external potential that prevents the system from going there and remaining trapped in the minimum. An example is provided below.

Example.
The following input file contains the interval condition in presence of soft walls:

```
HILLS HEIGHT 0.1 W_STRIDE 500
COORD LIST <g1> <g2> NN 8 MM 12 D_0 0 R_0 0.25 SIGMA 0.15
g1->
26 27
g1<-
g2->
29 30
g2<-
INTERVAL CV 1 LOWER_LIMIT 0.2 UPPER_LIMIT 2.8
UWALL CV 1 LIMIT 2.8 KAPPA 3000.0
LWALL CV 1 LIMIT 0.2 KAPPA 3000.0
ENDMETA
```

3.4.9 Inversion condition

The `INVERT` keyword activates the inversion condition near predefined boundaries `LIMIT1,LIMIT2`[27]. 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:

$$V(-s,t) \approx 2V(0,t) - V(s,t) \tag{3.3}$$

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 (Gaussian width for the selected CV; `SIGMA`).
- 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:

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

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

The second factor in Eq. (3.4) is approximately one for $\mid s \mid < \chi_2\chi_1$ and goes to zero for $\mid s \mid > \chi_2\chi_1$. This ensures that V goes smoothly to zero out of the boundaries. $\chi_2\chi_1$ thus defines the width of the inversion interval and χ_2 is a scaling factor regulated by the keyword `INVERSION`. χ_1 is regulated by the keyword `REFLECTION` (χ_1 is in σ units). The keyword `MAXHEIGHT` defines the largest $\mid w \mid$ 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 12 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 intrinsically 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
```

If `INVERT` is used in presence of soft walls it is recommended to use KAPPA larger than $KT/\sigma^{\texttt{EXP}}$, where `EXP` is the exponent defined for the wall (see section 3.10). Much smaller values for KAPPA in fact do not efficiently prevent the exploration of CVs regions that are located at more than σ beyond the bounduaries and this may lead to systematics errors. 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 (if not specified the program selects automatically 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.[27].

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. 4.28).
  - With DL_POLY, use the patched version of MakePAR to compile PLUMED.

3.6 Replica exchange methods

When combined with GROMACS (both version 4.0 and 4.5), PLUMED can perform replica–exchange simulations coupled with metadynamics in two

different ways: parallel tempering metadynamics (PTMetaD) [17, 28] and bias-exchange metadynamics (BE-META) [18].

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 has this feature implemented within a single parallel run, whereas the bias-exchange tool in the directory `utilities` allows to perform bias-exchange simulations with any MD engine via the linux shell (see Section 5.6).

The procedure for running bias-exchange simulations is similar to the one described earlier for parallel tempering (i.e., add `-multi nrep -replex nexch` to the mdrun command line, where `nrep` is the number of replicas and `nexch` is the number of steps between attempting exchanges). 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., `plumed0.dat`, `plumed1.dat`, ..., `md0.tpr`, `md1.tpr`, ...). Each plugin input file must contain the directive `BIASXMD` and the same list of CVs, in the same order. Each replica usually has only one (or few) of the CVs active: use a `NOHILLS` directive for each of the inactive CVs (or do not specify `SIGMA` for them). It is also possible to have replicas with all CVs inactive: such unbiased replicas can exchange with biased ones, jumping in this way beyond free-energy barriers, but since they are not subject to the hills they tend to populate the free-energy basins according to equilibrium statistics.

Example.
The following two input files define a bias-exchange metadynamics run:

```
# --- input file plumed0.dat ---
# switching on metadynamics and Gaussian parameters
HILLS HEIGHT 0.1 W_STRIDE 500
# switching on bias-exchange
BIASXMD
# instruction for CVs printout
PRINT W_STRIDE 50
# instruction for labeling of COLVAR and HILLS files
HILLS_LABEL A
# the CVs:  dihedral angles
TORSION LIST 1 5 11 13 SIGMA 0.314
TORSION LIST 11 13 15 21 SIGMA 0.314
# in this replica we bias only CV 1:
NOHILLS CV 2
# end of the input
ENDMETA


# --- input file plumed1.dat ---
# switching on metadynamics and Gaussian parameters
HILLS HEIGHT 0.1 W_STRIDE 500
# switching on bias-exchange
BIASXMD
# instruction for CVs printout
PRINT W_STRIDE 50
# instruction for labeling of COLVAR and HILLS files
HILLS_LABEL B
# the CVs:  dihedral angles
TORSION LIST 1 5 11 13 SIGMA 0.314
TORSION LIST 11 13 15 21 SIGMA 0.314
# in this replica we bias only CV 2:
NOHILLS CV 1
# end of the input
ENDMETA
```

In bias-exchange runs, the plugin will produce one `COLVAR` file and one `HILLS` file for each replica. What is actually exchanged between replicas are the atomic coordinates, not the bias, so that replica 0 (printing out `COLVAR0`) will always be biased by `HILLS0`, replica 1 (printing out `COLVAR1`) will always be biased by `HILLS1`,) and likewise for the other replicas.

In the example above, note the use of the keyword `HILLS_LABEL`: this will print a comment line "`#! ACTIVE 1 1 A`" in files `COLVAR0` and `HILLS0` and "`#! ACTIVE 1 2 B`" in files `COLVAR1` and `HILLS1`, indicating the number of active CVs, their index, and the chosen label. This facilitates the reconstruction of the multidimensional free-energy landscape from the bias-exchange simulation according to the weighted-histogram algorithm in Ref. [29], e.g.

employing the METAGUI program [30] downloadable from
`http://www.plumed-code.org/contributions`

See also Section 5.6 for informations on bias-exchange simulations with MD codes different from GROMACS.

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:

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

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 [31]. 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 [13] or Crooks [14] 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 MSD FRAMESET restrained.pdb
TARGETED TYPE MSD FRAMESET waters.pdb
TARGETED TYPE MSD 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[20, 32, 33].

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

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

where

$$\rho(t) = \left(CV(t) - TO\right)^2 \tag{3.7}$$

and

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

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
```

The corresponding COLVAR will look like:

```
#!  FIELDS time cv1 vbias vwall vext XX XX ABMD1
0.000 1.884703657 0.000000000 0.000000000 0.000000000 ABMD 1 1.243885933
0.200 2.353724409 0.000000000 0.007812004 0.000000000 ABMD 1 0.412082147
0.400 2.433164746 0.000000000 0.000000000 0.000000000 ABMD 1 0.321302206
0.600 2.608463835 0.000000000 0.004202352 0.000000000 ABMD 1 0.149200641
0.800 2.818935636 0.000000000 0.009463992 0.000000000 ABMD 1 0.026631583
1.000 2.831631093 0.000000000 0.147609095 0.000000000 ABMD 1 0.004049192
1.200 2.583559043 0.000000000 7.171877688 0.000000000 ABMD 1 0.004049192
1.400 2.841442463 0.000000000 0.121111568 0.000000000 ABMD 1 0.003130352
1.600 2.794514101 0.000000000 0.382087213 0.000000000 ABMD 1 0.003130352
1.800 2.634418597 0.000000000 4.258829096 0.000000000 ABMD 1 0.003130352
2.000 2.861303394 0.000000000 0.086113847 0.000000000 ABMD 1 0.000677239
```

where the last column is $\rho_m(t)$.

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:

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

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

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
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
42	PCA projection
45	Contact Map
55	SPRINT

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
```

3.12 Projection of gradients

The keyword PROJ_GRAD is enable one to calculate the projection of the gradient in this way:

$$P_{ij}(R) = \sum_{n}^{NAT} \nabla_n s_i(R) \cdot \nabla_n s_j(R) \tag{3.10}$$

where $s_i(R)$ and $s_j(R)$ are two collective coordinates defined in input. The output is stored into the COLVAR file in the form of upper diagonal matrix. The initial line of COLVAR file describes the output in detail.

Example.
The following line instructs plumed to calculate the projection of two collective variables s_1 and s_2.

```
DISTANCE LIST 1 2
TORSION LIST 1 2 3 4
TORSION LIST 2 3 4 8
PROJ_GRAD CV <g1>
g1->
2 3
g1<-
ENDMETA
```

In the COLVAR file you find a PROJ_GRAD keyword followed by $N(N+1)/2$ elements that constitute the upper diagonal matrix (diagonal elements included) of the projection. It is important to know that PROJ_GRAD is NOT supporting the LIST keyword therefore all the variables that one needs to use must be included in a "group" as reported in the example above.

3.13 Reconnaissance metadynamics

Reconnaissance metadynamics is a self-learning algorithm for accelerated dynamics that is able to work with a very large number of collective coordinates. Acceleration of the dynamics is achieved by constructing a bias potential in terms of a patchwork of one-dimensional, locally valid collective coordinates. To understand the details of the methodology please read reference [19]. Furthermore, as detailed in section 2.2, by default PLUMED compiles without reconnaissance metadynamics and so some slight modifications in compilation are required if you wish to run this type of simulation.

For a reconnaissance metadynamics simulation the PLUMED input must contain the definition of at least one CV (see chapter 4 for the required syntax) and the RECONNAISSANCE, BASINS, ONIONS and CLUSTER keywords. These keywords provide the parameters for reconnaissance metadynamics simulations and are described in sections 3.13.2 and 3.13.3.

3.13.1 Typical output

A reconnaissance metadynamics simulation produces at least four output files. These files are called BASINS, ONIONS, PPCA_DIAGNOSTICS and CLUSTER_DATA.0. The first of these files contains the locations of the various basins found during cluster analysis and their shapes. This data is formatted as follows:

- The first column contains the basin number
- The second column contains the timestep at which the basin was created
- The following d columns contain the values of the collective coordinates at the basin center
- The next d^2 columns contain the covariance of the collective coordinates inside the basin

- The final column contains the value of the "diffusion constant" in this particular basin.

The second file, `ONIONS`, gives the details of the bias. In reconnaissance metadynamics the bias consists of Gaussian functions that are added to the distance from basin centers. These Gaussian functions are only added when one is within a certain distance of the basin center, the basin size. This basin size changes as the simulation progresses and the `ONIONS` file stores the details of all basin expansion events as well as the details of the hill addition events. Each event is recorded on a single line of the ONIONS file, which is formatted as follows:

- The first column contains the time step at which the contribution to the bias potential was added.
- The second column contains the number of the basin to which this particular Gaussian was added
- The third column contains the distance from the basin center specified in column 2 at which the centroid of the Gaussian is to be found
- The fourth column contains the width of the Gaussian
- The fifth column contains the height of the Gaussian
- The final column contains the current size of the basin specified in column 2.

Basin expansion events can be distinguished from hill addition events as in the former case the height of the "Gaussian" in column 4 is zero.

The `PPCA_DIAGNOSTICS` file contains a information on how successful the PPCA algorithm, that is used for clustering, has been. By default fits of the data are performed with 1 to 10 Gaussian functions. Bayesian information criteria ($-\log(L) + n_p \log(M)$, $\log(L)$, n_p and M being the log likelyhood, number of parameters and number of datapoints respectively) and fuzzy volumes ($\sum_{n=1}^{N} \sqrt{|\mathbf{\Sigma}_n|}$) are reported for each fit. The best fit to the data is the one with the minimum value for the Bayesian Criterion and it is from this fit that the bias is constructed. In our early applications of the methodology we have found that the if a large number of Gaussian is regularly needed to fit the collected data the method will not be successful. However, a change

of cvs can often resolve this problem. After the PPCA_DIAGNOSTICS file has reported on the quality of the fits with different numbers of Gaussians the weights of the various Gaussians in the best fit to the data are reported along with information on which Gaussians have been accepted into the biassing strategy.

All the remaining files produced during a reconnaissance metadynamics simulation are only required to restart simulations.

Unlike in normal metadynamics one cannot obtain the free energy surface from an examination of the bias potential at the end of the simulation. However, a graphical tool that can be added to vmd is available from the PLUMED website. With this tool one can visualize the results from from reconnaissance metadynamics simulation.

3.13.2 Controlling the clustering

As discussed in reference [19] in reconnaissance metadynamics one periodically analyses the trajectory in order to obtain collective coordinates to bias. The frequency of these analyses are controlled by the CLUSTER keyword. The line containing this keyword should also contain an integer, which controls the frequency at which the values of the cvs are stored for subsequent cluster analysis and a integer than controls the frequency of clustering. The first of these quantities should be proceeded by STORE_FREQ and the second by RUN_FREQ. One may cluster over multiple timescales separately by including this line multiple times.

Example.
The following lines tell PLUMED to run cluster analyses every 10000 steps and every 100000 steps

```
RECONNAISSANCE
CLUSTER RUN_FREQ 100000 STORE_FREQ 100
CLUSTER RUN_FREQ 1000000 STORE_FREQ 1000
```

Once the clustering has finished PLUMED must make certain decisions on how these basins should be incorporated into the biassing strategy. The parameters controlling these decisions are given on the line starting with the BASINS keyword. This line provides three parameters:

- BASIN_TOL - the fraction of the trajectory that must be within a cluster in order for it to be used for biassing.

- `INITIAL_SIZE` - the initial size of the basin
- `EXPAND_PARAM` - the parameter that controls the basin expansion

Guidance on how to set the value of `BASIN_TOL` is provided in the supplementary information of reference [19]. The supplementary information also shows how, when the angular variables of a multivariate Gaussian are integrated out, the resulting probability distribution as a function of r (the distance from the center of the basin in the metric of the basin) is equal to a Gaussian with standard deviation $\frac{1}{\sqrt{2}}$ centered at $\sqrt{d-1}$. With this in mind the initial size of basins is set equal to $\sqrt{d-1}$ + the value of the initial size parameter. Therefore a sensible value for this parameter is around 1.5 - i.e. approximately two standard deviations.

The final parameter described above `EXPAND_PARAM` controls the way basins change size as a function of time. As discussed in the supplementary information of reference [19] we introduce a probabilistic criterion for basin expansion that is derived using ideas from the theory of diffusion. Since writing that paper we have slightly changed the way that this parameter functions. In particular we realized that the diffusion constant in a given basin could be also be fit from the data using:

$$D = \frac{1}{d\Delta t(M-1)} \sum_{i=2}^{M} |\mathbf{s}_i - \mathbf{s}_{i-1}|^2 \tag{3.11}$$

where M is the number of vectors used for clustering, d is the dimensionality, Δt is the `STORE_FREQ` parameter and the squared differences between vectors of CVs in the sum are calculated in the metric of the basin. The `EXPAND_PARAM` provided in the input should be a number between 0 and 1. It is used to scale this estimate value and expresses the level of confidence a user has in the fitting. In practice this parameter gives some control over the rate of exploration of phase space. If it is large phase space will be explored rapidly, while if it is small phase space will be explored more slowly.

3.13.3 Controlling the bias

The bias in reconnaissance metadynamics consists of a number of Gaussian functions that are periodically added to the potential. These Gaussians are different from those added in normal metadynamics because they act added

on the distance from the center of a basin. Hence, the keyword that controls the hill addition in reconnaissance metadynamics is `ONIONS` and NOT `HILLS`. However, the syntax for `ONIONS` is the same as the syntax for `HILLS`. That is to say one must provide:

- `HEIGHT` - The height of the Gaussian functions.
- `W_STRIDE` - The frequency with which to attempt hill addition.
- `WIDTH` - The width of the Gaussian functions.

Once again guidelines as to how to set these parameters are provided in the supplementary information of reference [19].

Bringing all of the above together a typical input file for a reconnaissance metadynamics simulation should look like this:

Example.
Typical input for a reconnaissance metadynamics simulation

```
RECONNAISSANCE
ONIONS HEIGHT 1.0 W_STRIDE 1000 WIDTH 1.5
BASINS BASIN_TOL 0.2 EXPAND_PARAM 0.3 INITIAL_SIZE 3.0
CLUSTER RUN_FREQ 500000 STORE_FREQ 250
TORSION LIST 13 15 17 19
TORSION LIST 15 17 19 21
```

3.13.4 Restarting a simulation

To restart a reconnaissance metadynamics simulation, the flag `RESTART` must be added on the line containing the directive `RECONNAISSANCE`. This allows one to restart the simulation after an interruption or after a run has terminated. At the start of the simulation the `BASINS`, `ONIONS` and `CLUSTER_DATA.*` files are read in so the bias can be restarted. Data from the restarted simulation will then be appended to the `BASINS` and `ONIONS` files. Please note that the `RESTART` flag is a directive to `PLUMED` to restart the reconnaissance metadynamics simulation. It not an instruction to restart the MD run. Hence, to restart the simulation one must also follow the usual procedure for restarting the MD run. This will depend on the particular MD engine you are using -

details can be found in the documentation for the particular MD code you are using.

Example.
A sample input file for restarting a reconnaissance metadynamics simulation

```
RECONNAISSANCE RESTART
ONIONS HEIGHT 1.0 W_STRIDE 1000 WIDTH 1.5
BASINS BASIN_TOL 0.2 EXPAND_PARAM 0.3 INITIAL_SIZE 3.0
CLUSTER RUN_FREQ 500000 STORE_FREQ 250
TORSION LIST 13 15 17 19
TORSION LIST 15 17 19 21
```

3.13.5 Using a subset of the defined cvs

By default reconnaissance metadynamics is performed using all the collective coordinates defined in the PLUMED input file. However, there may be occasions where it is desirable to include some constraints to prevent exploration of the uninteresting parts of phase space. In these cases it would also be desirable to not include these cvs in the reconnaissance metadynamics. Consequentially, a keyword exists which allows one to explicitly specify which of the CVs are to be used in the reconnaissance metadynamics. This keyword is CV_LIST and it should be added to the line containing the directive RECONNAISSANCE. It should be followed by a tag which gives the name of a list of the cvs to be used for reconnaissance. This list should be specified using the syntax described in section 4.

Example.
A sample input file for a reconnaissance metadynamics simulation using only three of the four collective coordinates specified in the PLUMED input file

```
RECONNAISSANCE CV_LIST <cvlist>
ONIONS HEIGHT 1.0 W_STRIDE 1000 WIDTH 1.5
BASINS BASIN_TOL 0.2 EXPAND_PARAM 0.3 INITIAL_SIZE 3.0
CLUSTER RUN_FREQ 500000 STORE_FREQ 250
TORSION LIST 13 15 17 19
TORSION LIST 15 17 19 21
TORSION LIST 17 19 21 23
TORSION LIST 19 21 23 25
cvlist->
1 2 3
cvlist<-
```

3.14 Driven Adiabatic Free Energy Dynamics (d-AFED)

The driven adiabatic free energy (d-AFED) algorithm [34] is a variant of the earlier AFED method [35, 36], which required cumbersome coordinates transformations. In the d-AFED method, an extra dynamical variable S is coupled to a collective variable $s(\mathbf{r})$, where $\mathbf{r}$ represents the coordinates of a number of atoms in the system. The coupling is mediated by a potential energy function with harmonic constant κ,

$$V(S, s(\mathbf{r})) = \frac{1}{2}\kappa \left(S - s(\mathbf{r})\right)^2 . \tag{3.12}$$

The dynamics of the S meta-variable is adiabatically decoupled from the dynamics of the underlying physical system by choosing a large mass $m_S \gg \bar{m}$, where $\bar{m}$ is a typical mass of the physical system. Thanks to the adiabatic separation, a temperature $T_S > T$ can be assigned to the S meta-variable. With this choice of m_s and T_s, the physical system will evolve fast at room temperature T around the instantaneous value of $s(\mathbf{r}) = S$. On the other hand, S will evolve slowly, but have a temperature large enough to drive the system over high free energy barriers.

In the limit of $\kappa \to \infty$, it can be shown that the free energy surface at temperature T can be recovered from the density $\rho^{\mathrm{adb}}(S)$ sampled at temperature T_S during the adiabatic d-AFED simulation using

$$\Delta G(S) = -k_B T_S \log\left(\rho^{\mathrm{adb}}(S)\right). \tag{3.13}$$

This result generalizes well to the case where more than one collective variable is used and $\Delta G(S)$ is a multi-dimensional free energy surface. Note that the d-AFED method is similar to the Temperature Accelerated Molecular Dynamics (TAMD) method devised by other authors[37].

The d-AFED method requires very efficient thermostatting of the meta-variable S. Therefore, in the present implementation, S is coupled to a Generalized Gaussian Moment Thermostat (GGMT) [38]. The meta-variable is coupled to two thermostatting variables p_η and p_ζ, with associated masses Q_η and Q_ζ, respectively. Given a typical time scale τ of the thermostated system, optimal masses are $Q_\eta = k_B T_S \tau^2$ and $Q_\zeta = \frac{8}{3}(k_B T_S)^3 \tau^2$. The order-2 GGMT dynamics for one degree of freedom is

$$\dot{p}_S \;=\; V(S, s(\mathbf{r})) - \frac{p_\eta}{Q_\eta} p_S - \frac{p_\zeta}{Q_\zeta}\left[k_B T_S + \frac{1}{3}\frac{p_S^2}{m_S}\right] p_S, \tag{3.14}$$

$$\dot{p}_\eta = \frac{p_S^2}{m_S} - k_B T_S, \tag{3.15}$$

$$\dot{p}_\zeta = \frac{1}{3}\left[\frac{p_S^2}{m_S}\right]^2 - (k_B T_S)^2, \tag{3.16}$$

$$\dot{S} = \frac{p_S}{m_S}, \quad \dot{\eta} = \frac{p_\eta}{Q_\eta}, \quad \dot{\zeta} = \frac{p_\zeta}{Q_\zeta}. \tag{3.17}$$

If multiple reaction coordinates are used, one separate GGMT thermostat is associated to each of them. The implemented integrator for the dynamics above is based on a Trotter decomposition of the corresponding Liouville operator [34]. The quality of the integration can be monitored using the quantity H_S, which would be conserved if the dynamics of S was decoupled from the physical system,

$$H_S(S, p_S, \eta, p_\eta, \zeta, p_\zeta) = \frac{p_S^2}{2m_S} + V(S) + \frac{p_\eta^2}{2Q_\eta} + \frac{p_\zeta^2}{2Q_\zeta} + k_B T_S\,(\eta + \zeta) \tag{3.18}$$

The heat transfer W_S from the meta-variable to the physical system can be calculated,

$$W_S = \int_0^t dt' \kappa\,(S - s(\mathbf{r}))\,\frac{p_S}{m_S} \tag{3.19}$$

The effective adiabaticity of the coupling can thus be asserted. In addition, for each collective variable, the quantity $H_S - W_S$ should be strictly conserved and provides a quality check for the simulation.

3.14.1 Input for d-AFED

For each CV, a `DAFED` directive is used to define the parameters of the corresponding dynamics. On the same line, the number of the CV to which the directive applies is specified after the keyword `CV`. The temperature T_S in K is given after keyword `TEMPERATURE`. The thermostat time constant τ is given in ps after keyword `TAUTHERMO`. The mass m_S and harmonic constant κ, are given after the keywords `MASS` and `KAPPA`, respectively. The units of κ and m_S depend on the nature of the CV. They should always be such that κS^2 and $m\dot{S}^2$ are both in units of energy (kJ/mol= amu nm^2 / ps^2), see the example below.

In addition, tow optional keywords can be used with the `DAFED` directive. First, for periodic CVs such as torsion angles, the S variable should also

evolve on a periodic interval. This is specified by the keyword `PERIODIC`, followed by two numbers for the lower and upper bounds. The numbers can be replaced by `MINUS_PI`, `PLUS_PI`, or `PLUS_2PI` to specify $-\pi$, $+\pi$, or $+2\pi$, respectively.

The optional keyword `JACOBIAN_FORCE` causes a bias force $F = -2k_BT/S$ to be applied to the dynamics of S. This is useful with distance CVs in order to counterbalance the effect of the Jacobian factor and sample a more uniform distribution along the CV.

Example.
The following lines couple CV 1 (a distance in nm) to a meta-variable of mass 10^5 amu with a harmonic constant of 10^6 kJ/mol/nm^2 and CV 2 (a unitless number) to a meta-variable with mass 10^3 amu*nm^2 with a harmonic constant of 10^4 kJ/mol. For both CV, the d-AFED temperature is 600 K and the GGMT thermostat time constant is 0.2 ps. See text for the optional keywords `JACOBIAN_FORCE` and `PERIODIC`.

```
DISTANCE LIST 1 34
TORSION LIST 5 15 29 36

DAFED CV 1 TEMPERATURE 600 MASS 1e5 KAPPA 1e6 TAUTHERMO 0.2 JACOBIAN_FORCE
DAFED CV 2 TEMPERATURE 600 MASS 1e3 KAPPA 1e4 TAUTHERMO 0.2 PERIODIC MINUS_PI PLUS_PI

DAFED_CONTROL RESTART checkpoint_file WRITE_STATE -1 N_RESPA 1

PRINT W_STRIDE 100
ENDMETA
```

A separate `DAFED_CONTROL` directive contains general controls for the d-AFED simulation. The d-AFED dynamics, including all variables described in Eqs. (3.14 - 3.17) can be restarted exactly from a previous run using a checkpoint file. Following the `WRITE_STATE` keyword appears the number of steps after which a checkpoint file is saved. A value of -1 implies that a checkpoint file is written only when GROMACS saves its own checkpoint file, i.e. at regular wall clock time intervals. The checkpoint file is saved in the current directory with default name `DAFED_STATE`. The optional keyword `RESTART` is used to specify the path to the checkpoint file from which to restart.

3.14.2 Typical output for d-AFED

WIth the d-AFED method, the `COLVAR` file will contain the following data, if d collective variables are used:

- time step
- value of the collective variable $s_1(\mathbf{r})...s_d(\mathbf{r})$

Then for each of the $S_j,\ j = 1...d$, appears a set of 4 columns with :

- the meta-variable S
- the instantaneous temperature of S in K
- the conserved quantity H_S, see Eq. 3.18, in kJ/mol
- the work W_S from S to the physical system, see Eq. 3.19, in kJ/mol

These fields are labeled `sj`, `T_sj`, `E_sj`, and `W_sj`, respectively, in the `COLVAR` header line, $j = 1...d$. Additional collective variables can be monitored during a d-AFED run, in which case more columns will appear before the first d-AFED keyword.

Note that for each extended variable S_j, the quantity $H_{S_j} + W_{S_j}$ should be conserved, $j = 1...d$. Let $\mathcal{H}$ be the pseudo-energy of the physical system including the associated thermostats and barostats. Then the total energy of the simulation, $\mathcal{H} + \sum_{j=1}^{d} H_{S_j}$, should be conserved as well. In addition, considering the physical system only, the quantity $\mathcal{H} - \sum_{j=1}^{d} W_{S_j}$ should be conserved.

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. 4.28.

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 if 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. 4.28.

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<-
```

It is also possible to use the sine or cosine of the angle as a collective coordinate by including the SIN or COS keywords respectively similarly to what is done for TORSION.

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 to 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 ith atom of group <g1> and the ith 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. 4.28.

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:

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

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

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.
- If `TYPE 5` is specified, only hydrogen bonds between atoms belonging to different residues are counted. Moreover, pairs with an index difference less than 5 are also discarded, so as to avoid counting H and a O that are in the same peptide group (NH-C=O) (contributed by M. Cuendet). This option requires the user to specify to which residue each atom is belonging using the RESLIST keyword (see example below).

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
```

In the following example, inter-residue hydrogen bond are counted

```
HBONDS LIST <H> <O> RESLIST <resH> <resO> TYPE 5
H-> 1 9 17 H<-
O-> 5 13 21 O<-
resH-> 1 2 3 resH<-
resO-> 1 2 3 resO<-
```

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. 4.28.

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:

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

The syntax of the command requires the user to specify three groups of atoms 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 :

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

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

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

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
```

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. 4.28.

4.9.1 Gyration tensor based CVs

The `RGYR` directive can be used also to access to a number of CVs based on gyration tensor [39]:

$$S = \frac{1}{N} \begin{pmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i z_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i z_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i^2 \end{pmatrix}, \tag{4.1}$$

which describes the spatial distribution of mass in a molecule or complex of molecules. Alternatively, tensor of inertia may be used for the same purpose:

$$\mathbf{I} = \begin{pmatrix} \sum m_i(y_i^2 + z_i^2) & \sum -m_i x_i y_i & \sum -m_i x_i z_i \\ \sum -m_i x_i y_i & \sum m_i(x_i^2 + z_i^2) & \sum -m_i x_i z_i \\ \sum -m_i x_i z_i & \sum -m_i y_i z_i & \sum m_i(x_i^2 + y_i^2) \end{pmatrix}. \tag{4.2}$$

The weighting of atomic contribution is controlled by keyword `MASS-WEIGHTED` (default) and `NO-WEIGHTED`. When `MASS-WEIGHTED` directive is applied, the individual atomic contribution to the gyration tensor is weighted by $\frac{m_i N}{\sum m_i}$ and the center of mass is used as origin of the system. If `NO-WEIGHTED` option is chosen, coordinates are related to the geometrical center of object and no mass-weighting is performed in calculation of CV.

Diagonalizing of gyration tensor provides the principal moments of gyration - S_1,S_2,S_3 and three eigenvectors corresponding to the principal axes of inertia. The individual gyration tensor based CVs are available by specifying the directive `RGYR` and the keyword `TYPE` followed by one of the following:

- `TRACE`: trace of inertia tensor
- `GTPC_1`: the largest eigenvalue of gyration tensor. The square root $S_1' = \sqrt{S_1}$ of principal moment S_1 is used as CV. If we approximate the shape and mass distribution of the object with an ellipsoid, S_1' corresponds to the largest radius of this ellipsoid.
- `GTPC_2`: the middle principal moment of gyration tensor, $S_2' = \sqrt{S_2}$.
- `GTPC_3`: the smallest principal moment of gyration tensor, $S_3' = \sqrt{S_3}$.
- `RGYR_1`: the largest radius of gyration around principal axes of inertia, $r_{g1} = \sqrt{S_1 + S_2}$.

- RGYR_2: the middle radius of gyration around principal axes of inertia, $r_{g2} = \sqrt{S_1 + S_3}$.
- RGYR_3: the smallest radius of gyration around principal axes of inertia, $r_{g3} = \sqrt{S_2 + S_3}$.
- ASPHERICITY: the deviation of mass distribution from spherical symmetry. The modified version of asphericity was implemented as a CV: $b' = \sqrt{b}$, where original Suter's [40] asphericity is given by $b = S_1 - \frac{1}{2}(S_2 + S_3)$.
- ACYLINDRICITY: the deviation from cylindrical symmetry. Again the modified version $c' = \sqrt{c}$ is used as a CV rather then original form [40] $c = S_2 - S_3$.
- KAPPA2: the relative shape anisotropy[40] $\kappa^2 = 1 - 3\frac{S_1S_2+S_1S_3+S_2S_3}{(S_1+S_2+S_3)^2}$ reflects both symmetry and dimensionality of an object. Limited between values of 0 and 1, κ^2 reaches 1 for an ideal linear arrangement and drops to zero in case of highly symmetric configurations (at least tetrahedral symmetry).

Example.
The following lines instruct PLUMED to use the trace of the inertia tensor of the group <g1> as a CV.

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

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

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

4.10 Dipole

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

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

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:

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

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 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 **??**). It is calculated using:

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

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:

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

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 $\{\mathbf{R}^0\}$:

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

See Ref.[41] 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.[41] 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.[41] 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 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:

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

where

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

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<-
```

4.17 Puckering coordinates

PUCKERING refers to the set of collective coordinates for 6-membered rings in polar coordinates [42]. 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<-
```

4.18 Path collective variables

One can instruct `PLUMED` to use path collective variables [43] 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, $d : \mathcal{X} \times \mathcal{X} \to \mathbf{R}_0^+$ is a metric on $\mathcal{X}$, and $\mathcal{Z} = \sum_{i=1}^{N} e^{-\lambda d(X_i, X(t))}$ is a normalization factor in which the prefactor, λ, should be chosen so as to have $\lambda d(X_i, X_{i\pm1}) \simeq 2.3$ 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 `MSD` (mean square deviation see section 4.18.1), `DMSD` (distance root mean squared deviation see section 4.18.2) or `CMAP` (the distance between contact matrices see section 4.18.3) [44].

- `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.* mean square displacement in Cartesian coordinates, MSD in distances or contact map distance).

An important change occurred from the version 1.2 to 1.3 : the convention for the names has changed the former `RMSD` has been replaced by `MSD`. Similarly `DRMS` has been changed into `DMSD`. This was done since many users noticed the inconsistencybetween the definitions and the related keywords.

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.

4.18.1 Mean square deviation

As already briefly mentioned, when using the path collective coordinates `S_PATH` and `Z_PATH`, the command `MSD` after the `TYPE` keyword instructs `PLUMED` to the the 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

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

where $M_{ij}(\{z\})$ is the roto-translation matrix calculated using the Kearsley [45] algorithm and w_j is the number specified (called the *displacement* parameter) for the beta column in the provided input PDB. If this is one then w_j=1 and w_{TOT}=N (while differently $w_{TOT} = \sum_i^N w_i$). The weights for the alignment are actually taken into account when forming the alignment matrix that compose the $M_{ij}(\{z\})$ and are denoted here with the set $\{z\}$ defined by the beta column in the reference PDB.

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 (¡basename¿) after the keyword `FRAMESET` and be aware that `PLUMED` then will expect to find N files named ¡basename¿.i.pdb, which contain the coordinates of the atoms in both the *displacement* and *alignment* sets for the ith 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 MSD.

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

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

```
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
```

4.18.2 Distance mean square deviation

Instead of using the MSD in the definition of the metric for the path in the S_PATH and Z_PATH CVs the command DMSD after the TYPE keyword allows one to use a metric based on the mean square deviation of the distances between a subset of the atoms in the system:

$$d(X_j, X_i) = \frac{2}{N_{\mathcal{A}}(N_{\mathcal{A}}-1)} \sum_{a=1}^{N_{\mathcal{A}}-1} \sum_{b=a+1}^{N_{\mathcal{A}}} (r_{ab}^{(j)} - r_{ab}^{(i)})^2,$$

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 4.18.1).

4.18.3 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:

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

where $\theta(x)$ is a step function 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 MSD and DMSD 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. 4.28.

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 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
```

4.18.4 Using path variables as MSD, DMSD 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 MSD, distance DMSD 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 MSD metrics.

```
PRINT W_STRIDE 10
TARGETED TYPE MSD 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 MSD/DMSD (see section 4.18.1).

```
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 CMPAVALUES file see section 4.18.3).

Additionally one can specify another keyword SQRT that transform the metrics in its square root, namely the mean square deviation would be

changed into the more commonly used root mean square deviation. Similarly it applies to DMSD and CMAP. It should be stressed that the use of this keyword is particularly dangerous especially for values close to zero since the square root has a cusp there. The former example would look like:

Example.

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

4.19 Contact Map

The Contact Map is defined as the sum of the contacts between a number of atom pairs specified by the user. A contact is defined in terms of the switching functions introduced for the coordination number CV in section 4.6. Each contact can have its own set of parameters, which are defined in a file specified by the keyword INDEX. See the documentation in paragraph 4.18.3 for a detailed explanation of the index file syntax.

As for PCV in contact map space (see paragraph 4.18.3), one can define contacts between the centers of mass of groups of atoms. The name of the file in which groups are defined is specified by using the GROUP keyword. See the documentation in paragraph 4.18.3 for additional details.

The optional flag NOPBC can be used to calculate distances without applying periodic boundary conditions.

Example.
The following command defines the CV Contact Map. Distances are calculated without periodic boundary conditions.

```
CMAP INDEX fr.ndx SIGMA 1.  NOPBC
```

The fr.ndx file should contain the details on how each of the switching functions in the contact matrix should 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
```

4.20 Energy

The ENERGY keyword instructs PLUMED to use the total potential energy of the system [46, 47, 48, 49] 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:

$$s = \sum_{i=2}^{N-1} \prod_{j=i-1}^{i+1} \frac{1}{4} \left[cos(\Phi_j - \bar{\Phi}_i) + 1 \right] \left[cos(\Psi_j - \bar{\Psi}_i) + 1 \right], \tag{4.3}$$

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 PCA projection

The PCA keyword instructs PLUMED to use as CV the projection of a set of atoms on a previously calculated Principal Components Analysis (PCA) eigenvector[50]. A typical application of this CV is to explore the system along its principal directions of fluctuations [51].

The current conformation X is first aligned to a reference structure X^{ref} (except if the optional NOALIGN keyword is used) and then projected on the specified eigenvector e^0. The set of atoms used for alignment must be the same as the one the eigenvector refers to. The user should specify the filename of the reference structure after the keyword FRAME and the filename of the eigenvector after the keyword EIGENVEC.

If the optional DIFF keyword is used, the difference between the current (aligned) conformation and the centered reference structure is projected on the eigenvector.

The CV is calculated as :

$$s_{PCA}(X, X^{ref}, e^0) = \sum_{i=1}^{N} < \mathcal{R}^{ref}(X_i - X^{CM}), e_i^0 >$$

or, when using the DIFF keyword, as :

$$s_{PCA}(X, X^{ref}, e^0) = \sum_{i=1}^{N} < \mathcal{R}^{ref}(X_i - X^{CM}) - X_0^{ref}, e_i^0 >$$

where N is the number of atoms used to align and project, $\mathcal{R}^{ref}$ is the 3×3 rotation matrix, calculated using the Kearsley [45] algorithm, that optimally overlaps the current centered set of atoms $(X_i - X^{CM})$ into the centered reference set of atoms X_0^{ref}, X^{CM} is the current centroid of conformation X (i.e. identical masses are assumed) and $<,>$ denotes the usual inner product.

If the optional NOALIGN keyword is used, no alignment is performed (i.e. the above rotation matrix $\mathcal{R}$ is the identity). This latter keyword is supported mainly for debug purposes.

Example.
The following command defines a CV as projection, along the eigenvector listed in `egv0.dat`, of the current conformation aligned to the reference structure specified in `ref.dat`:

```
PCA FRAME ref.dat EIGENVEC egv0.dat SIGMA 0.1
```

where `ref.dat` contains the reference structure in the format ATOMID X Y Z:

```
# my reference structure
2 1.324 1.045 1.550
5 1.316 1.174 1.469
6 1.345 1.174 1.350
7 1.287 1.286 1.538
```

and `egv0.dat` contains the eigenvector in the same format:

```
# my first PCA eigenvector
2 0.12 3.28 0.19
5 0.53 1.37 1.10
6 -0.23 1.93 0.33
7 5.32 -2.56 1.44
```

More eigenvectors can be used with additional `PCA` keywords as additional CVs, nevertheless, the set of atoms and the reference structure must be the same for all of them.

Example.
The following command defines two CVs as projections, along two distinct eigenvectors listed in `egv0.dat` and `egv1.dat`, of the difference between the current aligned conformation and the reference structure specified in `ref.dat`:

```
PCA FRAME ref.dat EIGENVEC egv0.dat DIFF SIGMA 0.1
PCA FRAME ref.dat EIGENVEC egv1.dat DIFF SIGMA 0.1
```

Some final notes :

- the atom and eigenvector coordinates should be in engine units (e.g. in nm for GROMACS)
- the alignment is not mass-weighted (identical masses for all the involved atoms are assumed), and since the routine used for the alignment is the same as the one used to calculate the RMSD, the same cautions hold. In particular, (i) users should ensure that the value of the hardcoded limit (`MAXATOMS_RMSD` in `metadyn.h`) is sufficiently large for their needs prior to compilation, (ii) the use of double precision code is recommended as well as (iii) the use of the directive `ALIGN_ATOMS`. See also 4.28.

- starting the simulation from the identical conformation used for alignment should be avoided since it could generate numerical instabilities in the calculation of the rotation matrix, possibly leading to a crash of the simulation.

- comments line beginning with '#' are allowed but blank lines must be avoided from the input files.

4.23 SPRINT topological variables

The `SPRINT` keyword instructs `PLUMED` to use as CV the "Social PeRmutation INvarianT" coordinates described in Ref. [52]:

$$S_i = \sqrt{N}\,\lambda^{\max}\,v_i^{\max,\text{sorted}}$$

where $\lambda^{\max}$ and $v_i^{\max}$ are the largest eigenvalue and the corresponding eigenvector of the $N \times N$ contact matrix among N atoms:

$$\sum_{j=1}^{N} C_{ij}\,v_j^{\max} = \lambda^{\max}\,v_i^{\max}$$

and C_{ij} is the same coordination function defined for the CV `COORD` in Section 4.6, with the corresponding parameters specified by the keywords `R_0`, `D_0` (both in units of the host MD code, e.g. Bohr for CPMD), `NN`, and `MM`. In S_i the eigenvector components are sorted from the smallest to the largest, so that $S_1 < S_2 < S_3 < ... < S_N$. Which S_i has to be used as CV is specified by the keyword `INDEX`: e.g. `INDEX 2` means that the 2^{nd} one has to be used, namely S_2. Note that the sorting of the eigenvector is performed only within sets of atoms of the same element (automatically identified in `PLUMED` by the atomic mass): if the list of N atoms includes e.g. two carbon and four hydrogen atoms, the `SPRINT` coordinates are sorted only within carbon atoms and within hydrogen atoms, without mixing the two elements (since atoms of different elements are not indistinguishable). Due to this, after keyword `R_0` it must be specified the number of different element pairs and the `R_0`-value for each possible pair of elements. The same holds for `D_0`. See below for an example.

Example.
The following command defines a SPRINT CV S_2 from a set of two carbon (1,2) and four hydrogen atoms (3,4,5,6):

```
SPRINT LIST <all> NN 6 MM 12 R_0 3 5.0 4.2 4.2 INDEX 2 SIGMA 1.0
all->
1 2 3 4 5 6
all<-
```

In the example above `R_0 3 5.0 4.2 4.2` refers to the three pairs of elements C-C, C-H, and H-H.

Note that it is desirable to keep a tail of the coordination function long enough so that the system appears formally as a single connected cluster of atoms. If a part is disconnected from the rest, the Perron-Frobenius theorem does not hold anymore and several S_i components will go to zero (see Ref. [52]).

4.24 Radial distribution function

The keyword `RDF` instructs plumed to use the number of distances between atoms in a given range as a collective variable. A recent paper [53] showed how a number of such collective variables could be used to define the instantaneous radial distribution function and how this description of the structure of small clusters could be used to enhance sampling in reconnaissance metadynamics simulations.

The number of distances within a given range is calculated using:

$$s = \sum_{i,j} \int_a^b w(r - r_{ij}) \mathrm{d}r \tag{4.4}$$

where r_{ij} is the distance between atoms i and j and w is a Gaussian window function with width σ. When using multiple such CVs calculating the set of distances separately for each bead in the RDF would be computational expensive and counterproductive. As such all RDF collective coordinates that involve the same atoms are calculated at the same time. In order that multiple RDF collective coordinates can be used in a single calculation RDF beads are labeled using the RDF_LABEL keyword. Obviously if all your RDF CVs come from the same RDF the RDF_LABEL should be 1 for all your RDF coordinates. An example input for this type of CV is as follows:

Example.
The following command instructs plumed to calculate two collective coordinates the number of distances between 3.0 and 3.5 and the number of distances between 3.5 and 4.0. The value of σ in this calculation is 0.25.

```
RDF RDF_LABEL 1 LIST <all> <all> RANGE 3.0 3.5 WIDTH 0.25
RDF RDF_LABEL 1 LIST <all> <all> RANGE 3.5 4.0 WIDTH 0.25
all->
1 2 3 4 ...
all<-
```

Example.
The following command instructs plumed to calculate two collective coordinates the number of distances between atoms in group ¡all¿ between 3.0 and 3.5 and the number of distances in group ¡all2¿ between 3.5 and 4.0. The value of σ in this calculation is 0.25.

```
RDF RDF_LABEL 1 LIST <all> <all> RANGE 3.0 3.5 WIDTH 0.25
RDF RDF_LABEL 2 LIST <all2> <all2> RANGE 3.5 4.0 WIDTH 0.25
all->
1 2 3 4 ...
all<-
all2->
5 6 7 8 ...
all2-<
```

4.25 Angular distribution function

Rather than using the radial distribution function as a collective variable one can use the distribution of angles between central atoms and the atoms in the first hydration sphere [53]. Once again angular distribution functions of this type have been combined employed successfully with the reconnaissance metadynamics algorithm. The number of angles within a given range is calculated using:

$$w = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sigma(r_{ij})\sigma(r_{ik})\int_a^b w(\theta-\theta_{jik})\mathrm{d}\theta \tag{4.5}$$

$$\text{where} \qquad \sigma(r) = \frac{1-\left(\frac{r-d_0}{r_0}\right)^n}{1-\left(\frac{r-d_0}{r_0}\right)^m} \tag{4.6}$$

where w is once again a Gaussian window function with width σ. The two switching functions $\sigma(r_{ij})$ and $\sigma(r_{ik})$ are used to make sure that one only takes angles in the first hydration sphere. A detailed description of how to set the parameters in these functions can be found in the section of this manual on coordination numbers. Much as was described above for the RDF the RDF_LABEL keyword ensures that the code does not waste time calculating all the angles involved in these ADF beads multiple times. If all your ADF CVs are based on the positions of the same atoms then RDF_LABEL should be 1 for all your ADF coordinates. In other words this keyword should be used to differentiate between the various angular distributions functions you are calculating in input.

Example.
The following command instructs plumed to use the number of angles in the first coordination sphere that are between 0.5 and 1.0 radians as a CV. The value of σ in this calculation is 0.25.

```
ADF RDF_LABEL 1 LIST <all> <all> <all> RANGE 0.5 1.0 WIDTH 0.25 R_0 3.0 NN 6 MM 12 D_0
1.5
all->
1 2 3 4 ...
all<-
```

4.26 Polynomial combination of CVs

Polynomial combinations of collective variables, given by the functional form

$$\sum_{i=1}^{k} c_i(\mathrm{CV}_i - s_i)^{n_i}$$

can be defined using the `POLY` directive, followed by the `TERMS` keyword that specifies the number k of terms to be included in the sum. In the next k lines, the compulsory keyword `CV` defines the number CV_i of the collective variable in the i-th term; optionally the values of c_i, s_i and n_i can be specified preceded respectively by the keywords `COEFF`, `SHIFT` and `EXP`. If not specified they assume the default values of $c_i = 1$ $s_i = 0$ $n_i = 1$. Fractional and negative values of the exponent are allowed, but singularities should be carefully avoided by imposing restraints on the regions explored by the collective variables involved.

Warning! To combine more than 20 collective variables ($k > 20$), the hardset allocation for `intpar` and `vecpar` in metadyn.h should be increased accrdingly and the code recompiled.

Example.
The following command defines a new CV as $\mathrm{CV}_1 - \mathrm{CV}_2^2$:

```
POLY TERMS 2 SIGMA 1.0
CV 1
CV 2 COEFF -1 EXP 2
```

More complicated functions might need multiple auxiliary CVs to be defined.

Example.
The following command defines a new CV as $\sqrt{3\mathrm{CV}_1 + \mathrm{CV}_2}$

```
POLY TERMS 2
CV 1 COEFF 3
CV 2
NOHILLS CV 3
POLY TERMS 1 SIGMA 1.0
CV 3 EXP 0.5
```

4.27 Function of CVs

General function of CVs with arbitrary form can be defined using the `FUNCTION` directive. You should define this function AFTER you defined the variables you are combining.

This function, still experimental, makes use of libmatheval library that you should download from `http://www.gnu.org/software/libmatheval/` and compile along with them. You may find useful infos on the syntax in `http://www.gnu.org/software/libmatheval/manual/`. It is still experimental so if you want to compile it you should hack your Makefile after patching. I tried in NAMD2.7. You should change a couple of lines in the Makefile.

Example.
in the Makefile (dots represent the arguments that you already find in the Makefile)

```
CXXBASEFLAGS = ... $(COPTI)/my/path/to/matheval/install_dir_x86_64/include
and later
namd2:  -L/my/path/to/matheval/install_dir_x86_64/include/lib -lmatheval  ...
```

while in SANDER (v10) I had to change

Example.
in $AMBERHOME/src/config_amber.h I added the needed stuff:

```
   AMBERBUILDFLAGS=-DAMBER -DHAVE_MATHEVAL -L/my/path/to/matheval/install_dir_x86_64/lib

-lmatheval -I/my/path/to/matheval/install_dir_x86_64/include
```

It is crucial to put the function of CVs after all the cvs you use to define it!!!!

Example.
The following command defines a new CV as function of the two angles

```
 PRINT W_STRIDE 1

ANGLE LIST 7 9 15
ANGLE LIST 9 15 17
FUNCTION " 2*CV_1 + (CV_2+CV_1)0.2 "
```

Additional note is that the function is not periodic.

4.28 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; 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
 -aver naver         time-average the bias profile over the last naver Gaussians
```

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_bT 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-processesing 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.

`-naver` is used to plot the time average of the bias profile over the last given number of hills (see Eq. 33 in Ref. [23])

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 that the CV is periodic between [-pi;pi] or [0;2pi] 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.

For GROMACS users: note that `driver` has some limitation in the choice of the simulation box shape. With GROMACS, it could be convenient to use the `-rerun` option of `mdrun` together with `PLUMED`.

5.2.1 Installation instructions

Before compiling `driver`, the files contained in the `common_files` directory must be linked to the `utilities/driver` directory (i.e. this can be done using `./getlinks.sh`).

To compile using g95/gcc, type:

```
make arch=g95
```

To compile with gfortran/gcc, type:

```
make arch=gfortran
```

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

```
make arch=gfortan_64
```

To compile with ifort/icc Intel compilers, type:

```
make arch=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 (one frame) 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 (only one frame, not a movie), 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 (i.e. this can be done using `./getlinks.sh`).

To compile using g95/gcc, type:

```
make arch=g95
```

To compile with gfortran/gcc, type:

```
make arch=gfortran
```

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

```
make arch=gfortan_64
```

To compile with ifort/icc Intel compilers, type:

```
make arch=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
⋮
```

5.5 Reweighting well-tempered metadynamics calculations

From a converged metadynamics run we can calculate directly the canonical probability distribution of the collective variables at a given temperature. On the contrary, the statistics of other degrees of freedom is somehow distorted by the application, during the simulation, of a time-dependent external potential on the CVs. Different possible techniques have been proposed to reconstruct the probability distribution of variables other than the CVs [54, 29, 21].

In well-tempered metadynamics, the reconstruction of the distribution of variables different from the CVs is particularly simple since for long times the amount of bias added decreases to zero and the system becomes closer and closer to equilibrium. The algorithm described in Ref. [21] consists of three different steps:

1. Accumulate the histogram of the CVs plus the other variables of interest between two updates of the bias potential;

2. When a new Gaussian is added, evolve the histogram following:

$$P(\boldsymbol{R}, t + \Delta t) = e^{-\beta(\dot{V}(\boldsymbol{S}(\boldsymbol{R}),t) - \langle \dot{V}(\boldsymbol{S},t) \rangle)\Delta t} P(\boldsymbol{R}, t), \tag{5.1}$$

   where $P(\boldsymbol{R}, t)$ is the biased probability distribution, $\dot{V}(\boldsymbol{S}(\boldsymbol{R}), t)$ the time derivative of the bias potential and the average in the exponent is calculated in the biased ensemble;

3. At the end of the simulation, the unbiased distribution $P_B(\boldsymbol{R})$ can be recovered from the histogram collected by using a standard umbrella sampling reweighting:

$$P_B(\boldsymbol{R}) \propto e^{\beta V(\boldsymbol{S}(\boldsymbol{R}),t)} \cdot P(\boldsymbol{R}, t). \tag{5.2}$$

5.5.1 Installation instructions

As `reweight.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 `reweight.f90` using the following command:

```
g95 -O3 reweight.f90 -o reweight
```

5.5.2 Usage

Example.
We performed a metadynamics run with 2 CVs and we are interested in reconstructing the distribution of a third variable.

```
reweight -colvar COLVAR -hills HILLS -ncv 2 -nvar 3 -stride 1 -fes 3 -temp 300
-welltemp

 -hills      HILLS filename
 -colvar     COLVAR filename
 -out        FES filename
 -ncv        number of variables in HILLS
 -nvar       number of variables in COLVAR
 -stride     ratio between COLVAR and HILLS stride
 -fes        ID of the variables for FES in output
 -temp       temperature in Kelvin
 -ngrid      histogram grid dimension
 -nreject    discard initial steps
 -timeout    stride for FES printout
 -pi         ID of the variables with [−π; π] periodicity
 -welltemp   control for well-tempered metadynamics
 -kjoule     energy in kjoule/mol
```

The code needs two files with the same format of the `PLUMED HILLS` and `COLVAR` files. In the latter, the metadynamics CVs should appear in the first d column followed by the variables whose distribution one wants to reweight. The ratio between the stride in `COLVAR` and `HILLS` must be constant and greater than 1. The more data you have for the histogram, the better.

Some important things to keep in mind:

- For the choice of the bin size, please follow the suggestions described in section 3.4.5;
- Eq. 5.1 is exact. However, at the beginning of the simulation the average of $\dot{V}(\boldsymbol{S}(\boldsymbol{R}), t)$ can be calculated only approximately. Luckily, a possible initial error is recovered for long times. Alternatively, one could discard the first part of the trajectory using the `-nreject` option. Please always check that your results are robust to a discard of initial parts of the trajectory;
- As for the calculation of the FES with `sum_hills`, remember to control the convergence by plotting the reconstructed distribution at different times by using `-timeout`;

5.6 Bias-exchange simulations via the linux shell

Bias-exchange simulations [18] can be performed using every MD engine, employing the tool in the directory `utilities/bias-exchange`. Each replica (walker) runs independently in a different directory for a duration τ_{exch} equal to the time between two exchanges of the bias. When all runs are finished, a script called `bias-exchange.sh` takes care of attempting exchanges of the bias and re-launching the simulation for each replica.

This procedure via the linux shell can be inefficient in terms of computer time (due to the need to stop and restart the simulations to perform exchanges) if τ_{exch} is chosen very short, e.g. < 100 timesteps. However in the literature times larger than 10 ps are typically employed for classical simulations of biomolecules, while for ab-initio simulations this should not be an issue already for $\tau_{exch} \sim 0.1$ ps. Note that for GROMACS `PLUMED` provides also bias-exchange within a single parallel run, which is computationally more efficient (see Section 3.6.2).

A simulation proceeds as follows (see also the directory `example`). First, the Fortran code `exchange-tool.f90` is compiled using the command `make` (modify the Makefile if necessary). The script `bias-exchange.sh` and the program `exchange-tool.x` must be copied in a chosen directory `BASE_DIR`, and the following variables inside the script `bias-exchange.sh` must be modified:

- `BASE_DIR`
- `NWALKER` is the number of replicas
- `KBT` is the Boltzmann factor $k_B T$ in units of the MD engine
- `NSIMULATIONS` is the maximum number of runs between exchanges for each walker

Then, inside `BASE_DIR` the user must create `NWALKER` directories called `walker1`, `walker2`, etc., and in each one the following files must be present:

- an input for restarting an MD simulation of length τ_{exch}
- a `PLUMED` input file called `plumed.dat` (see example below)

- a script `run-walker.sh` (see example below)

Finally, the bias-exchange simulation is started by launching the MD runs in each `walker1`, `walker2`, etc. directories using the command `./run-walker.sh &` (the latter script must be executable: it can be made so e.g. using command `chmod a+x run-walker.sh`). This script must launch the MD run in foreground (no `&` character at the end of the line) and it must end with the following lines:

```
touch READY
../bias-exchange.sh &
```

After a time τ_{exch} the runs finish and control passes to the `bias-exchange.sh` script (in background), which attempts to exchange the bias (i.e. files `HILLS` and `plumed.dat`, not atomic coordinates) between pairs of randomly chosen replicas. Detailed output is printed in the file `bias-exchange.log`.

It is required to employ the keyword `HILLS_LABEL` in each `plumed.dat` input file, with a different label for each walker: in this way `PLUMED` includes the information (necessary for the bias-exchange tool) about the active CVs in the `COLVAR` and `HILLS` files.

Example.
Example of `plumed.dat` input file in directory `walker1`. Note the keyword `HILLS_LABEL` and that only CV 1 is biased.

```
HILLS RESTART HEIGHT 0.001 W_STRIDE 100
PRINT W_STRIDE 10
HILLS_LABEL A

COORD LIST 1 <g234> NN 6 MM 12 R_0 5.0 SIGMA 0.1
g234->
2 3 4
g234<-

COORD LIST 2 <g134> NN 6 MM 12 R_0 5.0 SIGMA 0.1
g134->
1 3 4
g134<-

COORD LIST 3 <g124> NN 6 MM 12 R_0 5.0 SIGMA 0.1
g124->
1 2 4
g124<-

COORD LIST 4 <g123> NN 6 MM 12 R_0 5.0 SIGMA 0.1
g123->
1 2 3
g123<-

NOHILLS CV 2
NOHILLS CV 3
NOHILLS CV 4

ENDMETA
```

The other walkers will have a similar `plumed.dat` input file, but with different label (e.g. `HILLS_LABEL B` for `walker2`, etc.) and different active CV (e.g. the second one in `walker2`, etc.).

Example.
Example of `run-walker.sh` script file: the script must be executable.

```
#!/bin/bash
# note that CPMD is launched in foreground:
cpmd.x input .  -plumed >> output
# this creates and empty file called READY,
# telling the bias-exchange tool that the MD run has finished:
touch READY
# note that bias-exchange.sh is launched in background:
../bias-exchange.sh &
```

At the end of the bias-exchange simulation, the multidimensional free-energy landscape as a function of several CVs can be reconstructed according to the weighted-histogram algorithm in Ref. [29], e.g. employing the METAGUI program [30] downloadable from
`http://www.plumed-code.org/contributions`

The bias-exchange simulation can be continued, after `NSIMULATIONS` steps are reached and all replicas have finished their run, in the following way:

- enlarge `NSIMULATIONS` in the script `bias-exchange.sh`
- in each walker-directory, delete the file `READY`
- in each walker-directory, restart the run using `./run-walker.sh &`

Bibliography

[1] M. Bonomi, D. Branduardi, G. Bussi, C. Camilloni, D. Provasi, P. Raiteri, D. Donadio, F. Marinelli, F. Pietrucci, R. A. Broglia, M. Parrinello, PLUMED: a portable plugin for free energy calculations with molecular dynamics, Comp. Phys. Comm. 180 (2009) 1961.

[2] 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.

[3] 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.

[4] W. Smith, C. Yong, P. Rodger, Mol. Simulat. 28 (2002) 385–471.

[5] S. J. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comp. Phys. 117 (1995) 1–19.

[6] 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.

[7] M. Harvey, G. Giupponi, G. D. Fabritiis, ACEMD: Accelerated molecular dynamics simulations in the microseconds timescale, J. Chem. Theory and Comput. 5 (2009) 1632.

[8] 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. Sclauzero, 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.

[9] A. Laio, M. Parrinello, Escaping free energy minima, Proc. Natl. Acad. Sci. USA 99 (2002) 12562–12566.

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

[11] 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.

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

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

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

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

[16] 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.

[17] 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.

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

[19] G. A. Tribello, M. Ceriotti, M. Parrinello, A self-learning algorithm for biased molecular dynamics, Proc. Natl. Acad. Sci. USA 107 (41) (2010) 17509–17514.

[20] 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.

[21] M. Bonomi, A. Barducci, M. Parrinello, Reconstructing the equilibrium Boltzmann distribution from well-tempered metadynamics, J. Comput. Chem. 30 (11) (2009) 1615–1621.

[22] 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.

[23] A. Laio, F. L. Gervasio, Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science, Rep. Prog. Phys. 71 (2008) 126601.

[24] A. Barducci, M. Bonomi, M. Parrinello, Metadynamics, WIREs Comput. Mol. Sci. 1 (2011) 826.

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

[26] F. Baftizadeh, P. Cossio, F. Pietrucci, A. Laio, Protein folding and ligand-enzyme binding from bias-exchange metadynamics simulations, Current Physical Chemistry (in press).

[27] Y. Crespo, F. Marinelli, F. Pietrucci, A. Laio, Metadynamics convergence law in a multidimensional system, Phys. Rev. E 81 (2010) 055701(R).

[28] 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.

[29] F. Marinelli, F. Pietrucci, A. Laio, S. Piana, A kinetic model of trp-cage folding from multiple biased molecular dynamics simulations, PLoS Comput. Biol. 5(8) (2009) e100045.

[30] X. Biarnes, F. Pietrucci, F. Marinelli, A. Laio, Metagui. a vmd interface for analyzing metadynamics and molecular dynamics simulations, Comp. Phys. Comm. 183 (2011) 203.

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

[32] D. Provasi, M. Filizola, Putative active states of a prototypic g-protein-coupled receptor from biased molecular dynamics, Biophys. J. 98 (2010) 2347–2355.

[33] C. Camilloni, R. A. Broglia, G. Tiana, Hierarchy of folding and unfolding events of protein g, ci2, and acbp from explicit-solvent simulations, J. Chem. Phys. 134 (2011) 045105.

[34] J. B. Abrams, M. E. Tuckerman, Efficient and direct generation of multidimensional free energy surfaces via adiabatic dynamics without coordinate transformations, J. Phys. Chem. B 112 (2008) 15742–15757.

[35] L. Rosso, M. E. Tuckerman, An adiabatic molecular dynamics method for the calculation of free energy profiles, Mol. Sim. 28 (2002) 91–112.

[36] L. Rosso, P. Minary, Z. Zhu, M. E. Tuckerman, On the use of adiabatic molecular dynamics to calculate free energy profiles, J. Chem. Phys. 116 (2002) 4389–4402.

[37] L. Maragliano, E. Vanden-Eijnden, A temperature accelerated method for sampling free energy and determining reaction pathways in rare events simulations, Chem. Phys. Lett. 426 (2006) 168–175.

[38] Y. Liu, M. E. Tuckerman, Generalized gaussian moment thermostatting: A new continuous dynamical approach to the canonical ensemble, J. Chem. Phys. 112 (2000) 1685.

[39] J. Vymetal, J. Vondrasek, Gyration- and inertia-tensor-based collective coordinates for metadynamics. application on the conformational behavior of polyalanine peptides and trp-cage folding, J. Phys. Chem. A 115 (2011) 11455–11465.

[40] D. N. Theodorou, U. W. Suter, Shape of unperturbed linear polymers: polypropylene, Macromolecules 18 (6) (1985) 1206–1214.

[41] 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.

[42] M. Sega, E. Autieri, F. Pederiva, On the calculation of puckering free energy surfaces, J. Chem. Phys. 130 (22) (2009) 225102.

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

[44] 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.

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

[46] 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.

[47] 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.

[48] D. Donadio, P. Raiteri, M. Parrinello, Topological defects and bulk melting of hexagonal ice, J. Phys. Chem. B 109 (12) (2005) 5421–5424.

[49] M. Bonomi, M. Parrinello, Enhanced sampling in the well-tempered ensemble, Phys. Rev. Lett. 104 (2010) 190601.

[50] A. Amadei, A. B. M. Linssen, H. J. C. Berendsen, Essential dynamics of proteins., Proteins Struct. Funct. Genet. 17 (1993) 412–425.

[51] L. Sutto, M. D. Abramo, F. L. Gervasio, Comparing the efficiency of biased and unbiased molecular dynamics in reconstructing the free energy landscape of met-enkephalin, J. Chem. Theory Comput. 6 (12) (2010) 3640–3646.

[52] F. Pietrucci, W. Andreoni, Graph theory meets ab initio molecular dynamics: atomic structures and transformations at the nanoscale, Phys. Rev. Lett. 107 (2011) 085504.

[53] G. A. Tribello, J. Cuny, H. Eshet, M. Parrinello, Exploring the free energy surfaces of clusters using reconnaissance metadynamics, J. Chem. Phys 135 (11) (2011) 114109.

[54] G. Tiana, Estimation of microscopic averages from metadynamics, Eur. Phys. J. B 63 (2) (2008) 235–238.

Index