Software tools for free energy analysis based on the force-correction analysis method

Dec 16, 2021

Content

Introduction

The force correction analysis method (FCAM)[1] is designed to derive unbiased mean forces and free energies in collective variables (CVs) space from biased simulations. Building upon the umbrella integration approach[2] and variants thereof [3-6], the mean forces are calculated by subtracting the biasing forces to the total forces, each derived from a local average around a particular value, $\boldsymbol{\xi}$, of the CVs. The total forces are estimated using a density kernel which in this implementation is the product of Gaussian functions (one for each CV), leading to the following expression of the mean force component along CV ξ_i:

$$f_{\xi_i}(\boldsymbol{\xi}) \cong \frac{\sum_r \sum_t w_t^r(\boldsymbol{\xi}) \left[k_{\xi_i}\left(\xi_i^f(\boldsymbol{X}_t^r) - \xi_i\right) + \left(\frac{\partial V_r(\boldsymbol{\xi}', t)}{\partial \xi'_i}\right)_{\xi'=\xi^f(X_t^r)} \right]}{\sum_r \sum_t w_t^r(\boldsymbol{\xi})} \qquad (1),$$

where $w_t^r(\boldsymbol{\xi}) = exp\left\{-\beta \sum_{i=1}^N k_{\xi_i}\left(\xi_i^f(\boldsymbol{X}_t^r) - \xi_i\right)^2/2\right\}$ and $k_{\xi_i} = 1/\beta\sigma_{\xi_i}^2$ ($\beta = 1/k_B T$, where k_B is the Boltzmann constant and T the temperature), in which σ_{ξ_i} is the Gaussian standard deviation along ξ_i. The value of σ_{ξ_i} defines the resolution of mean forces and free energy surfaces, i.e. larger values increase the smoothing effect. The mean forces are calculated on a set of CVs configurations enumerated according to the index α, $(\xi_1^\alpha, \xi_2^\alpha, \ldots, \xi_N^\alpha)$, which can be placed for example on a regular grid. The latter forces can be then integrated to get the free energy on the same CVs configurations using e.g. a numerical integrator (for one dimension can be based on finite differences) or a Kinetic Monte Carlo (KMC) approach [1] (more suitable than the former in higher dimensionality).
The main tools for free energy analysis documented here include two python programs:

- ***calcf_vgauss.py*** allows to calculate mean forces based on FCAM
- ***graf_fes_kmc.py*** allows to derive the free energy landscape and minimum free energy paths from the mean forces based on a KMC approach

The previously mentioned programs allow:

- When biasing forces are available through external files, calculating mean forces and free energy for any CVs based enhanced sampling method (assuming is performed under equilibrium condition).
- When biasing forces are not available externally, calculating mean forces and free energy for metadynamics[7] (standard and well-tempered[8] based on "hills" files) and umbrella sampling simulations (based on harmonic potentials).
- Combine mean force estimates from different simulation trajectories and biasing schemes to

obtain the best estimate of the mean force (and associated free energy).

The output files provided by each program have default names that can be customized by adding a specific option in this regard. The complete list of options available is printed out by running the programs without any specification/option:

```
python3.8  calcf_vgauss.py
```

```
python3.8  graf_fes_kmc.py
```

Usability and Citation

These programs can downloaded and used freely from https://github.com/Faraldo-Gomez-Lab-at-NIH/Download, if you use them please cite ref. [1]:

Marinelli, F. and J.D. Faraldo-Gomez, *Force-Correction Analysis Method for Derivation of Multidimensional Free-Energy Landscapes from Adaptively Biased Replica Simulations*. J Chem Theory Comput, 2021. **17**(11): p. 6775-6788

Notes on python versions

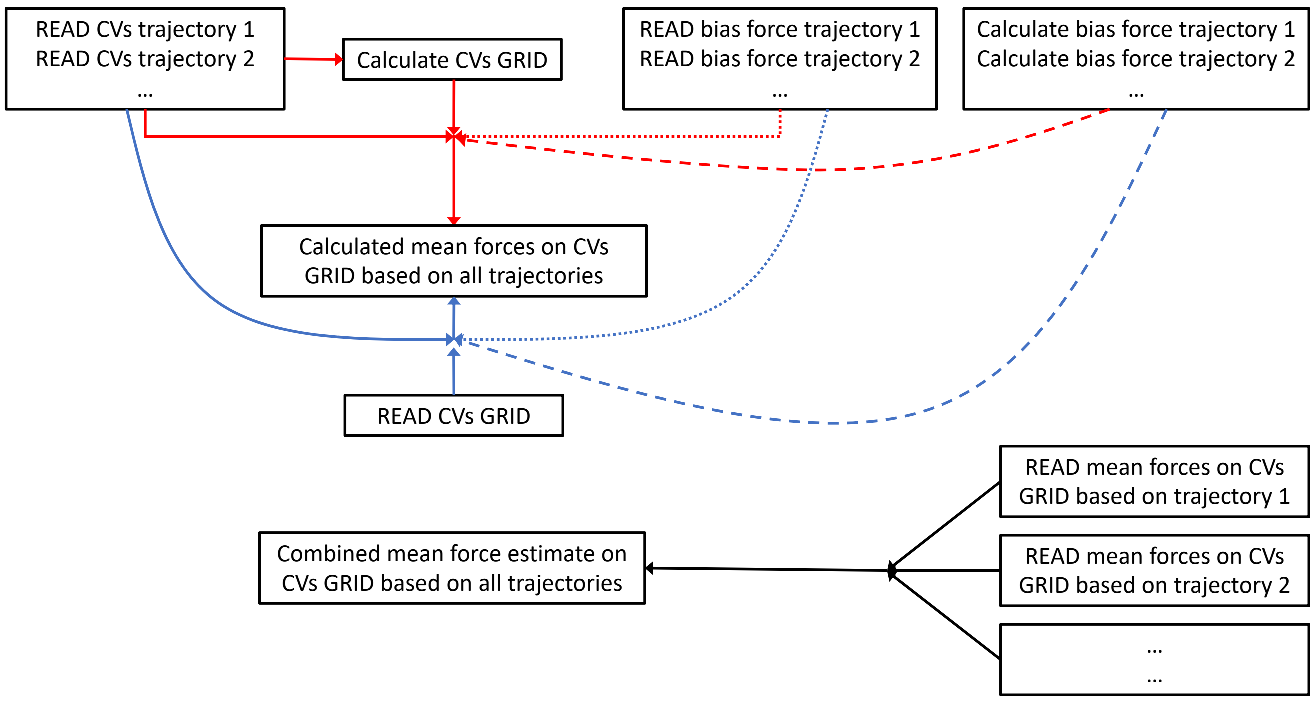
The previously mentioned programs have been tested using **python version 3.8** within **anaconda** 2020.02. Other python versions might work as well, e.g. a few tests show that the native python3.6 of CentOS 7 provides identical results. Note that the default version of ***graf_fes_kmc.py*** uses Numba (also available through anaconda) to speed up KMC calculations. A version without Numba is provided in the **tools** folder (*graf_fes_kmc_no-numba.py*), however is not recommended as KMC calculations are ~2 orders of magnitude slower in this case.
Other external programs can be also used for free energy integration from the mean forces, nonetheless they might require format conversion of the output file provided by ***calcf_vgauss.py*** containing the free energy gradient (negative of the mean force) calculated on CVs GRID points (see below).

Mean forces calculation based on FCAM

The main input information required in the program ***calcf_vgauss.py*** is (see Scheme 1 and eq. 1):

- the values of the CVs for each simulation frame (CVs trajectory)
- the value of the biasing force components, $(\partial V_r(\boldsymbol{\xi}',t)/\partial \xi'_i)_{\xi'=\xi^f(X_t^r)}$, for each simulation frame (bias force trajectory)
- a set of CVs configurations on which biasing forces are calculated (CVs GRID)

The biasing force trajectory can be either read from an external file (Scheme 1) or can be calculated by providing the necessary input information. The post-hoc derivation of the biasing forces trajectory is available for standard and well-tempered (WT) metadynamics as well as for umbrella sampling (US) simulations based on harmonic biasing potentials.
The main output of the program is the value of the free energy gradient (negative of the mean force) for each GRID point (Scheme 1). The CVs GRID points can be either inferred from the CVs trajectories and GRID definition (see below) or can be read from an external file. Note that for computational efficiency and for preserving memory only the GRID points that are explored during the simulations are included (and should be included if read externally, especially for high dimensional analysis).


READ CVs trajectory 1
READ CVs trajectory 2
…
Calculate CVs GRID
READ bias force trajectory 1
READ bias force trajectory 2
…
Calculate bias force trajectory 1
Calculate bias force trajectory 2
…
Calculated mean forces on CVs GRID based on all trajectories
READ CVs GRID
READ mean forces on CVs GRID based on trajectory 1
READ mean forces on CVs GRID based on trajectory 2
…
…
Combined mean force estimate on CVs GRID based on all trajectories

Scheme 1. Flow chart of the mean force calculation program ***calcf_vgauss.py***. Dotted and dashed lines as well as blue and red colors denote two alternative possibilities.

The previously mentioned information is provided through a general input file (e.g. *input.dat*) which is read when running the program according to the following syntax:

```
python3.8  calcf_vgauss.py  -if input.dat   (other specifications/options)
```

Format of the input file

The input file entails a set of keywords (highlighted in bold) by which the required information is provided:

Input file general format

COLVAR_FILE *colvar.traj (**HILLS_FILE** hills.traj **HILLS_CVS-CLS** [CVs list] **BF** [WT bias factor]) (**US_CVS-CLS** [CVs list] **US_C** [center] **US_K** [force constant])*
READ_BIAS_FORCE_TRJ *bias_force.traj*
READ_BIAS_GRAD_TRJ *bias_grad.traj*
CV-CL *[CV column on colvar.traj (starting from 0)] [CV lower bound] [CV upper bound] [CV number of GRID points] (PERIODIC)*
READ_EPOINTS *eff_points.out*
READ_GRAD_PMF *grad_on_eff_points.out*

Note that blank lines in the input file produce an error message and must be avoided. Hereafter we describe the quantities specified in the input file as well as associated specifications/options to be used in each case. The terms ending with *.traj* or *.out* in the example above are files provided by the user with customizable name and extension.

Reading CVs trajectories

The keyword ***COLVAR_FILE*** in the input file is used to read the multi columns file (named *colvar.traj* in the example above; a custom name can be provided by the user) containing CVs trajectory and optionally biasing force trajectory (see below). Blank spaces and comments (“#”) are automatically neglected. For two CVs (CV_1, and CV_2) for example the format of this file can be the following:

***Example of CVs trajectory* file:** dashed lines denote columns with quantities that are not relevant for the analysis

```
# step (or time)     CV_1           -----------------     biasing_force_1       CV_2           -----------------     biasing_force_2
      0          2.931417541e+01   ---------------------  0.000000000e+00  -1.797015831e+02   ---------------------  0.000000000e+00
     500         1.153575991e+02   ---------------------  0.000000000e+00  -1.239904418e+02   ---------------------  0.000000000e+00
      …             ……….. .          ………..                ………..            ………..               ………..                ………..
      …             ……….. .          ………..                ………..            ………..               ………..                ………..
29999000         9.778031483e+01   ---------------------  0.000000000e+00   2.547875483e+01   ---------------------  -1.331780967e-01
29999500         9.806234838e+01   ---------------------  0.000000000e+00  -4.698565314e+00   ---------------------  2.079940692e-02
```

Multiple trajectories can be analyzed simultaneously by specifying multiple ***COLVAR_FILE*** keywords on different lines:

Example of simultaneous analysis of multiple trajectories

```
COLVAR_FILE colvar.traj.1 (…)
COLVAR_FILE colvar.traj.2 (…)
…
```

The files *colvar.traj.1* and *colvar.traj.2***,** can have different number of lines (i.e. different simulation lengths) but must have the same format; i.e. CVs and eventually biasing forces written on the same columns.

CVs and GRID specification

The quantities after the keyword ***CV-CL*** in the input file specify:

- the column of the CV to be analyzed in the CVs trajectory file, starting from 0
- the GRID specification; lower boundary, upper boundary and number of GRID points for that CV
- whether the CV considered is **PERIODIC**. When the latter keyword is not specified the variable is considered not periodic. When it is specified, periodic conditions are applied at the GRID boundaries.

Multidimensional analysis can be carried out by specifying multiple ***CV-CL*** keywords on different lines. For the correct application of FCAM at least all CVs on which biasing forces are applied must be included in the analysis. An example in which both CV_1 (unbiased) and CV_2 (biased) in the CVs trajectory file reported above are considered for the analysis is the following:

Example of CVs and GRID specification based on the CVs trajectory example file reported previously

```
…
CV-CL 1 -180  180 144 PERIODIC
CV-CL 4 -180  180 144 PERIODIC
…
```

In the case above, the program will consider two **PERIODIC** CVs, the ones in column 1 and column 4 in the CVs trajectory file and will assume a GRID of 144X144 with boundaries -180 and 180. The GRID is generated by default with the specified parameters after reading the CVs trajectory file, GRID points that are not explored in the latter file are excluded. The GRID points are written in an output data file (named by default eff_points.out) having the following format:

Example of GRID points file

```
# GRID index        CV_1             CV_2    number of frames in the bin
      0            -178.75          -178.75          4
      1            -176.25          -178.75          6
      2            -173.75          -178.75          5
      ..            ……               …….             ..
      ..            ……               …….             ..
      ..            ……               …….             ..
    15260           173.75           178.75          7
    15261           176.25           178.75          6
    15262           178.75           178.75          7
```

The GRID points file, instead of being calculated from trajectory files, can also be read as input through the keyword ***READ_EPOINTS*** (see Scheme 1 and input file format). This can be useful to calculate the mean forces on the same GRID points from different trajectories and with different biasing force schemes using separate runs and then combine the estimates, which also provides a trivial parallelization scheme (see below for examples). The program can also read multiple GRID points files by specifying multiple ***READ_EPOINTS*** keywords on different lines. In that case ***calcf_vgauss.py*** calculates the mean forces on a single CVs GRID given by the sum of all the different GRID points files. The column with the number of frames per bin is required to correctly read the GRID points file. If you don't have the values for that column, it is sufficient to create it with arbitrary positive integer numbers (e.g. by setting all values to one).

Specific options for mean forces calculation

The options required when mean forces calculation is requested are (see eq. 1):

- ***-units*** *{kj or kcal}*
- ***-temp*** *[simulation temperature]*

Hence the typical running command is:

```
python3.8 calcf_vgauss.py -if input.dat -units {kj or kcal} -temp [sim temp] (other specifications/options)
```

The option ***-units*** specifies either kJ/(mol x CV) or kcal/(mol x CV) force units (and hence either kJ/mol or kcal/mol free energy units) which depend on the simulation program: kcal for NAMD[9, 10] and kj for GROMACS[11]. For other types of units, in alternative to ***-units*** the value of the Boltzmann factor (i.e. the ideal gas constant R) can be provided through the option ***-kb*** (with congruent units for *{sim temp}*).

Reading biasing forces externally

Specialized software's for enhanced sampling simulations based on collective variables, such as Colvars[12] and PLUMED[13-15], in many cases can provide directly in output the biasing forces applied to the CVs as a function of the simulation time. This can be done for example in Colvars[12] by enabling *outputAppliedForce* (https://colvars.github.io) and in PLUMED2[14] by using the keyword DUMPFORCES (https://www.plumed.org/doc). As described in the following sections, these biasing forces trajectories can be read directly by ***calcf_vgauss.py*** for derivation of mean forces based on FCAM. Importantly, this allows to correctly evaluate mean forces for any enhanced sampling method based on biasing forces applied to CVs as well as it allows to combine mean force estimates from different biasing schemes, provided that bias time-fluctuations are small enough so that trajectories can be considered under equilibrium condition[1].

Reading biasing forces from the CVs trajectory file

In the collective variables module (Colvars)[12] CVs and biasing forces (-$(\partial V_r(\boldsymbol{\xi}',t)/\partial \xi'_i)_{\xi'=\xi^f(\boldsymbol{X}_t^r)}$) can be written on the same CVs trajectory file (by enabling *outputAppliedForce*). In this case, the biasing forces can be read directly from that file using the options:

- ***-colvars* -colvarbias_column** *[bias_force column - CV column]*
  The term *[bias_force column - CV column]* specifies by which column from the associated CV the biasing forces are located. As this feature is specific for Colvars[12], the option ***-colvars*** must also be used, which assumes that files and parameters setting are those of Colvars.

For example the following command line instructs the program to calculate mean forces by reading the biasing forces at the second column from the associated CV, as in the CVs trajectory file example above:

```
python3.8 calcf_vgauss.py -if input.dat -units {kj or kcal} -colvars -colvarbias_column 2 -temp [sim temp]
```

Reading biasing forces from an external file

The biasing force as a function of the simulation time, $-(\partial V_r(\xi', t)/\partial \xi'_i)_{\xi'=\xi^f(X_t^r)}$, can be also provided through an external file using the keyword **READ_BIAS_FORCE_TRJ** *bias_force.traj* or ***READ_BIAS_GRAD_TRJ*** *bias_grad.traj.* In the latter case the bias gradient instead of the biasing force is considered (which have inverted sign respect to each other). For two CVs the format of those files is the following:

Example of biasing force trajectory file: dashed lines denote irrelevant columns

```
#  step (or time)    bias_force_or_grad_1    bias_force_or_grad_2              ---------
0.000000              0.0000000000              0.0000000000          --------------------
1.000000              0.0000000000              0.0000000000          --------------------
 …                     …                         …                    …
 …                     …                         …                    …
28749.00              50.4807821998            14.8035521414          --------------------
28750.00              48.1894266628            14.7787577675          --------------------
```

The biasing force components on each column of the previously mentioned files are those of the CVs specified in the input file through ***CV-CL*** keywords and with the same order; lines from top to bottom on the input file correspond to columns from left to right on biasing force (or bias gradient) trajectory file.
Multiple trajectories can be analyzed by specifying one **READ_BIAS_FORCE_TRJ** or ***READ_BIAS_GRAD_TRJ*** keyword for each trajectory, each placed on a different line and with the same order of the corresponding ***COLVAR_FILE*** keywords:

Example of multiple trajectories analysis by reading external biasing force trajectory files

```
COLVAR_FILE colvar.traj.1 (…)
COLVAR_FILE colvar.traj.2 (…)
…
READ_BIAS_FORCE_TRJ bias_force.traj_1
READ_BIAS_FORCE_TRJ bias_force.traj_2
…
```

Note that *colvar.traj.i* and *bias_force.traj_i* , where *i* denotes the trajectory index, must contain the same number of lines (except for comments and blank lines). Alternatively, a single keyword can be specified associated to one file, for example *bias_force.all_trajs* or *bias_grad.all_trajs*, containing the biasing forces (or bias gradient) for all trajectories appended together with the same order of the ***COLVAR_FILE*** lines (from top to bottom) on the input file:

Example of multiple trajectories analysis by reading a single biasing force trajectory file generated by appending data for each trajectory

```
COLVAR_FILE colvar.traj.1 (…)
COLVAR_FILE colvar.traj.2 (…)
…
READ_BIAS_FORCE_TRJ bias_force.all_trajs
…
```

Post-hoc derivation of the biasing forces for specific simulation types

In case biasing force trajectories are not available from external files, the program ***calcf_vgauss.py*** can calculate those internally for metadynamics simulations and umbrella sampling simulations based on harmonic potentials. The post-hoc calculation of the biasing forces requires accessory files reporting information on the bias potentials added as a function of the simulation time (for metadynamics) or the specification of parameters describing the shape of the bias potential (harmonic force constant and center for umbrella sampling).
Note that, unless biasing forces are read externally, ***calcf_vgauss.py*** does not allow mixing metadynamics and umbrella sampling analysis. Additionally, it does not allow calculating the biasing forces on one trajectory while reading the biasing forces for another trajectory.
The simultaneous analysis of different biasing schemes can be nevertheless achieved by using ***calcf_vgauss.py*** on each individual simulation type and then combining the obtained mean force

estimates.

Analysis of metadynamics simulations by derivation of the biasing force from the “hills” file

In case the biasing force trajectory is not available as input, ***calcf_vgauss.py*** allows to derive it for metadynamics simulations based on the file containing the Gaussians that have been added along the trajectory (“hills” file) with the typical format of PLUMED[13-15] and Colvars[12] software’s:

“hills” file example for 2D metadynamics on CV_1 and CV_2

```
#        step (or time)            CV_1                  CV_2                    sigma_CV_1    sigma_CV_2   Gaussian_height
    2.000000000000000    2.824348584082339   -0.4590819040057097              0.17           0.24            0.2
    4.000000000000000    2.851430730900406   -0.7171619217815017              0.17           0.24            0.2
    …                    …                    …                              …              …               …
    …                    …                    …                              …              …               …
    210058.0000000000    2.775681184477477    2.905182588387364               0.17           0.24            0.2
    210060.0000000000    2.443438281643501    2.807473333317552               0.17           0.24            0.2
```

The “hills” file must be specified in the input file (named *hills.traj* in the input example) after the keyword ***HILLS_FILE***, on the same line of the associated ***COLVAR_FILE*** *{CVs trajectory}*. Additionally, the user must also specify which are the columns of the CVs in CVs trajectory that are biased by the metadynamics, using the keyword ***HILLS_CVS-CLS***. In the CVs trajectory and “hills” file (named e.g. *hills.traj*) examples reported above this would be:

Input file example for 2D metadynamics requesting biasing force calculation based on the “hills” file

```
COLVAR_FILE colvar.traj HILLS_FILE hills.traj HILLS_CVS-CLS 1 4
…
```

Note that Colvars[12] and PLUMED[13-15] use different precision for Gaussians truncation. By default ***calcf_vgauss.py*** assumes the truncation value used in PLUMED (argument of the exponential<6.25), the parameters used in Colvars can be activated using the option ***-colvars***.
For well-tempered metadynamics simulations done with PLUMED[13-15] in which the Gaussian heights written in output are scaled according to the bias factor (BF; see https://www.plumed.org/doc), the latter must be also specified on the same line after the keyword ***BF***, e.g. if BF=15:

Input file example for 2D WT-metadynamics requesting biasing force calculation from “hills” file and Gaussians height is scaled according to BF

```
COLVAR_FILE colvar.traj HILLS_FILE hills.traj HILLS_CVS-CLS 1 4 BF 15.0
…
```

BF must not be used when using Colvars[12] as Gaussian heights in output are not scaled by the BF.

Note that *colvar.traj* and *hills.traj* can be written at different pace (typically *hills.traj* is written less frequently, if the opposite is true please add the option **-hlfl**), nonetheless the time column (column 0) must have the same units and total time lengths must be congruent in the two files. In most cases ***calcf_vgauss.py*** is able handle *colvar.traj* and *hills.traj* files with restarted trajectories appended to those files in which the simulation time is either put back to zero upon restarting, restarted to a previous time (especially in PLUMED[13-15] when simulations ends before requested steps) or to the same time it ends (repeated time values). In PLUMED (especially if compiled in single precision), the same time point may have slightly different values in the last digits for *colvar.traj* and *hills.traj* files, in that case is required to read the time column with reduced precision so that times match in both files and biasing forces are calculated correctly. The reading precision can be set by the option:

- **-colv_time_prec** *[precision value]*

By default, the program writes on output the calculated bias potential gradient in a file (by default named bias_grad.out) with the multicolumn format reported above:

Example of bias potential gradient file printed in output

```
#  step (or time)      bias_grad_1        bias_grad_2      Gaussian_energy    Gaussians_number   next_restart   previous_restart   replica_index
0.0                     0.0                0.0              0.0                0                  15000          0                  0
1.0                     0.0                0.0              0.0                0                  15000          0                  0
2.0                    -2.87832e-06        3.333135e-07     0.1999999          1                  15000          0                  0
…                       …                  …                …                  …                   …             …                   …
…                       …                  …                …                  …                   …             …                   …
29999.0                41.29764          -14.67957         20.37612           14999               15000          0                  0
30000.0               -17.91238           -9.14609         23.92791           15000               15000          0                  0
# trajectory restart: hills file appended
0.0                    44.44520           -6.81932         42.02874           15000               120030         15000              0
1.0                    65.51570            8.68860         27.19215           15000               120030         15000              0
…                       …                  …                …                  …                   …             …                   …
…                       …                  …                …                  …                   …             …                   …
210059.0                9.56577            1.04682        194.23776           120029              120030         15000              0
210060.0               41.09773          -10.72097        185.46729           120030              120030         15000              0
```

This file can be read in input on a different run (i.e. without the need of recomputing the bias gradient) using the keyword ***READ_BIAS_GRAD_TRJ***. The additional columns after the bias gradient components are not read by the program but report useful information to assess whether the bias gradient is calculated correctly. In particular, the *Gaussian_energy* column reports the value of the bias potential at that time point. Usually, the latter value is also reported on the last column of the COLVAR file in PLUMED[13-15] and should approximately match the one reported in the output bias gradient file. The following column reports the Gaussians that have been added up to that time point and must correspond to the ones of the *hills.traj* file read in input. The other columns are important to assess whether the trajectory restart has been handled correctly by the program. Specifically, the column named *next_restart* reports the Gaussian number of the next trajectory restart that has been detected from the time column (e.g. time going backward on the following line). The column *previous_restart* reports the Gaussians number of the previous trajectory restart.
The biasing forces for multiple trajectories can be derived as in previous cases by specifying multiple ***COLVAR_FILE*** and ***HILLS_FILE*** keywords on different lines which may entail different biased CVs:

Example of multiple trajectories analysis by calculating the bias gradient from "hills" files

```
COLVAR_FILE colvar.traj_1 HILLS_FILE hills.traj_1 HILLS_CVS-CLS {biased CVs for traj 1} (BF {BF for traj 1})
COLVAR_FILE colvar.traj_2 HILLS_FILE hills.traj_2 HILLS_CVS-CLS {biased CVs for traj 2} (BF {BF for traj 2})
…
```

As previously, the files *colvar.traj_1* and *colvar.traj_2* must report all the CVs with the same format, regardless that they are differently biased in each individual trajectory, e.g. the bias is applied on different subsets of the CVs for each trajectory.
The output bias gradient file is in this case a unique file in which the values for each trajectory are appended with the same order (from top to bottom) of the different ***COLVAR_FILE*** lines and can be again read in input on a different run using the keyword ***READ_BIAS_GRAD_TRJ*** as reported previously.
Note that often metadynamics is performed by updating the bias potential and associated gradients on a GRID. In this case calculating the bias gradient from the "hills" file as implemented in ***calcf_vgauss.py***, entails a discretization error which depends on the GRID spacing. Such discretization error is not present when biasing forces are printed directly by PLUMED[13-15] and Colvars[12] and read by ***calcf_vgauss.py***.
Useful options: use ***--justcalcmetabias*** to only calculate/print the metadynamics bias gradients and skip the other analysis steps.

Analysis of umbrella sampling simulations based on harmonic potentials

Mean forces based on FCAM can be derived also for umbrella sampling simulations comprising multiple windows in which harmonic biasing potentials are applied on selected CVs. Aside from reading the biasing forces externally, these types of simulations can be analyzed using ***calcf_vgauss.py*** by providing the list of biased CVs ($\xi_i^f(\boldsymbol{X_i})$, where $\boldsymbol{X_i}$ denotes the atomic configurations of window i), force constants (k_{ξ_i}), and CVs centers (ξ_i) of each window (i). Where the harmonic bias potential is assumed to be of type:

$$V_i = \frac{1}{2}\sum_i k_{\xi_i}\left(\xi_i^f(\boldsymbol{X_i}) - \xi_i\right)^2 \tag{2}$$

The latter information can be provided through the keywords ***US_CVS-CLS***, ***US_C*** and ***US_K*** specified for each window on the same line of the associated ***COLVAR_FILE*** keyword:

Example of input file for analysis of umbrella sampling windows by specifying biased CVs, harmonic centers and force constants

```
COLVAR_FILE colvar.window_1 US_CVS-CLS {biased CVs window 1}  US_C {CVs harmonic centers window 1}  US_K {CVs force constants window 1}
COLVAR_FILE colvar.window_2 US_CVS-CLS {biased CVs window 2}  US_C {CVs harmonic centers window 2}  US_K {CVs force constants window 2}
…
…
COLVAR_FILE colvar.window_N US_CVS-CLS {biased CVs window N}  US_C {CVs harmonic centers window N}  US_K {CVs force constants window N}
…
```

As for ***HILLS_CVS-CLS*** (see previous section) the keyword ***US_CVS-CLS*** specifies the columns of the CVs (starting from 0) that are biased by the harmonic potential in the *colvar.window_i* file. ***US_C*** and ***US_K*** provide the associated list (using same order from left to right) of centers and force constants for each CV.
By default, the program writes in output the bias gradient trajectory of all windows. As this might be a file with large number of lines it might be convenient to avoid writing in output the latter file using the option:

- ***-nopgradb***

Hence the typical running command for this type of analyses is the following:

```
python3.8 calcf_vgauss.py -if input.dat -units {kj or kcal} -temp [sim temp] -nopgradb
```

Output file of the free energy gradient for each GRID point

When mean forces calculation is requested, the ***calcf_vgauss.py*** program provides in output a file (named by default grad_on_eff_points.out) with the components of the free energy gradient (negative of the mean force) for each GRID point, which for two CVs has the following format:

Example of free energy gradient file for two CVs

```
# 2
# -180.0  2.5   144   1
# -180.0  2.5   144   1
-178.75      -178.75      0.03258107736330767    0.017226815057404192    36.961848888782384
-176.25      -178.75     -0.00422040496687694    0.003915166975661803    35.090191567022146
   …             …                 …                       …                        …
   …             …                 …                       …                        …
 176.25       178.75     -0.00994713230587822   -0.022722522986950495    43.82485172702444
 178.75       178.75      0.01969945193447550   -0.021510396309917593    42.54565870056456
  CV_1          CV_2     free_energy_gradient_1  free_energy_gradient_2       weight
```

The header of the file reports GRID information which is read by the program ***graf_fes_kmc.py*** for derivation of free energy integration and minimum free energy paths. The lines of the header report from top to bottom:

- number of CVs
- lower boundary, bin size, bins number and periodicity (1 is periodic, 0 is not periodic) for CV_1
- same information as above for CV_2
- same information as above for the other CVs

Such header has the same format of the one of GRID files in Colvars[12] (https://colvars.github.io), however at difference from the latter files the free energy gradient file of ***calcf_vgauss.py*** do not include blank spaces and GRID points that are not explored during the simulation (so as to facilitate multidimensional analysis), hence they must be converted accordingly in order to be read by Colvars analysis tools (e.g. alternative free energy integration programs [16]).

The last column of the file reports the total weight (or number of effective frames) of each grid point ($\sum_r \sum_t w_t^r(\boldsymbol{\xi})$, see eq. 1) which is also read by the program ***graf_fes_kmc.py*** to estimate the free energy difference between neighboring GRID points (eq. 12 in ref. [1]).

Evaluation of convergence and error through blocks analysis

Besides the main free energy gradient file, ***calcf_vgauss.py*** provides in output two additional free energy gradient files (named by default grad_on_eff_points1.out and grad_on_eff_points2.out) in which the same analysis is carried out by using first half and second half of the effective frames ($\sum_r \sum_t w_t^r(\boldsymbol{\xi})$, see eq. 1) for each GRID point respectively. The deviation between the latter gradient files provides an approximate estimate of the statistical error in the mean forces. An analogous error estimate for the free energy can be obtained by free energy integration (e.g. through the program ***graf_fes_kmc.py***) with those two free energy gradient files.
A more rigorous error analysis can be obtained by blocks analysis with different blocks sizes which can be performed using ***calcf_vgauss.py*** with the following options:

- ***-trfr1*** $frame_{frac}^{1}$
  The starting frame of each trajectory, i, that is considered for the analysis is $frame_{frac}^{1} \times N_{frames}^{i}$ where $frame_{frac}^{1}$ is the chosen value ranging from 0 to 1 and N_{frames}^{i} is the number of frames of trajectory *i*.

- ***-trfr2*** $frame_{frac}^{2}$
  The last frame of trajectory, i, that is considered for the analysis is $frame_{frac}^{2} \times N_{frames}^{i}$ where $frame_{frac}^{2}$ is the chosen value ranging from 0 to 1.

The latter options allow to analyze the desired portion of the trajectories included in the input file. Mean forces and free energy estimates (by integration of the formers) for different portions can be used to assess the convergence and perform blocks error analysis.

Combining free energy gradients estimates from individual sets of trajectories

The program ***calcf_vgauss.py*** allows combining the free energy gradients obtained by analysis of individual trajectories sets (each set analyzed independently) to obtain the best overall estimate among those. The requirements for this analysis are:

- Each estimate of free energy gradients must be carried out on the same GRID points.
- Each estimate of free energy gradients must entail an independent set of trajectories which do not overlap with those of the other sets.

This type of analysis allows:

- Trivial parallelization for multiple large trajectories, i.e. by running multiple independent analysis on independent subsets of the trajectories and then combining the obtained estimates.
- Combine estimates from mixed types of trajectories; e.g. metadynamics trajectories with biasing forces calculated with "hills" files with other types of trajectories/biasing schemes (e.g. trajectories where biasing forces are available from an external file or umbrella sampling simulations).

The free energy gradients for each independent run can be read using the keyword ***READ_GRAD_PMF*** *{free energy gradient file}* (see also input file general format).
To show how to do this in practice let us assume for example to have a *N* simulations. The analysis is carried out on *M* CVs whose trajectories are reported in different columns of files *colvar.traj_1*, *colvar.traj_2*,…, *colvar.traj_N*. The analysis can be performed using three steps:

- Calculate the GRID points from all trajectories. The prototypical input file (*input_grid.dat*) is the following:

```
COLVAR_FILE colvar.traj.1
COLVAR_FILE colvar.traj.2
…
COLVAR_FILE colvar.traj.N
CV-CL [CV_1 column] [CV_1 lower bound] [CV_1 upper bound] [CV_1 number of GRID points] (PERIODIC)
CV-CL [CV_2 column] [CV_2 lower bound] [CV_2 upper bound] [CV_2 number of GRID points] (PERIODIC)
…
CV-CL [CV_M column] [CV_M lower bound] [CV_M upper bound] [CV_M number of GRID points] (PERIODIC)
```

The running command is:

```
python3.8 calcf_vgauss.py -if input_grid.dat --justcalceffpoints
```

The option ***--justcalceffpoints*** instruct the program to stop after the GRID points are calculated and written (on a file named eff_points.out).

- Derive free energy gradients on the same GRID points from each individual trajectory using *N* independent runs. For trajectory *i* the input file (*input_traj_i.dat*) is the following:

```
COLVAR_FILE colvar.traj.i (specific keywords)
CV-CL [CV_1 column] [CV_1 lower bound] [CV_1 upper bound] [CV_1 number of GRID points] (PERIODIC)
CV-CL [CV_2 column] [CV_2 lower bound] [CV_2 upper bound] [CV_2 number of GRID points] (PERIODIC)
…
CV-CL [CV_M column] [CV_M lower bound] [CV_M upper bound] [CV_M number of GRID points] (PERIODIC)
(specific keywords)
READ_EPOINTS eff_points.out
```

The running command for trajectory *i* is:

```
python3.8 calcf_vgauss.py -if input_traj_i.dat -units {kj or kcal} -temp [sim temp] ( -colvars -colvarbias_column [bias_force - CV columns] ) -oeff grad_traj_i.out
```

At this stage *specific keywords* or options (***-colvarbias_column***) must be introduced to read or calculate the biasing forces of trajectory *i* (see sections above).

The option ***-oeff*** allows to specify the name of the free energy gradients file (i.e. to avoid overwriting in case all calculations are carried out in the same folder).

- Combine all free energy gradients estimates using the keyword ***READ_GRAD_PMF*** in the input file (input_comb.dat):

```
READ_GRAD_PMF grad_traj_1.out
READ_GRAD_PMF grad_traj_2.out
…
READ_GRAD_PMF grad_traj_N.out
CV-CL [CV_1 column] [CV_1 lower bound] [CV_1 upper bound] [CV_1 number of GRID points] (PERIODIC)
CV-CL [CV_2 column] [CV_2 lower bound] [CV_2 upper bound] [CV_2 number of GRID points] (PERIODIC)
…
CV-CL [CV_M column] [CV_M lower bound] [CV_M upper bound] [CV_M number of GRID points] (PERIODIC)
```

The running command is:

```
python3.8 calcf_vgauss.py -if input_comb.dat
```

The program produces in output the best estimate of the free energy gradient (on a file named by default grad_on_eff_points_comb.out), which is the weighted average of the individual estimates (as in eq. 1). The free energy gradient file entails in this case the weight for each component in the last columns (on the right), which is the same value for each component unless specific keywords are used (e.g. *REMOVE_COMP [list of components to discard (from 1 to number of CVs) from this estimate]* on the same line of ***READ_GRAD_PMF***):

Example of free energy gradient file for two CVs obtained by combining multiple estimates

```
# 2
# -3.14159265359   0.04363323129986111   144   1
# -3.14159265359   0.04363323129986111   144   1
-3.11977603794006     -3.119776037940069     -23.77963349913996     2.68601825130161     66.2871075242631     66.2871075242631
-3.07614280664020     -3.119776037940069     -31.16108169483828     3.24285225519108     66.7232251239851     66.7232251239851
…                     …                      …                      …                    …                    …
…                     …                      …                      …                    …                    …
3.076142806640208     3.119776037940069      -35.36356845050005     3.78058861312679     71.90796077157182    71.90796077157182
3.119776037940069     3.119776037940069      -28.54469126718131     2.02618726604002     74.23821116568745    74.23821116568745
      CV_1                  CV_2           free_energy_gradient_1 free_energy_gradient_2  weight_comp_1        weight_comp_2
```

Filtering options

In programs such as PLUMED[13-15] and Colvars[12] restarting is handled by appending the output to the same files, e.g. the same *colvar.traj* file. In some case (especially in PLUMED) such appended output may contain repeated frames (e.g. when restarting from a previous time point). To avoid frames double counting the following option can be used:

- ***-nfr*** *{frames to filter out before and after restart}*
  The specified number of frames around the restart time are discarded. To eliminate only the portion of those frames before the restart the following option can be added:
    - ***-backres***

Note that by default ***calcf_vgauss.py*** discards the sampling beyond the boundaries defined according to the ***CV-CL*** keywords. The latter sampling can be re-included using the option

- ***-nobound***

Trajectory labeling

The program ***calcf_vgauss.py*** allows to evaluate and write in output (on a file named by default label.out) the GRID point assigned to each configuration of the CVs trajectories. This functionality is enabled by the option:

- **-label**

Each GRID point is identified with the GRID index of the GRID points file (line number from top to bottom and starting from 0). When this option is enabled the program writes an output file having the following format:

Example of labelled trajectory file

```
# frame (or time)  GRID_index   trajectory_number
0.0.                    0                0
500.0                  56                0
…..                  ……             …….
…..                  ……             …….
29999000.0            205                0
29999500.0            175                0
```

Unassigned GRID points are those with GRID index *-999*.
The CVs GRID values can be added before the GRID_index column using the option:

- ***-writelabelscoord***.

Trajectory labeling is useful for ensemble averages and re-projecting the free energy along alternative CVs (see examples) based on reweighting [1]. Additionally, when applied to continuous trajectories, the labelled trajectory file file can be read by ***graf_fes_kmc.py*** to calculate the neighbors of each GRID point based on kinetic transitions.
Useful options: use ***-noforce*** to do only trajectory labeling and skip the other analysis steps.

Free energy and minimum free energy paths calculation based on KMC

The program ***graf_fes_kmc.py*** can read the free energy gradient file written in output by ***calcf_vgauss.py*** using the specification:

- ***-ff*** *{name of the free energy gradient file}*)

Based on the gradient file, ***graf_fes_kmc.py*** allows to calculate the free energy landscape using a KMC approach [1] (particularly efficient for multidimensional analysis). The KMC is based on the transition rate between neighboring GRID points α and β:

$$R_{\alpha\beta} = \left(\tau^0_{\alpha\beta}\right)^{-1} exp\left\{-\beta\frac{\Delta F_{\alpha\beta}}{2}\right\} \qquad (3).$$

Where $\Delta F_{\alpha\beta} = F(\boldsymbol{\xi}^{\boldsymbol{\beta}}) - F(\boldsymbol{\xi}^{\boldsymbol{\alpha}})$ is the free energy difference between those GRID points which can be calculated from the free energy gradients using finite differences [1]. The term $\beta = 1/k_B T_{KMC}$, where the KMC temperature, T_{KMC}, must be provided in input by the user (see below).
The program ***graf_fes_kmc.py*** can also calculate the minimum free energy path between two GRID points.

An estimate of the free energy can be also provided in input through an external file. This can be useful for filtering or as an alternative for the calculations of the rates in eq. 3. For example, for calculating minimum free energy paths based on an available free energy landscape. The free energy file can be read using the following options:

- ***-readfes -rfesfile*** *{name of the free energy file}*).
  By default, ***graf_fes_kmc.py*** attempts to read also a free energy gradient file and consequently assumes that the rates of eq. 3 are based on gradient file via finite differences. To use exclusively the free energy for all analyses and omit reading the gradient file, the following option must be added:
  - ***-noforces***
    Unless this option is specified, the free energy read from input is only used for filtering.

Specifications for neighbors search or reading

By default, ***graf_fes_kmc.py*** calculates the nearest neighbors of each GRID point (assuming a regular GRID: e.g. maximum 4 neighbors of a GRID point in 2D) which then are used for KMC calculations and other operations. Other options however can be used in this regard:

- ***-notnearest***: neighbors are those that share a border region (maximum 8 neighbors of a GRID point in 2D)
- ***-cutoff --cutoffval*** *[cutoff value]*: neighbors of a GRID point are those within the specified distance cutoff from it. The distance corresponds to the maximum norm (maximum of the absolute value of the components) where each CV is scaled by the bin size.
- ***-kcont -labelsf*** {name of label file}: neighboring GRID points are those for which direct kinetic transitions between each other are observed during continuous trajectories, based on a labelled trajectory file. The latter file may entail multiple continuous trajectories appended. GRID points that do not share a border region are excluded.
  The following option can be used to introduce a threshold on the number of transitions between GRID points to be considered neighbors:
  - ***-minbintrans*** *[number of transitions]*
- ***-rneighs -ineighf*** *{name of the neighbors file}:* neighbors of each GRID point are read from an external file

The list of neighbors for each GRID point are printed in an output file and can be read from an input file having the following format:

Example of neighbors' file

```
# GRID_index         number of neighbors               neighbors_GRID_index
0                              4                    1      125      126      15139
1                              4                    0      2        127      15140
…..                         ……                   …      …        …        …
…..                         ……                   …      …        …        …
15256                          3                   119    15255    15257    -1
15257                          4                   120    15133    15256    15258
```

Unassigned neighbors are those with index -1.

Calculation of the free energy from the mean forces using KMC

The running command to calculate the free energy from the gradient file (*grad_on_eff_points.out*) is:

> *python3.8* ***graf_fes_kmc.py*** *-ff grad_on_eff_points.out -units {kj or kcal} -temp [T_{KMC}] -nsteps {number of KMC steps} (additional options)*

The KMC simulation is started from the GRID point having the largest weight, according to the weight

column of the gradient file or to the minimal weight in case of multiple weight columns.
The program writes in output a file (named by default fes.out) with the free energy for each GRID point having the following format:

Example of free energy file for two CVs

```
# 2
# -180.0   2.5    144    1
# -180.0   2.5    144    1
-178.75         -178.75          0.6119651574747521
-176.25         -178.75          0.6467188103886569
   …               …                    …
   …               …                    …
 176.25          178.75          0.5315035921803015
 178.75          178.75          0.5437504610428038
 CV_1            CV_2             free_energy
```

This file entails the same header of the free energy gradient file produced by ***calcf_vgauss.py***. GRID points that are not explored by the KMC trajectory have undefined free energy giving rise to "nan" values. The convergence of the free energy depends on the *number of KMC* steps selected (typically in order of billions) and by T_{KMC}. Setting the latter parameter a few times larger than the simulation temperature enables to sample more efficiently large free energy barriers. Note however that extremely large temperatures leads to uniform sampling of the CVs space with consequent degradation of the free energy accuracy (starting from low free energy regions) hence a tradeoff range must be found (for the examples reported here between 500 and 3000K).
The accuracy of the free energy calculated through KMC simulations can be improved by trivial parallelization. Namely, using multiple independent runs and averaging the populations, $P_i(\boldsymbol{\xi}^{\boldsymbol{\alpha}}) = exp\{-F_i(\boldsymbol{\xi}^{\boldsymbol{\alpha}})/k_B T_{KMC}\}$, inferred from each run, where $F_i(\boldsymbol{\xi}^{\boldsymbol{\alpha}})$ is the free energy estimate obtained by run *i*. Assuming all runs have the same number of steps, the combined free energy is given by:

$$F(\boldsymbol{\xi}^{\boldsymbol{\alpha}}) = -k_B T_{KMC} log\left(\sum_i P_i(\boldsymbol{\xi}^{\boldsymbol{\alpha}})\right) \qquad (4).$$

For the latter equation to be valid, each individual KMC simulation must entail enough steps so that can be considered well equilibrated. Multiple KMC runs are generally recommended to evaluate the statistical accuracy of the free energy (through e.g. blocks analysis).

Calculation of minimum free energy paths

The calculation of the minimum free energy path between two GRID points ($\alpha_1$and α_N) implemented in ***graf_fes_kmc.py*** is based on the maximization of the pathway probability, $P^{KMC}_{\alpha_1,\alpha_N} = P^{KMC}_{\alpha_1,\alpha_2} P^{KMC}_{\alpha_2,\alpha_3} \dots, P^{KMC}_{\alpha_{N-1},\alpha_N}$, where $P^{KMC}_{\alpha_i,\alpha_j} = R_{\alpha_i,\alpha_j} / \sum_{\alpha_j} R_{\alpha_i,\alpha_j}$ [1]. This implementation does not allow yet to specify the pre-exponential term of the transitions rate, R_{α_i,α_j}, (e.g. for setting it according to the diffusion coefficients) which could affect the shape of the most probable pathway (although the exponential term is expected to provide the dominant contribution). The latter term is approximated by a geometric factor [1].
The two endpoints of the path are specified by:

- ***-sbmfepath*** *[GRID index of the starting point]*
- **-fbmfepath** *[GRID index of the final point]*

The GRID index is the same reported in the first column of the GRID points file and corresponds to the line number (starting from 0) of each a GRID point in the free energy gradient file.
The temperature of the transition rates, R_{α_i,α_j}, to define the pathway probability [1] is set by:

- ***-pathtemp*** *[path temperature]*
  To obtain the minimum free path this temperature must be small (default is 10K).

When minimum free energy path calculation is requested (see options below) the program writes the most probable path found on a file (named by default path_file.out) having the following format:

Example of minimum free energy path file for two CVs

```
#           CV_1                 CV_2          GRID_index        free_energy             delta_free_energy
-1.5053464798452    2.77071018754118     11109       0.365018396429295          0.0000000000000
-1.5053464798452    2.81434341884104     11208       0.406946751515946         -0.0419283550866
…                   …                     …          …                           …
…                   …                     …          …                           …
1.11264739814645   -2.02894525544354     2741        16.90701656299612          0.0562905674908
1.11264739814645   -2.07257848674340     2618        16.84814623475454          0.0588703282415
```

Two approaches are available for paths finding which are enabled by the following options:

- ***-smfepath***
  uses a systematic search across neighbors. This is a fast and accurate approach in low dimensionality but becomes unfeasible in large dimensionality.
- **-mfepath**
  Uses a stochastic search suitable for large dimensionality which is based on two stages:
  1. Global search consisting of a set of KMC simulations started from the initial point and stopped unpon reaching the final point. The trajectories are sorted according to the pathway probability.
  2. Starting from the trajectory with largest probability, alternative paths are generated by selecting two random GRID points along the current path and running KMC trajectories that connect those points. A new path is accepted according to a Metropolis-Monte Carlo criterion with probability $exp\{log(P_{\alpha_1,\alpha_N}^{KMC})/(N_{CVs}T_{MC})\}$, where N_{CVs} is the number of CVs (dimensionality) and T_{MC} is a dimensionless temperature factor defined with the option:
     - ***-mctemp*** *[T_{MC}]*

     Multiple runs can be performed by gradually reducing T_{MC} to converge to the most probable path. To do this, the paths generated in previous runs can be read as starting points for a subsequent run using the options:
     - ***-readpath -rpathfile*** *{name of path file}*

  The number of iterations for each stage, which regulates the convergence, is set by:
  - ***-npaths*** *[number of iterations]*
    The default value is 1000 (adequate for the first run. A larger value is recommended for subsequent runs at reduced T_{MC}).

  An example of simulated annealing protocol is reported

The program writes in output also a file (named by default per_iter_path_file.out) reporting, for each iteration, information useful to assess convergence, such as -$log(P_{\alpha_1,\alpha_N}^{KMC})$ (first two columns) and the path length.
Useful options: use ***-nofes*** to do only minimum free energy path calculation and skip the calculation of the full free energy landscape.

Filtering Options

A few options are available to exclude GRID points from the analysis based on specific features:

- ***-weth*** *[weight value]*
  GRID points having total weight ($\sum_r \sum_t w_t^r(\boldsymbol{\xi})$, see eq. 1) smaller than the specified value are excluded from the analysis. By default, all GRID points are included. This option is valid when a free energy gradient file is read in input (through the option ***-ff***), in which weight information is reported in the last column. When multiple weight columns are present, the minimal weight (on each line) is considered for excluding GRID points.
- ***-minneighs*** *[minimal number of neighbors]*
  GRID points having number of neighbors smaller than the specified value are excluded from the analysis. The default value is 0.
- ***-maxfes*** *[free energy threshold]*

GRID points with free energy larger than the specified value are excluded. This option can be useful for example to search for the smallest energetic barrier between free energy minima and exclude GRID points to speed up the search of minimum free energy paths. This option is relevant only when the free energy is read from an external file.

Additional tools

The **tools** folder includes bash/awk scripts that may be useful for analysis.

One dimensional free energy integration script based on finite differences

- *get_integral.sh*
  A simple awk/bash script for one-dimensional free energy integration using a numerical approach based on finite differences. It can be run using the command (for help type: *bash get_integral.sh -h*):

```
bash get_integral.sh {pmf_gradient_file} [CV lower bound] [CV upper bound] [CV number of GRID points]
```

Partitioning neighbors search into GRID points subsets

- *partition_neighs* folder
  Entails two awk/bash script designed to partition the neighbor search over subsets of GRID points that partially overlap. The rational is that the GRID points provided by ***calcf_vgauss.py*** are ordered so that proximal GRID indexes usually correspond to nearby GRID points. Partitioning the GRID point search into subsets can provide a significant speed up for high dimensional analysis although it can result in underestimation of the number of neighbors. The scripts provided for this analysis are;
  - *do_partitions.sh*
    awk/bash to split GRID points and gradient file in smaller and overlapping subsets. Neighbor search is carried out by ***graf_fes_kmc.py*** on each subset individually using independent runs (which can be run on different computer nodes). The results of each run are then combined to obtain a unique file with the neighbors of all GRID points named *neighsc_tot.out*. The latter neighbor file can be read on a subsequent run by ***graf_fes_kmc.py*** for free energy and minimum free energy path calculations.

  - *job_tmp.sh*
    template used by *do_partitions.sh* for running the neighbor search using ***graf_fes_kmc.py*** on a subset of GRID points. If you user slurm, modify this file to add the associated specifications.

  The running command is (for help type: *bash do_partitions.sh -h*):

```
bash do_partitions.sh {GRID points file} {free energy gradient file} {path of graf_fes_kmc.py} {partitions number} (slurm) ({1 if slurm done, else 0 })
```

  The option *slurm* can be added if the program is to be run on computers cluster using slurm for jobs submission (*sbatch job_name*). In that case the script must be used in two stages; first (options "*slurm 0*") the slurm jobs are submitted, second (options "*slurm 1*"), after slurm jobs are completed, the output files are combined to obtain the neighbor file.
  An example of partitioning the calculation of GRID points neighbors is provided for the case of concurrent metadynamics simulation of butyramide using two CVs:
  - folder *{FCAM main folder}/**tutorial**/colvars/buta_2d_concurrent/partition_neighs*

The specific steps of the analysis are provided in the *run.sh* bash script.

Examples

The ***tutorial*** folder entails a list of examples in which the software tools described here have been applied to different systems, simulation types and simulation software's. The **tutorial** folder contains three subfolders *colvars*, *plumed1.3* and *plumed2*, each including free energy analysis of simulations performed with the software naming the subfolder. Simulations using Colvars[12] were performed with NAMD[9, 10], whereas simulations based on PLUMED[13-15] were perfumed with GROMACS[11]. Each folder/subfolder entails:

- Input file of ***calcf_vgauss.py***.
- Trajectory files read by ***calcf_vgauss.py***.
- *run.sh* bash script with the list of running commands for ***calcf_vgauss.py*** and ***graf_fes_kmc.py***.
- *results* subfolder containing the main output files/results of the analysis.
- For metadynamics and its variants, *meta_inp* subfolder containing the main setup/input files of the simulation.

1D metadynamics

Reading biasing forces from the CVs trajectory file

- folder *{FCAM main folder}/**tutorial**/colvars/buta_1d*
  Metadynamics simulation (~ 60 ns) of solvated butyramide using the Φ (Fig. 1 in ref. [1]) dihedral angle as biased CV. The simulation is carried out with NAMD[9, 10] and Colvars[12] with CVs and biasing force trajectory written in the file *meta.1.equil.colvars.traj*. The latter file has the following format:

meta.1.equil.colvars.traj file for 1D metadynamics of butyramide

```
# step        dihe1              ft_dihe1            fa_dihe1             dihe2                ft_dihe2             fa_dihe2
        0    2.9314175415e+01  0.0000000000e+00  0.0000000000e+00  -1.7970158313e+02  0.0000000000e+00  0.0000000000e+00
      500    1.1535759912e+02 -1.3259568935e-01  0.0000000000e+00  -1.2399044189e+02  6.9562452128e-01  0.0000000000e+00
     1000    9.5401749700e+01  4.3135502067e-02  0.0000000000e+00  -5.8682636354e+01 -8.7555929331e-01  3.2635659581e-05
      …        …                  …                   …                    …                    …                    …
      …        …                  …                   …                    …                    …                    …
```

The biased CV correspond to the fourth column (starting from 0) whereas the biasing force (or applied force) is on column 6. Hence the input file for analysis (*input.dat*) is the following:

```
COLVAR_FILE meta.1.equil.colvars.traj
CV-CL 4 -180  180 360 PERIODIC
```

The GRID input entails lower and upper boundaries of -180 and 180 deg respectively and this range is subdivided in 360 GRID points.
The running command to derive the free energy gradients is (see also *run.sh* file):

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kcal -colvars -colvarbias_column 2 -temp 298
```

The free energy gradients are written on the file *grad_on_eff_points.out* (files *grad_on_eff_points1.out* and *grad_on_eff_points2.out* are also written and can be used for assessing the statical error) The free energy gradients can be integrated using KMC to obtain the free energy profiles:

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points.out -units kcal -temp 698 -nsteps 1000000000 -ofesf fes.out
```

where *fes.out* is contains the 1D free energy profile along Φ. For 1D, the analogous profile can be also obtained using a simple integrator based on finite differences (provided in the ***tools*** folder):

```
bash {FCAM main folder}/tools/get_integral.sh grad_on_eff_points.out -180 180 360
```

The output file containing the free energy profile is *fes_analytic.out*.

Multiple metadynamics trajectories

- As show in folder *{FCAM main folder}/**tutorial**/colvars/buta_1d/multiple_traj* two independent metadynamics trajectories, including CVs trajectories *../traj2/meta.1.equil.colvars.traj2* and *../ meta.1.equil.colvars.traj*, can be analyzed simultaneously using the following input file:

```
COLVAR_FILE ../traj2/meta.1.equil.colvars.traj2
COLVAR_FILE ../meta.1.equil.colvars.traj
CV-CL 4 -180  180 360 PERIODIC
```

  Mean forces and free energies can be calculated using the same running commands reported above.

The same results can be also obtained by first analyzing individually each trajectory (folders *{FCAM main folder}/**tutorial**/colvars/buta_1d* and *{FCAM main folder}/**tutorial**/colvars/buta_1d/traj2*) and then combining the free energy gradient files as shown in the folder:

- *{FCAM main folder}/**tutorial**/colvars/buta_1d/combine*

- folder *{FCAM main folder}/**tutorial**/colvars/deca-L-alanine*
  Same type of analysis for metadynamics (using Colvars[12]) in which the biased CV is the end-to-end distance of deca-L-alanine (Fig. 2 in ref. [1]). The end-to-end distance is on column 10 (starting form 0) of the *meta.1.equil.colvars.traj* file and the biasing force is on column 12:

meta.1.equil.colvars.traj file for 1D metadynamics on deca-L-alanine

```
# step        drmsd_alpha_pi       ft_drmsd_alpha_pi     fa_drmsd_alpha_pi      drmsd_alpha_310       ft_drmsd_alpha_310     fa_drmsd_alpha_310
drmsd_pi_310         ft_drmsd_pi_310       fa_drmsd_pi_310        endToEnd            ft_endToEnd          fa_endToEnd
         0      1.87516628886429e+00   0.00000000000000e+00   0.00000000000000e+00     3.51982869524296e+00   0.00000000000000e+00
0.00000000000000e+00          2.03608072641434e+00        0.00000000000000e+00        0.00000000000000e+00          1.44251677979842e+01
0.00000000000000e+00  4.96031918420945e-04
       500      1.91087436015728e+00   6.36446100776480e+01   0.00000000000000e+00     3.56656797086061e+00   2.67960944377219e+01
0.00000000000000e+00          1.98495083802446e+00     -4.77854358138410e+01       0.00000000000000e+00          1.40906501013127e+01
3.24828411854907e+01 -5.66240451777488e-03
      1000      2.02556124593402e+00  -4.12577960213198e-01   0.00000000000000e+00     3.57589297027775e+00   7.19430051084124e+00
0.00000000000000e+00         1.83166326888091e+00       7.45187328118181e+01       0.00000000000000e+00        1.42501523739237e+01    -
1.16508488444637e+01 -3.29787506163752e-03
      ….     ……………………… ………………………  ………………………   ………………………  ……………………..   ………………………   ……
…………………  ………………………  ………………………   ……………………..  ………………………   ………………………
```

In this case the CV is not periodic hence the input file (*input.dat*) is:

```
COLVAR_FILE meta.1.equil.colvars.traj
CV-CL 10 5  40 175
```

The GRID input entails lower and upper boundaries of 5Å and 40Å respectively and this range is subdivided in 175 GRID points. The PERIODIC keyword is omitted in this case as the end-to-end distance is not a periodic CV.
The running command to calculate the free energy gradient is:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kcal -colvars -colvarbias_column 2 -temp 298
```

The main output files have same name and meaning of the ones for the butyramide example and the free energy profile can then be obtained through KMC:

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points.out -units kcal -temp 2298 -nsteps 1000000000 -ofesf fes.out
```

Note that in this case a larger temperature was used for the KMC to be able to evaluate the large free energy difference between helical and extended states which is on the order of 20 kcal/mol (Fig. 2 in ref. [1]).
The same 1D free energy profile can also be calculated using the script provided in the ***tools*** folder:

```
bash {FCAM main folder}/tools/get_integral.sh results/grad_on_eff_points.out 5 40 175
```

Calculating the biasing force from the "hills" file

- folder *{FCAM main folder}*/**tutorial**/colvars/buta_1d_read_hills
  Same metadynamics simulation of butyramide reported above with biasing forces calculated from the "hills" file instead of reading them from the CVs trajectory file. The the hills file, *meta.1.equil.colvars.meta2.hills.traj*, has the following format:

"hills" file (meta.1.equil.colvars.meta2.hills.traj) for 1D metadynamics of butyramide

```
   1000   -5.86826363548143e+01   5.00000000000000e+00   2.50000000000000e-02
   2000   -7.33922178917330e+01   5.00000000000000e+00   2.50000000000000e-02
   3000   -6.23614978161429e+01   5.00000000000000e+00   2.50000000000000e-02
   …….    ……………………………...     ……………………………...     ……………………………
  Frame            Φ                 Gaussian_sigma         Gaussian_heigth
```

The "hills" file can be written through Colvars[12] by enabling *writeHillsTrajectory* (https://colvars.github.io). The input file (*input.dat*) for this analysis is:

```
HILLS_FILE meta.1.equil.colvars.meta2.hills.traj COLVAR_FILE ../buta_1d/meta.1.equil.colvars.traj HILLS_CVS-CLS 4
CV-CL 4 -180  180 360 PERIODIC
```

The running command to obtain the free energy gradients is:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kcal -colvars -temp 298
```

The free energy can be then obtained as shown previously.

2D Metadynamics

- folder *{FCAM main folder}*/***tutorial****/plumed2/AAPAK/meta-2d*
  2D Metadynamics of the cis-trans isomerization of the AAPAK peptide performed with GROMACS[11] and PLUMED2[14]. The Ω (describing the cis-trans transition) and Ψ (of the backbone) dihedral angles are the biased CVs (see *plumed.dat* file in the enclosed *meta_inp* folder). The mean forces can be derived from *COLVAR* (CVs trajectory with Ω and Ψ in columns 1 and 2) and *HILLS* ("hills" file) files using ***calcf_vgauss.py***. The HILLS file has the following format:

"hills" file (HILLS) for 2D metadynamics of the AAPAK peptide

```
 #! FIELDS time omega psi sigma_omega sigma_psi height biasf
 #! SET multivariate false
 #! SET min_omega -pi
 #! SET max_omega pi
 #! SET min_psi -pi
 #! SET max_psi pi
     2.000000094994903    2.824348584082339   -0.4590819040057097        0.17          0.24          0.2           1
     4.000000189989805    2.851430730900406   -0.7171619217815017        0.17          0.24          0.2           1
     ….                   …..                 …..                        ……            ……            ….            ..
     ….                   …..                 …..                        ……            ……            ….            ..
           Time                 Ω                     Ψ               sigma_Ω      sigma_Ψ   Gaussian_height   dummy
```

The input file (*input.dat*) for ***calcf_vgauss.py*** is:

```
HILLS_FILE HILLS COLVAR_FILE COLVAR HILLS_CVS-CLS 1 2
CV-CL 1 -3.14159265359  3.14159265359 144 PERIODIC
CV-CL 2 -3.14159265359  3.14159265359 144 PERIODIC
```

The GRID lower and upper boundaries for each dihedral angle are $-\pi$ and π rad respectively and the overall GRID entails 144x144 points.

The running command is:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kj -temp 300 -colv_time_prec
2
```

At difference with simulations performed with NAMD[9, 10] and Colvars[12] the previous command specifies kJ/mol units typical of GROMACS[11] and a reduced precision is used to read the time column (***-colv_time_prec*** option) to have same digits for *COLVAR* and *HILLS* files so that biasing forces are calculated correctly.

The free energy can be calculated through KMC using a sufficiently large temperature so as to sample sufficiently well the 2D landscape (free energy differences of ~20 kcal/mol):

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points.out -units kj -temp 2300
-nsteps 10000000000 -weth 1 -ofesf fes.out
```

Bias exchange metadynamics

- Folder *{FCAM main folder}/**tutorial**/plumed2/AAPAK/be*
  Bias exchange metadynamics simulations of the AAPAK peptide using two replicas (replica 0 and 1) each biased along either Ω or Ψ dihedral angles reported previously. The replicas produce CVs trajectory files *COLVAR.0* and *COLVAR.1* and "hills" files *HILLS.0* and *HILLS.1* (the first biased along Ω and the second along Ψ):

COLVAR0

```
#! FIELDS time cv1 cv2
#! SET min_cv1 -pi
#! SET max_cv1 pi
#! SET min_cv2 -pi
#! SET max_cv2 pi
 0.000000 2.546040 -0.403747
 1.000000 2.589227 -0.567716
…           …           …
…           …           …
Time        Ω           Ψ
```

COLVAR1

```
#! FIELDS time cv1 cv2
#! SET min_cv1 -pi
#! SET max_cv1 pi
#! SET min_cv2 -pi
#! SET max_cv2 pi
 0.000000 2.546040 -0.403747
 1.000000 2.749705 -0.437856
…           …           …
…           …           …
Time        Ω           Ψ
```

HILLS0

```
#! FIELDS time cv1 sigma_cv1 height biasf
#! SET multivariate false
#! SET min_cv1 -pi
#! SET max_cv1 pi
     2.000000094994903    2.857224334570646          0.17          0.2          1
     4.000000189989805    3.063882591383335          0.17          0.2          1
     …                    …                          …             …            …
     …                    …                          …             …            …
          Time                   Ω              sigma_Ω   Gaussian_height    dummy
```

HILLS1

```
#! FIELDS time cv2 sigma_cv2 height biasf
#! SET multivariate false
#! SET min_cv2 -pi
#! SET max_cv2 pi
    2.000000094994903   -0.7256785057897139          0.24          0.2          1
    4.000000189989805   -0.2978975265641424          0.24          0.2          1
    …                     …                          …             …            …
    …                     …                          …             …            …
           Time                   Ψ                sigma_Ψ   Gaussian_height   dummy
```

The following input file can be used to derive mean forces using ***calcf_vgauss.py***:

```
HILLS_FILE HILLS.0 COLVAR_FILE COLVAR.0 HILLS_CVS-CLS 1
HILLS_FILE HILLS.1 COLVAR_FILE COLVAR.1 HILLS_CVS-CLS 2
CV-CL 1 -3.14159265359  3.14159265359 144 PERIODIC
CV-CL 2 -3.14159265359  3.14159265359 144 PERIODIC
```

Mean forces and free energies can be derived running the commands:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kj -temp 300 -colv_time_prec 2
```

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points.out -units kj -temp 2300 -nsteps 10000000000 -weth 1 -ofesf fes.out
```

The resulting free energy can be compared with the one obtained with 2D metadynamics (see previous example).

A similar example in which biasing forces are read externally (written by PLUMED using DUMPFORCES keyword) is provided here:

- *{FCAM main folder}/**tutorial**/plumed2/AWPAK/be/read_bias_force_trj*
  similar bias exchange metadynamics simulation on the AWPAK peptide. The input file for ***calcf_vgauss.py*** is:

```
COLVAR_FILE ../COLVAR.0
COLVAR_FILE ../COLVAR.1
CV-CL 1 -3.14159265359  3.14159265359 144 PERIODIC
CV-CL 2 -3.14159265359  3.14159265359 144 PERIODIC
READ_BIAS_FORCE_TRJ ../forces.0
READ_BIAS_FORCE_TRJ ../forces.1
```

  where *forces.0* and *forces.1* are the biasing forces files of replica 0 and 1 respectively. The running command to derive the free energy gradient does not require the option ***-colv_time_prec*** as biasing forces are provided externally:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kj -temp 300
```

  The running command for free energy integration is instead the same as above.

Concurrent Metadynamics

This is a type of simulation in which multiple 1D metadynamics biasing potentials are applied to different CVs in the same simulation, with a total bias potential provided by the sum of the individual 1D biasing potentials[1, 17]. Note that this type of simulations can be directly analyzed through ***calcf_vgauss.py*** only by reading the biasing force externally as reported in the example below:

- Folder *{FCAM main folder}/**tutorial**/colvars/buta_2d_concurrent*
  Concurrent metadynamics simulation in which the biased CVs are Φ and Ψ dihedral angles of butyramide[1]. The format of the CVs trajectory file, *meta.1.equil.colvars.traj*, is the same reported in the previous 1D example in which the CVs are column 1 and 4 and the biasing forces column 3 and 6. Thus the input file for ***calcf_vgauss.py*** (*input.dat*) is:

```
COLVAR_FILE meta.1.equil.colvars.traj
CV-CL 1 -180  180 144 PERIODIC
CV-CL 4 -180  180 144 PERIODIC
```

The running command is:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kcal -colvars -colvarbias_column 2 -temp 298
```

The free energy can be calculated using KMC:

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points.out -units kcal -temp 1298 -nsteps 10000000000 -weth 1 -ofesf fes.out
```

Note that the automatic calculation of the biasing forces through “hills” files is not available for this type of simulations. If the biasing forces are not available externally one option to derive them from the “hills” files is to use two runs of ***calcf_vgauss.py***; each using the same CVs trajectory, but one of the two “hills” files associated with the respective CV (and specify in the input file which CV). This will give rise to two mono dimensional bias gradient files which must be then rearranged in a two-dimensional bias gradient file by pasting together the columns of the bias gradient of each CV. The multidimensional bias gradient can be read by ***calcf_vgauss.py*** through the keyword ***READ_BIAS_GRAD_TRJ***.

Umbrella sampling based on harmonic potentials

- Folder *{FCAM main folder}/**tutorial**/colvars/nanotube-umbrella_sampling/nanotube-big-water*
  Umbrella sampling simulations to investigate the free energy of solvation/de-solvation of the inner region of a nanotube. The simulations entail 151 different windows with harmonic biasing potentials on the number of water molecules (using a density variable [18]) in the inner region of the nanotube. The center of the harmonic potential varies of 0.2 water molecules between a window and the next so as to span a range of 30 water molecules globally and each window has a force constant of 12.5 kcal/(mol x number of water molecules). The CVs trajectory file for window *i* is *nt-water.us-00i.us.colvars.traj*, in which the biased CV is written in column 1. Based on this information, the input file for ***calcf_vgauss.py*** is:

```
COLVAR_FILE nt-water.us-000.us.colvars.traj US_CVS-CLS 1 US_C 0 US_K 12.5
COLVAR_FILE nt-water.us-001.us.colvars.traj US_CVS-CLS 1 US_C 0.2 US_K 12.5
COLVAR_FILE nt-water.us-002.us.colvars.traj US_CVS-CLS 1 US_C 0.4 US_K 12.5
...         ...                             ...        . ...  ..  ...  ..
...         ...                             ...        . ...  ..  ...  ..
...         ...                             ...        . ...  ..  ...  ..
COLVAR_FILE nt-water.us-148.us.colvars.traj US_CVS-CLS 1 US_C 29.6 US_K 12.5
COLVAR_FILE nt-water.us-149.us.colvars.traj US_CVS-CLS 1 US_C 29.8 US_K 12.5
COLVAR_FILE nt-water.us-150.us.colvars.traj US_CVS-CLS 1 US_C 30 US_K 12.5
CV-CL 0 0  30 1000
```

The GRID input entails lower and upper boundaries of 0 and 30 water molecules respectively and this range is subdivided in 1000 GRID points.
The running command is:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kcal -temp 300 -nopgradb -oeff grad_on_eff_points.out
```

The free energy can be obtained from the associated gradient (*grad_on_eff_points.out*) using KMC:

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points.out -units kcal -temp 698 -nsteps  100000000000 -ofesf fes.out
```

or the provided script for 1D free energy integration:

```
bash {FCAM main folder}/tools/get_integral.sh grad_on_eff_points.out 0 30 1000
```

Combining free energy gradient estimates from different trajectories

To provide an example of this type of analysis we consider bias exchange metadynamics simulations for dialanine on two replicas (CVs trajectories and "hills" files *COLVAR0*, *COLVAR1*, *HILLS0* and *HILLS1*), each biased on one of the two backbone dihedral angles. Instead of analyzing all replicas together as reported previously, here we analyze each replica individually and then we combine the two free energy gradient estimates.

- folder *{FCAM main folder}/**tutorial**/plumed1.3/diala-be/individual_plus_combine*
  1. Derive the GRID points using ***calcf_vgauss.py*** with input file (*input.dat*):

     ```
     COLVAR_FILE ../COLVAR0
     COLVAR_FILE ../COLVAR1
     CV-CL 1 -3.14159265359  3.14159265359 144 PERIODIC
     CV-CL 2 -3.14159265359  3.14159265359 144 PERIODIC
     ```

     The running command is:

     ```
     python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat --justcalceffpoint
     ```

     This will produce the GRID points file, *eff_points.out*, to be read in subsequent steps
  2. Derive free energy gradients for each replica in two separate folders:
     - *{FCAM main folder}/**tutorial**/plumed1.3/diala-be/individual_plus_combine/0*
     - *{FCAM main folder}/**tutorial**/plumed1.3/diala-be/individual_plus_combine/1*

     The input files for each calculation are:

     ```
     HILLS_FILE ../../HILLS0 COLVAR_FILE ../../COLVAR0 HILLS_CVS-CLS 2
     CV-CL 1 -3.14159265359  3.14159265359 144 PERIODIC
     CV-CL 2 -3.14159265359  3.14159265359 144 PERIODIC
     READ_EPOINTS ../eff_points.out
     ```

     and

     ```
     HILLS_FILE ../../HILLS1 COLVAR_FILE ../../COLVAR1 HILLS_CVS-CLS 1
     CV-CL 1 -3.14159265359  3.14159265359 144 PERIODIC
     CV-CL 2 -3.14159265359  3.14159265359 144 PERIODIC
     READ_EPOINTS ../eff_points.out
     ```

     Which can be processed with two running commands of the type:

     ```
     python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kj -temp 298
     ```
  3. The two gradients estimates can be combined in the folder *{FCAM main folder}/**tutorial**/plumed1.3/diala-be/individual_plus_combine* using the following input file (*input_combine.dat*):

     ```
     CV-CL 1 -3.14159265359  3.14159265359 144 PERIODIC
     CV-CL 2 -3.14159265359  3.14159265359 144 PERIODIC
     READ_GRAD_PMF 0/grad_on_eff_points.out
     READ_GRAD_PMF 1/grad_on_eff_points.out
     ```

     The running command to obtain the combined free energy gradient file (*grad_on_eff_points_comb.out*) is:

     ```
     python3.8 {FCAM main folder}/calcf_vgauss.py -if input_combine.dat
     ```

     The combined free energy gradient can be integrated using KMC to obtain the best estimate of the free energy (*fes_comb.out*):

     ```
     python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points_comb.out -units
     kj -temp 1298 -nsteps 10000000000 -weth 1 -ofesf fes_comb.out
     ```

The results can be compared with those obtained by analyzing trajectories all together, which are available here:

- *{FCAM main folder}/**tutorial**/plumed1.3/diala-be/results*

Minimum free energy paths

- Folder *{FCAM main folder}/**tutorial**/plumed1.3/diala-be/min_fes_path_c7eq-c7ax*
  Example of the derivation of the minimum free energy pathway between C_{7eq} and C_{7ax} free

energy minima of dialanine (Fig. 6 ref. [1]). The calculation is carried out using the systematic search approach (***-smfepath***) described previously by reading the mean forces (*grad_on_eff_points.out*) derived using ***calcf_vgauss.py*** (see also section above):

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -nofes -ff ../results/grad_on_eff_points.out -smfepath -sbmfepath 2618 -fbmfepath 11109 -temp 298 -units kj -pathtemp 1 -itpfile per_iter_path_c7eq-c7ax_ff.out -pfile path_c7eq-c7ax_ff.out
```

where *2618* and *11109* are the starting and ending GRID points indexes. The most probable pathway found is written on the file *path_c7eq-c7ax_ff.out*.
A similar pathway can be found by reading the free energy landscape (*fes.out*) instead of the mean forces:

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -noforces -nofes -readfes -rfesfile ../results/fes.out -smfepath -sbmfepath 2618 -fbmfepath 11109 -temp 298 -units kj -pathtemp 1 -itpfile per_iter_path_c7eq-c7ax.out -pfile path_c7eq-c7ax.out
```

- Folder *{FCAM main folder}/****tutorial****/plumed1.3/diala-be/min_fes_path_c7eq-c7ax/MC_sim_ann*
  Derivation of the same pathway using a Monte Carlo approach with a simulated annealing protocol. The first search is carried out at Monte Carlo temperature (T_{MC}) of 50 and reduced number of steps **(*-npaths*** *1000*) to obtain an initial approximate guess of the path (*path_c7eq_c7ax_50.out*):

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -noforces -nofes -readfes -rfesfile ../../results/fes.out -mfepath -sbmfepath 2618 -fbmfepath 11109 -temp 400 -npaths 1000 -pathtemp 1 -units kj -itpfile per_iter_path_c7eq_c7ax_50.out -pfile path_c7eq_c7ax_50.out -mctemp 50
```

The KMC temperature of 400K provides higher probability to overcome the free energy barrier between the two endpoints compared to the simulation temperature (298K) and thus ensures that an initial pathway can be found.
This initial path is read in the subsequent step that is carried out at T_{MC}=40 using a larger number of iterations ***(-npaths*** 100000):

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -noforces -nofes -readfes -rfesfile ../../results/fes.out -mfepath -rneighs -nopneighs -ineighf neighs.out -sbmfepath 2618 -fbmfepath 11109 -temp 400 -npaths 100000 -pathtemp 1 -units kj -itpfile per_iter_path_c7eq_c7ax_40.out -pfile path_c7eq_c7ax_40.out -tpaths 1 -mctemp 50 -readpath -rpathfile path_c7eq_c7ax_50.out
```

The calculation is continued by gradually reducing T_{MC} (up to T_{MC}=0.3, see *run.sh* script file) to gradually converge to the minimum free energy path.

Free energy as a function of alternative CVs through ensemble reweighting

The free energy as a function of alternative CVs is a particular case of ensemble average of an observable and can be estimated according to eq. 13, 14 of ref. [1]. This analysis requires:

- A file with the alternative CV as a function of the simulation frames
- A file with GRID point index for the same simulation frames
- The free energy for each GRID point

The latter information is used to derive the weight of each simulation frame, which can be used to calculate ensemble averages (through weighted averages) or weighted histograms of alternative CVs

(and hence the free energy across those CVs). The weight is given by $\exp\{-\beta F(\boldsymbol{\xi}^{\alpha(i)})\}/N_{\boldsymbol{\alpha}(i)}$, where $\alpha(i)$ is the GRID point index of frame i, $F(\boldsymbol{\xi}^{\alpha(i)})$ is the free energy of GRID point α and $N_{\boldsymbol{\alpha}(i)}$ is the cumulative number of frames in that GRID point.
An example of this analysis is provided for the 1D metadynamics of butyramide reported previously:

- folder *{FCAM main folder}/**tutorial**/colvars/buta_1d/free_energy_on_psi_reweight*
  In this folder it is shown how to derive the free energy profile as a function of the Ψ dihedral angle, which is not directly biased during the simulation. The CVs trajectory files, *meta.1.equil.colvars.traj*, reports on column 1 the value of Ψ as a function of the simulation frame. The GRID point index for each trajectory frame (label.out) can be calculated using the option **-label** of ***calcf_vgauss.py***, with input file (*input.dat*) given by:

```
COLVAR_FILE ../meta.1.equil.colvars.traj
CV-CL 4 -180  180 360 PERIODIC
READ_EPOINTS ../results/eff_points.out
```

  The running command is:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -label -noforce
```

  The file label.out can be used to count the number of frames in each GRID point ($N_{\boldsymbol{\alpha}(i)}$) and through the free energy file (*../results/fes_analytic.out*) to derive the weight for each simulation frame. The latter weight is used to calculate the weighted histogram along Ψ from which the associated free energy profile is derived.
  All the details for this analysis are provided in the *run.sh* bash/awk script reported in this folder.

Convergence and error evaluation

- Folder {FCAM main folder}/tutorial/colvars/buta_1d
  Let us take the example of the 1D metadynamics of butyramide reported previously. Upon calculating the mean forces the program ***calcf_vgauss.py*** writes in output also files *grad_on_eff_points1.out* and *grad_on_eff_points2.out*, corresponding to free energy gradients obtained using first and second half of the sampling for each GRID point, which provide an estimate of the error on the mean forces. These gradient files can be integrated to get the associated free energies (*fes1.out* and *fes2.out*) which can be used for assessment of the statistical error on the free energy:

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points1.out -units kcal -temp 698 -nsteps 1000000000 -ofesf fes1.out
```

```
python3.8 {FCAM main folder}/graf_fes_kmc.py -ff grad_on_eff_points2.out -units kcal -temp 698 -nsteps 1000000000 -ofesf fes2.out
```

Convergence analysis as well as more rigorous error estimates can be obtained by individual analysis of different portions of each trajectory (through options ***-trfr1*** and ***-trfr2***). An example of this type of analysis is provided in the following folders:

- *{FCAM main folder}/**tutorial**/colvars/buta_1d/first_half*
  Analysis of first half of the trajectory:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kcal -colvars -colvarbias_column 2 -temp 298 -trfr1 0.0 -trfr2 0.5
```

- *{FCAM main folder}/**tutorial**/colvars/buta_1d/second_half*
  Analysis of second half of the trajectory:

```
python3.8 {FCAM main folder}/calcf_vgauss.py -if input.dat -units kcal -colvars -colvarbias_column 2 -temp 298 -trfr1 0.5 -trfr2 1.0
```

For both runs the input file of ***calcf_vgauss.py*** is like the one used for analysis of the full trajectory, the only difference is that the GRID points previously calculated are read in input (so that mean forces for each portion of the trajectory are evaluated on the same GRID points):

```
COLVAR_FILE ../meta.1.equil.colvars.traj
CV-CL 4 -180  180 360 PERIODIC
READ_EPOINTS ../results/eff_points.out
```

The free energy evaluated from each portion of the trajectory can be calculated by integrating the corresponding gradient file (e.g. using ***graf_fes_kmc.py***). Different portions of the trajectories can be analyzed using appropriate values for options ***-trfr1*** and ***-trfr2***.

References

1. Marinelli, F. and J.D. Faraldo-Gomez, *Force-Correction Analysis Method for Derivation of Multidimensional Free-Energy Landscapes from Adaptively Biased Replica Simulations.* J Chem Theory Comput, 2021. **17**(11): p. 6775-6788.
2. Kastner, J. and W. Thiel, *Bridging the gap between thermodynamic integration and umbrella sampling provides a novel analysis method: "Umbrella integration".* J Chem Phys, 2005. **123**(14): p. 144104.
3. Zheng, L.Q. and W. Yang, *Practically Efficient and Robust Free Energy Calculations: Double-Integration Orthogonal Space Tempering.* Journal of Chemical Theory and Computation, 2012. **8**(3): p. 810-823.
4. Fu, H.H., et al., *Extended Adaptive Biasing Force Algorithm. An On-the-Fly Implementation for Accurate Free-Energy Calculations.* Journal of Chemical Theory and Computation, 2016. **12**(8): p. 3506-3513.
5. Lesage, A., et al., *Smoothed Biasing Forces Yield Unbiased Free Energies with the Extended-System Adaptive Biasing Force Method.* Journal of Physical Chemistry B, 2017. **121**(15): p. 3676-3685.
6. Marinova, V. and M. Salvalaglio, *Time-independent free energies from metadynamics via mean force integration.* J Chem Phys, 2019. **151**(16): p. 164115.
7. Laio, A. and M. Parrinello, *Escaping free-energy minima.* Proc Natl Acad Sci U S A, 2002. **99**(20): p. 12562-6.
8. Barducci, A., G. Bussi, and M. Parrinello, *Well-tempered metadynamics: A smoothly converging and tunable free-energy method.* Physical Review Letters, 2008. **100**(2).
9. Phillips, J.C., et al., *Scalable molecular dynamics on CPU and GPU architectures with NAMD.* Journal of Chemical Physics, 2020. **153**(4).
10. Phillips, J.C., et al., *Scalable molecular dynamics with NAMD.* Journal of Computational Chemistry, 2005. **26**(16): p. 1781-1802.
11. Hess, B., et al., *GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation.* J Chem Theory Comput, 2008. **4**(3): p. 435-47.
12. Fiorin, G., M.L. Klein, and J. Henin, *Using collective variables to drive molecular dynamics simulations.* Molecular Physics, 2013. **111**(22-23): p. 3345-3362.
13. Bonomi, M., et al., *PLUMED: A portable plugin for free-energy calculations with molecular dynamics.* Computer Physics Communications, 2009. **180**(10): p. 1961-1972.
14. Tribello, G.A., et al., *PLUMED 2: New feathers for an old bird.* Computer Physics Communications, 2014. **185**(2): p. 604-613.
15. Bonomi, M., et al., *Promoting transparency and reproducibility in enhanced molecular simulations.* Nature Methods, 2019. **16**(8): p. 670-673.
16. Hénin, J., *Fast and accurate multidimensional free*
energy integration. arXiv:2107.08511, 2021.
17. Gil-Ley, A. and G. Bussi, *Enhanced Conformational Sampling Using Replica Exchange with Collective-Variable Tempering (vol 11, pg 1077, 2015).* Journal of Chemical Theory and

Computation, 2015. **11**(11): p. 5554-5554.

18. Fiorin, G., F. Marinelli, and J.D. Faraldo-Gomez, *Direct Derivation of Free Energies of Membrane Deformation and Other Solvent Density Variations From Enhanced Sampling Molecular Dynamics*. Journal of Computational Chemistry, 2020. **41**(5): p. 449-459.