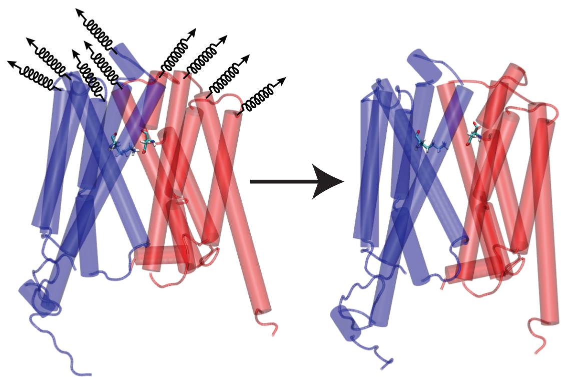
Department of Chemistry and Biochemistry
University of Arkansas, Fayetteville

Exploring Complex Conformational Transition Pathways



Tutorial by Mahmoud Moradi

September 18 2017

Contents

1 Introduction

The key objective of this tutorial is to walk you through various steps involved in the qualitative and quantitative study of complex protein conformational transitions using biased molecular dynamics (MD) simulations. The reader is encouraged to review the theoretical framework and practical applications of the methodology discussed here in the literature [1, 2, 3, 4]. Three specific stages are discussed in three subsequent Sections of the tutorial, namely, nonequilibrium pulling with steered MD (SMD) [5], transition path refinement with string method [6], and free energy calculations with umbrella sampling (US) [7]. All three methods have been discussed elsewhere in length both in the literature and within the NAMD tutorials. The aim of this tutorial, however, is to show (i) how these methods can be used with non-conventional collective variables such as *orientation quaternion* and (ii) how the three methods can be combined in order to study large-scale conformational changes of proteins, in particular those involving orientational changes of transmembrane helices (as illustrated in the example presented in the tutorial and discussed in more detail in the literature [1, 2, 3]).

Simulation Design and Convergence! This tutorial aims at demonstrating different stages of the methodology by setting up and performing relatively short simulations. However, a successful application of the methodology discussed here requires much longer simulations and/or larger number of replicas than those used in this tutorial. Although the convergence criteria and good practices in the design of simulations are discussed to some extent, they are not strictly followed in the tutorial to allow for its completion in a timely manner.

1.1 Methodology

The underlying assumption of many enhanced sampling techniques is that the dynamics of biomolecules (described by $\mathbf{x}^t$, the trajectory of the atomic coordinates) can be *effectively* mapped to a Brownian motion in a low-dimensional collective variable space [8, 9, 10, 11, 12, 4] (described by $\boldsymbol{\xi}^t$, the trajectory of the collective variables). The $\boldsymbol{\xi}^t$ represents the trajectory of a reduced system, where the other orthogonal degrees of freedom (with respect to $\boldsymbol{\xi}(\mathbf{x})$) are effectively integrated out. By adding a harmonic potential term centered at a given point (fixed or varied) in the collective variable space, one may sample the high energy states that are otherwise inaccessible or undersampled in an unbiased simulation. Examples of such methods include US [13, 14] as well as nonequilibrium pulling techniques such as SMD [5] and targeted MD [15]. The harmonic potential can be written as $U_c(\boldsymbol{\xi}) = \frac{1}{2}k\, d(\boldsymbol{\xi}, \boldsymbol{\xi}_c)$, where k and $\boldsymbol{\xi}_c$ are the force constant and the harmonic center, respectively, and $d(\boldsymbol{\xi}, \boldsymbol{\xi}_c)$ is the distance between $\boldsymbol{\xi}$ and $\boldsymbol{\xi}_c$. Once the collective variable space (or part of it) is sampled with biased simulations, a reweighting scheme can be used to reconstruct the unbiased thermodynamical properties of the system such as the free energy. The free energy or the potential of mean force (PMF) in terms of the collective variable (that is the effective potential of the reduced system) is defined as $G(\boldsymbol{\xi}_c) = -k_B T \log(\langle \delta(\boldsymbol{\xi}^t - \boldsymbol{\xi}_c) \rangle)$, where the ensemble average is over unbiased trajectories $\boldsymbol{\xi}^t$. The definition of collective variable has a direct impact on the sampling efficiency and the meaningfulness of the free energy maps.

- **Nonequilibrium pulling.** Nonequilibrium pulling simulation schemes such as SMD [5] often employ a simple intuitive collective variable (e.g., an interdomain molecular distance [5]) to generate an ensemble of trajectories, which is then used to estimate the free energies using nonequilibrium work relations [16, 17, 18, 19] (e.g., Jarzynski equality [16]). The same collective variable can be used for US simulations, where multiple copies of the system are biased at different points along the collective variable space, followed by free energy estimation using reweighting techniques (e.g., weighted histogram analysis method or WHAM [20]). Both methods could be prohibitively expensive to converge if possible at all for the study of realistic biomolecular processes. The nonequilibrium pulling simulations are generally less costly computationally on an individual basis, although many iterations may be necessary for free energy estimation. Alternatively, the pulling simulations can be used to guide the design of practical biasing protocols in a qualitative (or semi-quantitative) manner [1, 2, 3]. This approach is inherently system-specific; one should generally design such collective variables based on experimental data (e.g., variations in available crystal structures or distance distributions obtained from experiment) and biological insight (e.g., the alternating access principle in transporters [21]) regarding the nature of the conformational transition under study.

- **Path Refinement.** The complexity of free energy landscapes associated with proteins makes them virtually impossible to characterize when relying on conventional one-dimensional reaction coordinates. **String method with swarms of trajectories (SMwST)** [6] is one of the various types of string method [22, 10], a class of path-finding algorithms for multi-dimensional collective variable or atomic coordinate spaces. Using the scripting capability in NAMD [23, 24], a parallel version of SMwST is implemented [3], which exploits the NAMD's multiple-replica capability. NAMD has been enhanced to support extremely scalable multiple-replica simulations on massively parallelized computers [24]. Multiple concurrent NAMD instances are launched with internal partitions of Charm++ and located continuously within a single communication world. Messages between NAMD instances are passed by low-level point-to-point communication functions, which are accessible through NAMD's TCL scripting interface.

  In our fully parallel version of SMwST [24, 3], for a string of N *images*, each represented by M *copies*, $N \times M$ parallel *replicas* of the same system are simulated. Every iteration consists of (i) a short restrained simulation (for *equilibration*) to keep the system at the current *image center* in the collective variable space and (ii) a short free/released simulation (for *drifting*) to allow for the evolution of the string along its free energy gradient. Each iteration is then followed by a procedure to update the image centers (including *smoothing* and *reparametrization*). In our novel hybrid version of SMwST, every image is represented by M copies (where M could be 1); however, the image centers are updated every S iteration. This allows for using less number of replicas (e.g., only N replicas for $M = 1$), without losing the accuracy of the SMwST method.

- **Free energy Calculation.** Once an approximate minimum free energy path (MFEP) is obtained using the SMwST algorithm, one can calculate the free energy along the MFEP, which is the most relevant transition pathway. Conformational free energy calculations have proved difficult, particularly due to the limitations associated with conventional US [13] in sampling orthogonal degrees of freedom [25, 26]. US is significantly more successful when used in conjunction with a replica-exchange scheme [27], as in window/replica-exchange US [7, 28, 29], due to the mixing and diffusion of replicas along the sampled pathway, absent in conventional US simulations. Bias-exchange umbrella sampling (BEUS) [1, 2, 3] is a replica-exchange US scheme for free energy calculations along arbitrary curves in high-dimensional collective variable spaces.

  The BEUS scheme is a multiple-replica algorithm similar to the SMwST scheme, which uses the same NAMD TCL scripting capabilities discussed above to allow for coupling between independent NAMD simulations. In the replica-exchange US, the windows are represented by parallel replicas of the same simulations only with different bias centers, that are allowed to exchange their restraining centers according to a MetropolisâĂŞHastings type Markov chain Monte Carlo algorithm. In our novel implementation, however, each window is not necessarily represented only by one replica. To be consistent with the SMwST scheme, we use the term *image* instead of *window* (or use them interchangeably) in this tutorial. The BEUS scheme can use N images and M copies of each image, requiring $N \times M$ replicas in total (similar to the SMwST scheme). Given the availability of the suitable computer architecture, the multiple-copy scheme, where each image/window is represented with more than one replica, is advantageous for various reasons including: (i) a more reliable error estimate, (ii) a better exchange rate, and (iii) a better sampling overlap.

  Once the BEUS simulations converge [2], a non-parametric reweighting scheme [30, 31, 32, 33, 2] can be employed to not only reconstruct the free energy profiles but also estimate various other ensemble averages along the MFEP pathways. When multidimensional collective variables are used (as in the examples discussed in this tutorial), parametric reweighting methods such as conventional WHAM [20] are not practical; It is important to use a non-parametric reweighting scheme, where it is not necessary to explicitly build histograms in the collective variable space.

SMwST/BEUS Implementations! The SMwST implementation discussed in this tutorial is not the same as that discussed in String method with swarms of trajectories: A tutorial for free-energy calculations along a minimum-action path. The two algorithms are quite similar and both use NAMD's TCL scripting capabilities that give access to low-level communication functions between NAMD instances. However, the two implementations were developed independently and have some differences. Similarly, the BEUS scheme discussed here is similar but not the same as that discussed in the tutorial: One-dimensional replica-exchange umbrella sampling. The BEUS implementation discussed here is partly based on the TCL code used in the aforementioned tutorial; however, various capabilities have been added to our implementation, some of which are discussed in this tutorial and some will be discussed elsewhere. The above tutorial for one-dimensional replica-exchange US is highly recommended for getting familiar with a simplified version of the BEUS scheme. Our SMwST and BEUS implementations can be found in the supporting files of this tutorial in the codes directory.

1.2 Orientation Quaternion

Within the context of many transmembrane (TM) proteins, the orientations of TM helices are highly relevant to the large-scale conformational changes that are functionally important. TM helical orientations have proved [1, 2, 3] to be efficient collective variables describing the effective protein conformational landscapes of TM proteins. The orientation of helices can be described using the orientation quaternion formalism [34, 35, 36, 1]. The optimal rotation to superimpose one set of coordinates on another can be described by a unit quaternion, i.e., orientation quaternion $Q = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{u})$, in which θ and $\hat{u}$ (a unit vector) are the optimum angle and axis of rotation, respectively. The geodesic distance between orientation quaternions P and Q can be estimated by $\cos^{-1}(|P \cdot Q|)$, in which $P \cdot Q$ is their inner product [36].

It has been shown that the geodesic distance is a more appropriate measure to represent the distance in a harmonic potential [4]. Orientation quaternion is thus a mathematically rigorous choice for representing the orientation of a domain as a collective variable in a biased simulation. An orientation quaternion can thus be used as a collective variable to induce a rotational change on a given domain (e.g., a helix) or simply restrain its orientation. For a transmembrane protein consisting of N TM helices, one may use a set of N orientation quaternions to describe the entire orientational space of the TM helices. In the pulling simulations, however, the pulling protocol does not have to involve all N TM helices.

Required Software!

- VMD 1.9 or later http://ks.uiuc.edu/Research/vmd
- NAMD 2.10 or later http://ks.uiuc.edu/Research/namd You will need a Charm++ build NAMD with a low-level run-time system (LRTS) including netlrts, verbs, and mpi.
- Gnuplot for plotting, if you want to use the provided plotting scripts http://www.gnuplot.info

To complete this tutorial a working knowledge of NAMD software is required. In addition, the `colvars` module in NAMD is extensively used. The `colvars` module [36] is documented in the NAMD user's guide. This tutorial assumes that you are working on a Linux machine and are familiar with Linux command line interface, and shell scripting. Several Bash shell scripts are provided that may require alteration in different situations. Example VMD, awk, and gnuplot scripts are provided as well for the analysis and plotting of the results. For estimating free energies based on the US simulations, a c++ code is provided. The compiled binaries should work on most Linux machines. If not, you should recompile.

Mac Users. For Mac users, the majority of the scripts can be used with no change.

- Please make sure you have made a command-line link to VMD. Otherwise, replace `vmd` in any relevant scripts by the actual path to VMD.
- Gnuplot can be installed on Mac http://www.gnuplot.info/download.html if you want to use the plotting scripts provided (named `fig.sh` in various directories).
- If you do not have the `epstopdf` program, which is used at the end of all plotting scripts to generate pdf files, you can comment out the `epstopdf`-containing lines at the end of each `fig.sh` script. The ps files will be generated by gnuplot instead.
- The `npwham` code for extracting free energies from BEUS simulations is provided in `codes/npwham.0`. Please remove the linux-compatible `npwham` binaries and recompile:

```
cd codes/npwham.0
rm npwham
make
./npwham -h
```

  The last line should result in:

```
Non-Parametric Weighted Histogram Analysis Method
(npwham)
...
```

2 Nonequilibrium Pulling Simulations

The reader of this tutorial is assumed to be already familiar with steered molecular dynamics (SMD) simulations. In SMD simulations, time-dependent external forces are applied to induce a change (e.g., a slow conformational change) that is otherwise unlikely to be observed in a regular unbiased MD of similar simulation time. SMD has been successfully employed in a wide range of biological systems. Several tutorials already discuss the SMD method using different tools within NAMD such as NAMD Tutorial, QwikMD Tutorial, Stretching Deca-alanine, and Methods for calculating Potentials of Mean Force.

The most flexible implementation of SMD method in NAMD is provided by the *colvars* module [36], which allows the application of various *collective variables* (or colvars) for steering/pulling the system. Here to distinguish between the distance based steering (available through `smd` tool in NAMD) and the more general form of steering (available through the `harmonic` block in colvars module), we use "nonequilibrium pulling simulations"to refer to the type of simulations performed in this Section. Nonequilibrium pulling simulations are used in this tutorial to generate an initial conformational transition pathway for further refinement and eventually along-the-path free energy calculations as discussed in the next Sections.

2.1 Initial preparations

Before starting any nonequilibrium pulling simulation, one needs to prepare a fully equilibrated model of the system of interest in a known state, e.g., starting from a crystal structure. The pulling simulations can then be used to induce a particular transition to generate a conformational transition pathway. It is important to note that the pulling simulations do not necessarily require an atomic model of the end state. However, the pulling protocols must be designed based on some knowledge of the structure of the end state and the transition that leads to it. There is an empirical element to this approach, which requires several, if not many, attempts to result in a successful transition to a stable state. In each attempt, one may completely change the collective variables used for steering the system or simply change some of the parameters involved in the definition of the collective variables or the pulling protocol. The successful use of methodology, however, relies on both initial intuition/knowledge of the transition and the amount of work put on exploring potential pulling protocols.

2.1.1 Preparing an initial equilibrated model of protein

In this tutorial, we assume that the reader is already familiar with regular unbiased MD simulations and has generated an equilibrated model of protein. It is important, for the successful application of the methodology used here, that the initial conformation is a stable model. The model needs to be equilibrated until the system is stable and a convergence is reached. If the starting point is not a stable conformation (e.g., the initial equilibration is not performed sufficiently long), the pulling simulations will not be reliable and the remainder of the simulations in the next Sections will take much longer to converge. Therefore, it is important to perform a relatively long simulation on the initial model in order to ensure the initial conformation of the pulling simulations is well equilibrated.

In this tutorial, we use membrane transporter GlpT (PDB: 1PW4) as a test case. You can download the crystal structure model from the Protein Data Bank and follow the steps in Membrane Proteins Tutorial to generate an equilibrated model of *apo* GlpT in a lipid bilayer environment. Other tools such as QwikMD (see QwikMD Tutorial) and CHARMM-GUI (see http://www.charmm-gui.org/ can be used for this stage of simulations. We have already generated an equilibrated model of GlpT embedded in a POPE lipid bilayer and water box (with 150 mM NaCl) that can be found in the supporting files of this tutorial. The provided conformation is the last frame of a an unbiased equilibrium MD of simulation time 50 ns. The equilibration of a protein, however, could potentially take longer.

1 Locate the `system` folder in the supporting files, which includes:

- `glpt.psf:` protein structure file (psf) of the system.
- `eq.coor, eq.vel, & eq.xsc:` coordinates, velocities, and periodic cell information from the last frame of an equilibrium simulation.
- `glpt.inp:` namd input parameters common to all simulations, which will be *sourced* by all namd configuration files throughout this tutorial.
- `par_all36_lipid.prm par_all36_prot.prm par_water_ions.inp:` CHARMM parameter files.
- `do.sh:` a script to generate a pdb file (eq.pdb) from eq.coor and glpt.psf files.

Alternating Access Mechanism. To actively transport their substrates, membrane transporters are known to undergo large-scale conformational changes to alternate between two distinct functional states including the inward-facing (IF) and outward-facing (OF) conformations. In many cases, the available crystal structures are either in the IF or in the OF state, and the other state is unknown. A key task regarding the structural biology of membrane transporters is to structurally characterize the unknown states and elucidate the conformational changes associated with the alternating access mechanism. GlpT's only crystal structure is in the IF state. Here we set out to generate an OF model for GlpT and characterize its conformational transition pathway between the IF and OF states both qualitatively and quantitatively. The short simulations in this tutorial, however, are not enough to fully complete this task. The reader is encouraged to read Ref. [3] for a more complete story.

2.1.2 Orientation-based nonequilibrium pulling

While the colvars module in NAMD provides a variety of different collective variables, the collective variables used in this tutorial are all based on the "orientation quaternion"formalism. Several collective variables belong to this class including: `orientation, orientationAngle, orientationProj, spinAngle,` and `tilt`. In addition, the `rmsd` colvar implemented in the colvars module uses the orientation quaternion formalism to superimpose the coordinates onto a set of reference coordinates, prior to *best-fit* rmsd calculation. The `orientation` collective variable is a unit quaternion represented by four numbers (q_0, q_1, q_2, q_3), where $\sum_i q_i^2 = 1$. The orientation quaternion of a given set of coordinates $\{\mathbf{x_i}\}$ with respect to a reference set of coordinates $\{\mathbf{x_i^{(ref)}}\}$ represents the optimum rotation to superimpose $\{\mathbf{x_i}\}$ onto $\{\mathbf{x_i^{(ref)}}\}$. In this case the quaternion can be represented by $(\cos(\theta/2), \sin(\theta/2)\hat{\mathbf{u}})$, where θ is the optimum rotation angle and $\hat{\mathbf{u}}$ is a unit vector associated with the optimum axis of rotation.

Collective variable `orientation` is the most general collective variable among the orientation-based collective variables listed above. We will discuss some of its applications in this tutorial and use it as the primary colvar in most of the simulations. However, since `orientation` is a four-dimensional object, it is convenient to be replaced by a one-dimensional alternative in some of the applications. In this tutorial, first we start with the `spinAngle`, which has a very similar application to `orientation` and is easier to use. To use `spinAngle`, one should specify a particular axis of rotation ($\hat{\mathbf{u}}$) in the definition of the colvar. The angle of rotation can then be varied by steering, e.g., from $\theta(0)$ to $\theta(T)$, during a MD run of simulation time T. This is similar to using the `orientation` as the colvar and vary it from $(\cos(\theta(0)/2), \sin(\theta(0)/2)\hat{\mathbf{u}})$ to $(\cos(\theta(T)/2), \sin(\theta(T)/2)\hat{\mathbf{u}})$. There are some differences between the two approaches even in the context of the pulling simulations but more importantly in the SMwST simulations, which is discussed in the next Section. We note that other one-dimensional derivatives of `orientation` such as `tilt` and `orientationAngle` are also useful in pulling and BEUS simulations, depending on the context of the application.

2.1.3 Determining the pulling protocol

2 Load the initial model of GlpT in VMD.

NewCartoon representation of `eq.pdb` is shown in Fig. 1. You can load the equilibrated model of GlpT in VMD, using `eq.coor` and `glpt.psf` files. You should generate a pdb file, named `eq.pdb`, and place it in the `system` directory to be used later by the simulations as the GlpT reference structure. You can also generate `eq.pdb` from `eq.coor` and `glpt.psf` by using script `do.sh` provided in the `system` directory. `eq.coor` is the final coordinate file from the equilibrium simulations of GlpT.

3 Identify the molecular domains to be rotated in order to generate the OF state.

Roughly speaking, the N and C terminal halves of protein (blue and red domains, respectively as shown in Fig. 1) must rotate in opposite directions around an axis that sits at the interface of the two domains and is parallel to the membrane plane. A rigid body motion can be imposed on the two domains in opposite directions using the `harmonic` block in the colvars module by varying the `axisAngle` by time in opposite directions. This pulling protocol could be an initial trial protocol to generate an OF model.

4 Identify the axes of rotation for the molecular domains.

The tcl script `0-smd/0-spin/axis.tcl` finds the first principal axis (PA1 or yaw axis) of the helical parts of the protein, which approximately represents a common axis of rotation for the N and C domains, resulting in the opening of the extracellular/periplasmic side and closing of the intracellular/cytoplasmic side (forming the OF state):

0-smd/0-spin/axis.tcl:
```
mol new ../../system/eq.pdb
set pro [atomselect top "protein and helix"]
# axis of rotation = first principal axis
set u [lindex [measure inertia $pro] 1 0]
```

In order to visually see the PA1 axis (and other principal axes) in VMD, you can use the vmd_draw_arrow procedure:

0-smd/0-spin/draw.tcl:
```
proc vmd_draw_arrow {mol start end} {
    set middle [vecadd $start [vecscale 0.9 [vecsub $end $start]]]
    graphics $mol cylinder $start $middle radius 1
    graphics $mol cone $middle $end radius 2
}
...
```

Once you load the `eq.pdb` file in VMD and define the procedure above, you can draw the first principal axis using the following script:

0-smd/0-spin/draw.tcl:
```
...
set pro [atomselect top "protein and helix"]
set center [measure center $pro]
set u1 [lindex [measure inertia $pro] 1 0]
draw color yellow
vmd_draw_arrow top $center [vecadd [vecscale 40 $u1] $center]
```

Generalization! The definition of your colvar often requires examining various ideas since there is no general rule for designing collective variables. The way we define `spinAngle` for GlpT is not necessarily applicable to other proteins. Fortunately, the pulling simulations are relatively inexpensive computationally, allowing one to examine various colvar definitions to discover an appropriate colvar for the particular system and conformational transition under study.

5 Identify the angle of rotation.

At this point, we assume that we roughly need to rotate each N/C domain for about 20 degrees (in the opposite directions) to generate an OF state. By inspecting the orientation of each domain with respect to the axis of rotation (in VMD), it is clear that the angle must be positive for the N domain and negative for the C domain (Fig. 1). The exact amount of the total angle of rotation is not as crucial as the choice of the axis. If we start with an angle that is too small, we can extend the simulations and target a larger angle. If we start with an angle that is too large, we can stop the simulation once it reaches the desired opening (or discard the rest of the simulation).

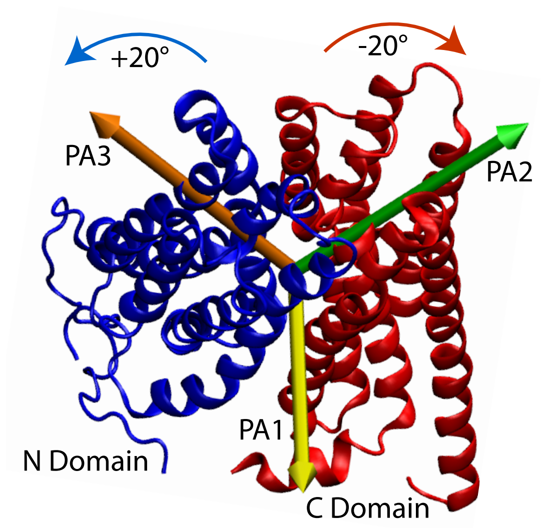


Figure 1: NewCartoon representation of equilibrated GlpT (`eq.pdb` file). The coloring method is `SegName`. The blue and red represent the N and C domains, respectively. The principal axes are calculated using `measure inertia` command in VMD using the atom selection `helix`. The first principal axis PA1 (colored yellow) approximates an axis of rotation for the N and C domains that may result in the opening of the periplasmic side (top) and closing of the cytoplasmic side (bottom), and generating an OF conformation. PA2 and PA3 (colored green and orange) are the other principal axes of the protein.

6 Determine the simulation time and force constant.

We start with a relatively short simulation time (i.e., 4 ns). At this point our simulations are more of a "shot in the dark"and require optimization (both in terms of the definition of collective variables and in terms of pulling protocol). Therefore, long simulations are not desired, although are eventually needed for more accurate results. For force constant, there are two criteria: (1) the force constant should be large enough such that the protein does not stay behind the *schedule* considerably, and (2) the force constant must be small enough such that the protein does not undergo undesired conformational changes such as unwinding of helices. Here we pick $3kcal/(mol(deg^2))$ as the force constant for the spin angles associated with the N and C terminal domains.

2.2 Generating the colvars configuration file

1 Generate a file to include all the information regarding the collective variables and pulling protocols.

`0-smd/0-spin/colvars.conf` is available in the supporting files that contains the pulling protocol described above.

2 Define the collective variables.

Two colvars are defined in `0-smd/0-spin/colvars.conf`. All C^α atoms associated with the transmembrane helices are included. `sa1_6` is the `spinAngle` of the N domain containing helices 1 to 6 and `sa7_12` is the `spinAngle` of the C domain containing helices 7 to 12. The reference structure is the pdb file associated with the initial conformation that you have already generated (`eq.pdb`). A common `axis` is used for both collective variables, as described above.

0-smd/0-spin/colvars.conf:
```
colvar {
   name sa1_6
   spinAngle {
      atoms {
         psfSegID PROA PROA PROA PROA PROA PROA
         atomNameResidueRange { CA 20-54 }
         atomNameResidueRange { CA 66-86 }
          atomNameResidueRange { CA 94-112 }
         atomNameResidueRange { CA 122-148 }
         atomNameResidueRange { CA 158-183 }
         atomNameResidueRange { CA 190-207 }
      }
      refPositionsFile ../../system/eq.pdb
      axis ( -0.36560463905 , 0.925897896289 , 0.09511154890060 )
   }
}
colvar {
   name sa7_12
   spinAngle {
      atoms {
         psfSegID PROB PROB PROB PROB PROB PROB
         atomNameResidueRange { CA 253-282 }
         atomNameResidueRange { CA 293-316 }
         atomNameResidueRange { CA 322-341 }
         atomNameResidueRange { CA 351-374 }
         atomNameResidueRange { CA 382-409 }
         atomNameResidueRange { CA 415-448 }
      }
      refPositionsFile ../../system/eq.pdb
      axis ( -0.36560463905 , 0.925897896289 , 0.09511154890060 )
   }
}
...
```

There are other ways to specify the atoms in the `atoms` block as discussed in *NAMD manual* (see Selecting atoms for colvars: defining atom groups). For instance, the *beta* or *occupancy* columns of a pdb file can be changed to a particular value for each domain to be identified using `atomsFile`, `atomsCol`, and `atomsColValue` in `atoms` block of the `colvar` definition.

3 Specify the pulling parameters.

Various biasing methods are available in the colvars module, among which `harmonic` is the method used in this tutorial for all the algorithms. For nonequilibrium pulling (or SMD), all or some of the collective variables defined in the colvars configuration file can be used for steering the system in one or multiple `harmonic` blocks. Here, both collective variables defined above are used and both follow a similar protocol, thus we use one `harmonic` block to specify the pulling parameters for both collective variables. The `center` is often chosen to be the actual collective variable value of the initial conformation. Since the reference structure here is the same as the initial conformation, the `spinAngle` is zero for both N and C domains. The rest of the parameters are already discussed above. The final line indicates the nonequilibrium work will be calculated and reported during the pulling process.

0-smd/0-spin/colvars.conf:
```
...
harmonic {
   name harm
   colvars { sa1_6 sa7_12 }
   centers { 0 0 }
   targetCenters { 20 -20 }
   forceConstant 3
   targetNumSteps 2000000
   outputAccumulatedWork on
}
```

2.3 NAMD simulation with orientation-based pulling

1 Generate the NAMD configuration file

The NAMD configuration file, `0-smd/0-spin/smd.conf`, can be found in the supporting files. The following commands activate the colvars module and call the colvars configuration file discussed above:

0-smd/0-spin/smd.conf:
```
...
colvars on
colvarsConfig colvars.conf
...
```

2 To perform the simulation, use NAMD 2.10 or above.

Note that the configuration file `0-smd/0-spin/smd.conf` contains the direction to a colvars configuration file, `colvars.conf`, and several files in the `system` directory. It is assumed that your working directory is `0-smd/0-spin`. The output files will be saved in `0-smd/0-spin/output`. To speed up the process we have already generated some sample output files that can be found in `0-smd/0-spin/sample-output`. These files include the colvars trajectory file (`smd.colvars.traj`), which contains the trajectory of the colvar and the accumulated work, and the final coordinates (`smd.coor`) to investigate whether the pulling protocol has been successful.

Restarting the pulling simulations:! If the simulation stops for any reason (e.g., reaching the specified Walltime), the simulation can be continued using the saved restart files, similar to a regular NAMD simulation. The only addition to the regular changes is that the colvars restart file (in this case `0-smd/0-spin/output/smd.restart.colvars.state`) must be specified in the NAMD configuration file. The keyword `colvarsInput` is used for this reason. An example of NAMD configuration file for restarting the interrupted simulations is provided in `0-smd/0-spin/smd.1.conf`.

3 Determine whether or not the pulling protocol has been successful.

Two trajectory files should be monitored, both while the simulations are running and at the end, to determine whether or not the pulling protocol has been successful:

- Colvars trajectory file. `0-smd/0-spin/output/smd.colavars.traj` is the colvars trajectory file. If you have not run your own simulations, you can find a sample file in `0-smd/0-spin/sample-output`. The time series of the two colvars, along with the accumulated work is reported every 500 steps. If successful, the angles should change from 0 to 20 and -20 (for N and C domain, respectively) almost linearly by time. It is acceptable if there is a slight delay in the response of the system (Fig. 2, top left) but if the delay significantly increases over the course of the simulation, the protocol should be modified. The first change to consider is the force constant. An increase in the simulation time could also help with this issue. In addition to the collective variable values, the accumulated work provides some information on the goodness of protocol (Fig. 2, top right). Large work values are typically indicative of bad protocols. This could simply be due to the short simulation time, large force constants, and finally the poor choice of collective variables. The work profiles can theoretically be used to estimate the free energy differences of the end states as well as the PMF [16, 17, 18, 19]. However, this often requires extremely large number of iterations to converge.

- Coordinate trajectory file. `0-smd/0-spin/output/smd.dcd` can be visualized (e.g., see Fig. 2, bottom left) and analyzed (e.g., see Fig. 2,bottom right) with VMD to ensure the pulling protocol has been successful (assuming the colvars trajectory file does not show any issues) or to detect the source of the problem. For instance, it is possible that a protocol seems successful from the colvars trajectory file but the pulling protocol results in undesirable artifacts such as unwinding of helices. The high amount of accumulated work (e.g., several hundreds of kcal/mol) may already indicate that an energetically unfavorable event has taken place.

4 Monitor the colvar trajectory and accumulated work.

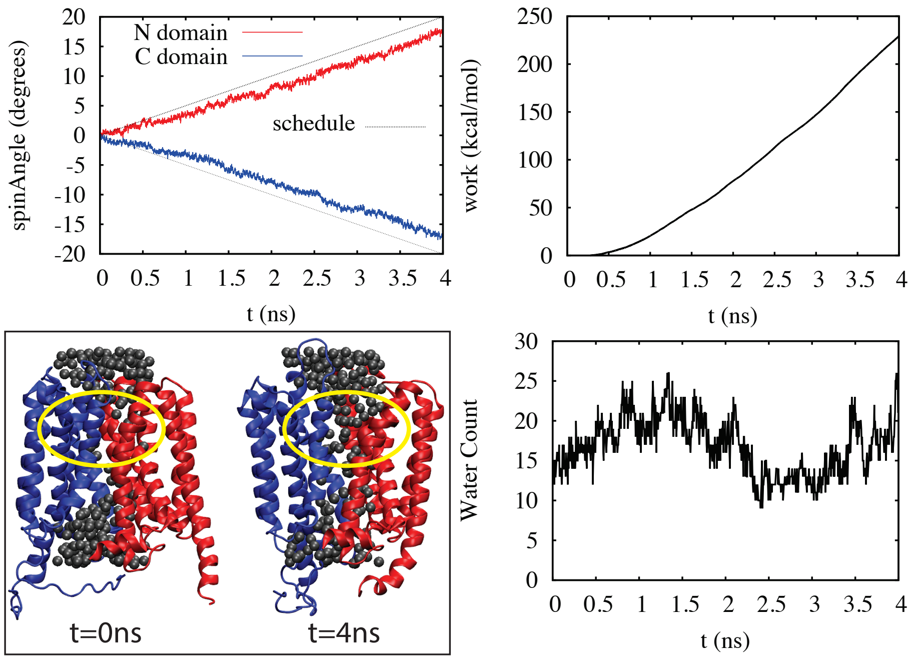


Figure 2: Results of pulling simulations using `spinAngle` (`0-smd/0-spin`). The spin angles more or less follow the schedule (top left). The nonequilibrium work reported (top right) is relatively high, which is potentially due to the short simulation time (i.e., fast pulling). There is no unusual conformational change (e.g., unwinding of helices) observed (bottom left). There is an increase in the number of water molecules in the periplasmic lumen due to the opening of the periplasmic side (bottom left, yellow mark). The quantification of the water count as time series is also shown (bottom right).

As soon as the simulation start, it is useful to monitor the colvar values reported in the colvar trajectory file. We have provided this file in `0-smd/0-spin/sample-output` (`smd.colvars.traj`). The time, the two colvar values and the accumulated work are reported in this file.

5 Visualize/analyze the simulation trajectory with VMD.

If you decide not to run the pulling simulations entirely, you can compare the initial and final conformations provided in the supporting files (`system/eq.pdb` and `0-smd/0-spin/sample-output/smd.coor`, respectively). We have also provided an example VMD script (`0-smd/0-spin/sample-output/analyze.tcl`) to extract the minimum donor-acceptor K46-D274 salt-bridge distance (which is known to be functionally important) and the number of water molecules in the periplasmic side of the lumen from the simulation trajectory file. We have not provided a sample dcd file but the output of the above analysis can be found in `0-smd/0-spin/sample-output/analyze.txt`.

6 Plot the results.

We have provided a gnuplot script (in `0-smd/0-smd/sample-output/fig.sh`) that uses both the colvars trajectory file and the `analyze.txt` file to plot different quantities. You may move the scripts such as this to your `output` directory to plot your own simulation results.

2.4 Working with the `orientation` colvar

While the `spinAngle` collective variable provides a good alternative to `orientation` in the GlpT case, we will not use it in the rest of the simulations. This is primarily due to the fact that `orientation` is more appropriate for the simulations of the next Section. We could have used the `spinAngle` in all of our pulling simulations; however, for the SMwST simulations, using the one-dimensional alternatives of `orientation` is not recommended as they do not allow for a full description of the orientation of a domain, whose evolution in a SMwST simulation is crucial for having the potential of capturing the MFEP. Therefore, we switch to `orientation` in order to keep the simulations consistent, although the consistency is not necessary in this case. You may choose to perform all your pulling simulations with one colvar and use another for your SMwST and BEUS simulations.

Here you will perform a simulation similar to that above only with a difference in the type of colvar used, i.e., `orientation` instead of `spinAngle`.

1 Generate colvars configuration file

`0-smd/1-orientation/colvars.conf` is available in the supporting files that contains the modified pulling protocol with the use of `orientation` colvar.

2 Define the collective variables.

Two `orientation` colvars are defined. `q1_6` and `q7_12` are quite similar to `sa1_6` and `sa7_12`. There is no axis of rotation needed in the definition of `orientation` collective variables.

0-smd/1-orientation/colvars.conf:
```
colvar {
   name q1_6
   orientation {
      atoms {
         psfSegID PROA PROA PROA PROA PROA PROA
         atomNameResidueRange { CA 20-54 }
         atomNameResidueRange { CA 66-86 }
          atomNameResidueRange { CA 94-112 }
         atomNameResidueRange { CA 122-148 }
         atomNameResidueRange { CA 158-183 }
         atomNameResidueRange { CA 190-207 }
      }
      refPositionsFile ../../system/eq.pdb
   }
}
colvar {
   name q7_12
   orientation {
      atoms {
         psfSegID PROB PROB PROB PROB PROB PROB
         atomNameResidueRange { CA 253-282 }
         atomNameResidueRange { CA 293-316 }
         atomNameResidueRange { CA 322-341 }
         atomNameResidueRange { CA 351-374 }
         atomNameResidueRange { CA 382-409 }
         atomNameResidueRange { CA 415-448 }
      }
      refPositionsFile ../../system/eq.pdb
   }
}
...
```

3 Specify the pulling parameters.

The initial values of `q1_6` and `q7_12` indicate no change from the reference structure as in the case of `sa1_6` and `sa7_12`. This is represented by orientation quaternion values $(1, 0, 0, 0)$. The target centers are determined from the formula $(\cos(\theta/2), \sin(\theta/2)\hat{\mathbf{u}})$, where θ is 20 and -20 for `q1_6` and `q7_12` and $\hat{\mathbf{u}}$ is the same vector used in the definition of `sa1_6` and `sa7_12`. Tcl script `0-smd/1-orientation/orientation.tcl` uses this formula to determine the targetCenters for `q1_6` and `q7_12`. The force constant is in the kcal/(mol$\times$rad^2) unit. Every 1 kcal/(mol$\times$deg^2) is almost 3,286 kcal/(mol$\times$rad^2) since 1 rad is almost $180/\pi$ degrees. Therefore, 3 kcal/(mol$\times$deg^2) used for the `spinAngle` protocol, is almost 10,000 kcal/(mol$\times$rad^2), which is used below:

0-smd/1-orientation/colvars.conf:

```
...
harmonic {
   name harm
   colvars { q1_6 q7_12 }
   centers { ( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 ) }
   targetCenters { ( 0.9848 , -0.0634 , 0.1607 , 0.0165 )
                  ( 0.9848 , 0.0634 , -0.1607 , -0.0165 ) }
   forceConstant 10000
   targetNumSteps 2000000
   outputAccumulatedWork on
}
```

We note that due to some fundamental differences between `orientation` and `spinAngle`, we cannot make an apple to apple comparison between their force constants as made above. The biasing potential used for `orientation` and the one used for `spinAngle` are both harmonic but with different functional forms due to differences of the two colvar types. It turns out 10,000 kcal/mol is too small to induce our desired transition with the `orientation` colvar. Figure 3 shows the orientation angle during the simulation, which stays behind the schedule considerably. We note that the `smd.colvars.traj` file (provided in `0-smd/1-orientation/sample-output`) only has the `orientation` trajectories (and accumulated work). To get the orientation angle, one can use the twice cosine of the first number in the orientation quaternion (q_0, q_1, q_2, q_3). The orientation angle will be $\theta = 2\cos^{-1}(q_0)$.

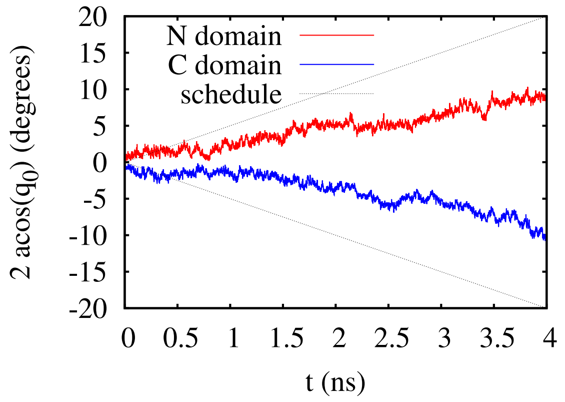


Figure 3: Results of pulling simulations using the `orientation` collective variable (see script `0-smd/1-orientation/sample-output/fig.sh`). The orientation angles stay behind the schedule due to the low force constant.

2.5 Optimizing the pulling protocol

Pulling simulations are relatively inexpensive and can be repeated several times with various definitions of collective variables and/or pulling parameters to achieve a reasonable initial transition pathway for more accurate calculations in the next stages. Here, we will not, however, try to optimize the protocol further. The final pulling simulation in `0-smd/2-long` is the same as `0-smd/1-orientation` with only one difference: the simulation time. You will see that by simply increasing the simulation time from 4 to 10 ns, the results improve significantly. Although the force constant of 10,000 kcal/mol was proven to be too small for the 4-ns pulling simulation, the increase in the simulation time could resolve the issue to a great extent without needing to resort to a higher force constant. Using longer simulations with lower force constants is more favorable than shorter simulations with higher force constant since the former stays closer to equilibrium.

Figure 4 shows the results of the our 10-ns pulling simulations. The force constant is still somewhat low and there is a delay in the response of the system to the scheduled change in the orientation. However, the transition takes place and the resulting conformation is an OF conformation. This is revealed by inspecting the resulting conformation suing VMD (the coordinate files are provided in `0-smd/2-long/sample-output`). The number of water molecules that make it to the periplasmic side of the lumen is considerably higher than that observed in shorter simulations (compare with Fig. 2). Interestingly, a functionally important salt bridge (K46-D274) breaks towards the end of the pulling simulation, which is known to stabilize the IF state. The breaking of this salt bridge was not observed in shorter simulations (compare with `0-smd/0-spin` and `0-smd/1-orientation`; this is evident from the `smd.coor` files provided). Note that in the simulations with `spinAngle`, the orientation of the N and C domains changed more than they did in these simulations. However, the simulation time was not long enough to allow for the breaking of the K46-D274 salt bridge or the entrance of more water molecules to the lumen.

The input files and sample output files are provided in `0-smd/2-long` folder. The dcd trajectory files of this simulation will be used to generate the initial conformations of the SMwST simulations in the next Section. A shortened version of the dcd file (including 20 frames) from this pulling simulation can be found in `1-smwst/0-parallel/extract/initial.dcd`.

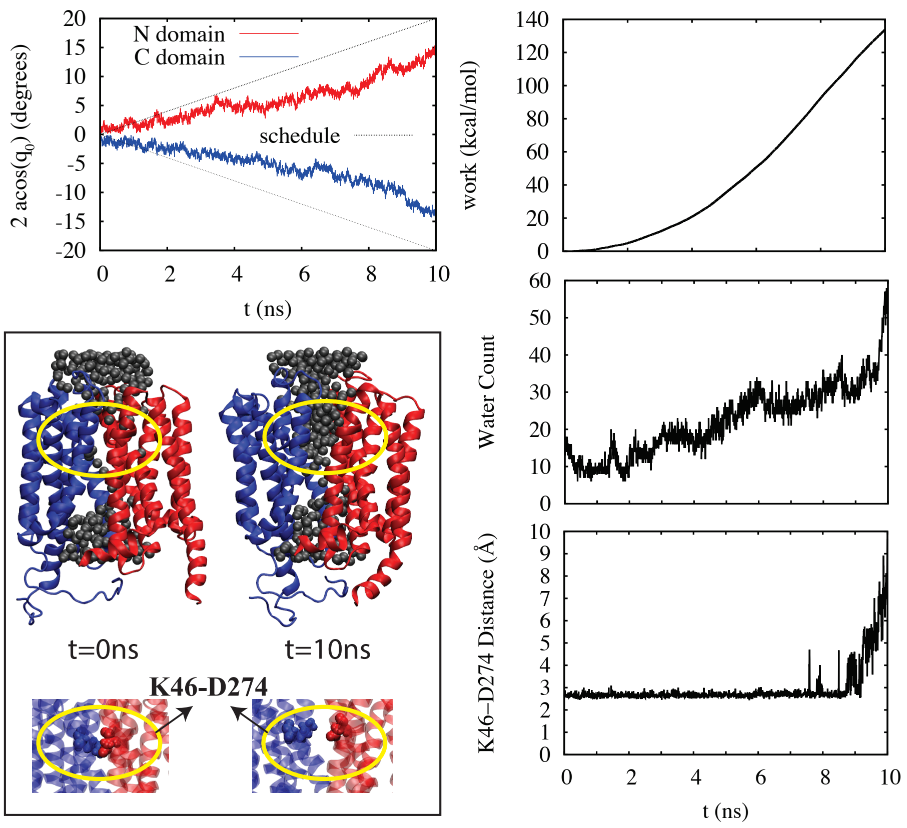


Figure 4: Results of pulling simulations using the `orientation` colvar (`0-smd/2-long`). The orientation angles stay behind the schedule to some extent due to the low force constant (top left) but the IF-to-OF transition takes place. The nonequilibrium work (top right) is considerably lower than that observed in the `spinAngle` simulation, primarily due to a longer simulation time, and an *effectively* lower force constant. There is an increase in the number of water molecules in the periplasmic lumen due to the opening of the periplasmic side (middle left, yellow mark), whose quantification (middle right) reveals a significant change in the number of water molecules towards the end of simulations, consistent with the breaking of the K46-D274 salt bridge (bottom).

Pulling Time! In order to avoid slow convergence of SMwST simulations, it is advantageous to run a longer pulling simulation and use it as a starting point for SMwST simulations. We recommend at least 100 ns of pulling. It is also important to equilibrate the resulting structure obtained from the pulling simulations before starting the SMwST simulations to ensure the transition has indeed resulted in a relatively stable conformation. The conformation of the system is expected to change during the equilibration since the structure is the result of a nonequilibrium simulation. However, if the desired conformation is not achieved (e.g., the periplasmic gate closes in the case of OF GlpT), it is somewhat pointless to perform SMwST simulations. SMwST simulations will not generate a stable OF state if the initial string does not contain a structure close enough to this state.

3 Refining the transition pathway

Nonequilibrium pulling simulations are relatively inexpensive and allow for exploring various collective variables. Transition pathways generated by these protocols, however, are *far from equilibrium.* The aim of this tutorial is to study the conformational transition pathways of proteins not only qualitatively (which was done in the previous Section) but also quantitatively (which will be done in the next Section using free energy calculations). String method with swarms of trajectories (SMwST) is a path optimization method that can be used to generate an approximate MFEP path from an initial pathway. Ideally, the initial pathway does not have to be close to the MFEP for SMwST algorithm to work; however, since the SMwST is quite costly, it is not practical to start with a path that is far from the MFEP in the collective variable space. We note that the nonequilibrium trajectories generated in the previous Section were primarily generated to be used as initial pathways for SMwST. Although nonequilibrium pathways are far from equilibrium, they are not necessarily far from the MFEP in the collective variable space. Therefore, if one uses a good pulling protocol and generates a pathway that is not far from the MFEP (although far from equilibrium), the SMwST algorithm can be used to relax and refine this pathway with a reasonable computational cost.

3.1 Extracting the initial pathway from a nonequilibrium trajectory

After performing the final nonequilibrium pulling simulation, you can generate the initial files to perform a SMwST simulation:

1 Generate a dcd file from your final nonequilibrium pulling simulation with the right number of frames.

You can use `catdcd` to combine the dcd files, e.g.:

1-smwst/0-parallel/extract/do.sh:
```
catdcd -o initial.dcd -stride 100 \
../../../smd/2-long/output/smd.dcd \
../../../smd/2-long/output/smd.1.dcd \
../../../smd/2-long/output/smd.2.dcd
...
```

The number of snapshots extracted will determine the number of images in the SMwST algorithm. Here, we pick 20 snapshots (as initial conformations of 20 images). These snapshots are selected using an appropriate stride to result in exactly 20 frames, while generating the combined dcd file (`initial.dcd`). This file is provided in the `1-smwst/0-parallel/extract` folder, so you do not have to run the entire `0-smd/2-long` simulations before you can continue the next steps of the tutorial.

2 Generate *.coor and *.xsc files from the combined dcd file.

The dcd file contains information both on the coordinates and the periodic cell. The VMD script `extract.tcl` is provided in the supporting files, which is executed using vmd:

1-smwst/0-parallel/extract/do.sh:
```
...
vmd -dispdev text -e extract.tcl
```

1-smwst/0-parallel/extract/extract.tcl:
```
package require pbctools
mol new ../../../system/glpt.psf
mol addfile initial.dcd type dcd first 0 last -1 step 1 waitfor all

set all [atomselect top all]
set num_frames [molinfo 0 get numframes]

for {set i 0} {$i < $num_frames} {incr i} {
    $all frame $i
    $all writenamdbin ../input/init.$i.coor
    pbc writexst ../input/init.$i.xsc -first $i -last $i
}

quit
```

The `*.vel` files cannot be extracted from the dcd file, but there is no need to have the velocity files to start the SMwST simulation. The velocities can be initiated randomly consistent with the temperature of the system.

3.2 Setting up the colvars configuration file

The colvars configuration file can contain one orientation quaternion per helix and all 12 helices can be included; therefore, we will have 12 orientations quaternions describing the orientations of all helices. Using the `fittingGroup` option (with `rotateReference on`) is recommended for the SMwST simulations since multiple conformations are in play and it is important to ensure the orientations of individual helices are calculated after aligning the protein with respect to the reference structure. Therefore, we use all 12 helices as the *fitting group* to align before calculating the orientation of each helix. The definition of the first `colvar` (orientation quaternion of helix 1), is given below:

1-smwst/0-parallel/colvars.conf:

```
 colvar {
    name q1
    orientation {
       atoms {
          psfSegID PROA
          atomNameResidueRange { CA 20-54 }
          centerReference on
          rotateReference on
          fittingGroup {
             psfSegID PROA PROA PROA PROA PROA PROA PROB PROB PROB
PROB PROB PROB
             atomNameResidueRange { CA 20-54 }
             atomNameResidueRange { CA 66-86 }
             atomNameResidueRange { CA 94-112 }
             atomNameResidueRange { CA 122-148 }
             atomNameResidueRange { CA 158-183 }
             atomNameResidueRange { CA 190-207 }
             atomNameResidueRange { CA 253-282 }
             atomNameResidueRange { CA 293-316 }
             atomNameResidueRange { CA 322-341 }
             atomNameResidueRange { CA 351-374 }
             atomNameResidueRange { CA 382-409 }
             atomNameResidueRange { CA 415-448 }
          }
          refPositionsFile ../../system/eq.pdb
          enableFitGradients off
       }
       refPositionsFile ../../system/eq.pdb
    }
 }
 ...
```

Note that the `enableFitGradients` is turned off since the *fit gradients* feature is not implemented for the `orientation` quaternion yet. In NAMD 2.12, you will receive an error if you do not turn off the `enableFitGradients`. In older versions, you will not get an error but the *fir gradients* feature will not be used. Another slight change in NAMD 2.12 is the addition of the `fittingGroup` keyword. For older versions, you should use: `refPositionsGroup`. We have provided an alternative colvars config file for versions older than NAMD 2.12:

1-smwst/0-parallel/colvars.old.conf:
```
 colvar {
    name q1
    orientation {
       atoms {
          psfSegID PROA
          atomNameResidueRange { CA 20-54 }
          centerReference on
          rotateReference on
          refPositionsGroup {
             psfSegID PROA PROA PROA PROA PROA PROA PROB PROB PROB
PROB PROB PROB
             atomNameResidueRange { CA 20-54 }
             atomNameResidueRange { CA 66-86 }
             atomNameResidueRange { CA 94-112 }
             atomNameResidueRange { CA 122-148 }
             atomNameResidueRange { CA 158-183 }
             atomNameResidueRange { CA 190-207 }
             atomNameResidueRange { CA 253-282 }
             atomNameResidueRange { CA 293-316 }
             atomNameResidueRange { CA 322-341 }
             atomNameResidueRange { CA 351-374 }
             atomNameResidueRange { CA 382-409 }
             atomNameResidueRange { CA 415-448 }
          }
          refPositionsFile ../../system/eq.pdb
       }
       refPositionsFile ../../system/eq.pdb
    }
 }
 ...
```

The harmonic block in either version looks like the following:

1-smwst/0-parallel/colvars.conf:
```
 harmonic {
    name harm
    colvars { q1 q2 q3 q4 q5 q6 q7 q8 q9 q10 q11 q12 }
    centers { ( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 ) (
1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 )
( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0 ) ( 1 , 0 , 0 , 0
) ( 1 , 0 , 0 , 0 ) }
 }
```

You can have other harmonic blocks and other colvars unrelated to the SMwST algorithm (e.g., to restrain part of the system) but all the colvars associated with the SMwST algorithm must be put in the same harmonic block. The name of the harmonic block and the names of the colvars that will be involved with the SMwST algorithm will be specified in the namd configuration file as discussed below. The `centers` above are not relevant as they will be overridden at the beginning of every SMwST iteration.

3.3 Setting up the SMwST simulations

The SMwST algorithm is implemented using the multiple-copy (or multiple-replica) capability in NAMD. The `smwst.namd` script is provided in the supporting files (in the `codes` folder) that can be sourced by namd configuration file for SMwST simulations, without any changes. However, there are several parameters that need to be specified in the NAMD configuration file in order to use the `smwst.namd` script. These parameters will be explained below:

1 Specify the job id.

Always start from 0 (which is the default value) for the first job, then if you need to restart the job due to interruption or to extend the simulations (to reach convergence), increment the job id by one.

1-smwst/0-parallel/job0.conf:

```
...
set job_id 0
...
```

2 Specify the desired total number of iterations.

This is a required field. When restarting, note that the number of previous runs will be determined from the reported simulation time in `*.restart.xsc` files; Whatever is performed already is subtracted from the specified `runs`. Here we start with 100 runs but continue the simulations to 1000 iterations in `1-smwst/0-parallel/job1.conf` to `1-smwst/0-parallel/job4.conf`. The number of iterations can be increased when restarting the SMwST simulations from the previously finished or interrupted SMwST simulations.

1-smwst/0-parallel/job0.conf:

```
...
set runs 100
...
```

3 Specify the number of images and copies.

The numbers of replicas and images are not in the configuration file. The only parameter specified here is the number of copies per image (`nc`), which has a default value: 1. The number of replicas is specified at the time of submission of your job as a charmm++ parameter (with `+replicas` switch), and the number of images is determined from the number of replicas divided by the number of copies. So the number of replicas must be a multiple of the number of copies.

1-smwst/0-parallel/job0.conf:
```
...
set nc 5
...
```

4 Specify the number of samples per copies.

Depending on the computational resources available to you, you may choose to use less number of copies, but sample each copy more than once before averaging the drift and updating the image centers. The number of samples is specified using `ns` variable, which is 1 by default. One may choose to use 1 copy per image and 5 samples per copy if the number of processors available is limited. Instead one may use 5 copies per image and only one sample per copy to get the results quicker. The two variables can also be combined to make the most efficient use of the time and resources. Note that the recommended number of `nc`×`ns` is at least 20. We use 5 here instead only to make the simulations easier to perform so one can complete the tutorial in a timely manner.

1-smwst/0-parallel/job0.conf:
```
...
set ns 1
...
```

5 Specify the simulation time for each stage.

`runE` and `runR` are the number of simulation time steps at each iteration, for the equilibration/restraining stage and drifting/release stage, respectively (with default values of 500 for each). Each iteration will thus have `runE`+`runR` steps. However, if `ns` is greater than 1, the centers will be updated only every `ns`× (`runE`+`runR`) steps.

1-smwst/0-parallel/job0.conf:
```
...
set runE 2500
set runR 2500
...
```

6 Specify the colvars configuration file, the harmonic block, and the collective variables used for the SMwST algorithm.

All required.

1-smwst/0-parallel/job0.conf:
```
...
set colvars_conf colvars.conf
set harm "harm"
set force 10000
set cvs { q1 q2 q3 q4 q5 q6 q7 q8 q9 q10 q11 q12 }
...
```

7 Specify the dimension and metric of collective veriables.

The dimension (`cvsd`) is 4 for quaternions. The metric (`cvsm`) is used for the distance calculations (each number provides a weight for each colvar in distance calculation; i.e., the given numbers are the components of a diagonal matrix that represents an isotropic metric tensor).

The default value is 1 for both `cvsd` and `cvsm`.

1-smwst/0-parallel/job0.conf:
```
...
set cvsd { 4 4 4 4 4 4 4 4 4 4 4 4 }
set cvsm { 1 1 1 1 1 1 1 1 1 1 1 1 }
...
```

If only one number is provided for `cvsd` or `cvsm`, all colvars will have the same dimension or metric (i.e., the number provided):

1-smwst/0-parallel/job0.conf:
```
...
set cvsd 4
set cvsm 1
...
```

8 Specify whether or not there should be reparametrization in the beginning of the simulation.

Allowed values are 0 (no initial reparametrization) and 1 (with an initial reparametrization), where the default is 0. If the initial reparametrization is not performed the initial image centers will be exactly the same as actual colvar values associated with initial conformations.

1-smwst/0-parallel/job0.conf:
```
...
set initial_reparam 1
...
```

9 Specify the smoothing parameter.

The `smooth` ranges between 0 and the number of images; recommended value is 1; 0 skips smoothing; The default value is 0.

1-smwst/0-parallel/job0.conf:
```
...
set smooth 1
...
```

10 Specify the output files.

The `output_root` is a required field specifying where the output files will be stored. Some examples include `output/glpt` and `output/%s/glpt`. The former will result in storing all output files in the same folder `../output`, while the latter will store the files associated with each image in a separate folder (e.g., `output/0` for all copies of image 0). If you choose this option, you need to make folders `0` to `19` in `../output` directory (e.g., using `seq ‘0 19‘`). The names of the output files will start with `glpt.job0.<id>`, where `<id>` is the replica id. Note that the replica id runs from 0 to 99 (for 100 replicas including 20 images, 5 copies each) and is unique for each replica (e.g., replica ids 0 to 4 belong to image 0 and so on).

1-smwst/0-parallel/job0.conf:
```
...
set output_root "output/%s/glpt"
...
```

11 Specify the initial coordinates, velocities, and periodic cell information.

These are only relevant when `<job_id>` is 0. Otherwise, the information is automatically determined from the `output_root` format, replacing the `<job_id>` with `<job_id>-1`, representing the previous job output files. In the `input_root`, the `%s` is replaced by the replica id as in the `output_root`. For instance if you set the `input_root` as `input/init.%s`, the coordinate file for all copies of image 0 will be `input/init.0.coor`. These files are already generated from the pulling simulation trajectories above. You have the option to provide the velocity files as well. In this case, you should set `usevel` to 1. Otherwise, you need to provide the temperature (e.g., `set temperature 310`). For the cell information, it is recommended to generate the *.xsc files like the `*.coor` files from the pulling simulation `*.dcd` files as done above. In this case you need to set `usexsc` to 1. Alternatively, you can set the `cellinfo` explicitly (e.g., `set cellinfo { {130 0 0} {0 130 0} {0 0 200} {0 0 0} }`, which contains the three `cellBasicVectors` followed by the `cellOrigin`).

1-smwst/0-parallel/job0.conf:
```
...
set input_root "input/init.%s"
set usevel 0
set temperature 310
set usexsc 1
...
```

12 Specify the rest of the namd parameters.

The rest of the namd parameters can be either provided here directly or simply sourced.

1-smwst/0-parallel/job0.conf:
```
...
set system_conf ../../system/glpt.inp
...
```

13 Source the smwst code.

1-smwst/0-parallel/job0.conf:

```
...
source ../../codes/smwst.namd
```

3.4 Running and analyzing the SMwST simulations

1 Run the SMwST simulation.

SMwST is a multi-copy algorithm and requires at least one processor per replica. A Charm++ NAMD build with a LRTS (netlrts, verbs, or mpi) is also needed. NAMD is then launched with `charmrun` (or `mpirun`) with `"+p <num_of_processors>"`, `"+replicas <num_of_replicas>"`, and `"+stdout <log_file_format>"`options added to divide the <num_of_processors> (utilized by the entire simulation) into <num_of_replicas> partitions that write to separate log files with `"%d"`in `"<log_file_format>"`replaced by the *replica id* of each replica. The number of replicas must evenly divide the number of processors specified. To run 100 replicas (each using 2 processors) writing to output/job0.log.0 through output/job0.log.99 log files:

Example run script:

```
charmrun namd2 job0.conf +p 200 +replicas 100 \
+stdout output/job0.log.%d > output/job0.log
```

Note that the number of copies was specified in the `job0.conf` file. The number of replicas is specified by `+replicas` argument. The number of images is not specified by the user. It is simply determined by dividing the number of replicas by the number of images. The number of replicas must be a multiple of the number of copies. Each replica will have its own log file if `+stdout` is used, however, the information regarding the image centers will be reported in the log file of the zeroth replica, i.e., `output/job0.log.0` in this case.

Serial SMwST! The provided script does not allow the use of serial SMwST. However, one may use one copy per image in order to limit the number of required processors to the number of images. The only change one needs to make is to set `nc` to 1. To ensure the validity of the SMwST algorithm, however, one should sample each copy multiple times. This can be done by setting the `ns` variable to whatever `nc` was set in the fully parallel version. We have the modified version in `1-smwst/1-serial`. The rest of the parameters and files are identical except for the runscript when the number of replicas (in `+replicas`) is equal to the number of images (20 in this case).

2 Analyze the simulations.

There is no mechanism implemented in the code for stopping the SMwST iterations due to convergence. The convergence can be determined by monitoring the evolution of the centers of images as reported in the log file of the zeroth replica. The drifted centers (determined from the average of copies of each image), the smoothed centers (if smoothing is used), and the reparametrized centers (after reparametrization) are reported at the beginning of each iteration.

The analysis script provided in `1-smwst/0-parallel/analysis` in the supporting files extracts the drifted, smoothed, and reparametrized centers from the log file of the zeroth replica and uses the reparametrized centers (can be altered to use the drifted or smoothed centers). One may monitor the orientation angles (twice the arccosine of the first element of each `orientation` colvar). The scripts also find the string root mean square distance (RMSD) of the centers at each iteration with respect to the initial, final, and previous iteration string centers (reported in columns 2, 3, and 4, respectively). This is reported in `s-rmsd-R.txt`. The string RMSD between two strings (of N images) is defined as the root mean square distance of the individual quaternions, i.e.,

$$r = \frac{1}{12N} \sum_{i=0}^{N-1} \sum_{j=1}^{12} |\cos^{-1}(Q_{i,j}.Q'_{i,j})|^2, \tag{1}$$

where $Q_{i,j}$ and $Q'_{i,j}$ are the j^{th} orientation quaternion of image i of the two strings. $|\cos^{-1}(Q_{i,j}.Q'_{i,j})|$ approximates the geodesic distance of the two quaternions. Pairwise string RMSD of all iterations are also calculated with the scripts in `1-smwst/0-parallel/analysis`.

Some sample output files (`1-smwst/0-parallel/sample-output`) are provided in the supporting files. You may change the `log` variable in `1-smwst/0-parallel/analysis/do.sh` (comment out line 6 and uncomment line 7) to use the provided sample log file to run the analysis. We have only provided the last log file (`1-smwst/0-parallel/sample-output/job4.log.0`), which includes information on the last 188 SMwST iterations. A gnuplot script is provided for plotting the results in `1-smwst/0-parallel/analysis/fig.sh`. Figure 5 summarizes some of the results (for all 1000 iterations), which shows the convergence of the SMwST method after about 600 iterations ($600\times(5+5\,\text{ps})=6\,\text{ns}$ per replica).

4 Free energy calculations along the refined transition path

Umbrella sampling (US) [13, 14] MD and similar biased MD based methods have been routinely used over the last two decades to calculate PMF along one- or two-dimensional collective variable spaces. For complex conformational transition pathways, it is difficult to find a low-dimensional space to properly define the reaction coordinate. Instead, free energy calculations along one-dimensional pathways, defined as curves in higher-dimensional collective variable spaces are

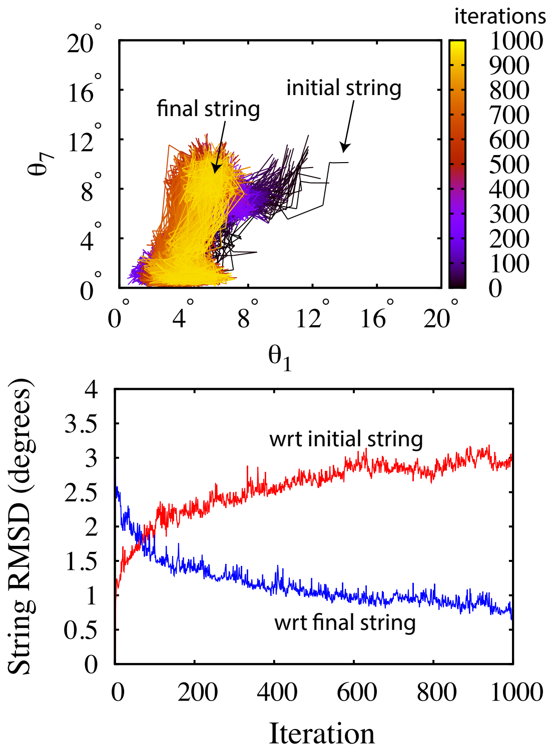


Figure 5: Examples of monitoring SMwST results. (Top) orientation angles with respect to the reference structure (`system/eq.pdb`) associated with orientations of helices 1 and 7 (θ_1 and θ_7, respectively). θ_i is calculated from the orientation quaternion of i^{th} helix (twice the arccosine of the first quaternion element) as reported for all images of all iterations (each color represents an iteration). (Bottom) the string RMSD with respect to the initial and final strings during 1000 iterations performed.

emerging [3] as better alternatives to traditional free energy calculation methods that rely on one- or two-dimensional collective variable spaces. The aim of this Section is to set up an US based simulation to calculate the free energy along the refined transition pathway obtained in the previous Section. To do so, we use the bias-exchange umbrella sampling (BEUS) method [1], which is implemented as a tcl code (`codes/beus.namd`) within NAMD's multiple-copy scheme, similar to the `codes/smwst.namd` code. The BEUS combines US and replica-exchange schemes similar to several other methods such as replica-exchange, window-exchange, and HREMD US methods. The BEUS method as implemented in the `beus.namd` code allows for defining arbitrary subspaces (e.g., a 1D curve) in high-dimensional collective variable spaces, to be sampled without the need to sample the entire high-dimensional space of collective variables used.

4.1 Setting up the BEUS simulation.

Before setting up the BEUS simulation parameters, you should decide whether or not you want to use the same number of images and copies for the BEUS simulations as you used for the SMwST simulations. It is generally convenient to choose the same number of images (or windows) as the image centers can be easily transferred and the initial conformations are already close to their relevant centers, so the initial equilibration in the BEUS simulations will be quite fast if needed. The number of copies does not have to be the same, however. The use of several copies for each image is necessary for the SMwST algorithm. However, for the BEUS scheme, one copy per image is generally enough. The image centers can then be transferred as is and the initial conformation of each image could be any of the final conformations associated with that image in the SMwST simulations. The use of several copies per image is advantageous for the BEUS scheme as it provides independent data points for error estimates as discussed later in more detail. Here, we will use 5 copies per image as in the SMwST simulations.

1 Transfer the final SMwST coordinates to the BEUS simulation.

The `2-beus/0-multiple/extract/do.sh` script transfers the final restart files (including `*.coor`, `*.vel`, and `*.xsc`) from the SMwST simulations to `2-beus/0-multiple/input` directory and renames them to `init.<replica_id>.*`, where `<replica_id>` is the replica id. The script can be easily modified to generate the desired number of copies per image.

2 Specify the centers of the BEUS windows.

The colvars configuration file for the BEUS simulation could be the same as the one used for the SMwSt simulations. The `centers` and `forceConstant` in the colvars configuration file will be overridden. The `2-beus/0-multiple/extract/do.sh` script extracts the final image centers from the SMwST log files (the final zeroth replica log file). The outcome is a tcl script which assigns a list to each element of array `center`. Each array element defines a specific image/window by assigning its `forceConstant` and `centers` to override the information in the colvars configuration file. The tcl list includes the name of the harmonic block used for the BEUS algorithm (e.g., `harm`), followed by the `forceConstant` and `centers` information. The 2-beus/0-multiple/extract/do.sh generates the following file, which is then used to set up the simulation configuration file:

2-beus/0-multiple/centers.tcl:
```
set centers(0) [list harm "forceConstant \{ 5000 \} centers \{ (
0.9989229671865218 , ...  \} "]
set centers(0) [list harm "forceConstant \{ 5000 \} centers \{ (
0.9994443474453633 , ...  \} "]
...
```

Note that the actual centers are omitted here but can be found once the above `do.sh` is executed. The `forceConstant` values could be different for different windows. This is sometimes useful to ensure a more even rate of exchange between neighboring windows. If you use the same force constant for all windows, it is possible to assign the desired force constant in the colvars configuration file and omit the `forceConstant` assignment in the `centers.tcl` file.

4.2 Setting up the configuration file

The `smwst/job0.conf` file contains the information needed to set up the BEUS simulation and will be passed to namd as the main configuration file with a similar run script (including `+replicas` and `+stdout` arguments as discussed in the previous Section). Several parameters are similar to those already discussed in SMwST simulations. These include `job_id`, `runs`, `nc`, `colvars_conf`, `system_conf`, `output_root`, `input_root`, `usevel`, `temperature`, `usexsc`, and `cellinfo`. The `input_root` here is slightly different from that of SMwST code. The `%s`, if specified will indicate the replica id rather than the image id. So the initial conformation could be unique for each copy.

Here `runs` indicates the number of iterations as in the SMwST case; however, each iteration is followed by an attempt of exchange between neighboring copies as discussed below:

1 Specify the number of steps per run.

The total simulation time will be `runs` multiplied by `runUS`. The exchange will be attempted every `runUS` steps. The default value is 500.

2-beus/0-multiple/job0.conf:
```
...
set runUS 500
...
```

2 Specify the number of runs per restart.

The number of iterations (exchange attempts) between checkpoints. The default value is 1.

2-beus/0-multiple/job0.conf:
```
...
set runs_per_restart 1
...
```

3 Specify the number of neighbors for each image.

The number of neighbors in each side of an image is 1 by default. For a 1D BEUS simulation, the total number of neighbors will then be 2 (unless at the boundaries) due to presence of two possible sides. For a 2D BEUS, there will be four neighbors (except for the boundary windows) by default. However, one may specify more neighbors; This could be advantageous in certain situations, when the non-immediate neighbors are close enough in the collective variable space to potentially exchange their centers. If `num_of_neighbors` is set to 2, the second closest image will be considered a neighbor as well.

2-beus/0-multiple/job0.conf:
```
...
set num_of_neighbors 1
...
```

The number of neighbors defines whether or not two given images are neighbors. For instance, with the default setting, image 2 is a neighbor of both image 1 and image 3 (one neighbor in each side). This will make all copies of each image neighbors with all copies of the neighboring image. Having various copies of each image increases the chance of exchange (and subsequently mixing) between neighboring images.

Exchange Scheme! Each replica has typically several neighboring replicas. For multiple-copy images, there could be many neighbors for each replica but even in a single-copy setup (i.e., each image is represented only with one copy), there are typically two neighboring replicas in a 1D BEUS setting. Whether a single- or multiple-copy setup is used, for each given replica, the exchange is not attempted for more than one neighboring replica at each iteration. The neighboring replica alternates between possible neighbors. The exchange attempt between a unique replica pair will occur every 2×`nc`×`number_of_neighbors` iterations for a 1D BEUS simulation (i.e., every 10 iterations for our example).

4 Specify the image centers.

The image centers are already generated using the `2-beus/0-multiple/extract/do.sh` script and stored in the `centers.tcl` file. This file must be assigned to the `centers_tcl` variable in namd configuration file.

2-beus/0-multiple/job0.conf:
```
...
set centers_tcl centers.tcl
...
```

5 Source the BEUS code

The `beus.namd` is a code similar to `smwst.namd` that must be sourced to perform the BEUS simulations. Note that various other BEUS schemes, such as 2D BEUS, and more arbitrary shaped BEUS schemes are supported by the `beus.namd` code that are not discussed here.

2-beus/0-multiple/job0.conf:
```
...
source ../../codes/beus.namd
```

4.3 Performing BEUS simulations

Running the BEUS simulations is similar to running the SMwST simulations. It is somewhat convenient to use the same number of images and copies in both simulations but it is not always efficient to do so. Having multiple copies per image is helpful for a reliable error estimate and increases the exchange rate but it is less crucial for BEUS simulations to have multiple copies per image as compared to SMwST simulations. Instead, one may use one copy for each image and perform each simulation longer, as an alternative approach. There is no need to set up a `ns` parameter here (unlike SMwST) as the image centers are not updated in the BEUS scheme. An advantage of using one copy per image is that it allows for using more images (with the same number of replicas), which improves the “sampling overlap” between neighboring images, an important requirement of any US scheme. Note that the number of images of a good SMwST setup is not necessarily the same as that of a good BEUS setup. In `2-beus/1-single`, we have used 96 images/replicas to sample the same space sampled by the `2-beus/0-multiple` simulations. The image centers are linearly combined to generate 96 unique image centers (out of 20 unique image centers from SMwST simulations). The initial conformations come from the SMwST simulations as well. The `2-beus/1-single/extract/do.sh` is the modified version of `2-beus/0-multiple/extract/do.sh`. One is encouraged to perform both sets of BEUS simulations and compare the results.

Minimal BEUS Run! To avoid storing large supporting files for this tutorial we have only provided the SMwST coordinates and xsc files of the first four images (or the first 20 replicas, since each image has 5 copies). So if you want to use the provided files as your initial string, you should have only 4 images in your BEUS simulations. The minimal simulation `2-beus/2-minimal` is quite similar to `2-beus/0-multiple` with only 4 rather than 20 images, and requiring 20 replicas for the simulations. We have provided some sample output files for this minimal simulation so you can follow the next steps of the tutorial without running the simulations.

4.4 Optimizing the BEUS protocol

The BEUS scheme similar to other simulation schemes is set up with various user-defined parameters. The poor choice of these parameters could result in slow convergence or systematic errors. The two common criteria to have a successful BEUS simulation are the "good overlap"(as in any umbrella sampling simulation) and the "good exchange rate"(as in any replica exchange simulation).

1 Monitor the exchange rate.

The accepted exchanges are reported in the main namd log file (`job0.log`, in this example). The `2-beus/*/monitor/exchange.sh` uses this log file to calculate the exchange rate between each image pair. We have provided sample log files in both `2-beus/1-single/sample-output` and `2-beus/2-minimal/sample-output` directories. To use the `exchange.sh` script with these log files instead of your own simulations, you need to comment out line 5 and uncomment line 6 of the script.

It is advantageous to have similar rates of exchange between all neighboring replicas. Having window-dependent force constants can potentially help with this condition, where the force constant is lowered for images with lower exchange rates and vice versa. The uniform exchange rate condition, in practice, could be hard to achieve. However, having zero exchange rates between neighboring replicas must be avoided with any means. To achieve this condition, one may lower the force constant for the non-exchanging replicas. However, this could potentially result in gaps in sampling, where higher energy states are not sampled due to the use of low force.

A more appropriate approach to avoid low exchange rates in certain regions is to add images. The equidistant reparametrization in the SMwST scheme is not necessarily advantageous for having a uniform exchange rate. A good practice is to reparametrize the string to result in a uniform exchange rate in the BEUS setting and this can be done empirically by trial and error before running the BEUS production runs.

4.5 Estimating the free energies

A nonparametric reweighting approach is recommended to be used for the estimation of free energies for the images from the BEUS simulations. Multistate Bennett acceptance ratio (MBAR) method and its Bayesian version (based on Gibbs sampling) are two potential methods. Our implementation is referred to as non-parametric WHAM (weighted histogram analysis method) that uses a non-parametric reweighting scheme to simultaneously estimate the image free energies and the probability of each sampled conformation. The former is used to estimate the so-called *perturbed free energy* , which is a good estimate for the PMF along the path within the stiff-spring approximation (i.e., large force constant limit) [4]. The latter can be used to reconstruct the PMF along the path and in any arbitrary well-sampled space. The `npwham` code (in `codes/npwham.0`), unlike traditional WHAM codes, does not build a histogram in the collective variable space. Instead, it requires as its input the list of biasing potential values that each sampled conformation would have felt under each window bias. We have provided example scripts to calculate these potentials from the BEUS colvars trajectories.

1 Extract the collective variable trajectories.

`2-beus/*/pmf/0-colvars/do.sh` is a script that combines the colvars trajectory files with the BEUS history files to assign the actual sampled collective variable values associated with each sample to its appropriate image id. The output (`/2-beus/*/pmf/0-colvars/cvs.txt`) will be useful to ensure the overlap criterion as it combines the image ids (column 3) and the colvar values (starting with column 4) in the same file. If you have not run your own simulations you can use `2-beus/2-minimal/pmf/0-colvars/do.sh` to generate the `cvs.txt` file of the minimal simulations. You need to comment out line 7 and uncomment line 8 to use the sample output files provided.

2 Calculate the hypothetical biasing potentials.

The calculations are based on the image centers (stored in `2-beus/*/centers.tcl`) and the actual sampled collective variables (stored in `2-beus/*/pmf/0-colvars/cvs.txt`). The biasing potentials are calculated the same way they are calculated in NAMD, that $U(Q_s, Q_c) = \frac{1}{2}k\Omega(Q_s, Q_c)^2$, where Q_s is a quaternion associated with a particular sample, Q_c is the same quaternion associated with a particular image center, and $\Omega(Q_s, Q_c) = \cos^{-1}(|Q_s \cdot Q_c|)$ is the approximate geodesic distance between Q_s and Q_c. The provided script (`beus/*/pmf/1-pot/do.sh`) calculates the potentials and store them in `2-beus/*/pmf/1-pot/pot.txt`. This file is the main input file for the *npwham* algorithm. In the case of `2-beus/2-minimal/pmf/1-pot/pot.txt`, each line contains the actual image id (0 to 3), the copy id (0 to 4), time, and the potentials based on individual image centers (from image 0 to 4).

3 Estimate the free energies from the potentials.

In addition to the potentials generated above you need to specify the number of images, the number of data points (determined from the number of lines of the potential file generated above: `2-beus/2-minimal/pmf/1-pot/pot.txt`), and the temperature.

2-beus/2-minimal/pmf/2-fe/do.sh:
```
awk '{
    for(i=1;i<=NF;i++) if(i!=2) printf "%s ",$i
    printf "\n"
}'../1-pot/pot.txt | \
../../../../codes/npwham.0/npwham -w 20 -l 20000 -t 310 \
> density.txt 2> density.err
```

The awk script is used to remove the second column, which is the copy id. The copy ids are used to distinguish different copies of each image. These will be used later for bootstrapping. For now, we only get average free energies. The generated `density.err` file contains the converged free energy estimates at the end of the file (see Fig. 6, black curve). The `density.txt` file contains the probability of each sampled conformation. One may use these probabilities to build weighted histograms in terms of various quantities. However, to see free energies we do not need to build any histograms. To learn more about the npwham code, you may use `codes/npwham.0/npwham -h`.

4 Use bootstrapping to estimate the errors.

You may use the *npwham* code to not only generate free energies but also error estimates. This can be done by using the copy ids already stored in the potential file as the *block* ids for block-based bootstrapping. The algorithm implemented in the GWHAM code is a Bayesian version of block bootstrapping and only requires a few blocks for each image/window. The `2-beus/2-minimal/pmf/1-pot/pot.txt` file can be used directly this time without the need to remove the copy ids. Two arguments are added to the *npwham* command line: `-b 5 -B 100`. The former indicates the number of blocks (or copies) and the latter is the number of bootstraps generated. The average free energies and standard deviations are reported at the end of the `stderr` file based on 100 bootstraps (see Fig. 6, red curve).

2-beus/2-minimal/pmf/2-fe/err.sh:
```
../../../../codes/npwham.0/npwham -w 20 -l 20000 -t 310 -b 5 -B 100 \
< ../1-pot/pot.txt > density.txt 2> density.err
```

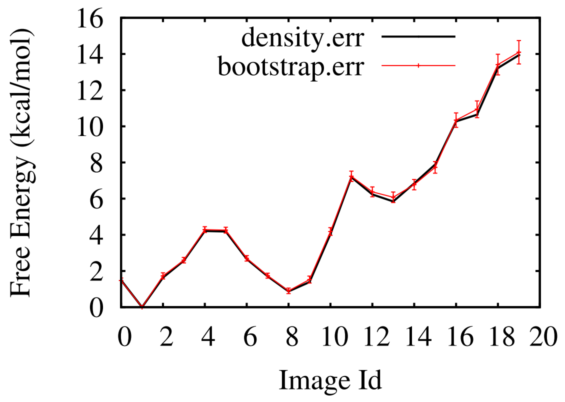


Figure 6: Free energies of different images as obtained using the *npwham* algorithm without (black) and with (red) the bootsrapping algorithm. The error bars are the standard deviations from 100 bootstraps.

5 Calculate the PMF along collective variables and other quantities.

The `density.txt` and/or the `bootstrap.txt` files in `beus/*/pmf/2-fe/` contain the probabilities of the samples that can be used to build weighted histograms. The log of the weighted histogram along any quantity is proportional to the PMF of that quantity. The script in `2-beus/*/pmf/3-pmf/` builds the PMF in terms of all orientation angles (calculated from the orientation quaternions). Each resulting column (except for the first that is the x axis) is a particular orientation angle (starting from first to the twelfth helix). The script in `2-beus/*/pmf/4-sb/` builds the PMF in terms of the K46-D274 salt bridge distance calculated from the dcd files of all BEUS simulations using VMD. We note that the PMF along arbitrary quantities may be calculated but they will not necessarily be reliable unless they are well sampled along the sampled pathway.

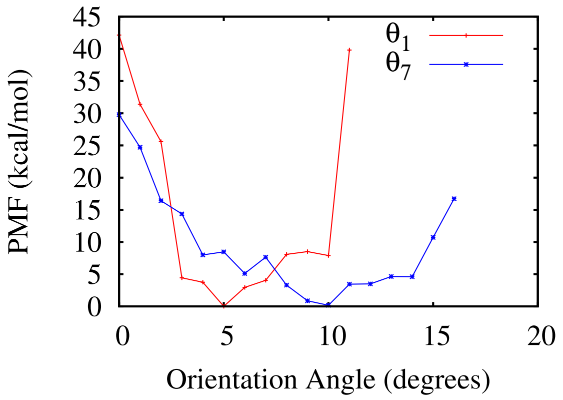


Figure 7: PMF along the orientation angles θ_1 and θ_7 as obtained using the weighted histogram method based on the *npwham* estimates of the probability of individual samples (`density.txt`).

5 Closing Remarks

The three-step sampling protocol described in tutorial is both general and limited. It is general since in principle it can be applied to various systems from channels and transporters to other transmembrane proteins and even soluble proteins. The limitation, however, is due to the fact that the definition of collective variables and the pulling protocols that induce the transition of interest for a given system are intrinsically system-specific. The system specificity of the biasing protocols prevents the user from applying the methodology to arbitrary systems. The user should find the appropriate collective variables and pulling protocols, that may or may not be similar to those used here. On the other hand, the success of the methodology is partly due to the system specificity of its design, which requires some knowledge of the system under study and its transition. The knowledge-based nature of the methodology, which is manifested in its system-specificity, is the driving force behind the selective, focused, and target-oriented sampling of the phase space, which otherwise is too large to search. The choice of the collective variables, however, is much more flexible than a conventional SMD or US simulation, allowing for sampling of more complex and realistic conformational transition pathways. We also note that there is an empirical side in the methodology from the choice of collective variables and pulling protocol to the choice of the number of images and copies. This empirical nature is both beneficial (by allowing better sampling) and limiting (by preventing the transfer of parameters to other systems). Even in the example provided here, the transition pathway generated, optimized, and quantified is based on a quite simple initial pulling protocol, which is chosen due to its simplicity. Multiple attempts are typically necessary to optimize the initial pulling protocol as discussed in more detail for GlpT [3] and another transproter [1, 2]. It is recommended that the reader uses the protocols provided in the tutorial with extra care. Every step should be followed by careful examination of the results both qualitatively, using the visualization of the trajectories, and quantitatively, using the statistical tools such as those discussed in Ref. [2].

References

[1] M. Moradi and E. Tajkhorshid. Mechanistic picture for conformational transition of a membrane transporter at atomic resolution. *Proceedings of the National Academy of Sciences, USA*, 110(47):18916–18921, 2013.

[2] M. Moradi and E. Tajkhorshid. Computational recipe for efficient description of large-scale conformational changes in biomolecular systems. *Journal of Chemical Theory and Computation*, 10(7):2866–2880, 2014.

[3] M. Moradi, G. Enkavi, and E. Tajkhorshid. Atomic-level characterization of transport cycle thermodynamics in the glycerol-3-phosphate:phosphate transporter. *Nature Communications*, 6:8393, 2015.

[4] A. Fakharzadeh and M. Moradi. Effective Riemannian diffusion model for biomolecular simulations. *Journal of Physical Chemistry Letters*, 7:4980–4987, 2016.

[5] S. Izrailev, S. Stepaniants, M. Balsera, Y. Oono, and K. Schulten. Molecular dynamics study of unbinding of the avidin-biotin complex. *Biophysical Journal*, 72:1568–1581, 1997.

[6] A. C. Pan, D. Sezer, and B. Roux. Finding transition pathways using the string method with swarm of trajectories. *Journal of Physical Chemistry B*, 112(11):3432–3440, 2008.

[7] Y. Sugita, A. Kitao, and Y. Okamoto. Multidimensional replica-exchange method for free-energy calculations. *Journal of Chemical Physics*, 113(15):6042–6051, 2000.

[8] P. Das, M. Moll, H. Stamati, L. E. Kavraki, and C. Clementi. Low-dimensional, free-energy landscapes of protein folding reactions by non-linear dimensionality reduction. *Proceedings of the National Academy of Sciences, USA*, 103(26):9887–9890, 2006.

[9] A. L. Ferguson, A. Z. Panagiotopoulos, P. G. Debenedetti, and I. G. Kevrekidis. Systematic determination of order parameters for chain dynamics using diffusion maps. *Proceedings of the National Academy of Sciences, USA*, 107(31):13597–13602, 2010.

[10] L. Maragliano, A. Fischer, E. Vanden-Eijnden, and G. Ciccotti. String method in collective variables: Minimum free energy paths and isocommittor surfaces. *Journal of Chemical Physics*, 125(2):024106, 2006.

[11] W. E and E. Vanden-Eijnden. Transition-path theory and path-finding algorithms for the study of rare events. *Annual Review of Physical Chemistry*, 61:391–420, 2010.

[12] M. E. Johnson and G. Hummer. Characterization of a dynamic string method for the construction of transition pathways in molecular reactions. *Journal of Physical Chemistry B*, 116:8573–8583, 2012.

[13] G. M. Torrie and J. P. Valleau. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella Sampling. *Journal of Chemical Physics*, 23(2):187–199, 1977.

[14] S. H. Northrup, M. R. Pear, C. Y. Lee, J. A. McCammon, and M. Karplus. Dynamical theory of activated processes in globular proteins. *Proceedings of the National Academy of Sciences, USA*, 79:4035–4039, 1982.

[15] J. Schlitter, M. Engels, P. Krüger, E. Jacoby, and A. Wollmer. Targeted molecular dynamics simulation of conformational change — application to the T $\leftrightarrow$ R transition in insulin. *Molecular Simulation*, 10(2–6):291–308, 1993.

[16] C. Jarzynski. Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78:2690–2693, 1997.

[17] G. E. Crooks. Path-ensemble averages in systems driven far from equilibrium. *Physical Review E*, 61:2361–2366, 2000.

[18] G. Hummer and A. Szabo. Free energy reconstruction from nonequilibrium single-molecule pulling experiments. *Proceedings of the National Academy of Sciences, USA*, 98:3658–3661, 2001.

[19] C. Jarzynski. How does a system respond when driven away from thermal equilibrium? *Proceedings of the National Academy of Sciences, USA*, 98:3636 – 3638, 2001.

[20] S. Kumar, D. Bouzida, R. H. Swendsen, P. A. Kollman, and J. M. Rosenberg. The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method. *Journal of Computational Chemistry*, 13:1011–1021, 1992.

[21] O. Jardetzky. Simple allosteric model for membrane pumps. *Nature*, 211(5052):969–970, 1966.

[22] W. Ren, E. Vanden-Eijnden, P. Maragakis, and W. E. Transition pathways in complex systems: Application of the finite-temperature string method to the alanine dipeptide. *Journal of Chemical Physics*, 123(13):134109, 2005.

[23] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26:1781–1802, 2005.

[24] W. Jiang, J. Phillips, L. Huang, M. Fajer, Y. Meng, J. Gumbart, Y. Luo, K. Schulten, and B. Roux. Generalized scalable multiple copy algorithms for molecular dynamics simulations in NAMD. *Computer Physics Communications*, 185:908–916, 2014.

[25] P. A. Kollman and D. A. Pearlman. The lag between the Hamiltonian and the system configuration in free-energy perturbation calculations. *Journal of Chemical Physics*, 91(12):7831–7839, 1989.

[26] L. Zheng, M. Chen, and W. Yang. Random walk in orthogonal space to achieve efficient free-energy simulation of complex systems. *Proceedings of the National Academy of Sciences, USA*, 105(51):20227–20232, 2008.

[27] Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314:141–151, 1999.

[28] S. Park, T. Kim, and W. Im. Transmembrane helix assembly by window exchange umbrella sampling. *Physical Review Letters*, 108:108102–108105, 2013.

[29] S. Park and W. Im. Two dimensional window exchange umbrella sampling for transmembrane helix assembly. *J. Chem. Theory Comput.*, 9:13–17, 2013.

[30] C. Bartels. Analyzing biased Monte Carlo and molecular dynamics simulations. *Chemical Physics Letters*, 331:446–454, 2000.

[31] M. Habeck. Bayesian reconstruction of the density of states. *Physical Review Letters*, 98:200601, 2007.

[32] M. R. Shirts and J. D. Chodera. Statistically optimal analysis of samples from multiple equilibrium states. *Journal of Chemical Physics*, 129(12):124105, 2008.

[33] Z. Tan, E. Gallicchio, M. Lapelosa, and R. M. Levy. Theory of binless multi-state free energy estimation with applications to protein-ligand binding. *Journal of Chemical Physics*, 136(14):144102, 2012.

[34] E. A. Coutsias, C. Seok, and K. A. Dill. Using quaternions to calculate RMSD. *Journal of Chemical Physics*, 25(15):1849–1857, 2004.

[35] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[36] G. Fiorin, M. L. Klein, and J. Hénin. Using collective variables to drive molecular dynamics simulations. *Molecular Physics*, 111(22–23):3345–3362, 2013.