Conducting and Analyzing Window-based Umbrella Sampling and Milestoning Simulations with NAMD

Justin Spiriti

June 17, 2025

1 Umbrella Sampling Simulations

Conventional molecular dynamics simulations involve simulating the motion of atoms using Newton's laws in a potential given by the force field

$$\begin{aligned} U_{\mathrm{FF}}(\mathbf{x}) &= \sum_{\text{all bonds}} \frac{1}{2} K_b \left(b - b_0\right)^2 + \sum_{\text{all angles}} \frac{1}{2} K_\theta \left(\theta - \theta_0\right)^2 + \sum_{\text{all torsion angles}} K_\phi \left[1 - \cos\left(n\phi + \delta\right)\right] \\ &+ \sum_{\text{all nonbonded pairs}} \varepsilon \left[\left(\frac{r_0}{r}\right)^{12} - 2\left(\frac{r_0}{r}\right)^{6} \right] + \sum_{\text{all nonbonded pairs}} \frac{K_{\mathrm{coul}} q_i q_j}{r} \end{aligned} \tag{1}$$

The concept behind umbrella sampling is to add an additional term that forces the simulation to sample in different regions of a space defined by a reaction coordinate. The reaction coordinate is a function of the Cartesian coordinates that defines structural changes in the system. In this case we use the coordinates of the center of mass of the ligand relative to the protein after first aligning the protein-ligand complex via backbone with a reference structure. A harmonic term in terms of this center of mass is added to the force field potential.

$$U(\mathbf{x}) = U_{\mathrm{FF}}(\mathbf{x}) + \frac{k}{2} \left(\mathbf{r}_{\mathrm{CM}} - \mathbf{r}_{\mathrm{CM},0}\right)^2 \tag{2}$$

The harmonic term is centered on a position taken from a previously performed simulation designed to obtain at least one pathway for dissociation; this can be from a steered molecular dynamics simulation, a τ-RAMD simulation, a simulation from the "pathways" method described below, or some other technique.

This attempts to document how I performed the umbrella sampling simulations for PYK2 and the Markov state analysis using TRAM. I have tried to collect together all the scripts I used in the directory. Most of the scripts identify the simulation by a `name` parameter. The name I use is a combination of the name of the protein, the ligand, and the force field used, for example `pyk2-cpd1-amber`.

Needed software includes `tleap` from AmberTools, VMD and NAMD. If compounds need to be parameterized, it will be necessary to use Schrodinger Maestro, Gaussian, and `antechamber` from AmberTools as well.

1. It may be necessary to construct a force field for the compound. The scripts for doing this are in the `pathways` directory. This requires performing a quantum chemistry optimization using Gaussian to obtain the electrostatic potential, to which the partial charges can then be fitted using the RESP method. The scripts require a `mol2` file for the compound, which is best obtained by loading a reference PDB file into Schrodinger Maestro, deleting everything except the ligand, making any needed chemical modifications, and saving the result as a `mol2` file. This ensures that atom names in the force field match those in the PDB file. the `set-up-gaussian3` script uses `antechamber` to set up two Gaussian input files. The first performs an optimization at the HF/MIDIX level (MIDIX is a small basis set optimized for reproducing geometries) and the second performs an optimization at the HF/6-31g* level using the output of the first as an initial guess. The `do-gaussian3.qsub` submission script runs both calculations. Finally, the `finish-parameterization` script completes the parameterization, using `antechamber` once again to fit the charges to the electrostatic potential. (The `do-cubes.qsub` and

draw-orbitals.tcl scripts may be used to draw a picture of the orbitals.) These scripts are in the params/ directory.

2. The "pathway" method is a simplified version of the weighted ensemble method designed to find initial pathways for ligand dissociation. It is an alternative to SMD or τ-RAMD for this purpose. It works by running a number of simulations in parallel for a short length of time, then selecting those in which the ligand has moved the most and starting new simulations from them. The scripts for this method are located in initial-pathways/. The run-pathways.qsub script is the master script that controls everything. The number of simultaneous simulations is set through the nseg variable. It should divide the total number of processors used (set by the ntasks option in the script header). The number of new simulations started at each iteration (which should divide the total number of simulations) is set through the nstart variable. The total number of iterations is given in the for iter in ‘seq 1 200‘ statement. The segment-explicit.inp and segment-gb.inp scripts are NAMD scripts that actually run the MD simulations; select-new-simulations.py is a Python script that selects the new simulations and writes this information to a log file.

3. The first step is to choose the centers of the umbrella potentials from among the configurations sampled in a previous SMD or τ-RAMD simulation. For τ-RAMD simulations the get-com-all script reads each τ-RAMD trajectory and constructs a file giving the ligand center of mass for each frame in each trajectory (calling VMD with the get-relative-com3.tcl script to accomplish its work. The get-windows.py script actually selects the windows. This script is important because it selects the spacing and extent of the windows. The parameters are

   ```
   ./get-windows.py list-of-coms interval limit > list-of-windows
   ```

   where list-of-coms is the list of centers of mass generated by get-com-all, interval is the minimum distance between centers, and limit is the maximum distance for a center from the initial center of mass (which the script takes to be the center of mass in the original list of centers of mass, which comes from first frame of the first trajectory file originally considered). For the PYK2 simulations, the windows were 1.0 Å apart, and the limit was 15 Å. The output of get-windows.py is a list containing the window number, name of the trajectory and frame from which each center is selected, and coordinates of the center of mass . The get-windows script (which relies on the align-frame.tcl script) uses this file to extract the frames, align each one to a reference structure, and write them as individual PDB files in a directory named frames. All of the above-mentioned scripts can be found in the umbrella-sampling/extract/ directory.

4. Each of the extracted frames needs to be solvated, placing water molecules in a box around the protein and adding sodium chloride to make the system neutral and to give a salt concentration of 150 mM. This is accomplished using the solvate-all script (located in the umbrella/solvate directory) which takes the following parameters:

   ```
   ./solvate-all name cpd nwindow
   ```

   The cpd parameter is the specific name of the ligand (for example cpd1); the script uses this to construct the names of the frcmod and lib parameter files that it will use (cpd.frcmod and cpd.lib respectively). The script calls tleap from AMBER with the tleap-explicit-template.in script, processing PDB files in the frames directory and placing solvated PDB files and corresponding inpcrd and prmtop files in the data directory. The sed statement in the script modifies each PDB file so that it can be processed by tleap, by relabeling hydrogen atoms in the amino group at the N-terminus and placing a TER marker between the atoms of the protein and atoms of the ligand. It must be modified to do these correctly for a new protein otherwise tleap will give errors. The solvate-all script also calls the setup-ref.awk script, which marks all non-hydrogen atoms in the protein and ligand with a 1.0 in the B-factor column (this is important for the restraints during the heating and equilibration phase), creating a "reference structure" that is placed in the ../ref directory. All of these scripts can be found in the solvate/ directory.

5. The systems need to be minimized, heated, and equilibrated. The scripts for doing this are in the `umbrella/min-heat-equil` directory. This is done with harmonic restraints on the heavy atoms of the protein and ligand, so that the water equilibrates to the protein-ligand complex and not the other way around. The systems are minimized in five phases with harmonic restraints on the heavy atoms of the protein and ligand and with force constants that descend in factors of 10 from 10^4 kcal/mol Å^2 to 1 kcal/mol Å^2; each phase consists of 1000 steps of minimization. Heating then takes place over 600 ps with the harmonic restraints at 1 kcal/mol Å^2, and with the temperature increasing by 10 K every 20 ps. Equilibration takes place over 400 ps in five phases with the harmonic restraints of 0.75, 0.5, 0.25, 0.1 and 0 kcal/mol Å^2. The NAMD scripts that perform these tasks are `pyk2-umbrella-minimize-amber.inp`, `pyk2-umbrella-heat-amber.inp`, and `pyk2-umbrella-equil-amber.inp`. The `pyk2-umbrella-heat.qsub` script may be used to submit to the Foundry queue (or modified for another cluster). These scripts can be found in the `min-heat-equil/` directory.

6. Finally, the production run can begin. It is necessary to run the `fix-ref.awk` script on each of the reference structures in order to re-mark each backbone atom in the PDB file with a 1.0 in the B-factor column. The main NAMD script is `pyk2-run-umbrella.inp` and the definition of the umbrella potential is in `colvar3.inp`. The force constant k for the umbrella potential is given by the `forceConstant` setting in the script. The MD simulation is run in 1-ns sections coordinated by the `do5` script. This script calls NAMD to run a section of dynamics, then strips the water molecules from the trajectory file for that section to save disk space. The master control script is `pyk2-run-umbrella-foundry.qsub`. The overall length of the simulation can be adjusted via the loop in this script. The These scripts are located in the `umbrella-sampling/umbrella/` directory.

7. The trajectory segments need to be joined together to form complete trajectories. The `join-amber` script performs this. The center of mass of the ligand in each frame also needs to be determined. This is done using the `get-relative-com3.tcl` (which is slightly different from the one in the `umbrella-sampling/extract/` directory).

8. Finally the free energy surface can be derived using WHAM. The scripts and programs to do this are in the `umbrella-sampling/analysis/free-energy-surface` directory. The source code for the WHAM program is `wham3d-1d3.f90` (it may be compiled with any Fortran compiler before use). The command line for this program is as follows:

   ```
   ./wham3d-1d3.f90 control-file text-output dx-output
   ```

   The `create-control-file` script generates a "control file" that is the primary input for the WHAM program. The first line of this control file (which is output by `create-control-file`) contains several parameters, including the total number of data points for each window, the resolution of the grid for the three-dimensional histograms, starting and ending times, and the maximum number of WHAM iterations and convergence tolerance. The control file also contains the force constant for the umbrella potential, which must match what was specified in `colvar3.inp`. The program produces a DX file containing the three-dimensional free energy surface that may be visualized using VMD. This is done by importing it into VMD as a volumetric data set alongside the reference structure used for alignment, then adding a representation with the drawing style set to Isosurface and the surface style set to Solid Surface. The level of the contour may be set through the Isovalue box. Isosurfaces may also be drawn using the (The WHAM program also outputs the values of $\ln Z_i/Z_0$; however there might be a minor bug that causes these to come out with the wrong sign.) The `draw-fe-surfaces4c.tcl` script may be used with VMD to draw free energy surfaces.

9. An MM/GBSA analysis of protein-ligand interactions as a function of ligand center-of-mass distance from the binding site (as described in the Supporting Information for the paper and shown in figure S5) may be conducted by running the scripts in the `umbrella-sampling/analysis/gb-anal` directory. The `do-mmpbsa` script conducts the analysis using the `MMPBSA.py` tool in the AmberTools package and the `mmpbsa.in` input file. The `parmed.in` file is used to change the type of dielectric solvation radii used for the analysis. The `plot-inte-by-distance2b.py` script averages the results in bins by distance. It takes two parameters: the frame interval and the bin size in distance space.

2 Milestoning simulations

The scripts to determine an optimized pathway using the free energy surface and use this pathway for milestoning are located in the `milestoning/` directory. They require a `ref` directory that is coordinate to all of them, as well as a scratch directory that contains the following subdirectories:

`prmtop`	Amber parameter-topology files
`heat-data`	Heating coordinate and velocity files
`heat-dcd`	Heating trajectories
`equil-data`	Equilibration coordinate and velocity files
`equil-dcd`	Equilibration trajectories
`start-frames`	Starting frames for each milestoning trajectory
`data`	Coordinate and velocity files for milestoning trajectories
`dcd`	Initial trajectory files for milestoning trajectories
`dcd-nowater`	Trajectory files for individual sections after stripping out water
`dcd-joined`	Final trajectory files for each milestoning trajectory, after joining sections and stripping out water

In addition, there are two important files: the pathway (list of anchor positions) and list of milestones. These files are needed by several scripts throughout the process and must generally be in the same directory as the scripts using them. Throughout the scripts, the system is identified with a `name` parameter, individual anchor points are identified by sequence numbers, and milestones are identified by a pair of anchor numbers identifying the two Voronoi cells along whose boundary the milestone lies.

1. The scripts for determining an optimized pathway are in the `milestoning/find-pathway` directory. The `zero-temperature-string3.py` script performs the optimization. The parameters are the free energy surface file, its resolution, the number of points and total length, the three components of the vector along the initial direction for the path, a separation k and minimum distance r_0 for the self-avoiding potential in the second term of equation 2 in the 2024 paper, and the step size in τ and number of iterations for the relaxation. The separation k refers to the minimum separation $|j-i|$ between points i and j along the string for the self-avoiding potential to be effective. The scripts `get-zts-scan-list` and `zts-scan.qsub` may be used to systematically scan a range of initial directions for the pathway. The optimized pathways are placed in a directory called `sample-pathways-energy` and given a name ending in a number identifying the pathway. The `check-pathways` script may be used to determine the maximum energy along each pathway, which may assist in selecting the optimized pathway with the lowest barrier height, passing through a saddle point on the free energy surface. There are also various drawing scripts, for drawing figures such as figures 3(a) and 3(b) in the 2024 paper. The `select-frames-initial-milestones2.py` script is used to identify the frames from umbrella sampling simulations with the ligand center of mass closest to the milestone midpoints, and the `select-frames2` script to extract them as separate PDB files. The `select-frames-initial-milestones2.py` also writes the milestone list; this along with the pathway file (list of anchor positions) is needed by subsequent scripts.

2. The `solvate-all` script in the `milestoning/solvate` directory solvates all the selected frames using the `tleap` program in AmberTools. The parameters are the name of the simulation, the name of the compound, and starting and ending milestone numbers.The script makes substitutions in the `tleap-explicit-template.in` script. It also relies on parameter files in the `params/` directory. The `setup-ref.awk` script marks all non-heavy atoms in the system with a "1" in the B-factor column, and all other atoms with "0", to facilitate the application of restraints later with NAMD. Starting structures for each milestone are placed in a directory called `ref` that is coordinate to the `milestoning/solvate` directory and used by the later stages. AMBER `prmtop` files are initially stored in the subordinate `data` directory but need to be copied to the `prmtop` subdirectory within the scratch directory. The `solvate-all` script needs to have access to the list of milestones.

3. The `pyk2-milestoning-minimize-amber.inp` and `pyk2-milestoning-heat-amber.inp` scripts, found in the `milestoning/min-heat` directory, are NAMD scripts for carrying out minimization and heating of each system using the AMBER force field with harmonic restraints on all non-hydrogen atoms in the protein-ligand system. Similar scripts are available for CHARMM. The `pyk2-milestoning-heat-foundry.qsub` is a job submission script for performing minimization and heating for all systems.

4. The `milestoning/equil` directory contains the scripts for running equilibration simulations with a restraining potential to restrain the system to each milestone. The `do-equil-milestoning` script is the master script that calls the `generate_colvar_file.py` script to write a colvars file describing the reaction potential, and then uses NAMD to run the `pyk2-equil-milestoning.inp` script. The `if` statement in this script must be edited to reference the pathway, milestone list, and reference structure for the system. There is a job submission script `pyk2-milestoning-equil-foundry.qsub` for performing equilibration across all milestones. The same directory also contains the scripts for selecting initial frames for the production trajectories. Each initial frame must be on the corresponding milestone; that is to say, the ligand center of mass must be closer to the two anchor points defining the initial milestone than to any others. This script is `select-starting-frames2.py`, called from the job submission script `select-frames2.qsub`. The large `if` statement in `select-frames2.qsub` must be adjusted to reference the pathway and milestone list for the particular system, as well as the reference structure (which should be the same as that used to calculate the original free energy surface using umbrella sampling). The `nsim` variable within the script selects the number of starting frames to choose for each milestone, which should be at least the number of trajectories per milestone.

5. The `milestoning/run` directory contains the scripts for running the production milestoning simulations and logging the results. The main script is `do-milestoning-simulations`, which contains a large `if` statement that references the pathway, milestone list, and reference structure for the system. Each trajectory is run in short sections (e.g. 10 ps) and checked to see if it has left both of the Voronoi cells corresponding to its initial milestone (and thus struck another milestone – see fig. 3(c) in the paper). This script calls `pyk2-run-umbrella.inp`, a NAMD script to run the individual sections of each trajectory, and `check-trajectory.py` which checks to see if the trajectory has hit another milestone. Once it has, all of the sections are automatically joined together and stripped of solvent; the resulting trajectory is placed in the `dcd-joined/` subdirectory of the scratch directory. In addition to the trajectories themselves, a "log file" is produced, which contains information about the starting and ending milestones and the time taken to reach another milestone for each trajectory. This log file is the input for the next scripts, which calculate the dissociation rate, free energy for each milestone, committor values, and other quantities. It has the following format:

   ```
   name start_ianchor start_janchor id end_ianchor end_janchor pdb-id time
   ```

   where `name` is the name of the system, the `(start_ianchor, start_janchor)` pair identifies the initial milestone, the `id` identifies the individual simulation , the `(end_ianchor, end_janchor)` pair identifies the initial milestone, and the `time` is the time in ns to reach the ending milestone.

6. The `milestoning/analysis` directory contains the scripts for analyzing the log file and calculating the free energy profile and dissociation rate. Determining a dissociation rate requires identifying the starting and product milestones for the transition. The starting milestone is usually (1,2), but is specified in the invocation of the script. There may be more than one product milestone, so a list is provided, which may be generated using the `detect-product-milestones.py` script in the `milestoning/analysis` directory, which identifies all milestones whose reference structures have a ligand center of mass greater than a certain distance from the reference structure. The invocation of this script is as follows:

   ```
   ./detect-product-milestones.py name milestone-file reference-pdb distance-cutoff
   ```

   . Alternatively, the product milestones may be selected manually and listed in the file, which is a simple list of anchor pairs.

7. The main script for calculating the free energy profile and dissociation rate is `calculate-dissoc-rate-with-bootstrap-m`
   The invocation is as follows:

   ```
   ./calculate-dissoc-rate-with-bootstrap-multiproduct-connected6.py name log-file milestone-file \
     start_ianchor start_janchor product-milestones-file num-bootstraps
   ```

The `(start_ianchor, start_janchor)` pair identifies the initial milestone; the product milestones file can be generated as described above. The script produces several files:

File name	Description
`{name}-bootstrap-distributions.p`	A Python pickle file containing data on the bootstrap distributions of various quantities
`{name}-fe-from-milestoning-bootstrap`	A file containing the calculated free energies for every milestone, together with bootstrap confidence intervals
`{name}-rate-dist.png`	A plot of the bootstrap distribution of the dissociation rate, similar to figure S4 in the paper.
`{name}-rate-dist-cdf.png`	A plot of the CDF of the dissociation rate bootstrap distribution

A similar script exists for calculating the committor values. The command line is simpler:

```
./calculate-dissoc-rate-with-bootstrap-committors2.py name log-file milestone-file \
 product-milestones-file num-bootstraps
```

The script assumes the first milestone is the starting milestone. The number of bootstraps should be specified as zero, since the script is not really designed for bootstrapping. The script plots the committors in several different ways, but the file `{name}-committors-product-milestones.png` contains a plot similar to figure 7 of the paper. A similar file without the `png` suffix contains the corresponding data in text form.

The script produces a number of plots that reflect figures 7, S3, and S4 in the paper.

8. Once the milestoning simulation is completed and the committors calculated, the transition state milestones may be identified and the corresponding frames from the umbrella sampling simulations selected for transition state analysis. The scripts for doing this are in the `/umbrella-sampling/analysis/transition-state/` directory. The `get-transition-state3.py` script constructs bound and transition state trajectories. The `draw-bound-transition-state.tcl` script draws bound and transition state ensembles as shown in figure 10. The `do-hbonds.tcl` and `organize-hbonds-ts.py` scripts perform a protein-ligand hydrogen bond analysis and organize the results into a table. A similar contact analysis (shown in figure 8 of the paper) can be carried out using the `contact-anal.py` script; this script takes the cutoff for defining contacts as a parameter.

9. The `plot-milestones.py` script identifies new milestones that were discovered during the simulation. It makes a plot portraying old and new milestones, but also produces a file `{name}-new-milestones` with a list of the new milestones. This file can be appended to the original milestones list to create a new milestones list. The `milestoning/run/extract-frames-new-milestones` script may be used to obtain starting structures for these milestones, which can then be solvated in the manner described above.

3 Table of scripts

The following table shows the path of the script used to generate each of the results containing figures or tables, except for tables 3 and 4 and figure 8, which were constructed manually.

Figure	Script
Figure 4	`umbrella-sampling/analysis/free-energy-surface/draw-fe-surfaces4c.tcl`
Figure 5	`other-plots/plot-barrier-height-vs-rate2.py`
Figure 6	`other-plots/overall-rate-plot.py`
Figure 7	`milestoning/analysis/calculate-dissoc-rate-with-bootstrap-committors2.py`
Figure 9	`umbrella-sampling/analysis/transition-state/contact-anal.py`
Figure 10	`umbrella-sampling/analysis/transition-state/draw-bound-transition-state-align.tcl`
Figure 11	`umbrella-sampling/analysis/transition-state/rmsd-hist-transition-state2.py`
Figure 12	`umbrella-sampling/analysis/transition-state/rama-chisq-transition-state2.py`
Table 5	`umbrella-sampling/analysis/transition-state/organize-hbonds-ts.py`
Figure S1	`initial-pathways/plot-iteration-info2.py`
Figure S2	`umbrella-sampling/analysis/draw-sampling2.tcl`
Figure S3	`milestoning/analysis/plot-fe.py`
Figure S4	`milestoning/analysis/calculate-dissoc-rate-with-bootstrap-multiproduct-connected6.png`
Figure S5	`umbrella-sampling/analysis/gb-anal/plot-inte-by-distance2b.py`

4 Sample data

The `sample-data/pyk2-cpd1-amber-tramd-solvated.zip` file contains `prmtop` files and starting structures for the solvated systems corresponding to the PYK2-compound 1 system. The structures need to have their occupancy and B-factor columns marked using either the `setup-ref.awk` or `fix-ref.awk` scripts. The `sample-data` directory also contains the pathway files, milestone files, product milestone files, and log files for each of the three ligands, and reference structures for all five ligands.

5 TRAM scripts

These are in the `tram/` directory. (Should I include these or leave them for another time?)

1. `tica-lag-time-validation-unbiased.py` – performs TICA analysis.

2. `do-cluster.py` – K-means clustering in TICA space

3. `get-ligand-com.py` – determine ligand center of mass in NumPy format for the benefit of the other scripts

4. `pyemma-tram-unbiased2.py` – construct the TRAM model

5. `plot-flux-colored2.py` – calculate and plot dissociation rates as a function of distance cutoff between unbound and bound states

6. `plot-flux-colored-pcca4.py` – construct macrostates from the TRAM model using PCCA and select representative frames