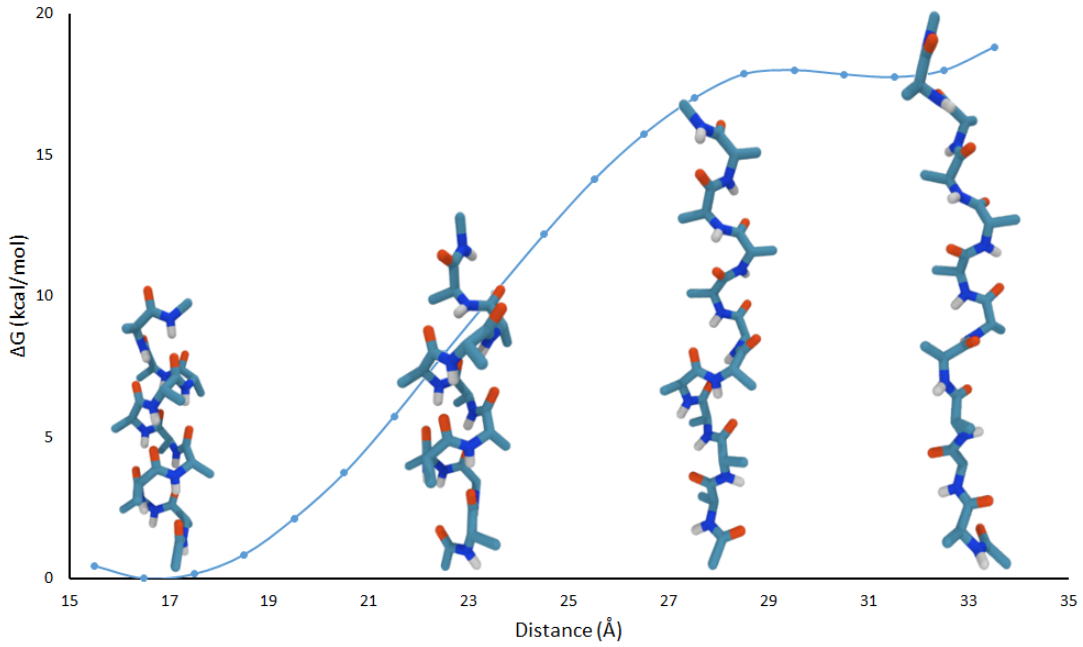
Department of Biochemistry and Molecular Biology
Gordon Center for Integrative Science
The University of Chicago

One-dimensional replica-exchange umbrella sampling



Lu Hong
Mikolai Fajer
Benoît Roux

August 21, 2015

Please visit www.ks.uiuc.edu/Training/Tutorials/ to get the latest version of this tutorial, to obtain more tutorials like this one, or to join the `tutorial-l@ks.uiuc.edu` mailing list for additional help.

Abstract

This tutorial introduces replica exchange umbrella sampling (REUS) simulation method using NAMD. 16 independent replicas of deca-alanine in vacuum is created and then to each replica a harmonic biasing potential is added to the distance between the two termini of the polypeptide. From the results of the simulation the free energy of the system is calculated as a function of the distance using WHAM. This tutorial assumes familiarity with the basics of NAMD, VMD, Tcl and the Linux CLI.

1. Introduction

This tutorial is designed to familiarize you with using NAMD to setup a replica exchange umbrella sampling (REUS) MD simulation, also known as umbrella sampling with Hamiltonian replica exchange molecular dynamics (US/H-REMD).

A central issue in using MD simulations to estimate PMF is obtaining a representative sample of the conformational space. Given the currently available computational powers, however, sufficient sampling is usually difficult to achieve, especially when systems has a large number of degrees of freedom and relevant conformational states are separated by large energy barriers. As a result, the system under simulation may be trapped in a few conformational states throughout the simulation, which results in PMF estimations that are dependent on the starting conformation of the system.

One method used to overcome the problem of unrepresentative sampling is REUS. In a REUS simulation, a sequence of n independent copies of the system is created, and in the ith system, a harmonic biasing potential $V(\xi, i)$ is added to the Hamiltonian of the system. The biasing potential is of the form

$$V(\xi, i) = \frac{1}{2}k(\xi - \xi_i)^2$$

where i is an integer ranging from 0 to $n-1$, k is the force constant, ξ is a collective variable along which the PMF is calculated, and ξ_i is the center of the biasing potential. During the simulation, the energy of adjacent replicas are compared periodically and their biasing potential exchanged with a specific transition probability. When the simulations are complete, the unbiased PMF can be recovered using the Weighed Histogram Analysis Method (WHAM). By sufficient "mixing" of the biasing potential in the replica space during the simulation and restricting the size of the conformational space that each replica can access, the REUS method improves the efficiency and quality of the sampling process.

In this tutorial, you will examine the energetics of stretching a deca-alanine α-helix using REUS. You will simulate a deca-alanine molecule in vacuum with a harmonic biasing potential of the form $V(d, i) = 1/2k(d - d_i)^2$, where d is a collective variable defined by

$$d(t) = \|\mathbf{r}_C(t) - \mathbf{r}_N(t)\|$$

In other words, d is the distance between the C- and N-termini of the deca-alanine molecule. The simulation creates 16 independent replicas of the deca-alanine molecule, numbered consecutively from 0 to 15. The biasing potential in each replica has a force constant of 1 kcal/mol·Å^2,

and for the ith replica, the biasing potential is centered at $(17 + i)$ Å. Each replica is simulated for 10 ns.

This tutorial assumes that you are working on a Linux machine and are familiar with basic Linux CLI and running conventional MD simulations with NAMD. For the analysis part, you should be familiar with the basics of shell scripting, Tcl programming, and VMD scripting. To complete this tutorial, you need

1. `umbrella_1d.zip`, which is provided with this document,
2. NAMD, version 2.10 or later (`www.ks.uiuc.edu/Research/namd/`),
3. VMD, version 1.9 or later (`www.ks.uiuc.edu/Research/vmd/`),
4. An implementation of WHAM, such as `membrane.urmc.rochester.edu/content/wham`.

2. Compiling NAMD

Running replica exchange with NAMD requires Charm++ 6.50 or later built with a low-level runtime system (LRTS). See the "Compiling NAMD" section in the file `notes.txt` that comes with the NAMD source code if you need help on how to compile NAMD. A NAMD of version 2.6 or higher is needed to support the replica exchange module.

In this tutorial, we uses Open MPI for the LRTS (`http://www.open-mpi.org/`). On Debian style Linux systems, Open MPI can be installed by typing

```
sudo apt-get install libopenmpi-dev openmpi-bin
```

into a terminal.

3. Running the REUS simulation

To run the deca-alanine simulation, unzip `umbrella_1d.zip` to your preferred directory. Open `job0.conf` and `job1.conf` and locate the line

```
if { ! [catch numPes] } { source ../umbrella.namd }
```

Change `../umbrella.namd` to `<your_NAMD_directory>/lib/replica/umbrella.namd`. Type in a terminal the following commands:

```
cd <your_working_directory>
mkdir output; cd output
mkdir {0..15}; cd ..
mpirun -np 16 namd +replicas 16 job0.conf +stdout output/%d/job0.%d.log
```

The first three lines set up the `output` folder, which will be used to store the output files from the simulation. Each replica is given a separate folder in `output` with a name matching the index of the replica. The fourth line initiates the simulation. The command `mpirun` calls Open MPI, and the option `-np 16` specifies that the number of MPI ranks is 16. The number of ranks must always be a multiple of the number of replicas. The command `namd +replicas 16` calls NAMD with the replica exchange module and specifies the number of replicas. The command `+stdout output/%d/job0.%d.log` indicates that the session log for the ith replica will be directed to the file `output/i/job0.i.log`.

The simulation can be restarted if desired by using the command:

```
mpirun -np 16 namd +replicas 16 job1.conf +stdout output/%d/job1.%d.log
```

Depending on the local setup or network environment of your machine, a different set of command may be needed to submit your job to the MPI. Consult with your system administrator for more information.

4. The set-up of the REUS simulation

This section explains how to set up a REUS simulation by a detailed discussion of the configuration of the deca-alanine example. The files relevant to the REUS simulations in `umbrella_1d.zip` are

- `alanin.params`
- `alanin.pdb`
- `alanin.psf`
- `alanin_base.namd`
- `colvars.conf`
- `input/`
- `job0.conf`
- `job1.conf`
- `stretch_alanin.conf`

The files `alanin.params`, `alanin.pdb`, and `alanin.psf`, which deal with the struc-

ture and connectivity of the deca-alanine molecule, are common to most MD simulations, and therefore will not be discussed further. The rest of the files are discussed in the order in which they are read by NAMD during the simulations.

4.1. job0.conf

As seen from the last section, when the REUS simulation is submitted to the MPI, NAMD first reads the job0.conf file, which contains two commands:

```
source stretch_alanin.conf
if { ! [catch numPes] } { source ../umbrella.namd }
```

The first line instructs NAMD to read in stretch_alanin.conf, and the second line instructs NAMD to read in umbrella.namd, which contains the NAMD code for running REUS simulations.

4.2. stretch_alanin.conf

The file stretch_alanin.conf sets up the following parameters necessary for the REUS simulations:

- num_replicas: the number of replicas used in the simulation.
- temperature: the temperature used in the simulation.
- steps_per_run: the number of simulation timesteps between exchange attempts. The value of steps_per_run defines the length of a run.
- num_runs: the number of runs before stopping, which should be a multiple of runs_per_frame × frames_per_restart. The product of step_per_run and num_runs is the total number of time steps that each replica will be simulated.
- runs_per_frame: the number of runs between trajectory outputs.
- frames_per_restart: the number of runs between restart outputs.
- namd_config_file: the NAMD configuration file that contains the parameters needed for the conventional MD simulations in between exchanges.

– `output_root`: the directory for output files.

– `input_root`: the directory for the initial configurations of the replicas. If you wish to initiate all replicas with the same initial configuration, you can comment out this line.

The file `stretch_alanin.conf` also defines two new Tcl commands (functions). The first command is called `replica_bias`:

```
proc replica_bias { i } {
  return [list lenpot "centers [expr 17 + $i]"]
}
```

The command `replica_bias` takes in `i`, the index of the replicas, and returns the center of the ith replica, which is $(17+i)$Å. Note that `lenpot` is the name of the harmonic restraint (see section **4.5.**) This function `replica_bias` controls the centers of the biasing potentials. You can change how the biasing potentials are positioned along the collective variable by replacing `17 + $i` with another function of `i`. In general, the centers of the biasing potentials should be positioned to allow sufficient overlap between neighboring biasing potentials.

In some situations it may be advisable to vary the force constant of each biasing potential. This can be done by changing `replica_bias` to

```
proc replica_bias { i } {
 return [list lenpot "centers [expr 17 + $i] forceConstant function_of_i"]
}
```

The second command is `replica_neighbors`:

```
proc replica_neighbors { i } {
  global num_replicas
  if { $i % 2 } { set s -1 } { set s 1 }
  set result {}
  foreach { d } { $s -$s } {
    set j [expr $i + $d]
    if { $j < 0 || $j >= $num_replicas } {
      lappend result $i ; # swap with self
    } {
      lappend result $j
    }
  }
  return $result
}
```

This command takes on one parameter, `i`, the index of the replicas, and returns a list of the indices of the two adjacent replicas. For example, if $i = 3$, then `replica_neighbors` returns 2 and 4. The exceptions are the first and last replicas. For the first replica (replica #0),

the function returns 0 and 1, and for the last replica (in this example, replica #15), the function returns 14 and 15.

4.3. alanin_base.namd

The file alanin_base.namd is read after stretch_alanin.conf. This file contains conventional MD simulation parameters and some performance tuning parameters that are shared by all replicas.

The file alanin_base.namd also sets up the collective variable module:

```
colvars         on
colvarsConfig   colvars.conf
```

The first line enables the collective variables module, which is necessary for the definition of the biasing potentials. The second line provides the name of the configuration file for the collective variables.

4.4. input/

This folder contains the velocity (.vel), position coordinates (.coor), and extended system configuration (.xsc) files for each replica. These files are already equilibrated with the biasing potentials and are used to initiate each replica in the REUS simulation if the input_root parameter in stretch_alanin.conf is pointed to "input/alanin.initial.%d".

4.5. colvars.conf

The file colvars.conf contains parameters used to create the collective variables and biasing potentials for the REUS simulations. Recall that in the deca-alanine example, a harmonic biasing potential is defined on the distance between the C- and N-termini of deca-alanine.

The file colvars.conf has three blocks of code. First, it defines the parameter colvarsTrajFrequency, which is the number of steps in the simulation between which the value of collective variables (colvars) are written to the colvars.traj files.

The second block of code defines a colvar:

```
colvar {
  name length
  distance {
    group1 { atomNumbers 1 }
    group2 { atomNumbers 66 }
  }
}
```

The keyword `name` specifies the name of the colvar. Lines 3 to 6 define the component of `length`, which specifies the functional form of the colvar and the atoms involved in the colvar. The keyword `distance` indicates that the colvar is a distance function. This keyword is followed by two groups of atoms whose distance between each other is to be measured: `atomNumbers 1` and `atomNumbers 66`. These numbers refer to the atom IDs in the PDB file.

The `colvars.conf` file ends with a definition of the harmonic restraint potential:

```
harmonic {
  name lenpot
  colvars length
  centers 17.0
  forceConstant 1.0
}
```

The keyword `name` defines the name of the harmonic restraint. The keyword `colvars` specifies the name of the colvar to which the biasing potential will be applied. On the fourth line, the keyword `centers` defines the center of the biasing potential, and on the fifth line, the keyword `forceConstant` defines the force constant of the biasing potential. Although it is required to specify the center and force constant when defining a harmonic restraint, these values are not used in a REUS simulation because it is overridden by the `replica_bias` function defined in `stretch_alanin.conf`.

The collective variable module of NAMD is quite versatile and its functions well-documented. See chapter 10 of the NAMD User's Guide (Version 2.10) at `www.ks.uiuc.edu/Research/namd/2.10/ug/` for a more detailed discussion of this module.

5. Analysis

In this section you will compute the Gibbs free energy of the deca-alanine system as a function of the distance between the C- and N-termini of deca-alanine using WHAM. This section

assumes that you have completed the simulation in section **3.**

To start the analysis, navigate to the `analysis` folder and type in the command

```
./analysis.sh
```

Make sure that the shell file has the correct permission to execute. If the binary directories of NAMD, VMD, and WHAM are not already added to the PATH environmental variable, you need to edit `analysis.sh` so that the bash shell can locate the needed binary files.

5.1. The set-up of the analysis procedure

The `analysis.sh` file does three tasks, which can also be done manually if you do so desire. First, it calls the program `sortreplicas`, which is found in the NAMD binary directory. This program takes three arguments:

```
sortreplicas <job_output_root> <num_replicas> <runs_per_frame>
```

The program sorts through all the 16 trajectories generated by the replicas and produces 16 sorted trajectories so that the ith sorted trajectory contains only frames with the biasing potential $V(d, i)$. This sorting is necessary for the following WHAM analysis.

Second, `analysis.sh` loads the script `time_series.vmd` into VMD:

```
vmd -dispdev text -e time_series.vmd -args <current_replica> <current_step>
```

This script measures the distance between atom 1 and 66 (i.e., the distance between the two termini) in each frame, and writes the list of distances to files in `time_series/`. The script takes two arguments, the first argument, `current_replica`, is the index of the replicas, which is looped over in `analysis.sh`, and the second argument, `current_step`, is the number of times the simulation has been restarted. The command

```
echo "`cat -n time_series/alanine_0_$i.anal`">time_series/alanine_0_$i.anal
```

adds a column of line numbers to the distance time series, so that the format conforms with the requirement of WHAM. For more information on VMD scripting, see the VMD User's Guide (Version 1.9.2) at `www.ks.uiuc.edu/Research/vmd/current/ug/`.

Third, `analysis.sh` calls WHAM with the command

```
wham 10 40 30 0.01 300 0 meta PMF.out
```

This command instructs WHAM to divide the reaction coordinate d into 30 bins for analysis, each 1 Å long, starting at 10 Å and ending at 40 Å. The convergence tolerance for the WHAM

calculations is 0.01. The temperature of the simulations is 300 K. 0 indicates that no "padding" values should be printed for the PMF. Then WHAM reads in the `meta` file, which contains the location of the distance time series, the center of the biasing potential, and the force constant of the biasing potential for each replica. In the end WHAM writes out its analysis results to the `PMF.out` file. For a complete description of the available arguments and file requirements, see the WHAM manual at `membrane.urmc.rochester.edu/sites/default/files/wham/doc.html`.

5.2. Results

Open `PMF.out`, and the first table contains the PMF we wished to estimate:

```
#Coor Free +/-Prob +/-
10.500000 inf -nan 0.000000 -nan
11.500000 inf -nan 0.000000 -nan
12.500000 inf -nan 0.000000 -nan
13.500000 inf -nan 0.000000 -nan
14.500000 inf -nan 0.000000 -nan
15.500000 0.450333 -nan 0.187453 -nan
16.500000 0.000000 -nan 0.399633 -nan
17.500000 0.161082 -nan 0.304834 -nan
... ...
```

The first column of the table gives the center of the 30 bins requested in the WHAM analysis, and the second column contains the $\Delta G(d)$ function. Some values are labeled `inf` because the simulation does not contain frames where the distance between the two termini falls within the range of the bin. Because no error analysis is requested, the column headed by `+/-` contains only `-nan`. A plot of $\Delta G(d)$ is shown in Figure 1. The plot shows that stretching destabilizes the α-helix conformation of deca-alanine, and the energy difference between the α-helix conformation and the stretched random coil conformation is approximately 18 kcal/mol.

6. Concluding remarks

Replica exchange umbrella sampling is a useful technique for improving PMF estimation. In this tutorial we explored how to use NAMD to implement a simple one-dimensional REUS simulation of deca-alanine in vacuum. The method described in this tutorial can be easily adapted for studying more complex and realistic biological systems and processes, possibly with reaction coordinates other than distance.

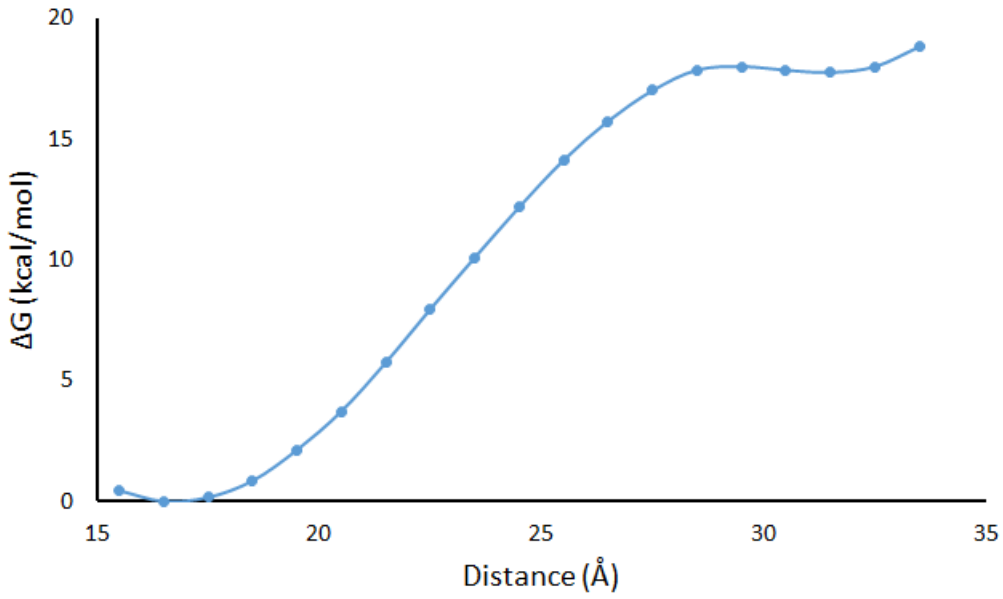


Figure 1: Changes in free energy as a function of the distance between the C- and N-termini of deca-alanine.

One immediate extension of the one-dimensional REUS method is multidimensional REUS simulation, where several colvars are constrained at the same time, creating a replica space of several dimensions. This requires some simple modifications of the `replica_bias` and `replica_neighbors` functions, as well as the colvars and biasing potential definitions. You can find an example of two-dimensional REUS simulation in `<your_NAMD_directory>/lib/replica/umbrella2d`. In this example, two sets of biasing potentials are applied to the length of the deca-alanine molecule, with one set applied to the upper half of the molecule, and the other set to the lower half of the molecule. Higher dimensional REUS simulations allow a more detailed description of the energetics of the system, but also demand significantly more computational resources and time for sufficient "mixing" of the replicas.

One critical issue in doing PMF calculation is convergence. Although REUS allows the system to thoroughly sample one (or a few) degrees of freedom, MD simulations with REUS method may still require long simulation time before convergence, especially for large systems with complicated structures such as a proteolipid complex. This is in part due to the fact that the colvars under investigation may be coupled with some other slow varying degrees of freedom. Many methods are available for error analysis and assessing convergence, such as block averaging and the Monte Carlo Bootstrap error analysis method available in WHAM.