THE JOURNAL OF PHYSICAL CHEMISTRY B
A JOURNAL OF THE AMERICAN CHEMICAL SOCIETY

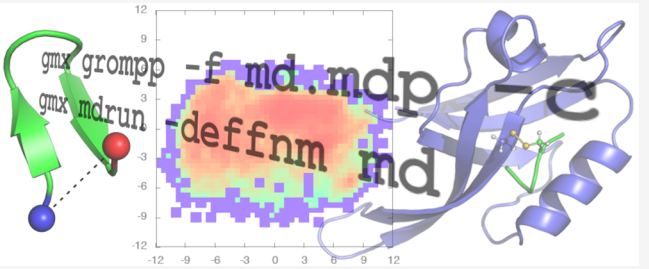
pubs.acs.org/JPCB
Article

Introductory Tutorials for Simulating Protein Dynamics with GROMACS

Justin A. Lemkul*

Cite This: J. Phys. Chem. B 2024, 128, 9418−9435

ABSTRACT: Atomistic molecular dynamics (MD) simulations have become an indispensable tool for investigating the structure, dynamics, and energetics of biomolecules. Continual optimization of software algorithms and hardware has enabled investigators to access biologically relevant time scales in feasible amounts of computing time. Given the widespread use and utility of MD simulations, there is considerable interest in learning essential skills in performing them. Here, we present a set of introductory tutorials for performing MD simulations of proteins in the popular, open-source GROMACS package. Three exercises are detailed, including simulating a single protein, setting up a protein complex, and performing umbrella sampling simulations to model the unfolding of a short polypeptide. Essential features and input settings are illustrated throughout. The purpose of these tutorials is to provide new users with a general understanding of foundational workflows, from which they can design their own simulations.



BACKGROUND AND THEORY

Molecular dynamics (MD) simulations offer the ability to investigate systems of interest with spatial and temporal resolution that exceeds most experimental methods. The accuracy of any MD simulation depends on the quality of the underlying force field, the equation for the energy of a given configuration of atoms and the associated parameters, and the adequacy of sampling. Classical force fields apply principles of Newtonian mechanics to compute the energies of an atomic configuration, from which forces are derived to predict the evolution of the configuration over time.[1] The issue of sampling is approached in a number of ways, from the "brute force" approach of running extremely long, unbiased simulations, to enhanced sampling methods and external biasing potentials along predetermined degrees of freedom. Therefore, familiarity with advantages and limitations of different force fields,[2] knowledge of routine simulation protocols, assessing convergence, and experience in enhanced sampling methods are all important for performing robust and scientifically meaningful MD simulations.

Many software packages have been developed to perform MD simulations and analyze their outcomes, including AMBER,[3−5] CHARMM,[6] GROMACS,[7−10] LAMMPS,[11] NAMD,[12] OpenMM,[13,14] and Tinker,[15] among others. Each of these programs offers different features, force field compatibility, and optimizations for achieving sufficiently fast simulations for systems ranging from thousands to millions of atoms. This tutorial article focuses on guiding the reader through core functions of the GROMACS simulation package in the simulation of polypeptides and proteins.

PREREQUISITES AND INSTALLATION

The exercises described here are designed for use with the current version of GROMACS at the time of writing the article, which is version 2024.1. While there is a general expectation that future versions will retain compatibility and functionality, features and command-line options may change in future versions. GROMACS versioning denotes the year the software was released, with "minor" version changes as a decimal value after the year of the release. Minor versions correspond only to bug fixes in a given release cycle. Thus, it is anticipated that the instructions provided here will be relevant to any version in the 2024 series. Installation instructions for compiling GROMACS on a variety of operating systems and hardware can be found at https://manual.gromacs.org/current/install-guide/index.html. For the purposes of this article, it is assumed that the user has successfully installed an appropriate version of GROMACS. It is also expected that the user is proficient with Unix/Linux file manipulation and navigation, and use of plain-text editors.

These exercises also use the CHARMM36 force field,[16−27] which is a package that is external to GROMACS. It can be obtained from http://mackerell.umaryland.edu/charmm_ff.shtml#gromacs. Download the most recent version of the force field port,[28] which as of writing this article was the July

Received: July 20, 2024
Revised: September 6, 2024
Accepted: September 11, 2024
Published: September 21, 2024

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

2022 version. All the systems presented here are for proteins in an aqueous environment, therefore the specific version of the force field is CHARMM36m, the most recent version of the protein parameter set.[18] Future versions of the force field will be released as it is refined and extended, and their availability will be communicated via the GROMACS user forum (https://gromacs.bioexcel.eu/).

To install the CHARMM36 force field, it is a good practice to place the `charmm36-jul2022.ff` directory in the working directory where the user will be executing all GROMACS commands. It is also possible to install it system-wide, a task that requires administrator privileges.

GROMACS analysis tools produce output files that are plain text and either as raw values or formatted for plotting in the Grace program (https://plasma-gate.weizmann.ac.il/Grace/). Additional analysis described below will be performed with standard Python3 utilities and the Gnuplot program (http://www.gnuplot.info/). Exercise 2 will require the use of PyMOL (https://pymol.org/) to build capping groups and reconstruct missing side chain atoms in the proteins being simulated.

EXERCISES

Here, we present three tutorial exercises to introduce the user to core features of GROMACS in preparing, simulating, and analyzing simulations of polypeptides and proteins. The first exercise is the most detailed, providing step-by-step instructions for simulating a protein in an aqueous solution. The second exercise deals with a protein dimer. The third exercise details how to apply an external biasing potential to unfold a small β-hairpin peptide. This article begins with a brief description of GROMACS file types, conventions, and the correct use of the CHARMM36 force field in GROMACS.

GROMACS File Types.

1. `.gro`: a fixed-format coordinate file with coordinates given in units of nm
2. `.pdb`: a fixed-format coordinate file used by the Protein Databank with coordinates in units of Å
3. `.top`: a system topology, defining the complete contents of a system
4. `.itp`: an "included" topology, defining a specific molecule type, auxiliary parameters, or other topological directives
5. `.mdp`: "molecular dynamics parameter" file that specifies all relevant settings for performing a calculation or simulation
6. `.tpr`: a binary run input file that combines coordinates, topology, all associated force field parameters, and all input settings defined in the `.mdp` file
7. `.edr`: a binary file containing energy data from the calculation or simulation
8. `.xtc`: a binary trajectory file in compressed format containing time, box vector, and coordinate information
9. `.trr`: a high-precision trajectory file containing time, box vector, coordinate, velocity, and force information

GROMACS Command-Line Conventions. All GROMACS programs are executed as modules of the `gmx` program, as will be illustrated below. Input and output options are provided as command-line arguments that start with a dash, e.g. `-f` typically refers to an input file of some sort, `-o` as an output, *etc*. Some arguments are boolean, e.g. `-v` for verbose mode and `-nov` to turn off verbose output. Some arguments require a numerical argument, such as in specifying salt concentration, whereas others take on file names. All GROMACS commands have default values and file names that can be referenced from the manual or by issuing `gmx help <program>`, where `<program>` is the name of the command being executed. The full documentation for GROMACS command-line tools can be found at https://manual.gromacs.org/current/user-guide/cmdline.html.

Notes on Proper Use of the CHARMM36 Force Field. Each force field has certain assumptions inherent in it, which are tied to parametrization methodology and target data, as well as the settings associated with the model physics. It is the latter concern that is worth mentioning here. Shown below are standard settings for using the CHARMM36 force field related to treatment of nonbonded interactions and the use of constraints.

```
; Constraints are applied to bonds involving H atoms
constraints             = h-bonds
; Nonbonded settings
; van der Waals and neighbor searching
cutoff-scheme           = Verlet
vdwtype                 = cutoff
vdw-modifier            = force-switch
rvdw-switch             = 1.0
rvdw                    = 1.2
rlist                   = 1.4
dispcorr                = no
; electrostatics
coulombtype             = PME
rcoulomb                = 1.2
```

These settings should be considered a part of the force field and not altered unless compelling evidence suggests that alternate values should be used. As such, in this exercise, these settings are applied uniformly in all stages of the protocol. The value of `rlist` is shown here as the "conventional" value that is used across simulation packages, but with the GROMACS Verlet nonbonded method, this value is automatically tuned to ensure conservation of energy. The value of `rlist` is only used directly if the user sets `verlet-buffer-tolerance = -1`.

Exercise 1: Protein in Water. *Prepare the Protein Topology.* The first system we will construct is a single protein in water with ions. In this case, the protein is ubiquitin, taken from PDB: 1UBQ.[29] Download the coordinates in PDB format from https://www.rcsb.org/structure/1UBQ. The first step in preparing the system is to define the topology of the protein, set protonation states of side chains and termini, specify disulfide linkages, and build in any missing hydrogen atoms. These processes are carried out in the context of one of the force fields provided with GROMACS, or in our case, downloaded from another source. In classical MD simulations, the bonded structure of a molecule is fixed, therefore there are no protonation or deprotonation events, and cysteine residues will remain in their oxidized (disulfide) or reduced (free thiol) forms. These properties are all defined by the GROMACS tool `pdb2gmx`. This program prompts the user to make several

```
# generate a topology
gmx pdb2gmx -f lubq.pdb -o ubiquitin.gro
```


https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

selections, including choice of force field, treatment of termini, and the water model to be used. Issue the following command:

When prompted, choose the CHARMM36 force field, the position of which may vary depending on installation location. In the working directory, it will appear as choice 1, but if in the top-level GROMACS force field directory, it will be choice 9. Enter the corresponding number and Enter to continue.

Next, choose the water model. In the absence of compelling evidence otherwise, the default water model should always be used as it is the water model for which electrostatic parameters are calibrated. The default water model for CHARMM is the TIP3P model,[30] with modifications to include Lennard-Jones terms on hydrogen atoms.[31,32] The default water model for each force field in GROMACS is always listed first among the available choices, so choose 1 and press Enter.

In the absence of any other command-line options to specify termini, side chain protonation states, or disulfide linkages, `pdb2gmx` will use default settings. Assuming that most biomolecular simulations are performed at approximately neutral pH, `pdb2gmx` will assign charged termini to the N- and C-terminal groups (e.g., $-NH_3^+$ and $-COO^-$) and will assume canonical pK_a values for all amino acids. For ubiquitin, these assumptions are fine but may not be universal for all proteins a user might wish to model. Disulfide bonds are also detected automatically if two cysteine Sγ atoms are within 2 Å, with a distance threshold of 10%, meaning disulfide linkages will be created if these two atoms are within 1.8−2.2 Å.

`pdb2gmx` will also process any crystallographic water molecules that are present in the input coordinate file. In the case of PDB: 1UBQ, there are 58 water molecules. These water molecules will be added to the topology that is written. Upon completion of `pdb2gmx`, the user will have the following files in the working directory:

1. `ubiquitin.gro`: a force field-compliant structure that is protonated according to the assumptions described above.
2. `topol.top`: the topology of the protein, and in this case, the crystallographic water molecules.
3. `posre.itp`: an auxiliary topology file that specifies atoms that are restrained by default and their force constants. This file and its use will be discussed in detail later.

Define the Periodic Cell and Add Solvent. Globular proteins are typically simulated in an aqueous environment that approximates their natural solution environment. To this end, a volume of space around the protein must be defined and subsequently filled with water, ions, and any other soluble molecules of interest. Because condensed-phase simulations are performed under periodic boundary conditions to prevent edge effects and allow for proper energy conservation, the user must define the size and shape of a central image (also called a "unit cell," though this term is distinct from its connotation in X-ray crystallography). Here, we will solvate ubiquitin in a rhombic dodecahedral cell. The two GROMACS programs to achieve this task are `editconf` to define the cell and `solvate` to add water to the accessible volume around ubiquitin.

```
# define the unit cell size and shape
gmx editconf -f ubiquitin.gro -o box.gro -c -d 1.2 -bt dodecahedron
# add solvent (water) to the unoccupied volume
gmx solvate -cp box.gro -cs spc216.gro -o solv.gro -p topol.top
```

When invoking `editconf`, the coordinates of ubiquitin are centered within the defined box (with the `-c` option), and the size of the box is determined from a buffer distance from the maximum and minimum coordinates of the protein; here the buffer is set to 1.2 nm (`-d 1.2`). While a different box shape like a cube may be more convenient to visualize, the use of a rhombic dodecahedron saves on the required volume of the unit cell and number of water molecules, while still achieving the same periodic distance. The shape ("box type") is set with `-bt dodecahedron`.

The `solvate` program takes the solute coordinates (`-cp`, historically for "coordinates of the protein") and uses a pre-equilibrated box of water as the solvent (`-cs spc216.gro`) and tessellates it through the available volume, deleting any water molecules that overlap with protein atoms. The number of water molecules added to the system is stored and then written to the topology file if the user supplies the `-p` option, as shown here. It is important to note that the source of solvent coordinates sometimes causes confusion. While in this example, we are topologically representing water with the TIP3P model, the coordinate file is called `spc216.gro`, because it contains 216 molecules of SPC water.[33] This coordinate file is reasonable for use with any three-point water model, as the solvent will quickly redistribute upon energy minimization and equilibration to achieve the intrinsic structural and dynamic properties of the chosen water model.

Simulations of proteins in aqueous solution also typically include some amount of salt that is added to the water around the protein. Counterions are often added to offset the net charge of the protein to yield an electrically neutral simulation box. The reason for this convention is that the particle mesh Ewald (PME) method[34,35] for computing long-range electrostatic forces requires such neutrality, and artifacts may arise if explicit ions are not added.[36,37] In this case, ubiquitin carries no net charge, so mobile ions may not be strictly necessary. In practice, simulations are often performed to model some experimental or biological conditions, which almost always involve ions.

The `genion` program requires a binary description of the molecular topology, which is assembled by the GROMACS preprocessor, `grompp`. The `grompp` program takes instructions for performing a calculation (energy minimization, MD simulation, *etc.*) and constructs a single input file that contains the coordinates (passed via the `-c` argument), the topology (specified with `-p`) and maps the bonded and nonbonded parameters to each of the atoms. Thus, a single input file, with the extension `.tpr`, contains all the necessary information to carry out a calculation. When adding ions, no calculation is actually performed, but the `.tpr` file contains all the information that `genion` needs to determine which molecules are water, such that they can be replaced by ions. By providing the topology via `-p` to `genion`, the number of added ions is automatically populated and the number of water molecules is similarly updated. The user also specifies the name of cations (via `-pname`, for "positive ion name") and anions (via `-nname` for "negative ion name"), as well as how many ions are to be added. The latter can be accomplished by explicitly specifying how many of each ion to add, or to allow `genion` to calculate these quantities based on a desired concentration (via `-conc`, given in M). Neutralizing ions can automatically be added as part of this concentration with the `-neutral` option, but it is not necessary here. For this example, we will add 100 mM (0.1 M) NaCl to the system with the following sequence of commands:

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

```
# add ions (100 mM NaCl)
gmx grompp -f inputs/ions.mdp -c solv.gro -p topol.top -o ions.tpr
# choose group 13 (SOL) to replace water molecules with ions
gmx genion -s ions.tpr -o solv_ions.gro -p topol.top -pname NA -nname
   CL -conc 0.1
```

The constructed system is illustrated in Figure 1A. It is important to note that the shape of the unit cell is wrapped into a triclinic representation and therefore appears misaligned with

```
# rewrap unit cell
gmx grompp -f inputs/ions.mdp -c solv_ions.gro -p topol.top -o wrap.tpr
gmx trjconv -s wrap.tpr -f solv_ions.gro -o rewrap.gro -pbc mol -ur
   compact
```

the rendered box. It is possible to reconstruct the dodecahedral shape in the following way:

Select 0 (System) for output from `trjconv`. The result of this operation (Figure 1B) is that the unit cell is rewrapped to yield the true rhombic dodecahedral shape. Note that this process is useful only for visualization; the actual physics in any simulation are unaffected by the representation and for optimal performance, coordinates should not be rewrapped in this way prior to performing any calculation or simulation.

Perform Energy Minimization. The process of packing regularly ordered cubic volumes of solvent and subsequently replacing random water molecules with ions may introduce artificiality into the system. Moreover, for a protein structure that has been resolved via X-ray crystallography, it is being exposed to a very different environment than when packed in a lattice. Therefore, bad contacts may exist within the system, as well as suboptimal hydrogen bonding among water molecules and between the water and protein. To resolve these issues, the first molecular mechanics calculation is energy minimization, which moves the atoms along the computed gradient until a threshold maximum force is achieved. Doing so does not guarantee that a system will arrive at a global energy minimum, but it does provide a reasonable starting point for an MD simulation.

Key Concepts and Common Pitfalls

- ☐ `pdb2gmx` writes a topology and prepares a force field-compliant coordinate file
- ☐ `editconf` defines the shape and size of a unit cell
- ☐ `solvate` adds solvent
- ☐ `genion` adds ions, requiring assembly of an input file by `grompp`
- ☐ Updating the coordinates by adding or deleting species requires a corresponding update to the topology, which GROMACS tools handle automatically, but only if requested by the user

We invoke `grompp` to build the input file, and subsequently execute `mdrun` to perform the calculation. The `mdrun` program is the main physics engine in GROMACS and is capable of performing energy minimization, molecular dynamics, and normal modes calculations. As in the generation of a `.tpr` file for the addition of ions, `grompp` will take some set of input instructions from an `.mdp` file; in this case, however, it is important that the settings specify a physically sound model. Aspects such as the nonbonded cutoffs, which are linked to the force field used, must be sensible values. This information is again coupled with the coordinates and topology (passed via `-c`

```
# prepare input .tpr file and execute the calculation
gmx grompp -f inputs/minim.mdp -c solv_ions.gro -p topol.top -o em.tpr
gmx mdrun -deffnm em
```

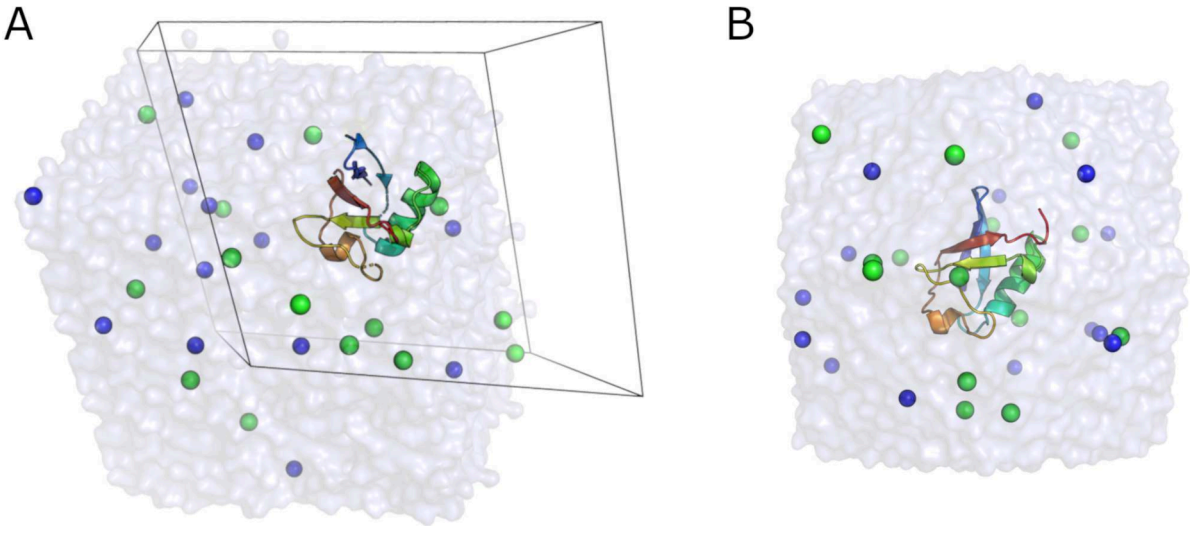


Figure 1. Solvated ubiquitin system, with the protein represented as a cartoon and all water and ions (Na^+ and Cl^- as blue and green spheres, respectively) shown. (A) The default representation with the unit cell shown as a thin black outline. (B) The rewrapped coordinates, showing the rhombic dodecahedral shape.

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

and -p, respectively) to give the binary .tpr file. This file will now be used by mdrun to perform the energy minimization.

The -deffnm option of mdrun stands for "define file name," therefore setting a base file name for all input and output files. Rather than individually specifying names for every type of input and output, this approach is very convenient. After running the commands above, one will achieve a result similar to the following, which is printed both to em.log and to the

```
Steepest Descents converged to Fmax < 500 in 673 steps
Potential Energy  = -3.5638659e+05
Maximum force     =  4.0737012e+02 on atom 447
Norm of force     =  2.1949146e+01
```

terminal. Coordinates of the energy-minimized structure are printed to the em.gro file.

The exact values of potential energy and the number of steps required to converge below the specified force tolerance (here, 500 kJ mol^{-1} nm^{-1}) may vary due to some randomness in adding water and the behavior of the compilers used to compile GROMACS.

The potential energy as a function of minimization step can be extracted from a binary energy file (extension .edr) with the

```
# extract potential energy vs. EM step
gmx energy -f em.edr -o potential.xvg
```

energy command, as shown below. When prompted, select 11 (Potential) and 0 (to terminate input).

Figure 2A shows the progression of the potential energy of the system as a function of energy minimization step. The value drops dramatically in the first few steps as water molecules rearrange to optimize hydrogen bonding. Figure 2B shows an overlay of the crystal structure and the minimized coordinates. There are only subtle changes in the structure of ubiquitin, despite the precipitous drop in the energy of the system. This outcome indicates that minor changes in the protein structure are coupled with much more extensive movements in the water to give a lower energy. As a final note, energy minimization is a nondynamical process. That is, there are no velocities and therefore no sense of kinetics. The atoms move along the gradient to reach a lower-energy configuration.

Perform Equilibration. Having reached a suitably low potential energy, the system must now be equilibrated under the desired statistical mechanical ensemble prior to data collection. That is, the thermodynamic variables of interest must stabilize. In this example, we will apply an isothermal–isobaric (*NPT*) ensemble and will equilibrate in two phases, first under a canonical (*NVT*) ensemble, and then *NPT*. During these simulations, it is typical to apply a harmonic restraining force to the non-hydrogen solute atoms. Given that equilibration is initiated from random velocities, there is a chance that unphysical interatomic collisions could distort the structure and produce an unreliable simulation.

It is in this step that we invoke the posre.itp file written by pdb2gmx. This topology file specifies each of the non-hydrogen atoms in ubiquitin and a force constant in the *x*-, *y*-, and *z*-dimensions (the default value for which is 1000 kJ mol^{-1} nm^{-2}). The application of restraints is determined by a macro in the topology that follows C preprocessor syntax:

```
#ifdef POSRES
#include "posre.itp"
#endif
```

With this construct, the user can selectively choose whether or not to invoke position restraints. When specifying define = -DPOSRES in the .mdp file, the topology macro is satisfied (the POSRES keyword is defined) and the restraint topology is included. When compiling the .tpr file with grompp, an additional coordinate file is provided as input via the -r option. This coordinate file sets the origin of the restraint potential, i.e. where the restraint potential is zero. It is typical that the files specified by -c and -r are identical.

```
# prepare input .tpr file and execute NVT equilibration
gmx grompp -f inputs/nvt.mdp -c em.gro -r em.gro -p topol.top -o nvt.
   tpr
gmx mdrun -deffnm nvt -nb gpu
```

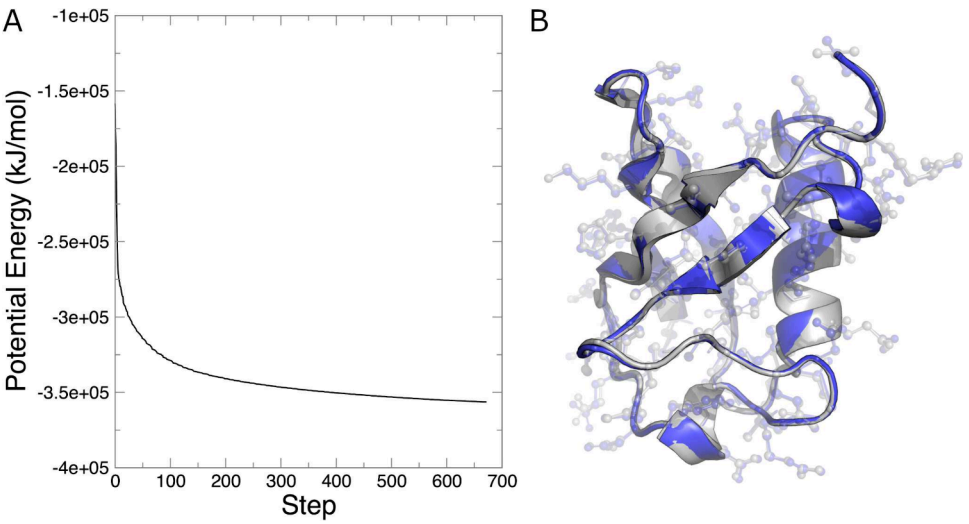


Figure 2. (A) Potential energy as a function of energy minimization step. (B) Overlay of the crystal structure of ubiquitin (gray) and the energy-minimized structure (blue).

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

As before, we invoke `grompp` and `mdrun` to perform the equilibration steps. First, we perform an *NVT* equilibration to stabilize the temperature in the system.

Note that the `mdrun` command also invokes `-nb gpu`, which offloads nonbonded calculations to a graphical processing unit (GPU) to accelerate the simulation. Most modern GPUs are supported by GROMACS and can therefore be used to speed up simulations. This 100 ps *NVT* simulation finished in under 30 s using an NVIDIA Titan Xp GPU (a performance of over 300 ns per day).

```
# extract temperature during NVT equilibration
gmx energy -f nvt.edr -o temperature.xvg
```

Extract the temperature over the course of the *NVT* equilibration by again invoking the `energy` command, selecting 16 and 0 for Temperature and to terminate input, respectively.

With the thermostat settings given here (the velocity rescaling method of Bussi et al.),[38] the temperature converged quickly and oscillated around the desired value of 298 K during the final 50 ps of the simulation (Figure 3A).

Proceed to equilibrating the pressure under an *NPT* ensemble in the same manner. Note here that the previous state is passed

```
# prepare input .tpr file and execute NPT equilibration
gmx grompp -f inputs/npt.mdp -c nvt.gro -r nvt.gro -t nvt.cpt -p topol.
   top -o npt.tpr
gmx mdrun -deffnm npt -nb gpu
```

to `grompp` via the `-t` option, which reads a checkpoint file to process state variable information. Using this option ensures an exact continuation between the two equilibration phases, even when changing input settings as we are here.

Similar to the *NVT* equilibration, extract the pressure over time with the `energy` command, this time selecting 17 and 0 for Pressure and to terminate input, respectively. Pressure is a quantity that fluctuates to a much greater extent than temperature, as is clear from Figure 3B. Over 500 ps, an average

```
# extract pressure during the NPT equilibration
gmx energy -f npt.edr -o pressure.xvg
```

pressure of -10 ± 189 bar was obtained. Over the final 400 ps of the equilibration, this value was -1 ± 170 bar. Both values are statistically indistinguishable from the target value of 1 bar and therefore suggest adequate equilibration.

Note that there is no strict requirement for equilibration to be performed in two phases as shown here. It is possible to equilibrate directly under the desired ensemble, however simultaneously generating random velocities and attempting to regulate pressure can sometimes lead to instability. Therefore, it is often beneficial to first stabilize the temperature, followed by the pressure, as is the protocol here.

Perform an Unbiased MD Simulation. Having sufficiently equilibrated the system, restraints on the protein heavy atoms can now be removed and data collection can begin. This phase of the simulation is sometimes referred to as "production MD" as it is the actual production of usable data. The time frame for the simulation depends on the dynamics of interest; fast motions like side chain rotations may be sampled within a few ns but larger domain motions or folding may require μs−ms of time. Here, we will perform a total of 100 ns of unrestrained MD on ubiquitin in two intervals of 50 ns to illustrate a few key features of GROMACS.

As in previous steps, the input is assembled with `grompp`, reading in the previous state and producing a `.tpr` file that is passed to `mdrun`. Here, the file nomenclature reflects the time interval of the simulation. The input file specifies a simulation length of 50 ns, hence the base of all file names will be `md_0_50`.

Modern simulations are typically performed for much longer times, and GROMACS offers a convenient mechanism to continue a simulation that has previously completed. Having decided that 50 ns is not enough sampling, we extend the simulation to 100 ns with the `convert-tpr` program. As input, it takes the previous `.tpr` file and writes a new one as output that is modified in some way. Here, we change the amount of time specified for the calculation, using `-until`

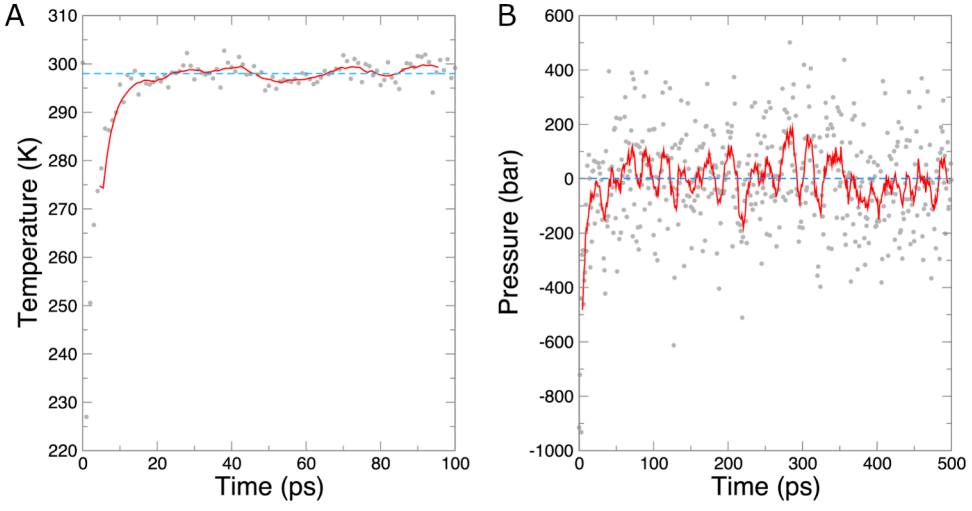


Figure 3. (A) Temperature over time during *NVT* equilibration. Data points are instantaneous values of temperature, the solid red line is a 10-ps running average of the temperature, and the dashed blue line is the target temperature of 298 K. (B) Pressure over time during the *NPT* equilibration. Data points are instantaneous values of pressure, the solid red line is a 10-ps running average of the pressure, and the dashed blue line is the target pressure of 1 bar.

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

Key Concepts and Common Pitfalls

- ☐ The potential energy during energy minimization is generally negative for a condensed-phase system, and its average is not a meaningful quantity
- ☐ Equilibration is intended to reach certain thermodynamic conditions; it does not achieve equilibrium sampling
- ☐ Position restraints require the specification of an origin set of coordinates
- ☐ Some quantities, like pressure, fluctuate more dramatically than other quantities, like potential energy or temperature
- ☐ The ensembles and length of time necessary for equilibration depend on the system itself and the conditions for modeling the system

```
# prepare input .tpr file and execute the MD simulation
gmx grompp -f inputs/md.mdp -c npt.gro -t npt.cpt -p topol.top -o
   md_0_50.tpr
gmx mdrun -deffnm md_0_50 -nb gpu
```

`100000`, meaning the simulation should continue until a time of 100,000 ps (100 ns). It is important to note that while we name the new `.tpr` file `md_50_100.tpr`, it actually contains the state at t = 0 ns, with instructions to perform a full 100 ns of simulation time.

This naming style is not strictly necessary but is somewhat convenient. By default, `mdrun` will append to existing output files, whose names are encoded in the checkpoint `.cpt` files that it writes. If `mdrun` cannot find these files, it will fail with an error. By specifying separate file names, loading the simulation from a previous checkpoint (`-cpi md_0_50.cpt`, corresponding to t = 50 ns), and telling `mdrun` to not append to any files (via `-noappend`), we will obtain output files with the base

```
# increase the number of steps in the .tpr file, then run
gmx convert-tpr -s md_0_50.tpr -o md_50_100.tpr -until 100000
gmx mdrun -deffnm md_50_100 -cpi md_0_50.cpt -nb gpu -noappend
```

name of `md_50_100` that actually correspond only to that time interval. Execute the second half of the simulation as follows:

Analyze the Simulation. The first step in analyzing the simulation outcome is assembling and reimaging the trajectory so that it can be conveniently visualized. That is, we need to concatenate the two `.xtc` files into one continuous file, and subsequently account for periodic boundary effects. These effects include molecules "jumping" from one edge of the central image to another and molecules appearing "broken."

To concatenate trajectory files, use the GROMACS utility `trjcat`. It accepts any number of trajectory files as input (to the `-f` option) and outputs a single trajectory file after removing duplicate frames (the default behavior, which can be changed with the `-cat` option). Following concatenation, periodic boundary effects are removed with the `trjconv` program by putting all of the molecules in the central image (with `-pbc mol`, wrapping the unit cell into a compact shape (here, a rhombic dodecahedron) with `-ur compact`, and centering a desired selection in the unit cell (with `-center`). Note that for more complex systems, additional steps may be necessary to properly reimage the system, but for a system of a protein (or any single solute) in water, this step should solve any imaging issues. After reimaging the system, the solute (here, ubiquitin) may still be rotating around in the center of the unit cell, so applying a fitting algorithm is useful for visualization convenience. Again, we invoke `trjconv` on the reimaged trajectory, this time fitting to a reference structure (contained in the `md_0_50.tpr` file, e.g. the equilibrated coordinates) to remove global rotational or translational motion (with `-fit rot+trans`). The first invocation of `trjconv` centers on the protein (by selecting group 1, Protein) and outputs the entire system (group 0, System). The second `trjconv` step retains only ubiquitin, stripping out all the water and ions because they are not relevant to the analysis demonstrated here.

```
# Concatenate trajectory files
gmx trjcat -f md_0_50.xtc md_50_100.part0002.xtc -o md_0_100.xtc

# Re-image: select 1 (Protein) for centering and 0 (System) for output
gmx trjconv -s md_0_50.tpr -f md_0_100.xtc -o md_0_100_reimage.xtc -pbc
    mol -ur compact -center

# Fitting: select 4 (Backbone) for fitting and 1 (Protein) for output
gmx trjconv -s md_0_50.tpr -f md_0_100_reimage.xtc -o md_0_100_fit.xtc
   -fit rot+trans
```

For fitting, global rotation and translation are removed by fitting to the backbone (group 4), eliminating the influence of high-frequency movements of side chains, and writing out only the protein coordinates (group 1).

Given that the reimaged, fit trajectory now only contains protein atoms, it is convenient to generate matching reference coordinate and `.tpr` files. To do so, we perform a similar `trjconv` operation on `npt.gro`, which holds the

```
# Make matching (protein-only) coordinate and .tpr files
# Select 1 (Protein) for both centering and output
gmx trjconv -s md_0_50.tpr -f npt.gro -o protein.gro -pbc mol -ur
   compact -center

# Select 1 (Protein) for group to be written out
gmx convert-tpr -s md_0_50.tpr -o protein.tpr
```

coordinates at t = 0 ns, and we use `convert-tpr` to extract the protein coordinate and topology information from the existing `.tpr` file. These files can be used for subsequent visualization and analysis.

Visual inspection of the trajectory should always be the first step in analyzing any simulation, to understand what happened over the course of the trajectory. What one observes at this stage

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

may motivate future analysis. Here, we present some general analysis methods that are commonly employed, but the user should always have a plan for analysis that will answer the scientific question(s) of interest.

One may first want to assess how much the structure of the protein changed over time, which is conventionally calculated with root-mean-square deviation (RMSD) against some reference structure (e.g., the equilibrated coordinates or the crystal structure). The `rms` program in GROMACS performs this calculation. Choose group 4 (Backbone) for both fitting and calculation groups. Typically, RMSD does not include side chain atoms, which are often very flexible and whose contributions to

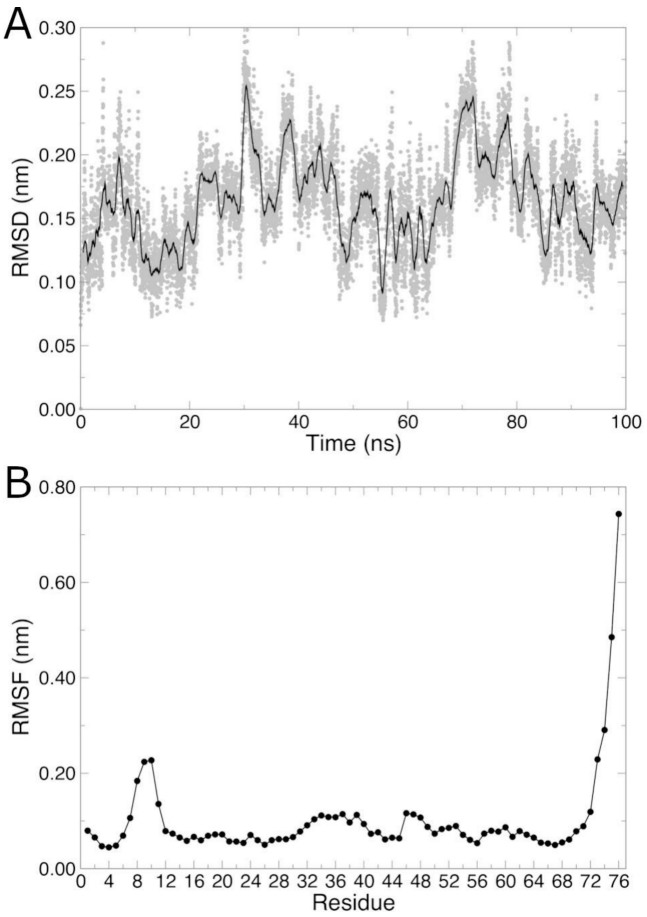


Figure 4. (A) Backbone RMSD over time. Data points are instantaneous values of RMSD, and the solid black line is a 1-ns running average. (B) Cα RMSF for each residue.

RMSD are not necessarily indicative of any real change in the protein structure. Analyzing the backbone RMSD is a direct way of quantifying any deviations in secondary and tertiary structure of the protein. Results of this calculation are shown in Figure 4A.

Similarly, the flexibility of the protein can be assessed using root-mean-square fluctuation (RMSF) calculations, with the `rmsf` program. Here, we will compute the RMSF of Cα atoms (group 3 when prompted), and printing the output per residue (rather than by atom number) with the `-res` option. Formally, the `-res` option causes the atoms of each residue to be averaged to give a single RMSF value per residue, but since we are only choosing one atom per residue, the average is the same as the actual value of the RMSF for that atom, and the output is more conveniently interpreted as residue number, rather than atom number (Figure 4B).

From the RMSD analysis, the protein structure changed very little over time, sampling values between 0.1−0.2 nm (1−2 Å, Figure 4A). Much of that deviation is likely attributable to the very flexible C-terminal residues (Figure 4B). While RMSF formally quantifies the oscillation of coordinates about an average position, residues with high RMSF values often indicate

```
# RMSD: select 4 (Backbone) for both fitting and calculation
gmx rms -s protein.tpr -f md_0_100_fit.xtc -tu ns -o rmsd.xvg

# RMSF: select 3 (C-alpha) for calculation
gmx rmsf -s protein.tpr -f md_0_100_fit.xtc -res -o rmsf.xvg
```

areas of structural change throughout the simulation. Visualization of the trajectory confirms that the last few residues in ubiquitin are highly mobile and deviate from the reference coordinates, but the protein otherwise undergoes very little conformational change, as expected.

Assessing protein structure during an MD simulation often includes characterization of its secondary and tertiary structures. Counting hydrogen bonds can provide insight into both aspects of protein structure. The `hbond` program in GROMACS counts hydrogen bonds based on geometric criteria (donor−acceptor distance $\leq$3.5 Å and hydrogen-donor−acceptor angle $\leq$30° by default). The `hbond` program prompts the user for two selections of atoms containing donors and/or acceptors; these groups must be exactly identical or completely non-overlapping. Here, we will analyze the total number of hydrogen bonds in ubiquitin and those involving just the backbone. It is important to note the nomenclature that GROMACS uses for the polypeptide backbone. The "Backbone" group contains only N, Cα, and C atoms. The "MainChain" group includes all of the Backbone atoms but also includes the O atoms. Lastly, the "MainChain+H" group contains the MainChain atoms as well as the amide H atoms. Given that hydrogen bonds are determined explicitly through the use of an angle involving hydrogen atoms,

```
# Hydrogen bonds
# Choose 1 (Protein) for both groups
gmx hbond -s protein.tpr -f md_0_100_fit.xtc -tu ns -num hb_all.xvg
# Choose 7 (MainChain+H) for both groups
gmx hbond -s protein.tpr -f md_0_100_fit.xtc -tu ns -num hb_bb.xvg
```

computing "backbone" hydrogen bonds actually requires the use of the "MainChain+H" group, not "Backbone." We will also analyze all hydrogen bonds in ubiquitin over time by issuing the following two commands. Results of these calculations are shown in Figure 5A.

Recent versions of GROMACS implement the DSSP algorithm[39] for secondary structure assignment in the `dssp` program.[40] This algorithm assigns secondary structure elements in proteins based on hydrogen-bonding patterns, providing a time evolution of secondary structure throughout a trajectory, as

```
# secondary structure (select 7 for MainChain+H)
gmx dssp -s protein.tpr -f md_0_100_fit.xtc -tu ns -hmode dssp -o dssp.
   dat -num num.xvg
```

shown in Figure 5B. Given the nature of secondary structure assignment, the group analyzed must also be MainChain+H as was the case for analyzing backbone hydrogen bonds. Issuing the

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

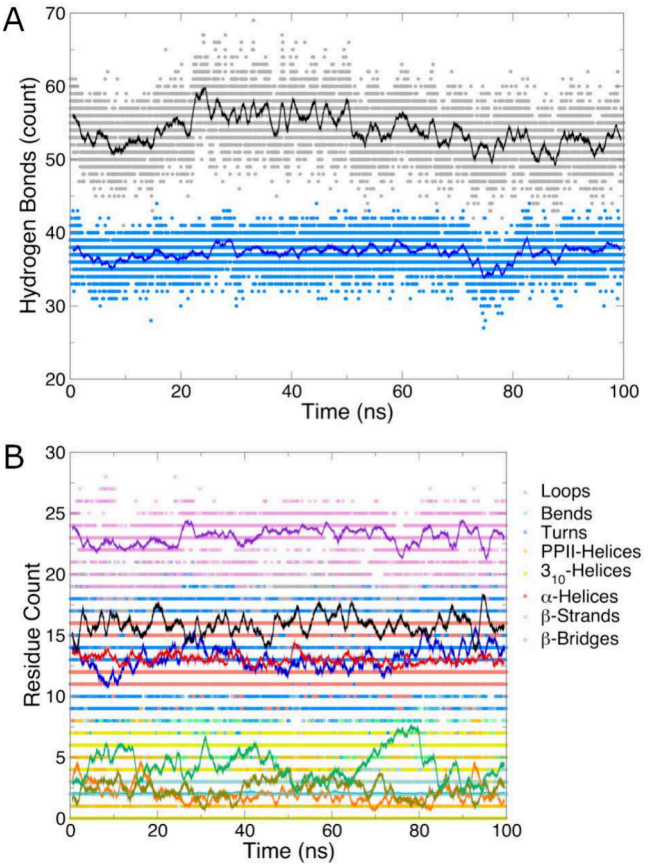


Figure 5. (A) Number of hydrogen bonds over time for all protein atoms and the backbone. (B) Number of residues in each secondary structure type, as assigned by DSSP. For both plots, data points are instantaneous values, and the solid lines are 1-ns running averages. No π-helices were detected in the trajectory, so this time series was omitted for clarity.

following command will analyze secondary structure in the trajectory.

```
# calculate averages of secondary structure elements
gmx analyze -f num.xvg
```

The GROMACS program `analyze` can be beneficial to compute averages and error estimates for time series data.

Table 1. Secondary Structures Assigned by the DSSP Algorithm[a]

Structure	Residue Count
Loops	16 ± 2
Bends	4 ± 2
Turns	13 ± 3
PPII-Helices	2 ± 2
π-Helices	0 ± 0
3$_{10}$-Helices	2 ± 2
α-Helices	13 ± 2
β-Strands	23 ± 1
β-Bridges	2.1 ± 0.3

[a]Listed as the average number of residues of each type. The error bars are the "standard deviation" produced by `analyze` but given the strongly correlated nature of frames saved at 10-ps intervals, are more correctly referred to as a root-mean-square fluctuation.

Calculate the average number of residues of each type by issuing the following command:

The resulting averages and associated error bars are listed in Table 1.

The last analysis that will be demonstrated here is principal component analysis (PCA), a dimensionality reduction technique that in MD simulations is frequently used to determine low-frequency (fundamental) motions in biomolecules. The positional covariance matrix of a selection of atoms is computed, and the matrix is subsequently diagonalized to determine the characteristic eigenvectors and their associated eigenvalues. These eigenvectors can be used to visualize the slow motions in biomolecules. The GROMACS `covar` command computes the covariance matrix; in this example, we will choose the backbone atoms (group 4, "Backbone") rather than including flexible side chains. The matrix is subsequently diagonalized by the `anaeig` command, which produces

```
# PCA: choose Backbone (4) for both fitting and covariance analysis
gmx covar -s protein.tpr -f md_0_100_fit.xtc -o eigenval.xvg -v
   eigenvec.trr -av average.pdb -mwa

gmx anaeig -v eigenvec.trr -f md_0_100_fit.xtc -s protein.tpr -proj
   proj.xvg -2d 2dproj.xvg -extr extreme.pdb -filt filtered.xtc -first
   1 -last 2
```

various outputs, such as the coordinates of extreme positions along each eigenvector and a projection of each trajectory frame along the selected principal components (here, the first two, as specified by `-first 1 -last 2`).

Analysis of the covariance matrix produces 3N eigenvectors, where N is the number of atoms analyzed. In this case, from the 228 backbone atoms, 684 eigenvectors will be produced. Many of these eigenvectors will have very small magnitudes, therefore contributing little to the dynamics of the protein. As shown in Figure 6A, the first eigenvector has a much larger magnitude than all subsequent eigenvectors. It is for this reason that the subsequent projection of trajectory frames and extraction of extreme coordinates along each of these eigenvectors was limited to only the first and second eigenvectors when executing `anaeig`.

The trajectory frames are mapped onto the first two principal components to give an impression of free energy barriers between different conformations. This free energy surface is shown in Figure 6B. For this analysis, a two-dimensional histogram of the eigenvalues along the first two principal components was constructed using a bin width of 0.5 u$^{1/2}$·nm. The resulting histogram was subsequently Boltzmann weighted and offset to zero at the bin of maximal occupancy to produce the free energy surface.

According to this analysis, the dominant motions in ubiquitin reside in the disordered C-terminal region. Principal component 1 corresponds to the C-terminal region swinging in one plane, and principal component 2 corresponds to a twisting in a plane orthogonal to the first (Figure 6C). This finding reinforces the flexibility indicated by RMSF analysis (Figure 4B).

CONCLUDING REMARKS

In this exercise, we have prepared a simulation system of the protein ubiquitin in an aqueous solvent, with mobile ions, performed equilibration and unrestrained MD, and analyzed the simulation outcomes. The general method shown here is applicable to any similar system of a biomolecule in water, but

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

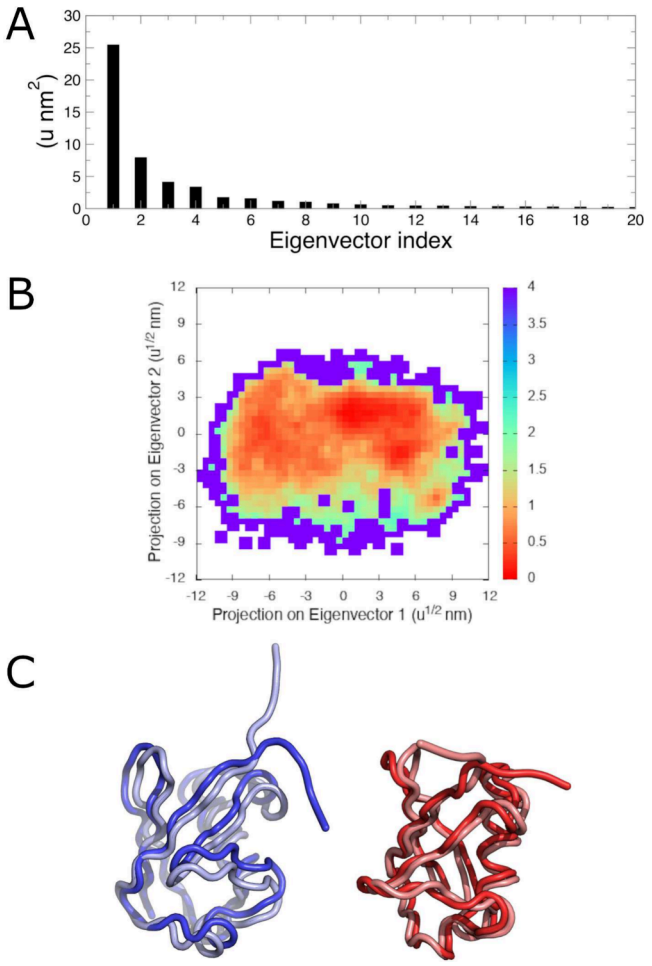


Figure 6. Principal component analysis of ubiquitin. (A) Eigenvalues for each of the first 20 eigenvectors. (B) Free energy landscape of all trajectory frames projected onto the first two principal components. The color bar corresponds to relative free energy in units of kcal mol^{-1}. Relative free energies were computed as $\Delta G = -k_B T \ln[P_i/P_{max}]$, where P_i is the probability in a given bin and P_{max} is the probability in the bin of maximum occupancy. The two-dimensional histogram was generated over a range of {−12,12} with a bin width of 0.5 u$^{1/2}$·nm in both dimensions. (C) Extreme structures along eigenvector 1 (blue) and eigenvector 2 (red).

should not be viewed as the only way to set up, equilibrate, or analyze the system. We did not consider the important issue of convergence, that is, deciding how many replicate simulations to perform and when to terminate each simulation after having achieved adequate sampling. Convergence is a critically important consideration but is well beyond the scope of the tutorial presented here. Performing replicate simulations is also considered compulsory as any given trajectory may be a statistical outlier rather than a true indicator of equilibrium behavior and many independent simulations may be necessary.[41] Additionally, the analysis presented here showcases the routine tasks one may wish to accomplish with GROMACS and should not be viewed as a recipe that needs to be, or even should be, followed. Indeed, we did not define true scientific questions or hypotheses here; it is these considerations that should motivate the analysis of MD trajectories. The workflow given here should provide a reasonable starting point for learning core GROMACS functions and performing simulations of a single biomolecule in water.

Key Concepts and Common Pitfalls

- ☐ MD simulations can be extended using checkpoint (extension `.cpt` files)
- ☐ Periodicity effects must be accounted for prior to visualization; molecules are not actually "broken" as they may initially appear
- ☐ Analysis should generally be planned prior to performing simulations based on the scientific question(s) of interest, but can be adapted as results are obtained
- ☐ GROMACS provides a suite of analysis tools that use similar syntax

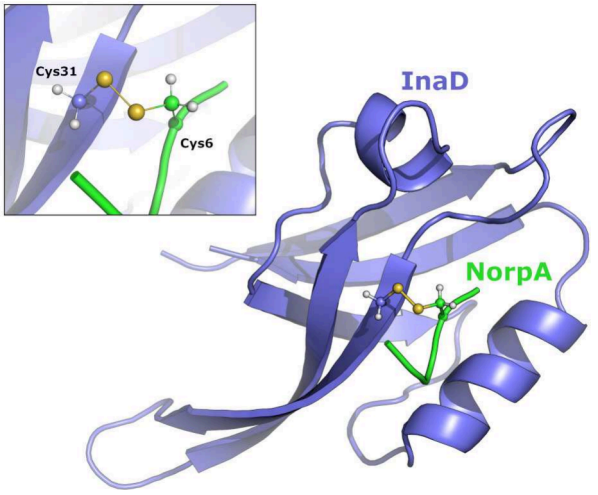


Figure 7. Structure of the InaD:NorpA complex, with InaD colored in blue and NorpA in green. The inset shows the detail of the intermolecular disulfide linkage between Cys31 of InaD and Cys6 of NorpA.

Exercise 2: Protein Dimer in Water. In this exercise, we will prepare a simulation system that is a protein dimer. We will use the N-terminal PDZ domain of InaD in complex with a peptide from the C-terminal region of NorpA, taken from PDB: 1IHJ.[42] As in Exercise 1, download the coordinates in PDB format from https://www.rcsb.org/structure/1IHJ as a starting point.

This system presents several additional levels of complexity compared to the ubiquitin system. First, the PDB file contains four polypeptide chains, because the InaD:NorpA complex crystallizes as a dimer of dimers. Therefore, we must separate out one dimer (either chain A/D or chain B/C) for the simulation. Second, the N- and C-termini of the InaD PDZ domain and the N-terminus of the NorpA peptide are missing residues, so we will have to carefully consider how to treat them. Third, the complex involves a disulfide linkage between the two

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

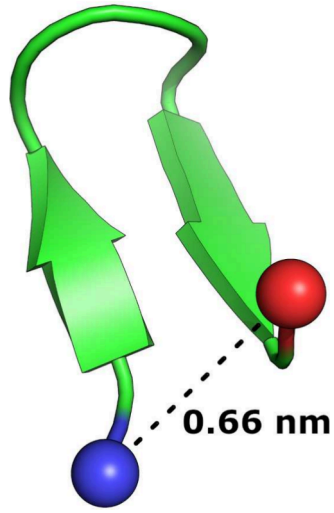


Figure 8. Structure of chignolin, from PDB: 1UAO. The reaction coordinate for umbrella sampling is indicated by the dashed line between the Cα atom of Gly1 and the Cα atom of Gly10, shown as blue and red spheres, respectively.

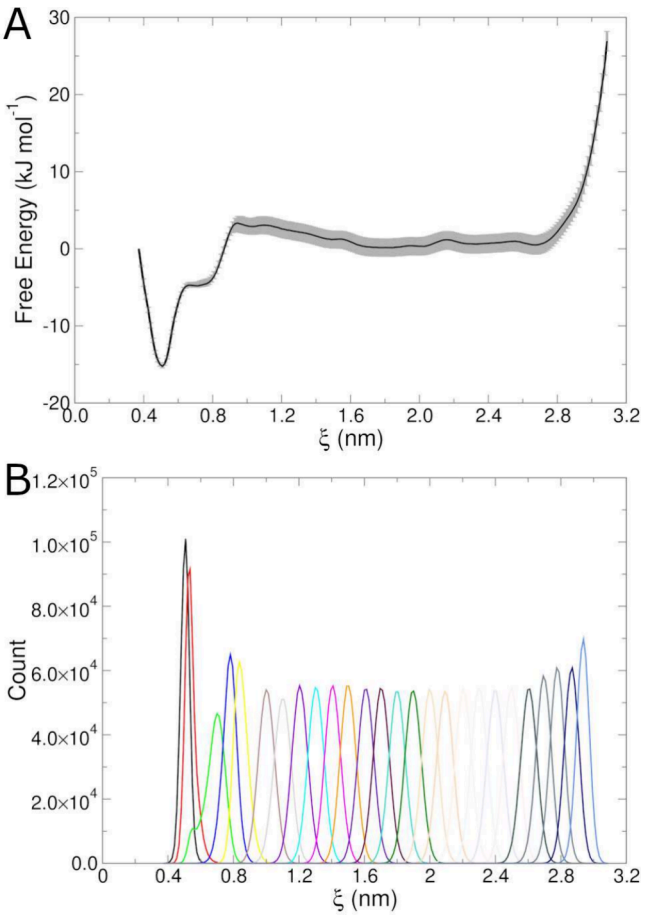


Figure 9. Umbrella sampling results for chignolin unfolding. (A) Free energy surface along the reaction coordinate, ξ, with gray shading indicating error bars from bootstrapping. (B) Histograms of sampling within each window along ξ.

polypeptide chains (Figure 7). Fourth, there are missing side chain atoms in several of the residues.

The fact that the protein complex contains multiple chains is no problem for the `pdb2gmx` program. Provided that each chain is assigned a unique chain identifier (letter or number) or is separated by a `TER` card, `pdb2gmx` can easily generate a topology for any complex of biomolecules. We must ensure that any input coordinate file provided to it meets these requirements.

```
# isolate the chains of interest
grep " A " 1ihj.pdb | grep "ATOM" > 1ihj_chainAD.pdb
echo "TER" >> 1ihj_chainAD.pdb
grep " D " 1ihj.pdb | grep "ATOM" >> 1ihj_chainAD.pdb
echo "TER" >> 1ihj_chainAD.pdb
```

Prepare the Structure. The first step in preparing the InaD:NorpA dimer for simulation is to extract two chains from the PDB file. For this example, we will extract the dimer that is formed by chains A and D. Use standard Linux commands `grep` and `echo` as shown below to isolate these chains.

The crystal structure of InaD:NorpA is missing several terminal residues and a few side chain atoms. These missing entities will need to be modeled in. Rather than model in disordered terminal residues, which are unlikely to impact the dynamics in any functional way, it is commonplace to instead add "capping groups." That is, rather than model the N- and C-termini in their ionized forms, which would be artificial in the case of incomplete chains, neutral groups can be added to mimic the presence of peptide bonds. N-termini are typically capped with acetyl groups and C-termini are typically capped with amide or *N*-methyl amide groups. Here, chain A (corresponding to InaD) is missing residues at both termini and chain D (corresponding to NorPA) is missing residues only at its N-terminus; its C-terminus is the true end of the polypeptide chain and therefore will not be altered.

The crystal structure is also missing several side chain atoms, in residues Asp95 and Phe105. It is common for flexible regions of a protein to have poor electron density in regions with mobile side chains. We will use the PyMOL Mutagenesis Wizard to reconstruct the missing atoms, and the residue building tool to add appropriate capping groups.

To reconstruct missing atoms in Asp95 and Phe105, open the Mutagenesis Wizard in PyMOL (from the top-level menu, Wizard → Mutagenesis → Protein). Click anywhere on Asp95 to select it, and from the right-hand menu, click "No Mutation" and toggle it to `ASP`... and choose `ASP` to correspond to the deprotonated (anionic) form of Asp. Repeat this procedure for Phe105 to build in its missing side chain atoms.

Next, add the capping groups. While this task can be completed with the Mutagenesis Wizard, it will attempt to uniformly cap all chains with the same entitites, which is not what is desired here. Instead, use the residue building feature. From the top-level menu, select Build → Residue → Acetyl. Doing so places an acetyl group in the middle of the viewing

```
fuse /ace///ACE`1/C, /1ihj_chainAD//A/GLY`12/N, mode=1
dele ace
```

area. This group must be translated to a position that is suitable for connecting to the N-terminal residue of InaD (in this case, Gly12). Issue the following command in the PyMOL command prompt to move the acetyl group appropriately and merge its atoms into the chain definition of InaD.

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

```
fuse /nme///NME`24/N, /1ihj_chainAD//A/PHE`105/C, mode=1
dele nme
```

Perform a similar process for the C-terminus of InaD. From the top-level menu, select Build → Residue → *N*-Methyl. Move this group to a suitable position for connection to Phe105 with the following command.

Repeat the process of adding an acetyl group to the N-terminus of NorpA. Save the modified coordinates in PDB format with File → Save Molecule → PDB format (name 1ihj_chainAD_capped.pdb).

The last step of preparation is to modify the added atom names to conform to the expectations of the force field. Atoms added to the amino acid side chains of Asp95 and Phe105 will be correctly named. The acetyl and *N*-methyl groups, however,

```
ATOM      1  N   GLY A  12       2.867   0.935 -21.408  1.00 38.47
ATOM      2  CA  GLY A  12       4.215   0.909 -20.769  1.00 37.68
ATOM      3  C   GLY A  12       4.116   0.957 -19.259  1.00 35.86
ATOM      4  C01 GLY A  12       2.347  -0.436 -21.513  1.00  0.00
ATOM      5  O   GLY A  12       3.438   1.817 -18.703  1.00 39.12
ATOM      6  O02 GLY A  12       3.090  -1.340 -21.917  1.00  0.00
ATOM      7  CH3 GLY A  12       0.912  -0.721 -21.127  1.00  0.00
ATOM      8 1HH3 GLY A  12       0.881  -1.440 -20.321  1.00  0.00
ATOM      9 2HH3 GLY A  12       0.429   0.189 -20.801  1.00  0.00
ATOM     10 3HH3 GLY A  12       0.373  -1.119 -21.973  1.00  0.00
```

require modification. Not only do the atoms have to be renamed, but they must appear as their own residues, named `ACE` and `NME`, to be properly processed by `pdb2gmx`. Open the PDB file obtained from PyMOL in a plain-text editor (like Vim or Emacs) and examine the atoms that have been added at the N-terminus of InaD, which will look something like the following:

```
ATOM      1  CH3 ACE A  11       0.912  -0.721 -21.127  1.00  0.00
ATOM      2 HH31 ACE A  11       0.881  -1.440 -20.321  1.00  0.00
ATOM      3 HH32 ACE A  11       0.429   0.189 -20.801  1.00  0.00
ATOM      4 HH33 ACE A  11       0.373  -1.119 -21.973  1.00  0.00
ATOM      5  C   ACE A  11       2.347  -0.436 -21.513  1.00  0.00
ATOM      6  O   ACE A  11       3.090  -1.340 -21.917  1.00  0.00
ATOM      7  N   GLY A  12       2.867   0.935 -21.408  1.00 38.47
ATOM      8  CA  GLY A  12       4.215   0.909 -20.769  1.00 37.68
ATOM      9  C   GLY A  12       4.116   0.957 -19.259  1.00 35.86
ATOM     10  O   GLY A  12       3.438   1.817 -18.703  1.00 39.12
```

Rename these atoms to conform to the requirements of the force field definition, found in `aminoacids.rtp` in the `charmm36.ff` directory (`C01` becomes `C` and `O02` becomes `O`), rename the residue to `ACE`, and change the residue number from 12 to 11. When finished, the new N-terminus of the InaD protein (chain A) should look like the following:

Similar adjustments are needed at the C-terminus of InaD to account for the *N*-methyl amide cap, which is called `NME` in the GROMACS implementation of the force field. When finished, the C-terminal residues of InaD should look like the following:

Repeat the procedure for fixing the acetyl cap at the N-terminus of NorpA, which is contained in chain D.

Write the Topology and Build the System. At this point, proceeding with preparing the simulation system is almost identical to that of ubiquitin in Exercise 1. In fact, `pdb2gmx` can handle any number of biomolecule chains, and protein complexes therefore require no special consideration in many

```
ATOM    718  N   PHE A 105       0.883   1.451 -14.503  1.00  0.00
ATOM    719  CA  PHE A 105       0.572   1.606 -15.921  1.00  0.00
ATOM    720  C   PHE A 105       0.439   0.262 -16.597  1.00  0.00
ATOM    721  O   PHE A 105      -0.598   0.078 -17.297  1.00  0.00
ATOM    722  CB  PHE A 105       1.699   2.417 -16.630  1.00  0.00
ATOM    723  CG  PHE A 105       1.824   3.899 -16.243  1.00  0.00
ATOM    724  CD1 PHE A 105       0.775   4.525 -15.560  1.00  0.00
ATOM    725  CD2 PHE A 105       2.944   4.645 -16.618  1.00  0.00
ATOM    726  CE1 PHE A 105       0.838   5.885 -15.273  1.00  0.00
ATOM    727  CE2 PHE A 105       3.010   6.004 -16.321  1.00  0.00
ATOM    728  CZ  PHE A 105       1.956   6.623 -15.652  1.00  0.00
ATOM    729  N   NME A 106       1.438  -0.805 -16.466  1.00  0.00
ATOM    730  HN  NME A 106       1.372  -1.397 -17.306  1.00  0.00
ATOM    731  CH3 NME A 106       2.691  -0.147 -16.136  1.00  0.00
ATOM    732 HH31 NME A 106       2.678   0.872 -16.492  1.00  0.00
ATOM    733 HH32 NME A 106       2.836  -0.142 -15.065  1.00  0.00
ATOM    734 HH33 NME A 106       3.515  -0.670 -16.599  1.00  0.00
```

cases. Only a few modifications are necessary to the procedure detailed above for the InaD:NorpA dimer system. Because an intermolecular disulfide linkage is formed between the two proteins, they must be considered in the same topological molecule definition. This convention does not mean the two proteins have to be one molecule in a chemical sense, rather their topologies have to be unified so the proper bonded interactions can be written to reflect the disulfide. Doing so is handled with the `-merge` option of `pdb2gmx`. The user can interactively merge specific chains, or choose to have all chains automatically merged into one topological molecule definition (called a "molecule type" in GROMACS).

Invoke `pdb2gmx` as follows, making the same selections as above for the force field and water model. The command specifies interactive selection of terminal patching via `-ter`,

```
# generate a topology
gmx pdb2gmx -f 1ihj_chainAD_fixed_capped.pdb -o conf.gro -merge all -
   ter
```

which is essential here. Without this option, `pdb2gmx` will attempt to treat the termini in their free amino and carboxylic acid forms, which will lead to an error because no such patching should be applied to capping groups. When prompted, choose `None` for the N-termini of both chains and the C-terminus of InaD. Choose `COO-` for the C-terminus of NorpA.

The terminal output of `pdb2gmx` includes an important distance matrix that is printed in all cases. We use it here as an illustrative example of the disulfide linkage that is present in this structure. The Sγ atoms of Cys31 in InaD and Cys6 in NorpA are at a distance of 0.204 nm. GROMACS uses a database file called `specbond.dat` that defines "special bonds" such as disulfides and linkages to prosthetic groups like heme. The

```
                 HIS16   MET17   CYS31   HIS60   CYS62
                 NE245    SD52   SG156  NE2372   SG386
 MET17    SD52   0.832
 CYS31   SG156   1.824   2.125
 HIS60  NE2372   1.599   2.224   1.320
 CYS62   SG386   0.721   1.199   1.550   1.079
  CYS6   SG773   1.979   2.218   0.204   1.498   1.706
```

reference distance for a disulfide bond is 0.2 nm, and GROMACS uses a 10% tolerance for all distances of this type.

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

Therefore, any two Cys Sγ atoms that are within 0.18−0.22 nm apart will be assigned as being in a disulfide linkage unless

```
gmx editconf -f conf.gro -o box.gro -c -d 1.2 -bt dodecahedron
gmx solvate -cp box.gro -cs spc216.gro -o solv.gro -p topol.top
gmx grompp -f inputs/ions.mdp -c solv.gro -p topol.top -o ions.tpr
gmx genion -s ions.tpr -o solv_ions.gro -p topol.top -neutral -conc 0.1
    -pname NA -nname CL
```

overridden by the user via the `-ss` option, which prompts interactive assignment of cysteine oxidation states.

At this point, the protocol for constructing the system and performing an MD simulation is essentially the same as in Exercise 1, with one small difference. Proceed with defining a unit cell, filling it with water, and adding ions:

The only difference here is the use of the `-neutral` option when running `genion` to add 100 mM NaCl. This option is necessary because the InaD:NorpA complex carries a net −1 charge, unlike ubiquitin, which was electrically neutral. Therefore, `genion` will add one more Na^+ ion than the number of Cl^- ions within the requested 100 mM NaCl to neutralize the charge of the system. The remaining protocol is identical as in the case of ubiquitin in Exercise 1, and therefore will not be repeated here.

Key Concepts and Common Pitfalls

- ☐ `pdb2gmx` can handle multiple biopolymer chains natively
- ☐ `pdb2gmx` cannot build missing atoms (aside from H), therefore sidechains must be complete; it further requires that all atoms present be named according to the expectations of the residue definitions
- ☐ Capping groups are often applied to termini with missing residues to better model their chemical properties; these groups must be provided as separate residues for correct interpretation by `pdb2gmx`
- ☐ Simulating a protein dimer is functionally identical to a single protein or polypeptide chain
- ☐ Counterions should be added to balance the net charge of the solute, and often more ions are added to better model *in vitro* or *in vivo* conditions; these addtions can be made simultaneously

Exercise 3: β-Hairpin Unfolding. *Introduction to Umbrella Sampling.* Unbiased MD simulations may not fully sample the configurational space for any system of interest, rendering it impractical, if not impossible, to calculate free energy differences between states. A central problem is the sparse sampling of high-energy states defining barriers between configurations, or the complete lack thereof. One strategy that has been developed to overcome this limitation is called umbrella sampling.[43] In this method, a reaction coordinate, ξ, is defined (e.g., a distance or angle) and discrete regions along ξ are simulated. These regions are referred to as "windows" and a biasing potential is applied during MD simulations to restrict sampling to a narrow region of space around a target value at the center of each window.

Given the need to sample several windows along the reaction coordinate, umbrella sampling relies on independent MD simulations at each target value. Subsequent analysis is required to extract the potential of mean force (PMF)[44] or a free energy surface that approximates the PMF. A common method for postprocessing umbrella sampling data is the Weighted Histogram Analysis Method (WHAM),[45] which was implemented in GROMACS in 2010.[46] The WHAM method discretizes the reaction coordinate to construct histograms of forces or coordinates in each window, which can subsequently be used to extract the unbiased probability distributions and the free energy values. Given that there are two unknowns, the unbiased probability distributions and the free energies, WHAM solves a system of equations iteratively. Interested readers are directed to an excellent article by Roux on this method and the relationship between umbrella sampling simulations and key concepts in statistical mechanics.[47]

```
head -n 281 1uao.pdb > 1uao_model1.pdb
```

Build the System. For this exercise, we will use the structure of chignolin (PDB: 1UAO, Figure 8).[48] Download the coordinates in PDB format from https://www.rcsb.org/structure/1UAO. This PDB entry contains multiple models from an NMR ensemble, so we will extract the coordinates of the first deposited model with the Linux `head` utility:

As with any other system, the first step in preparing the simulation system is to generate a topology. This task is performed as before with `pdb2gmx`. Note here the use of the `-ignh` option. In some cases, deposited structures do not adhere to typical nomenclature for hydrogen atoms, causing `pdb2gmx` to fail. A convenient solution is to ignore the presence of any hydrogen atoms (via `-ignh`) and have

```
gmx pdb2gmx -f 1uao_model1.pdb -ignh
```

`pdb2gmx` rebuild them according to standard geometry rules for different functional groups. The nonstandard nomenclature for some hydrogen atoms in the chignolin structure prompts the use of this option. Otherwise, the hydrogen atoms have each be renamed in accordance with the expectations of the force field, which can be a tedious and error-prone task.

A key consideration in preparing the unit cell for a system that will be subjected to umbrella sampling is the length of the reaction coordinate that will be sampled. Each configuration that is simulated has to satisfy the minimum image convention, and the box must be sufficiently large such that the reaction coordinate length is less than half of the shortest box vector. This last requirement is due to the need to unambiguously define a distance between the two groups that define the reaction coordinate, rather than attempt to distinguish between distances

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

within the central image and across periodic boundaries. In this case, we will be following a similar protocol as Sumi and Koga,[49] sampling along a reaction coordinate with a maximum value of 3.0 nm.

For efficiency and to reflect the intrinsic (spherical) symmetry of the solute as it extends along the reaction coordinate, we will

```
gmx editconf -f conf.gro -o box.gro -c -box 7 -bt dodecahedron
```

again employ a rhombic dodecahedral box shape. We will set a periodic image distance of 7.0 nm, which should remain at least

```
gmx solvate -cp box.gro -cs spc216.gro -p topol.top -o solv.gro
```

twice as long as the longest reaction coordinate value and be sufficiently large to accommodate the protein without violating the minimum image convention. This value is set with the `-box` option.

Solvating the unit cell proceeds as normal, with no special considerations.

```
gmx grompp -f inputs/ions.mdp -c solv.gro -p topol.top -o ions.tpr
gmx genion -s ions.tpr -o solv_ions.gro -p topol.top -pname NA -nname
   CL -neutral
```

Here, we demonstrate the addition of neutralizing counterions with `genion`. In Exercise 2, we combined the

```
gmx grompp -f inputs/minim.mdp -c solv_ions.gro -p topol.top -o em.tpr
gmx mdrun -nb gpu -deffnm em
```

```
gmx grompp -f inputs/nvt.mdp -c em.gro -r em.gro -p topol.top -o nvt.
   tpr
gmx mdrun -nb gpu -deffnm nvt
```

```
gmx grompp -f inputs/npt.mdp -c nvt.gro -t nvt.cpt -r nvt.gro -p topol.
   top -o npt.tpr
gmx mdrun -nb gpu -deffnm npt
```

`-neutral` option with a specified total concentration of mobile ions. Here, we use the `-neutral` option by itself to demonstrate that it can function without adding more ions beyond what are needed to neutralize the solute charge. In principle, one could specify any desired ionic strength here.

Minimization and equilibration are carried out as in the previous exercises.

Generate Windows. The next step in the protocol is to generate the initial configurations needed to define the sampling windows along the chosen reaction coordinate. In the case of a linear reaction coordinate, such as the Cα-Cα distance we are using here, generating these configurations can be done by applying a biasing potential along this reaction coordinate to induce separation. That is, we will start with the native state in the PDB file (now minimized and equilibrated) and force the polypeptide to elongate along this vector.

To apply this bias, we need to define the two groups that will define the ends of the reaction coordinate, namely the Cα atoms of Gly1 and Gly10. Here, we introduce the concept of "index

```
gmx make_ndx -f npt.gro
```

groups," which GROMACS tools generically use for various input settings and analysis. The program best suited to generating static index groups is called `make_ndx`, which reads in a coordinate or binary topology file and prompts the user to make selections of atoms. Typing "help" at the `make_ndx` prompt gives examples and general syntax for making selections.

```
> r 1 & a CA
> name 17 Gly1_CA
> r 10 & a CA
> name 18 Gly10_CA
> q
```

Here, we will be selecting one atom from each of two residues, assigning each atom to a group. We will also rename the groups for convenience. Below is the `make_ndx` prompt (indicated by the > sign) and the syntax for these selections and renaming

```
; Pull options
pull                    = yes
pull-ngroups            = 2         ; 2 groups, 1 reaction coordinate
pull-ncoords            = 1         ; one reaction coordinate
pull-group1-name        = Gly1_CA
pull-group2-name        = Gly10_CA
pull-coord1-groups      = 1 2
pull-coord1-type        = umbrella  ; harmonic potential
pull-coord1-geometry    = distance
pull-coord1-dim         = Y Y Y     ; 3D bias
pull-coord1-start       = yes
pull-coord1-rate        = 0.005
pull-coord1-k           = 2000
```

operations. The `r` token indicates the residue being selected, `a` indicates the atom name, and the use of `&` signifies the union of these selections (the overlap, i.e. residues that satisfy both selections).

Having created these groups (which are written to a file called `index.ndx` by default), we can use them to apply a biasing force between them to generate initial configurations. The input `.mdp` file settings are shown below.

We are specifying that two groups (`pull-ngroups = 2`) will be used in total to define one pulling coordinate (`pull-ncoords = 1`). It is possible to define multiple biases simultaneously, among any arbitrary number of groups, although here we will use only one bias. The vector along which the bias is defined is specified by the names of the pulling groups (`pull-group1-name` and `pull-group2-name`) and assigning them to a coordinate (`pull-coord1-groups = 1 2`). We will apply a harmonic biasing force (also called an "umbrella potential") via `pull-coord1-type = umbrella` that acts along all three Cartesian dimensions of the vector defining the distance between the two selected Cα atoms (`pull-coord1-geometry = distance` and `pull-coord1-dim = Y Y Y`). We will use the starting distance between the two atoms as the initial value for the pulling with `pull-coord1-start = yes`, but we will set this option differently for umbrella sampling (see below). The biasing force is exerted via an imaginary spring connecting the groups, and it will be elongated at a rate of 0.005 nm ps^{-1} and a force constant of 2000 kJ mol^{-1} nm^{-2} (via `pull-coord1-rate = 0.005` and `pull-coord1-k = 2000`, respectively).

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

```
gmx grompp -f inputs/pull.mdp -c npt.gro -t npt.cpt -n index.ndx -p
   topol.top -o pull.tpr
gmx mdrun -nb gpu -deffnm pull
```

Again, invoke `grompp` to build the binary run input file, this time also supplying the index file that contains the custom groups. Without these definitions, `grompp` will fail with an

```
gmx distance -s pull.tpr -f pull.xtc -n index.ndx -select 'com of group
    17 plus com of group 18' -oall
```

error related to undefined groups. Then, perform the pulling simulation with `mdrun`.

When the simulation is complete, measure the distance between the two Cα atoms over time with the `distance` command. This program allows for a selection of atoms that follows its own syntax that is more complex and flexible than that of `make_ndx`. For examples on the syntax for the `distance`

```
# find_windows.py - search a distance time series and find the
# closest value to a supplied target distance, then print the
# time value corresponding to that distance

import numpy
import os

infile = open('dist.xvg', 'r')
lines = infile.readlines()

count = 0
for d in numpy.arange(0.5,3.1,0.1):
    diff = 100000
    keeptime = 0
    for line in lines:
        if not (line.startswith('#') or (line.startswith('@'))):
            tmp = line.split()
            time = float(tmp[0])
            distance = float(tmp[1])
            tmpdiff = abs(d - distance)
            if (tmpdiff < diff):
                diff = tmpdiff
                keeptime = time
                keepdist = distance
    cmd = "echo 0 | gmx trjconv -quiet -s pull.tpr -f pull.xtc -dump %f
        -o window%d.gro" % (keeptime,count)
    os.system(cmd)
    print ("target value: %.1f best value: %.3f at time: %.1f" % (d,
       keepdist, keeptime))
    count += 1

exit()
```

```
python find_windows.py
```

command or any other GROMACS program that allows for dynamic selections, use `gmx help selections`.

Each frame from the pulling simulation becomes a candidate for the starting structure of an umbrella sampling window. We will be sampling along the reaction coordinate from 0.5−3.0 nm, in intervals of 0.1 nm. Therefore, we need to identify snapshots that have Cα-Cα distances that are as close as possible to the desired values. This search can be scripted using any language of the user's choosing, but here the following Python script is used:

The script also calls the `trjconv` program to write out a coordinate file at the time that corresponds to the best match of the desired distance. These coordinates will subsequently be equilibrated and subjected to the umbrella sampling simulations.

Perform Umbrella Sampling. A total of 26 windows define the desired reaction coordinate distance in the chignolin system. The approach shown here will run the simulations in a loop.

```
pull-coord1-start       = no
pull-coord1-init        = XX
pull-coord1-rate        = 0.0
pull-coord1-k           = 1000
```

There is no requirement to perform the simulations in this way, as each system can be simulated independently.

The `.mdp` settings that define the umbrella potential are nearly identical to those of the pulling simulation performed in the previous section. There are a few notable exceptions. Because we want to uniformly define 0.1 nm intervals for the umbrella windows, we will define these values explicitly. We will use a template file that has a replaceable field to generate real `.mdp` files; note the following settings that differ from the input settings shown above.

Key Concepts and Common Pitfalls

- ☐ A reaction coordinate must be defined in the system that serves as a progress indicator for the behavior of interest
- ☐ In the case of a linear reaction coordinate, the maximum length of the reaction coordinate cannot exceed half the shortest box vector, often requiring larger than normal simulation unit cells
- ☐ Configurations along the reaction coordinate serve as input for independent simulations
- ☐ A linear reaction coordinate can be defined as the vector connecting two groups, which are often defined using index groups
- ☐ The force constant assigned for umbrella sampling simulations determines the width of sampling along the reaction coordinate, but there are no *a priori* methods to know what this value should be

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

Execute the shell script below to replace the XX placeholder with real values. The use of pull-coord1-start = no tells grompp to ignore the initial distance between the defined groups. The reference distance for the biasing potential will thus be set to what is specified in pull-coord1-init, which will take regularly spaced values such as 0.5, 0.6, *etc.* This approach allows us to use coordinate files that are close, but not perfect, matches for the desired distances. For example, if in window 0, the reference distance between the Cα atoms is 0.507 nm, it can still be specified as being the reference for the desired value of 0.5 nm, allowing for uniform separation of window centers.

```
#!/bin/bash

mkdir umbrella
cd umbrella

for (( i=0; i<=25; i++ ))
do
    echo "## RUNNING WINDOW ${i} ##"

    # equilibrate in each window
    gmx grompp -f ../inputs/nvt.mdp -c ../window${i}.gro -r ../window${
       i}.gro -p ../topol.top -o nvt${i}.tpr
    gmx mdrun -nb gpu -deffnm nvt${i}

    gmx grompp -f ../inputs/npt.mdp -c nvt${i}.gro -t nvt${i}.cpt -r
       nvt${i}.gro -p ../topol.top -o npt${i}.tpr
    gmx mdrun -nb gpu -deffnm npt${i}

    # generate .mdp file for umbrella sampling from template
    dist=`echo 0.5 + $i*0.1 | bc | sed "s/^\./0\./g"`
    sed "s/XX/${dist}/g" ../inputs/us.tmpl > ../inputs/us${i}.mdp

    # Equilibrate
    gmx grompp -f ../inputs/us${i}.mdp -c npt${i}.gro -t npt${i}.cpt -n
        ../index.ndx -p ../topol.top -o us${i}.tpr
    gmx mdrun -nb gpu -deffnm us${i}
done
```

The value of pull-coord1-rate is set to zero, as we do not wish to apply any net force to the system to impart displacement. We only want to use the harmonic potential to restrain the distance around a set target. We also use a lower force constant here than what was applied during pulling; the magnitude of this force is somewhat of an empirical choice but as will be shown below, 1000 kJ mol^{-1} nm^{-2} yields good sampling. The larger value applied during window generation was necessary to induce the desired elongation in a practical amount of time.

Analyze Results. The principal output from a series of umbrella sampling simulations is a free energy surface. In this case, we generate the profile from the WHAM algorithm.[45,46] The wham program in GROMACS takes text files as input, each of which must specify the file names of the input .tpr files and either the output pull coordinate or pull force files. The program reads the biasing geometry and reference lengths from the .tpr files and the coordinate or force data from the outputs of mdrun to solve the WHAM equations. The input files can be prepared via a script like the following.

```
# write_wham_files.py - writes the text files needed by gmx wham
# to perform the WHAM analysis.

tprfile = open('tpr-files.dat', 'w')
ffile   = open('pullf-files.dat', 'w')

for i in range(0,26):
    tprfile.write('umbrella/us%d.tpr\n' % (i))
    ffile.write('umbrella/us%d_pullf.xvg\n' % (i))

exit()
```

```
python write_wham_files.py
```

```
gmx wham -it tpr-files.dat -if pullf-files.dat -bsprof -bsres -
   nBootstrap 200
```

```
Determined boundaries to 0.364667 and 3.095285

Warning, poor sampling bin 197 (z=3.06115). Check your histograms!
Warning, poor sampling bin 198 (z=3.07481). Check your histograms!
Warning, poor sampling bin 199 (z=3.08846). Check your histograms!
```

Once the output files are generated, execute the wham program.

Note any messages from wham regarding areas of poor sampling. These warnings can indicate regions that will require additional sampling windows, an adjustment to window spacing, or changes to the force constant used. Here, we receive three warnings that there are windows with poor sampling at the extreme end of the defined reaction coordinate, and actually greater than the desired ending value of 3.0 nm.

The free energy surface generated by the WHAM algorithm is shown in Figure 9A. The global minimum corresponds to roughly 0.5-nm separation, which is expected based on the experimental structural ensemble. Error bars (shaded areas on the free energy surface) come from the default Bayesian bootstrapping algorithm built into the wham program, though

Key Concepts and Common Pitfalls

- ☐ Histograms should be checked to determine if sufficient overlap has been achieved between neighboring windows
- ☐ In the event of poor overlap between neighboring windows, the force constant can be refined or more windows added
- ☐ There is no requirement that all windows use the same force constant
- ☐ Bootstrapping underestimates error values in the free energy profile

https://doi.org/10.1021/acs.jpcb.4c04901
J. Phys. Chem. B 2024, 128, 9418−9435

they are likely an underestimate of the true error.[46] The corresponding sampling histograms are shown in Figure 9B. The distributions are all overlapping to some extent, suggesting adequate coverage along the reaction coordinate. The sharp increase in free energy beyond ξ = 3.0 nm is a result of the few data points in that range, as the `wham` program warned. The chignolin peptide is fully elongated and somewhat strained at this distance, and these values are not likely to be physically meaningful. The overall shape of the curve, as well as the depth and position of the energy minimum, is similar to the result obtained in a previous study.[49]

CONCLUSIONS

Here, we have illustrated three basic workflows in the GROMACS simulation package. Exercise 1 detailed a routine procedure for simulating a globular protein in water. Exercise 2 expanded upon the first protocol and added intermolecular disulfide bonds and capping groups to the preparation steps and planning. Exercise 3 illustrated the use of a biasing potential to unfold a short polypeptide and calculate the free energy change associated with this process. The GROMACS software is capable of many more types of simulations and supports other enhanced sampling methods, but these core skills provide the foundation for other advanced applications.

ASSOCIATED CONTENT

Data Availability Statement

All input files necessary to reproduce the protocols here are provided via GitHub at https://github.com/Lemkul-Lab/gmx_tutorials_jpcb.

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jpcb.4c04901.

A compressed archive of all input files needed to reproduce the protocols described here (ZIP)

AUTHOR INFORMATION

Corresponding Author

Justin A. Lemkul − *Department of Biochemistry and Center for Drug Discovery, Virginia Tech, Blacksburg, Virginia 24061, United States;* orcid.org/0000-0001-6661-8653; Phone: +1 (540) 231-3129; Email: jalemkul@vt.edu

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jpcb.4c04901

Notes

The author declares no competing financial interest.

ACKNOWLEDGMENTS

The author thanks Dr. Anne M. Brown for critical reading of the manuscript, and the many GROMACS users who have provided feedback on similar tutorials and who have expressed the need for updated tutorials on additional topics, which motivated some of the content of this tutorial article. This work was supported by the National Institutes of Health, grant GM133754 and USDA-NIFA (project VA-160211).

REFERENCES

(1) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Oxford University Press: Oxford, 2017.

(2) Lemkul, J. A. *Computational Approaches for Understanding Dynamical Systems: Protein Folding and Assembly*; Elsevier, 2020; p 71.

(3) Case, D. A.; Cheatham, T. E., III; Darden, T.; Gohlke, H.; Luo, R.; Merz, K. M., Jr.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. The Amber biomolecular simulation programs. *J. Comput. Chem.* **2005**, *26*, 1668−1688.

(4) Case, D. A.; Aktulga, H. M.; Belfon, K.; Cerutti, D. S.; Cisneros, G. A.; Cruzeiro, V. W. D.; Forouzesh, N.; Giese, T. J.; Götz, A. W.; Gohlke, H.; et al. AmberTools. *J. Chem. Inf. Model.* **2023**, *63*, 6183−6191.

(5) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An overview of the Amber biomolecular simulation package. *WIREs Computational Molecular Science* **2013**, *3*, 198−210.

(6) Brooks, B. R.; Brooks, C. L., III; MacKerell, A. D., Jr.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; et al. CHARMM: The biomolecular simulation program. *J. Comput. Chem.* **2009**, *30*, 1545−1614.

(7) Berendsen, H.; van der Spoel, D.; van Drunen, R. GROMACS: A message-passing parallel molecular dynamics implementation. *Comput. Phys. Commun.* **1995**, *91*, 43−56.

(8) Lindahl, E.; Hess, B.; van der Spoel, D. GROMACS 3.0: A package for molecular simulation and trajectory analysis. *J. Mol. Model.* **2001**, *7*, 306−317.

(9) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *J. Chem. Theory Comput.* **2008**, *4*, 435−447.

(10) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **2015**, *1*−*2*, 19−25.

(11) Thompson, A. P.; Aktulga, H. M.; Berger, R.; Bolintineanu, D. S.; Brown, W. M.; Crozier, P. S.; in 't Veld, P. J.; Kohlmeyer, A.; Moore, S. G.; Nguyen, T. D.; et al. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comput. Phys. Commun.* **2022**, *271*, 108171.

(12) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781−1802.

(13) Eastman, P.; Swails, J.; Chodera, J. D.; McGibbon, R. T.; Zhao, Y.; Beauchamp, K. A.; Wang, L.-P.; Simmonett, A. C.; Harrigan, M. P.; Stern, C. D.; et al. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology* **2017**, *13*, e1005659.

(14) Eastman, P.; Galvelis, R.; Peláez, R. P.; Abreu, C. R. A.; Farr, S. E.; Gallicchio, E.; Gorenko, A.; Henry, M. M.; Hu, F.; Huang, J.; et al. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *J. Phys. Chem. B* **2024**, *128*, 109−116.

(15) Rackers, J. A.; Wang, Z.; Lu, C.; Laury, M. L.; Lagardère, L.; Schnieders, M. J.; Piquemal, J.-P.; Ren, P.; Ponder, J. W. Tinker 8: Software Tools for Molecular Design. *J. Chem. Theory Comput.* **2018**, *14*, 5273−5289.

(16) MacKerell, A. D., Jr.; Bashford, D.; Bellott, M.; Dunbrack, R. L.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; et al. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *J. Phys. Chem. B* **1998**, *102*, 3586−3616.

(17) Best, R. B.; Zhu, X.; Shim, J.; Lopes, P. E. M.; Mittal, J.; Feig, M.; MacKerell, A. D., Jr. Optimization of the Additive CHARMM All-Atom Protein Force Field Targeting Improved Sampling of the Backbone ϕ, ψ and Side-Chain χ_1 and χ_2 Dihedral Angles. *J. Chem. Theory Comput.* **2012**, *8*, 3257−3273.

(18) Huang, J.; Rauscher, S.; Nawrocki, G.; Ran, T.; Feig, M.; de Groot, B. L.; Grubmüller, H.; MacKerell, A. D., Jr. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nat. Methods* **2017**, *14*, 71−73.

(19) Beglov, D.; Roux, B. Finite Representation of an Infinite Bulk System: Solvent Boundary Potential for Computer Simulations. *J. Chem. Phys.* **1994**, *100*, 9050−9063.

(20) Denning, E. J.; Priyakumar, U. D.; Nilsson, L.; MacKerell, A. D., Jr. Impact of 2′-Hydroxyl Sampling on the Conformational Properties

of RNA: Update of the CHARMM All-Atom Additive Force Field for RNA. *J. Comput. Chem.* **2011**, *32*, 1929−1943.
(21) Hart, K.; Foloppe, N.; Baker, C. M.; Denning, E. J.; Nilsson, L.; MacKerell, A. D., Jr. Optimization of the CHARMM Additive Force Field for DNA: Improved Treatment of the BI/BII Conformational Equilibrium. *J. Chem. Theory Comput.* **2012**, *8*, 348−362.
(22) Klauda, J. B.; Venable, R. M.; Freites, J. A.; O'Connor, J. W.; Tobias, D. J.; Mondragon-Ramirez, C.; Vorobyov, I.; MacKerell, A. D., Jr.; Pastor, R. W. Update of the CHARMM All-Atom Additive Force Field for Lipids: Validation on Six Lipid Types. *J. Phys. Chem. B* **2010**, *114*, 7830−7843.
(23) Klauda, J. B.; Monje, V.; Kim, T.; Im, W. Improving the CHARMM Force Field for Polyunsaturated Fatty Acid Chains. *J. Phys. Chem. B* **2012**, *116*, 9424−9431.
(24) Hatcher, E. R.; Guvench, O.; MacKerell, A. D., Jr. CHARMM Additive All-Atom Force Field for Acyclic Polyalcohols, Acyclic Carbohydrates, and Inositol. *J. Chem. Theory Comput.* **2009**, *5*, 1315−1327.
(25) Hatcher, E.; Guvench, O.; MacKerell, A. D., Jr. CHARMM Additive All-Atom Force Field for Aldopentofuranoses, Methyl-aldopentofuranosides, and Fructofuranose. *J. Phys. Chem. B* **2009**, *113*, 12466−12476.
(26) Raman, E. P.; Guvench, O.; MacKerell, A. D., Jr. CHARMM Additive All-Atom Force Field for Glycosidic Linkages in Carbohydrates Involving Furanoses. *J. Phys. Chem. B* **2010**, *114*, 12981−12994.
(27) Vanommeslaeghe, K.; Hatcher, E.; Acharya, C.; Kundu, S.; Zhong, S.; Shim, J.; Darian, E.; Guvench, O.; Lopes, P.; Vorobyov, I.; et al. CHARMM General Force Field: A Force Field for Drug-like Molecules Compatible with the CHARMM All-Atom Additive Biological Force Fields. *J. Comput. Chem.* **2010**, *31*, 671−690.
(28) Wacha, A. F.; Lemkul, J. A. charmm2gmx: An Automated Method to Port the CHARMM Additive Force Field to GROMACS. *J. Chem. Inf. Model.* **2023**, *63*, 4246−4252.
(29) Vijay-kumar, S.; Bugg, C. E.; Cook, W. J. Structure of ubiquitin refined at 1.8 Å resolution. *J. Mol. Biol.* **1987**, *194*, 531−544.
(30) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.* **1983**, *79*, 926−935.
(31) Durell, S. R.; Brooks, B. R.; Ben-Naim, A. Solvent-Induced Forces between Two Hydrophilic Groups. *J. Phys. Chem.* **1994**, *98*, 2198−2202.
(32) Neria, E.; Fischer, S.; Karplus, M. Simulation of activation free energies in molecular systems. *J. Chem. Phys.* **1996**, *105*, 1902−1921.
(33) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; Hermans, J. In *Intermolecular Forces*; Pullman, B., Ed.; 1981; p 331. DOI: 10.1007/978-94-015-7658-1_21.
(34) Darden, T.; York, D.; Pedersen, L. Particle mesh Ewald: An N·log(N) method for Ewald sums in large systems. *J. Chem. Phys.* **1993**, *98*, 10089−10092.
(35) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G. A smooth particle mesh Ewald method. *J. Chem. Phys.* **1995**, *103*, 8577−8593.
(36) Hub, J. S.; de Groot, B. L.; Grubmüller, H.; Groenhof, G. Quantifying Artifacts in Ewald Simulations of Inhomogeneous Systems with a Net Charge. *J. Chem. Theory Comput.* **2014**, *10*, 381−390.
(37) Kopec, W.; Gapsys, V. Periodic boundaries in Molecular Dynamics simulations: why do we need salt? *BioRxiv* **2022**, DOI: 10.1101/2022.10.18.512672.
(38) Bussi, G.; Donadio, D.; Parrinello, M. Canonical sampling through velocity rescaling. *J. Chem. Phys.* **2007**, *126*, 014101.
(39) Kabsch, W.; Sander, C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* **1983**, *22*, 2577−2637.
(40) Gorelov, S.; Titov, A.; Tolicheva, O.; Konevega, A.; Shvetsov, A. DSSP in GROMACS: Tool for Defining Secondary Structures of Proteins in Trajectories. *J. Chem. Inf. Model.* **2024**, *64*, 3593−3598.
(41) Knapp, B.; Ospina, L.; Deane, C. M. Avoiding False Positive Conclusions in Molecular Simulation: The Importance of Replicas. *J. Chem. Theory Comput.* **2018**, *14*, 6127−6138.
(42) Kimple, M. E.; Siderovski, D. P.; Sondek, J. Functional relevance of the disulfide-linked complex of the N-terminal PDZ domain of InaD with NorpA. *EMBO Journal* **2001**, *20*, 4414−4422.
(43) Torrie, G.; Valleau, J. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *J. Comput. Phys.* **1977**, *23*, 187−199.
(44) Kirkwood, J. G. Statistical Mechanics of Fluid Mixtures. *J. Chem. Phys.* **1935**, *3*, 300−313.
(45) Kumar, S.; Rosenberg, J. M.; Bouzida, D.; Swendsen, R. H.; Kollman, P. A. The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method. *J. Comput. Chem.* **1992**, *13*, 1011−1021.
(46) Hub, J. S.; de Groot, B. L.; van der Spoel, D. g_wham − A Free Weighted Histogram Analysis Implementation Including Robust Error and Autocorrelation Estimates. *J. Chem. Theory Comput.* **2010**, *6*, 3713−3720.
(47) Roux, B. The calculation of the potential of mean force using computer simulations. *Comput. Phys. Commun.* **1995**, *91*, 275−282.
(48) Honda, S.; Yamasaki, K.; Sawada, Y.; Morii, H. 10 Residue Folded Peptide Designed by Segment Statistics. *Structure* **2004**, *12*, 1507−1518.
(49) Sumi, T.; Koga, K. Theoretical analysis on thermodynamic stability of chignolin. *Sci. Rep.* **2019**, *9*, 5186.