Running Gaussian-accelerated Molecular Dynamics Simulations in NAMD [Article v0.1]

Haley M. Michel[1], **Marcelo D. Polêto**[1], **Justin A. Lemkul**[1,2]

[1]Department of Biochemistry, Virginia Tech, Blacksburg, Virginia, 24061, United States; [2]Center for Drug Discovery, Virginia Tech, Blacksburg, Virginia, 24061, United States

This LiveCoMS document is maintained online on GitHub at https://github.com/Lemkul-Lab/gamd_tutorial.git; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated January 21, 2025

Abstract Gaussian-accelerated molecular dynamics (GaMD) simulations are an advanced technique that enhances the sampling of configurational space by applying biasing potentials that reduce energy barriers, enabling faster exploration of the free energy landscape. This tutorial demonstrates the application of GaMD to the alanine dipeptide, serving as an accessible model system, and guides users through all GaMD simulation stages: conventional MD, GaMD equilibration, GaMD production, and reweighting. Users will gain practical insights into the preparation of input files, monitoring of GaMD convergence, and analysis of free energy profiles using PyReweighting. We make a particular effort to connect the underlying theory with the GaMD workflow. This tutorial is intended for users with prior molecular dynamics experience, Linux and command-line navigation, and with basic Python knowledge. The step-by-step instructions and accompanying scripts aim to streamline the GaMD workflow, making it accessible for the broader research community to explore enhanced sampling for a range of biomolecular systems.

***For correspondence:**
jalemkul@vt.edu (JAL)

1 Introduction

Molecular dynamics (MD) simulations play a key role in studying the dynamics and chemical properties of molecular systems with atomistic resolution. Most published MD studies employ unbiased simulations, that is, they do not apply any external biasing forces to the system and thus, the dynamics of the system freely sample accessible conformational states across the free energy landscape without external influence. However, a major limitation in this approach is the timescale of relevant conformational transitions, requiring a great computational cost to sample motions of interest in the μs - ms range, which are especially common for large or complex molecular systems. To overcome this limitation, *i.e.* enhanced sampling methods have been developed to add biases to aid in sampling rare states. The vast majority of these methods, such as metadynamics and umbrella sampling, require a detailed understanding of the system's characteristics and a clear description of the transition path between states, also known as a collective variable (CV) or reaction coordinate. By biasing across a CV, the motion dimensions are simplified, constraining the conformational pathway but allowing for more sampling along the chosen CV. Although these methods are generally effective, the transition events of some systems are difficult to describe through one or even a few CVs. As a potential solution, enhanced sampling methods such as accelerated molecular dynamics (aMD) and more recently its derivative, Gaussian-accelerated MD (GaMD), have been developed to

accelerate conformational sampling without having to limit the simulation to a predetermined CV pathway.

During GaMD simulations, biases are added to the potential energy (or subsets of the potential energy) of the system, thereby lowering energy barriers and facilitating transitions between conformational states. As a result, GaMD simulations allow for sampling of rarer states in shorter simulation times than otherwise would be necessary in unbiased simulations, typically referred to as "conventional" simulations in previous GaMD literature and in this tutorial. Originally implemented in the AMBER MD suite [1], GaMD is implemented in NAMD [2], OpenMM [3], and TINKER [4], and has now been combined with other enhanced sampling methods such as replica exchange [5] and weighted-ensemble approaches [6]. The original GaMD algorithm has also been extended to apply selective boosting to enhance ligand binding in protein-ligand complexes in LiGaMD [7] and now also boosts protein residues with direct ligand interactions in LiGaMD2 [8], bound peptides in Pep-GaMD [9], and protein-protein interactions PPI-GaMD [10].

Despite these advances, GaMD, like any enhanced sampling algorithm, requires a detailed understanding of not only how to execute the simulations but what the underlying theory is to get the most benefit from applying the technique to the system of interest. As such, here, we present a tutorial for employing the GaMD method to alanine dipeptide using the NAMD simulation suite. After reading and completing the tutorial, the user should be able to:

1. Identify key elements of the theory underlying GaMD simulations
2. Prepare and run GaMD simulations using NAMD
3. Reweight and analyze the GaMD free energy profiles using the PyReweighting toolkit

2 Prerequisites

2.1 Background Knowledge and Experience

The GaMD tutorial presented here is not intended for beginners wanting to learn the basics of MD simulations. Users should already have familiarity interacting with Linux or Unix command-line interfaces, and using plain-text editors such as VIM or EMacs to edit and view files. This tutorial is focused on introducing theoretical concepts of GaMD and how to execute such simulations. As such, this tutorial expects users to be familiar with routine simulations in NAMD prior to attempting this tutorial. The reweighting process will require basic knowledge of Python, including how to set up a conda environment and install common packages such as NumPy [11] and SciPy [12]. Additional knowledge in analyzing MD simulations will be beneficial.

Throughout this article, user commands that are to be issued via the terminal are written in `monospace font` with `$` to indicate the command prompt, which should not be entered as part of the command itself. Program and file names will also be written in `monospace font` to distinguish them from the remainder of the content in the article. There will be examples of code and data in files represented in blocks that look like this:

Listing 1. example_code_box.sh

```
# helpful comment here
command line here
```

2.2 Software Requirements

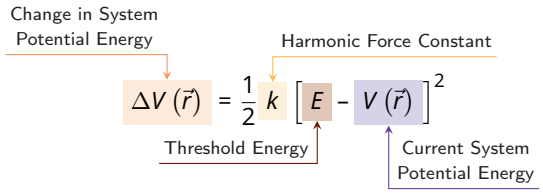
The GaMD algorithm is implemented in NAMD versions 2.12 and above, which can be freely downloaded for academic use from the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign (https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=NAMD). NAMD versions prior to 2.12 will be unable to run the simulations and therefore cannot be used with this tutorial. Once simulations are complete, access to CatDCD (https://www.ks.uiuc.edu/Development/MDTools/catdcd/) and an MD analysis software will be required. In this tutorial, we will reference AmberTools version 22 [13], CPPTRAJ [14], and CHARMM [15]. Free energy surfaces will be constructed using the PyReweighting Toolkit, which can be obtained from the Miao lab GitHub page (https://github.com/MiaoLab20/pyreweighting). A conda environment will be required for reweighting and should be built with Python version 3.10. Several other Python packages will need to be installed within the environment. NumPy 1.23.3, SciPy 1.9.1, and Matplotlib 3.6.1 will be the most important but we have also exported the conda environemnet used in this tutorial as a YAML (.yml) file in the associated GitHub repository. We note that the tutorial has been tested with these versions, and that while others may work just fine, we cannot make any guarantees.

3 Theory

3.1 Basic GaMD Theory

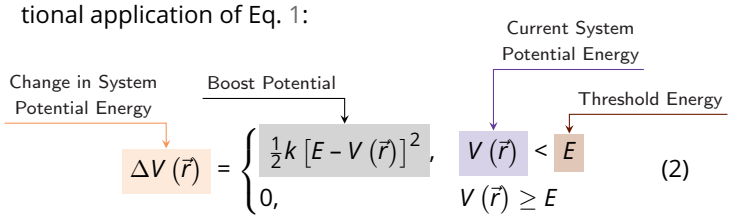
GaMD is based on the same theory as accelerated MD (aMD) [16], a technique in which a biasing potential is applied to the system to lower energetic barriers and increase the probability of conformational changes [2, 17]. The biasing term, called the boost potential or $\Delta V_{aMD}(\vec{r})$, is applied to the potential energy of the system, $V(\vec{r})$, only when it falls below a specified threshold energy, E. Due to the large magnitude of the aMD boost potential, recovery of

the unbiased potential energy surface is dominated by simulation frames corresponding to larger boost magnitudes, resulting in higher statistical noise upon reweighting than other enhanced sampling methods [18, 19]. To overcome this limitation, GaMD makes use of a boost potential that is restricted to a Gaussian, or normal, distribution in which data points are clustered more tightly around the mean value, making statistical outliers less likely. As a result, the boost is harmonic and the shape of the free energy landscape is better conserved, allowing for more accurate recovery of the unbiased free energy landscape. The boost potential can be calculated via Eq. 1:



(1)

where k is the harmonic force constant and E is the threshold energy. The boost is applied based on the relationship of the potential energy of the system at the current simulation step, $V(\vec{r})$, to E. When the system potential drops below E, the boost is applied and added to $V(\vec{r})$ at that step. If $V(\vec{r})$ is equal to or above E, no boost is applied to the system. This relationship is shown in Eq. 2, which is essentially the conditional application of Eq. 1:



(2)

The modified potential energy at any point in the simulation can thus be defined as $V^*(\vec{r}) = V(\vec{r}) + \Delta V(\vec{r})$. Both the threshold energy, E, and the harmonic force constant, k are calculated based on statistics obtained from short, conventional MD simulations that will be discussed in Section 3.2. E and k are then refined during a GaMD equilibration stage prior to a production stage, during which E and k are held constant. The change in system potential energy, $\Delta V(\vec{r})$, is then dependent on the difference between E and the current system potential energy, $V(\vec{r})$.

The statistical information gathered consists of the minimum and maximum potential energies, V_{min} and V_{max} respectively; the average potential energy, V_{avg}; and the standard deviation of the potential energy, $\sigma_{\Delta V}$. Using these statistical values, E can be calculated in two different ways dependent on user-specified options (Eq. 3):

$$\begin{aligned} E &= V_{max} && \text{“lower bound”} \\ E &= V_{min} + \frac{1}{k} && \text{“upper bound”} \end{aligned} \tag{3}$$

The choice of E is system-dependent, with the primary consideration being the system size. In theory, systems containing many particles $\left(> 10^6\right)$ may need larger boost values to more comprehensively sample the free energy landscape and therefore can benefit from setting E equal to $V_{min} + \frac{1}{k}$. This approach has been found useful in studying the dynamics of large CRISPR-Cas systems [20]. However, the majority of systems may only require E to be set to V_{max}, to avoid adding too large of a boost to the system, resulting in a flattened energy surface and subsequent inaccurate reweighting. More information on how these threshold energy equations are derived, as well as how the harmonic force constant is determined, can be found in the Section 3.2.

The boost can be applied to the potential energy equation in a few different ways. The potential energy equation (Eq. 4) consists of several terms describing the interactions within the system that are summed to determine the total potential energy of the system at a given time during the simulation.

$$\begin{aligned} V(\vec{r})_{total} = &\sum_{bonds} V_r + \sum_{angles} V_\theta + \sum_{dihedrals} V_\tau \\ &+ \sum_{nonbonded} V_{van\ der\ Waals} + \sum_{nonbonded} V_{electrostatics} \end{aligned} \tag{4}$$

The current version of GaMD in NAMD allows users to apply the boost to: (1) only the total potential energy, $V(\vec{r})_{total}$ (also denoted ΔV_p in some references); (2) only the dihedral term, V_τ, (also denoted ΔV_D in some references); or (3) both the total and dihedral terms, which is called “dual-boost” in the existing literature. Most studies employing GaMD employ the “dual-boost” as it has been shown to provide greater acceleration and thus more sampling compared to the other biasing approaches [16].

3.2 Not-So-Basic Mathematical Details

Section 3.1 presented the general mathematical framework behind how the GaMD procedure actually works, however, there are a few other mathematical considerations for how some of these equations and variables are actually used.

As discussed above, the goal of the GaMD method is to smooth the potential energy surface such that the system more readily overcomes energy barriers while maintaining the overall shape of the surface. Doing so allows high and low energy states to be distinguished upon recovery of the original free energy surface (FES). For this to work, the boost potential applied to the system must meet several criteria: it

must (1) ensure that the biased FES maintains the same overall shape as the original FES, (2) lower the energetic barriers on the surface, and (3) maintain a sufficiently narrow distribution that allows for robust reweighting. To break this concept down further, if we consider two arbitrary points, $V_1(\vec{r})$ and $V_2(\vec{r})$ on the original energy surface (Figure 1), we can think of the first two criteria as follows.

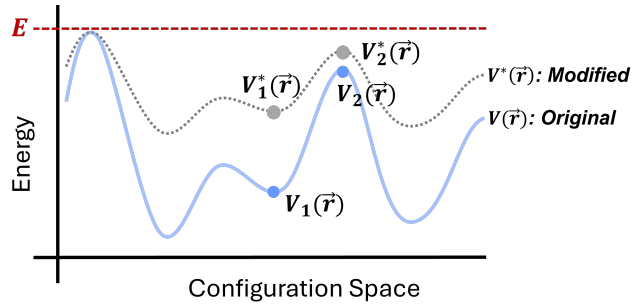


Figure 1. Overview of the how boost potential criteria manifest in GaMD simulations. The original FES (light blue) is modified by boost potentials for any energy state below the threshold energy (red).

If $V_1(\vec{r}) < V_2(\vec{r})$, then the respective potential energies on the boosted surface will also follow the same trend, $V_1^*(\vec{r}) < V_2^*(\vec{r})$, to maintain the shape of the original FES.

$$(1) \quad V_1(\vec{r}) < V_2(\vec{r}) \implies V_1^*(\vec{r}) < V_2^*(\vec{r})$$

In addition, the difference between the boosted energies should also be less than the difference of the original energies, $V_1^*(\vec{r}) - V_2^*(\vec{r}) < V_1(\vec{r}) - V_2(\vec{r})$. This criterion ensures we are lowering the energy barriers.

$$(2) \quad V_1(\vec{r}) < V_2(\vec{r}) \implies [V_1^*(\vec{r}) - V_2^*(\vec{r})] < [V_1(\vec{r}) - V_2(\vec{r})]$$

If the first two criteria are combined in terms of $V^*(\vec{r})$, the value of E must exist between V_{max} and $V_{min} + \frac{1}{k}$, in which case $k \leq \frac{1}{V_{max} - V_{min}}$. If we define $k \equiv k_0 \cdot \frac{1}{V_{max} - V_{min}}$, k_0 must be between 0 and 1. This requirement will be important for monitoring simulation convergence during GaMD equilibration and determines the extent to which conformational sampling is accelerated.

Finally, the distribution of potential energies needs to be sufficiently narrow to allow the system to be properly reweighted. More details on reweighting will be provided in Section 3.3).

(3) Narrow ΔV distribution

$$\sigma_{\Delta V} = k(E - V_{avg})\sigma_V \leq \sigma_0$$

where $\sigma_0 = k_B T$

The value of $\sigma_{\Delta V}$ is the standard deviation of the boost potential, while σ_V and σ_0 are the standard deviations of the potential energy in the current simulation step and the thermal energy, $k_B T$, at the simulation temperature, respectively. Enforcing $\sigma_0 = k_B T$ ensures that the distribution of ΔV is within the desired statistical mechanical ensemble.

The potential energy standard deviation is used alongside V_{max}, V_{min}, and V_{avg} to calculate k_0 in a manner that depends on whether E is set to the "lower bound" ($E = V_{max}$) or "upper bound" $\left(E = V_{min} + \frac{1}{k}\right)$ condition. When E is set to lower bound, k_0 is calculated according to Eq. 5:

$$k_0 = \min\left(1.0, k_0'\right) = \min\left(1.0, \frac{\sigma_0}{\sigma_V} \cdot \frac{V_{max} - V_{min}}{V_{max} - V_{avg}}\right) \quad (5)$$

such that k_0 is set to 1 or $k_0 = \frac{\sigma_0}{\sigma_V} \cdot \frac{V_{max} - V_{min}}{V_{max} - V_{avg}}$, whichever is lower. In the case of an upper bound condition, k_0 is calculated via Eq. 6:

$$k_0 = k_0'' \equiv \left(1.0 - \frac{\sigma_0}{\sigma_V}\right) \cdot \frac{V_{max} - V_{min}}{V_{avg} - V_{min}} \quad (6)$$

only if k_0'' is between 0 and 1. If k_0'' is not between 0 and 1 then k_0 will be calculated via Eq. 5. k_0 can range from any value between 0 and 1, with higher k_0 values corresponding to larger boost potentials being added to the energy surface and thus greater acceleration of conformational sampling. k_0 is then used to calculate k and ultimately used to determine the boost using Eq. 1 and 2.

3.3 Reweighting

Once GaMD simulations are complete, the modified energy surface is reweighted to recover the original energy surface. There are many different methods that can be used to reweight enhanced sampling simulations, but in the case of GaMD, it has been shown that cumulant expansion to the second order is the most efficient way to obtain the original FES as it was found to be less noisy than exponential reweighting and the use of Maclaurin series expansion [19]. To perform reweighting, the PyReweighting toolkit of Python scripts has been developed for reweighting aMD and GaMD simulations using either the exponential average, Maclaurin series expansion, or cumulant expansion. In the case of GaMD, we are using cumulant expansion, an expansion of mathematical expressions to approximate a distribution of the boosts applied across the simulation time to determine the probability, and associated free energy, of each state in the system.

GaMD simulations generate a biased probability distribution of states sampled in the trajectory. This biased distribution has energetic barriers that are generally lower

than what would be found in the original distribution and are determined by the change in system potential energy or the boost, $\Delta V(\vec{r})$, applied at the specified frame in the simulation (Figure 1). The amount of boost applied can then be converted to weights that relate the biased probability distribution to the original probability distribution over a specified CV used to describe transitions in the system. For large biomolecules (e.g., with millions of atoms), 0.5-1 million frames are suggested to ensure that there is enough boost data to accurately reweight, which corresponds to saving simulation frames every 0.1 ps [2, 3, 21]. This tutorial will not go over CV selection as this issue has been discussed elsewhere [22], however, it is essential that the CV best describes the system characteristics to answer the underlying research question.

Reweighting begins by reducing the dimensionality of the simulation data to correspond to the CV that is chosen to best describe the different states of a transition to be characterized. The range of possible CV values is divided into bins, or discrete intervals of the entire CV data range. As such, the simulation frames are sorted into these bins to determine how frequently different values of the CV are visited during the simulation. In each of these bins, the CV value and associated boost value are stored for that specific simulation frame. Since the GaMD algorithm is pushing the simulation to visit low-probability states, there needs to be a correction factor to offset the boost that was added. For each simulation frame, the boost is used to calculate this correction, which is then averaged across all bins to give an ensemble-averaged reweighting factor (This is shown in Eq. 7):

Ensemble-Averaged Reweighting Factor

$$\left\langle e^{\beta \Delta V(\vec{r})} \right\rangle = \exp\left\{ \sum_{k=1}^{\infty} \frac{\beta^k}{k!} C_k \right\} \tag{7}$$

where $\beta = 1/k_B T$ and the angle brackets, $\langle\rangle$, signify the averaging that occurs across the simulation ensemble. Essentially, the summation ($\sum$) can be expanded to however many mathematical terms the user deems necessary. Previous work has determined that using two "cumulants" (terms) was the most efficient method for GaMD reweighting [19]. For example, the first three expanded terms that can be substituted for C_k are shown in Eq. 8:

$$\begin{aligned} C_1 &= \Delta V, && \text{"first order"} \\ C_2 &= \left\langle \Delta V^2 \right\rangle - \left\langle \Delta V^2 \right\rangle = \sigma_V^2 && \text{"second order"} \\ C_3 &= \left\langle \Delta V^3 \right\rangle - 3 \left\langle \Delta V^2 \right\rangle \langle \Delta V \rangle + 2 \left\langle \Delta V^2 \right\rangle && \text{"third order"} \end{aligned} \tag{8}$$

Thus, if using cumulant expansion to the second order, the reweighting factor becomes equal to the summation of the series for cumulants 1 and 2 in Eq. 8. The reweighting factor is calculated for every frame in the simulation and determines how much that simulation frame should contribute to the original probability distribution. The biased probability distribution, $p^*(A_j)$, is simply frequency of CV sampling without considering the boost potential. For each bin, j, the reweighting factors of all corresponding simulation frames are averaged to determine the ensemble-averaged reweighting factor. The ensemble-averaged reweighting factor is then multiplied by $p^*(A_j)$ to rescale the distribution based on the magnitude of the boost potential in each bin. The result is then normalized by dividing by the total reweighting factor to obtain the original probability distribution, $p(A_j)$, as shown in Eq. 9:

Original Probability Distribution; Ensemble-Averaged Reweighting Factor; Bin Number; Biased Probability Distribution; Total Boltzmann Reweighting Factor Across All Bins

$$p(A_j) = \frac{p^*(A_j)}{\sum_{j=1}^{M} \left\langle p^*(A_j)\, e^{\beta \Delta V(\vec{r})} \right\rangle_j} \left\langle e^{\beta \Delta V(\vec{r})} \right\rangle_j, \quad j = 1, \ldots, M \tag{9}$$

Doing so redistributes the probability density based on the GaMD boost, correcting for over-sampling of high energy states and under-sampling of low-energy states. In addition to the probability distribution, the free energy is also calculated using cumulant expansion to the second order shown in Eq. 10:

Reweighted Free Energy; Biased Free Energy; Force Constant

$$F(A_j) = F^*(A_j) - \sum_{k=1}^{2} \frac{\beta^k}{k!} C_k + F_c \tag{10}$$

Finally, the reweighted energy profile for each bin, j, can be calculated using the reweighted free energy, $F(A_j)$, and the original probability distribution, $p(A_j)$, using Eq. 11:

Reweighted Free Energy; Original Probability Distribution

$$F(A_j) = -\frac{1}{\beta} ln\, p(A_j) \tag{11}$$

The reweighted free energy and the associated CV values can then be plotted and used to evaluate the FES.

Conventional MD		Gaussian-accelerated MD		
Prep Steps	Conventional Simulation	Prep Steps	Equilibration	Production
Adaptation to simulation environment	Gathering statistics: V_{max}, V_{min}, V_{avg}, and σ_V	Adaptation to simulation environment	Boost added and boost parameters updated each step: k_0, E, and k	Boost parameters are constant, and the boost is applied based on the difference between E and the system energy

Figure 2. Overview of the GaMD process in NAMD.

4 Workflow of Running a GaMD Simulation

The GaMD procedure in the NAMD suite consists of three distinct stages. First, a short conventional MD simulation gathers initial energetic information and calculates thermodynamic statistics of the system such as V_{max}, V_{min}, V_{avg}, and σ_V. Then, an equilibration step is used to update these statistics and refine the harmonic force constant, k using Eqs. 5 and 6; the threshold energy, E using Eq. 3; and the boost potential using Eq. 1. Finally, a GaMD production simulation of user-defined length is performed using the established threshold energy from equilibration and boosts are applied to enhance the conformational sampling. A simplified overview of these steps and their general function is given in Figure 2.

4.1 Overview of the Run Scripts

As mentioned previously, we will focus on how to specify a GaMD simulation within a NAMD run script. Templates for the inputs can be found on the Miao lab website (https://www.med.unc.edu/pharm/miaolab/resources/gamd/tutorial-of-gamd-in-namd/) and our full scripts can be found in the GitHub repository accompanying this article (https://github.com/Lemkul-Lab/gamd_tutorial.git). In general, the GaMD section has the following form:

Listing 2. Conventional and Equilibration Setup

```
#########################################
## GAMD
#########################################
accelMD                 <on/off>
accelMDdual             <on/off>
accelMDdihe             <on/off>
accelMDG                <on/off>
accelMDGiE              <1/2>
accelMDGcMDPrepSteps    <number of steps>
accelMDGcMDSteps        <number of steps>
accelMDGEquiPrepSteps   <number of steps>
accelMDGEquiSteps       <number of steps>
accelMDOutFreq          <number of steps>

accelMDGStatWindow      <number of steps>

accelMDGsigma0P         <value>

accelMDGsigma0D         <value>

accelMDGRestart         <on/off>
accelMDGRestartFile     <file name>
```

A full description of all GaMD parameters and their default values for NAMD can be found in the NAMD manual (https://www.ks.uiuc.edu/Research/namd/2.14/ug/node70.html). Here we will provide a synopsis of each term:

accelMD Specifies whether accelerated MD is to be turned on or off. GaMD is an extension of aMD, so this setting must be on if you are using GaMD.

accelMDdual Specifies whether to apply "dual-boost" GaMD.

accelMDdihe Specifies whether to apply dihedral boost. If "dual-boost" is on, then this keyword must be set to "on" as well. Doing so will provide the greatest sampling acceleration.

accelMDG Specifies the use of GaMD as the accelerated MD method.

accelMDGiE Sets E to either the lower bound (1) or upper bound (2). Lower bound (1) sets E to V_{max} and is the typical setting.

accelMDGcMDPrepSteps Number of preparatory conventional MD steps.

accelMDGcMDSteps Number of conventional MD steps after preparatory steps.

accelMDGEquiPrepSteps Number of preparatory GaMD equilibration steps.

accelMDGEquiSteps Number of GaMD equilibration steps after preparatory steps.

accelMDOutFreq Specifies the output frequency of the GaMD algorithm.

accelMDGStatWindow Sets how often to calculate the average and standard deviation of the boost potential during GaMD. This value should be set to $\sim 4\times$ the number of particles in the simulation system.

accelMDGsigma0P Upper limit of the standard deviation of the total boost potential.

accelMDGsigma0D Upper limit of the standard deviation of the dihedral boost potential.

accelMDGRestart Specifies whether or not this simulation is being restarted from a previous one. Ensures that the `*.gamd` file is read to restart.

accelMDGRestartFile Specifies the name of the GaMD restart file.

4.2 Running Conventional Simulations and GaMD Equilibration

Initially, GaMD starts with a specified number of conventional simulation steps during which no boost is applied to the system. This step gathers statistics related to the potential energy of the system, including V_{max}, V_{min}, V_{avg}, σ_V, E, k_0, and k. To start the simulations, NAMD will require information about the structure of interest and the topology. Using this starting information, a short segment of preparatory steps (`accelMDGcMDPrepSteps`) is conducted before the conventional simulation segment to allow the biomolecule to adapt to its simulation environment, basically to ensure the simulation will not crash. The user can specify any amount of steps for these different segments. Generally, the preparatory steps require less simulation time and therefore fewer steps since they are not used to collect statistics and refine the boost potential. This tutorial will not recommend a specific number of steps for each segment but previous studies using GaMD have employed 200,000 steps for the preparatory segment and 200,000 steps for the conventional segment when simulating larger biomolecules. Our chosen number of steps for alanine dipeptide will be discussed in Section 5.

Once the conventional segment is complete, equilibration preparatory steps will be conducted, followed by the equilibration segment. During the equilibration segment, the statistics are refined and E is allowed to increase until it levels off. Boosts are also applied during this stage and refined based on the changes to E. Prior work states that k_0 should stabilize or reach 1 throughout equilibration to determine if your equilibration simulation has reached convergence [2]. However, we have found that it may also be beneficial to track V_{max}, as it still can increase even when k_0 reaches 1. We will discuss more on interpreting the output files in and monitoring the simulations in Section 4.4. If equilibration does not reach convergence in the designated number of steps, it can be extended using the GaMD restart file and by setting `accelMDGcMDPrepSteps`, `accelMDGcMDSteps`, and `accelMDGEquiPrepSteps` to `0`. Both conventional MD and GaMD equilibration are executed from the same script, which we have named `gamd_equil.in` and can be found in each force field directory in the GitHub repository.

4.3 Running GaMD Production

Once it has been established that the GaMD equilibration has converged by monitoring k_0 and V_{max}, the system can proceed to the production stage. At the end of equilibration, the value of E is set and does not change throughout production. Here, boosts are applied based on the difference between E and the potential energy at the current step in the simulation, $V(\vec{r})$. The larger the difference in E and $V(\vec{r})$, the larger the magnitude of boost applied.

The run script is generally the same as the one used for conventional and equilibration, however, no conventional or equilibration steps will be performed. As such, `accelMDGcMDPrepSteps`, `accelMDGcMDSteps`, `accelMDGEquiPrepSteps`, and `accelMDGEquiSteps` are all set to `0`. The `run` parameter can be adjusted, as well, to suit a user's specific needs in terms of production simulation length. In addition, `accelMDGRestart` will need to be set to "on" so that the GaMD statistics and E can be read in from the restart file, `gamd_equil.gamd`. An example of the GaMD parameters for production is as follows:

Listing 3. md.*.in

```
#########################################
## GAMD
#########################################
accelMD                 on
accelMDdual             on
accelMDdihe             on
accelMDG                on
accelMDGiE              1

accelMDGcMDSteps        0
accelMDGEquiSteps       0
accelMDGcMDPrepSteps    0
accelMDGEquiPrepSteps   0
accelMDOutFreq          500

accelMDGStatWindow      10000

accelMDGsigma0P         6.0
accelMDGsigma0D         6.0

accelMDGRestart         on
accelMDGRestartFile     <BASENAME>.gamd
```

In principle, the GaMD production can be performed for any length of time, but for practicality, we suggest running the GaMD production simulations in strides. This way, trajectories and all output files are saved in increments and in the case of crashes or file corruption, only short intervals of data are lost rather than the entire simulation. Generally, the production input files will look the same but variables are read in from a separate bash script to automate the setup of short strides. We will overview how we generate the strides in more detail in Section 5.4.2. Both the submis-

sion script and the template script can be found on GitHub and are named `sub_gamd_prod.sh` and `md.template`.

4.4 Interpreting GaMD Output Files

Throughout the GaMD process, several output files are created to monitor progress and continue running the simulation. Many of these files are generated by NAMD, and not the GaMD algorithm itself, including:

- `.coor` - PDB-formatted atomic coordinates.
- `.xsc` - the extended system restart information.
- `.vel` - PDB-formatted atomic velocities.
- `.xst` - all the extended system information over time.

The GaMD algorithm produces two files: the log file, `gamd_equil.out` or `md.*.out`, that contains all the thermodynamics statistics of your system throughout the GaMD process; and the GaMD restart file, `gamd_equil.gamd` or `*.gamd`, that contains information about the current stage of GaMD and is required to continue running a simulation from where it left off. GaMD equilibration will create one of each of these files, whereas every stride during production will have its own set of these output files.

Taking a closer look at `gamd_equil.out` in Listing 4, this file contains the step by step output of GaMD and is a critical output file to inspect. All of the statistical data collected by the GaMD algorithm for each simulation step is found within this file. The top portion of this file details the NAMD environment setup for the simulation. The simulation has started running when the `run` keyword is executed and the number of steps for the simulation is printed to the `.log` files (Listing 4, first line). Below this line, the GaMD algorithm will start printing the system potential energy statistics to the output file. These lines are used to track the statistics over time, verify that each stage is running correctly, and to monitor statistics throughout the simulation. An example of this section at the start of the conventional preparatory steps is shown in Listing 4.

The aMD and GaMD algorithms report on the state of the system at the current step. The example shown in Listing 4 represents step 0, or the initial state of the system in the current GaMD run. On the `ACCELERATED MD` line, `dV` is the boost at the current step, which, for dual-boost, is actually the sum of the dihedral and total potential boosts as described in Eq 12:

$$\underbrace{\Delta V}_{\text{Overall Boost}} = \underbrace{\Delta V_T}_{\text{Total Potential Boost}} + \underbrace{\Delta V_D}_{\text{Dihedral Boost}} \tag{12}$$

ΔV_T and ΔV_D are calculated using Eqn. 1 and the energy values corresponding to the total potential energy, $V(\vec{r})$, and the dihedral energy term, V_τ.

$$\Delta V_T = \frac{1}{2}k\left\{\underbrace{E_T}_{\text{Threshold Energy}} - \left[\underbrace{V(\vec{r})}_{\text{Current System Potential Energy}} - \underbrace{V_\tau}_{\text{Dihedral Energy}}\right]\right\}^2 \tag{13}$$

$$\Delta V_D = \frac{1}{2}k\left\{E_D - V_\tau\right\}^2 \tag{14}$$

On the same line `dVAVG` is the average boost up to the current point in the simulation. The remainder of this line reports the contributions of the various terms in the potential energy equation. For "dual-boost" GaMD, separate lines are printed to track statistics for the dihedral energy, `GAUSSIAN ACCELERATED MD: DIHED`, and the total potential energy, `GAUSSIAN ACCELERATED MD: TOTAL`. In these lines, `iE` indicates the setting of the threshold energy, and the remainder of the printed fields are the maximum, minimum, and average energy, the variation in energy, the threshold energy, `k0`, and the harmonic force constant for the current step.

During conventional and equilibration preparatory steps, no potential energy statistics are updated, and as such `sigmaV`, `E`, `k0`, and `k` all remain `0`. Without statistics, no boost can be calculated so `dV` and `dVAVG` also remain `0` throughout the preparatory step segments. The statistics start to be reported during the conventional segment while the boost values remain zero until the equilibration stage. During equilibration, E and the boost are updated in their respective field as the system explores the conformational landscape. Once equilibration convergence has been observed through monitoring of k and E (see Section 5), E is held constant for production, remaining consistent in the output files, and the boost is calculated each step using Eq. 1.

Another output file, `*.gamd`, contains all the GaMD information from the last complete step. An example is shown in Listing 5. In it are two lines of information for the dihedral and total energies indicated by the single letter codes in the first column of the row, `D` and `T`. Following are the current step, `Vn`; the maximum energy, `Vmax`; the minimum energy, `Vmin`; the average energy, `Vavg`; the standard deviation, `sigmaV`; the variability from the mean, `M2`; the threshold energy, `E`; and `k`. For equilibration, only one `gamd_equil.gamd` is produced, compared to production, during which a file is produced for each stride in our approach.

4.5 Reweighting

As mentioned previously, the PyReweighting toolkit can be found at https://github.com/MiaoLab20/pyreweighting. The PyReweighting toolkit allows for one-, two-, or three-dimensional reweighting, depending on how many CVs are chosen for this analysis. Within the `pyreweighting`

Listing 4. gamd_equil.out

```
TCL: Running for 26100000 steps

""

ACCELERATED MD: STEP 0 dV 0 dVAVG 0 BOND 0.000398329 ANGLE 0.758706 DIHED 4.07314 IMPRP 0.074434 ELECT
    -9757.12 VDW 1176.49 POTENTIAL -8575.72
GAUSSIAN ACCELERATED MD: DIHED iE 1 Vmax 4.07314 Vmin 4.07314 Vavg 4.07314 sigmaV 0 E 0 k0 0 k 0
GAUSSIAN ACCELERATED MD: TOTAL iE 1 Vmax -8579.8 Vmin -8579.8 Vavg -8579.8 sigmaV 0 E 0 k0 0 k 0
```

Listing 5. system.gamd

```
# NAMD accelMDG restart file
# D/T Vn Vmax Vmin Vavg sigmaV M2 E k
D 10000 23.42610900347244 2.4217065108565792 8.5355531287135644 2.4186399083415688 5.84981900622251123e+04
    23.42610900347244 0.04760906673501196
T 10000 -8188.84384705514 -9062.2482195580287 -8296.0966588262636 20.988071042745403 4.40499126095328107e+06
     -8188.84384705514 0.00114494503517807
```

directory, these options are represented by scripts named `PyReweighting-1D.py`, `PyReweighting-2D.py`, and `PyReweighting-3D.py`. Each of these scripts is executed by `reweight-1D.sh`, `reweight-2D.sh`, and `reweight-3D.sh`, respectively. The execution scripts determine what reweighting scheme will be used. In the `reweight-*.sh` scripts, there are four sections of code representing the types of reweighting schemes: cumulant expansion (`amdweight_CE`), no weight (`noweight`), weight (`amdweight`), histogram (`histo`), and the boost potential (`amd_dV`). As noted above, previous studies concluded that it is best to use cumulant expansion to the second order to reweight GaMD simulations [18, 19], which is carried out using the (`amdweight_CE`) argument. In addition to reweighting, it is also important to assess the anharmocity of the system to determine if the boosts applied follow a Gaussian distribution. The anharmonicity calculations are performed using the `amd_dV` argument. All commands are already present in the `reweight-2D.sh` and an example is shown in Listing 6.

The Python interpreter is used to execute the reweighting algorithm in `PyReweighting-2D.py`, which we use as an example in the tutorial that follows. All command-line arguments denoted with `$` are variables that are specified in the `reweight-2D.sh` shell script. A complete list of the arguments present in this example as well as any additional arguments that may be beneficial in obtaining the most descriptive set of potential of mean force values for the chosen CVs are outlined below.

input (required) The file containing the chosen CVs, separated by column

job (required) Choice of reweighting method

weight (required) The file containing the weights from GaMD

Xdim (optional) Specify dimensions for the first CV (x-axis)

Ydim (optional) Specify dimensions for the second CV (y-axis)

discX (optional) Discretion size (bin width) of first CV

discY (optional) Discretion size (bin width) of second CV

cutoff (optional) Minimum number of simulation frames required per bin

T (optional) Temperature

Emax (optional) Maximum free energy

fit (optional) Fit ΔV distribution

order (optional) Order of Maclaurin series to use in reweighting. May only be needed if not using cumulant expansion to the second order.

Each of the optional arguments is dependent on the system and is employed at the discretion of the user. In the alanine dipeptide tutorial that follows, we chose to add in `Xdim` and `Ydim`, which are set to the possible dimensions of ϕ and ψ angles of the alanine dipeptide backbone.

The `-job` argument determines the type of reweighting and ultimately what output files are obtained. Setting `-job amdweight_CE` provides the user with the potential of mean force (PMF) values for c1, c2, c3 using Eq. 8. Each file name contains the name of the data file used and the bin width specified, making it easier to track the parameters specified. For example, the file name `pmf-2D-c2-cv_500.dat-reweight-discx15-discy15.xvg` indicates this file contains the energy values (`pmf`), from 2D reweighting performed using cumulant expansion to the second order (`c2`) across the CVs contained within

Listing 6. reweight-2D.sh

```
python $dir_codes/PyReweighting-2D.py -input $data -T $T -Emax $Emax -cutoff $cutoff -discX $binx -discY
    $biny -job amdweight_CE -weight weights.dat | tee -a reweight_variable.log
```

cv_500.dat with a bin width for x and y of 15 (discx15 and discy15). Files beginning with pmf-2D-c2 will be the primary files for GaMD reweighting. The file is organized with the first CV, x, and the second CV, y, in the first and second columns, followed by the PMF in the third column. This file will be used to generate the FES.

An additional quantity, "anharmonicity" or γ, can be used to characterize the distribution of the applied boosts. When γ is zero, the boost is considered harmonic (*i.e.* it follows a Gaussian distribution) and confirms that the reweighted free energy profiles are accurate. As γ increases, $\Delta V(\vec{r})$ becomes less harmonic and indicates that the reweighted free energy profile deviates from the original energy profile. This value can help evaluate the quality of the FES and we will discuss this value in context in Section 5.5.2.

5 Tutorial

We will demonstrate the GaMD protocol with a small system of the alanine dipeptide in water. To illustrate several important principles and outcomes, we will compare results of this protocol that arise from using the AMBER ff19SB [23], CHARMM36m [24], and Drude-2019 [25] force fields.

5.1 Introduction

Alanine dipeptide has long been used as a model system for validating force field parameters and simulation methods, and has been used as the published test system for GaMD [2, 3, 21]. This tutorial will focus on instructing users on how to set up, reweight, and analyze alanine dipeptide GaMD simulations, with a comparison among the results of the three different force fields. The reweighting process using the PyReweighting toolkit will also be demonstrated along with a discussion on how to adjust specific GaMD parameters for the system of interest. The GitHub repository associated with this article (https://github.com/Lemkul-Lab/gamd_tutorial.git) contains all scripts and input files required to complete the tutorial.

5.2 GaMD System Preparation

5.2.1 The System

We are interested in the conformational sampling of alanine dipeptide. Three systems were prepared using the AMBER ff19SB, CHARMM36m, and Drude-2019 force fields which we will refer to as ff19SB, C36m, and Drude, respectively. Details on how to construct each system with their respective FF will not be given here, and the user should construct and minimize their system in whatever manner is appropriate. Briefly, the AMBER ff19SB system was constructed using AmberTools [13] and then converted to C36m. The Drude system was then constructed from the C36m system using the CHARMM program [15]. Initial C36m and Drude structures were obtained from a brief minimization using a previously published protocol [26].

5.2.2 Setting Up the Working Directories

Topology and coordinate files for each system are located in the directory named according to the force field used (ff19sb, c36, and drude in our case). For AMBER ff19SB, the parameter and topology file (ala.parm7) and the coordinate file (ala.rst7) should be placed within the ff19sb directory while for C36m and Drude, ala.c36.psf and ala.c36.pdb should be placed in c36m and ala.drude.psf and ala.drude.pdb should be placed in drude. A directory for GaMD equilibration and production should be created in the directory corresponding to each system. For clarity, we named these directories equil and prod. Dynamics input files for equilibration, (gamd_equil.in) and production (md.template), and submission scripts for SLURM queueing (sub_gamd_equil.sh and sub_gamd_prod.sh) can be found in the associated force field directories on GitHub and should also be placed in the respective equilibration and production directories. All force field parameters for ff19SB are found within the parm7 file while the parameter files for C36m and Drude systems are listed in the run scripts and can be obtained from the MacKerell lab website (https://mackerell.umaryland.edu/charmm_drude_ff.shtml), if required.

5.3 Contents of the Run Scripts and Parameters for Equilibration

We have provided complete run scripts for conventional MD/GaMD equilibration and GaMD production for each force field as they each require distinct NAMD keywords. We encourage users to ensure that they have the correct script for the force field applied to the system. In general, the scripts provided will need minimal adjusting to reflect system input files, designated force field, system properties, and the length of time for each of the GaMD steps. The preparatory, conventional MD, and GaMD equilibration steps will all be initialized from the gamd_equil.in input

script, while the production step will be run in short intervals (or "strides") from a template script, which will be described in more detail in Section 5.4.2. As a general suggestion,users should review all scripts for any NAMD parameters that need to be adjusted for their specific system prior to running the actual simulations. The input files for this tutorial should only require adjustment on specifying locations of force field files. Here, we will provide an overview of the sections of the NAMD input files and provide information regarding the NAMD parameters or keywords that we will use for alanine dipeptide. This first section will focus on NAMD keyword selections for conventional MD and GaMD equilibration, while the information for GaMD production will be provided in Section 5.4.1.

5.3.1 Coordinate, Topology, Force Field and Other Input Files

This section specifies the input topology and coordinate files. C36m and Drude use NAMD keywords `coordinates` and `structure` for the PDB and PSF files respectively, while ff19SB systems are specified with NAMD keywords `ambercoor` and `parmfile` for the rst and parm7 files. In the associated script, we have specified the force field parameter files using `parameters` for C36m and Drude. NAMD also requires the format of the force field parameters to be stated through keywords `amber` for ff19SB and `paraTypeCharmm` for C36m and Drude. There are several other keywords in this section that can be adjusted depending on what system is running; `mergeCrossterms` for C36m, Drude, and ff19SB, and `readexclusions` and `scnb` for ff19SB. The details of each of these keywords can be found in the NAMD manual. Temperature (K) and time step (in femtoseconds, fs) are also set in this section using the `temperature` and `timestep` keywords. These settings generally depend on the system of interest but for alanine dipeptide we will simulate at 298 K and use a timestep of 2.0 fs for ff19SB and C36m and a timestep of 1.0 fs for Drude.

5.3.2 Output Files and Output Frequency

The output section specifies what type of output files are desired as well as what to name them. We set NAMD keywords `outputname`, `dcdfile`, and `restartname` such that all outputs will have the prefix `gamd_equil` for the GaMD equilibration stage.

Output parameters specify how often the respective properties are written to output files. The user indicates a number of steps for these keywords, which can then be multiplied by the time step to determine how often the output files are updated. As mentioned previously, saving 0.5 – 1 $\times$ 10^6 frames is recommended to ensure there is enough simulation data to accurately reweight. Since alanine dipeptide is a small system, we have set our scripts to update the output every 1 ps (500 steps for systems with 2 fs timestep, and 1000 steps for systems with 1 fs time step).

5.3.3 Temperature and Pressure Control

This section details the type of thermostat and barostat to use for the simulations. For all systems, we specify the use of Langevin dynamics (`Langevin`) via the Langevin thermostat and barostat [27] (`LangevinPiston`), and the temperature at which the system is maintained (`LangevinTemp`).

5.3.4 Non-bonded Setup

This section contains information on how each system handles non-bonded interactions. The `switching` keyword will turn on the switching function for C36m and Drude and `vdwForceSwitching` should be specified for C36m only. Other parameters will depend on the force field used. The short-range non-bonded cutoff, switching function distance, exclusions, neighbor searching radius, 1-4 interaction scaling, and dielectric can all be set in this section. This section also contains optional keywords to specify a Drude system, with keyword `drude`, and other Drude-specific settings to turn on or off the hard wall constraint, specify the Drude-atom bond distance at which the hard wall acts, set the through-space Thole cutoff, etc.

5.3.5 Constraints

This section contains keywords for bond constraints. `rigidbonds` determines what, if any, atoms are constrained. For these systems, all the bonds connecting hydrogens to their parent atoms are constrained and so are all water molecules. `rigidTolerance`, `rigidIterations`, and `useSettle` determine the maximum bond-length error, the number of attempts to constrain bond lengths, and employ the SETTLE algorithm to water molecules.

5.3.6 Electrostatics

This section details the settings related to the use of the particle mesh Ewald algorithm for electrostatics [28]. Here, the use of `PME` is specified along with the number of Fourier lattice grid points in each direction using `PMEGridSizeX`, `PMEGridSizeY`, and `PMEGridSizeZ`.

5.3.7 Unit Cell Definition

This section details the size of the simulation box and the cell origin. `wrapAll` indicates whether or not periodic boundary conditions should be active for all atoms and is typically used for simulations during which no restraints are applied.

5.3.8 GaMD Settings

This section contains the GaMD-specific settings previously described in Section 4. Here, `accelMD`, `accelMDdual`, `accelMDdihe`, `accelMDG` control what type of simulation to run. If you wish to run a GaMD simulation using "dual boost," all of these keywords should be set to `on`. `accelMDdual` and `accelMDdihe` can be toggled off to apply only dihedral or total potential boosts. `accelMDGiE` determines how the threshold energy is set. Because alanine dipepetide in water is a small system, we will set it to 1, which corresponds to the "lower bound" condition, or $E = V_{max}$. For equilibration, there is no prior state from which to resume, so `accelGRestart` is set to `off`. Next, we designate the number of steps for each segment of the GaMD simulation. For this tutorial, we will be running 100000 steps (200 ps) of conventional MD prep, 1000000 (2 ns) of conventional MD, 100000 steps (200 ps) of equilibration prep, and 25000000 steps (50 ns) of equilibration. The number of steps shown here are used in conjunction with a time step of 2 fs. If using a time step of 1 fs, as with the Drude force field, the number of steps for each segment need to be multiplied by 2 to achieve the same sampling times. This difference is reflected in the run scripts associated with the Drude force field on GitHub.

5.3.9 Run Settings

The `run` keyword instructs the NAMD integrator how many integration steps to perform. For the process here, we set the number of steps to be the sum of the conventional MD prep, conventional MD, equilibration prep, and GaMD equilibration steps. Thus, we have specified 26200000 steps to match what we specified in the GaMD section. As before, the number of steps will need to be multiplied by 2 if using a time step of 1 fs.

Once the run script has been edited, the submission script can also be edited to use whatever number of CPU cores requested for the resources to which the job will be submitted. Our submission script is designed to be used with the SLURM job manager but can be edited if necessary.

5.4 Running and Monitoring the Simulations

5.4.1 Equilibration

Once `gamd_equil.in` and `sub_gamd_equil.in` have been edited, the equilibration job can be submitted to a SLURM queueing system using:

```
$ sbatch sub_gamd_equil.sh
```

During equilibraton, it is important that the simulation is monitored to determine when it converges. We have found it useful to monitor E in addition to k_0 to assess convergence. To do so, we can extract both values from the `gamd_equil.out` and graph them as a function of the equilibration time. The `grep` command below is an example of how we can create a new file containing only the values of k_0 for the dihedral boost from the 18th column of the `GAUSSIAN ACCELERATED MD: DIHED` line:

```
$ grep "GAUSSIAN ACCELERATED MD: DIHED"
    gamd_equil.out | awk '{print NR/1000,
    $18}' > k0_dihe.dat
```

This command can be used to extract any of the statistics into their own files; we provide the column codes here for reference. The associated GitHub repository contains a Bash script that can be used to obtain all statistic files, named `gen_stats_files_gamd.sh`

Table 1. List of GaMD statistics and their associated column codes for use with `grep` and `awk` commands.

Statistic	Column Code
V_{max}	\$8
V_{min}	\$10
V_{avg}	\$12
σ_V	\$14
E	\$16
k_0	\$18
k	\$20

These files will have a two columns that list the time point in nanoseconds and the desired statistic across all preparatory, conventional, and equilibration steps, which can be graphed as a time series. We plotted each time series with XmGrace, where the x-axis corresponds to the combined time point of conventional and equilibration simulations. We note that the math within the `gen_stat_files_gamd.sh` is written specifically to accurately convert the time points based on our time step and our output interval. This script may need to be adjusted if a user alters those values. The time series of k_0, E, and k for alanine dipeptide during the GaMD equilibration stage with each of the three force fields are shown in Figure 3.

Ideally, k_0 will reach 1 during equilibration, indicating that the system has the highest dynamic acceleration, however, we note that even when k_0 reaches 1, E may still be increasing. As E is directly involved in the boost equation shown in Eq. 1, we suggest also monitoring E during equilibration to ensure that the system reaches its highest possible energy. Such an outcome is indicated by E leveling off towards the end of the equilibration steps. As shown in Figure 3, each force field reached the highest acceleration indicated by k_0 reaching a value of 1, however, there are differences in the time series of E for each FF. The differences in system potential energy are due to the inherent differences in the potential energy terms in each force field. The time series also show

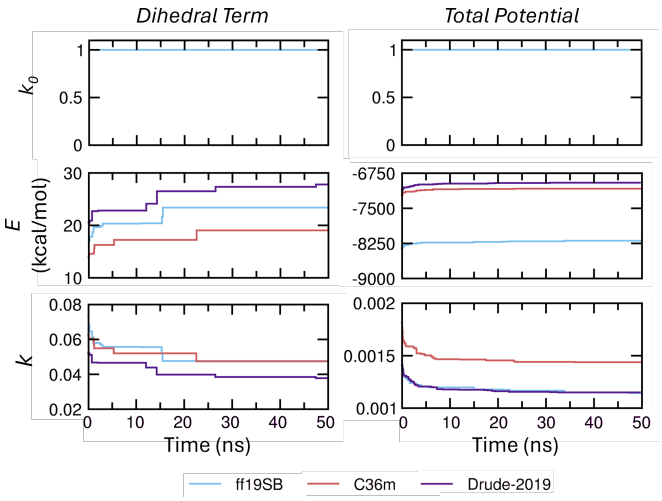


Figure 3. GaMD parameters as a function of time during equilibration.

how quickly the system reaches a converged *E* for the dihedral term with ff19SB and C36m, whereas Drude may need additional time to reach its maximum possible energy. Use of a polarizable force field with GaMD is still relatively new, thus the time required to equilibrate should be studied further. A particularly important point is that the Drude force field has inherently slower kinetics than C36m and most non-polarizable force fields. This difference arises from the more realistic diffusion of the SWM4-NDP water model [29] compared to the (CHARMM-modified) TIP3P model [30–32] used in conjunction with C36m. This same trend is shown with the total potential energy but is less obvious due to the scale of energy differences across force fields.

The time series for *k* is also shown as it is involved in the boost calculation, however, we do not regularly monitor *k* as it is inversely correlated to *E* and thus exhibits the same behavior as *E*.

5.4.2 Production

Once the system has reached equilibration convergence, the output files can then be used to set up the GaMD production run. In `prod`, we will need the system topology, `ala.parm7` for ff19SB, `ala.c36.psf` for C36m, or `ala.drude.psf` for Drude. A few different files from the end of equilibration will be required for production. The coordinate file (`gamd_equil.coor`), the atomic velocity file (`gamd_equil.vel`), and the file containing the box size (`gamd_equil.xsc`), should all be copied to the production folder. Additionally, the file containing the GaMD parameters from the last step of equilibration, `gamd_equil.gamd`, will also need to be copied to this directory. The job submission script (`sub_gamd_prod.sh`) and the production run template file (`md.template`) will also need to be present in this directory.

Only slight changes are required to the the file, `md.template` which serves as our template for the GaMD production input scripts. All prep, conventional, and equilibration steps should be set to 0 and `accelMDGRestart` should be turned on to read in the GaMD restart file from equilibration. The production phase is set to run in strides of 10 ns or 10000000 steps for Drude and 5000000 steps for C36m and ff19SB. To do so, the template file will be used to generate nearly identical input files for each stride, named `md.*.in`, with * indicating the stride number. The generated input files will be populated with variables such as <BASE>, <BASENAME>, and <CURR>, which are specified in the `sub_gamd_prod.sh`. <BASE> will be populated by the system name the user chooses, while <BASENAME> and <CURR> are automatically populated based on the current stride. <BASENAME> will be set to `gamd_equil` if the current stride is 1, otherwise will be set to the system name, force field, and previous stride number. For example, the <BASENAME> for stride 2 of the Drude alanine dipeptide system would be `ala.drude.2`. Finally, <CURR> is set to the current stride number that is being simulated. These variables will be used as the prefixes for output files that will be read in to begin the next stride. Once the topology and coordinate files, submission script, template files are in the working directory, submit the production run submission script, e.g. via SLURM:

```
$ sbatch sub_gamd_prod.sh
```

For each stride, the coordinate, velocity, box size, and GaMD restart files will be output in a manner similar to equilibration. All the integral boost information required for reweighting will be found in the output files named `md.*.out`.

5.5 Constructing the Free Energy Profile

Once the production simulations have reached the desired simulation length, we can start to extract the information required to reweight our simulations and obtain a free energy profile. The following subsections will detail this process. All steps described here can be carried out using the `gamd_postprocess.sh` script from our GitHub.

5.5.1 Collating Weights and Collective Variables

All important GaMD information is printed to the output files, named `md.*.out`, with the * indicating the numeric stride of production. For reweighting, we first need to combine the data from all strides together into one file, which we will name `gamd_md.log` file. If using the command line, this task can be completed using the `cat` command.

```
$ cat md.{1..50}.out > gamd_md.log
```

Using a command described in the PyReweighting tutorial (https://mccammon.ucsd.edu/computing/amdReweighting/), we can extract the boost potential values for each step and calculate the weights in k_BT using the equipartition theorem, $E_{kinetic} = \frac{3}{2}RT$ (see below, \$6/(0.001987*298)).

```
$ grep "ACCELERATED MD" gamd_md.log | grep
    -v "GAUSSIAN" | grep -v "ACTIVE" | awk
    '{print $6/(0.001987*298)" " $4 " " $6
    " "$8}' > raw_weights.dat
```

This command will created a file called `raw_weights.dat` (Listing 7), containing the boost in k_BT (column 1) alongside the associated simulation step (\$4, column 2), the boost in kcal/mol of that step (\$6, column 3), and the average boost (\$8, column 4).

Listing 7. raw_weights.dat

```
13.3554    0        7.90805     7.90805
16.0597    10000    9.50935     7.69249
13.2625    20000    7.85307     7.5384
14.7218    30000    8.71715     8.167
14.4863    40000    8.57769     7.78162
16.9263    50000    10.0225     8.13965
19.0174    60000    11.2607     8.59088
13.4737    70000    7.97811     8.4957
```

Note that if you are running production in strides like this tutorial, the simulation steps in the weight file will range from 0 to 10000000 for a single stride, then start over at step 0 for the next stride. To use this file for reweighting, the steps will need to be renumbered continuously from 0 through the final step of the simulation. For our simulations of 500 ns, we use a Python script called `fix_time_gamd.py` that will read the `raw_weights.dat` file and renumber the steps to produce a usable `weights.dat` file for reweighting.

The next step is to generate the CVs. Since this tutorial is focused on alanine dipeptide, we chose to use a 2-dimensional reweighting scheme using the backbone ϕ and ψ torsions. Since the output files for production are written in strides of 10 ns, to analyze our systems, we need to generate one trajectory file for the entire simulation. NAMD writes the trajectories in DCD format, so we use the `catdcd` program to compile them all.

```
$ catdcd -o full.dcd ala.19sb.{1..50}.dcd
```

This command will generate a single DCD-formatted trajectory file named `full.dcd` that can then be used for subsequent analysis. For all systems, water was stripped and each system was oriented to remove global rotation and translation of the alanine dipeptide. For Drude and C36m, a in-house script (`orient.inp`) was used to orient the systems with the CHARMM program. We then generated ϕ and ψ angles using another CHARMM script. For ff19SB, an in-house script was used to orient with CPPTRAJ and a Python script was used to generate ϕ and ψ using AmberTools22. To ensure we can have a functional CV file for reweighting, both ϕ and ψ must be in the same file (`cv.dat`) containing only two columns, the first of which contains the time series of ϕ values and the second column contains the time series of ψ values.

At this point, we have a file containing the boost values (`weights.dat`) and a file containing our CVs (`cv.dat`) for the full simulations. Both the weight file and the CV file should be copied to the local workstation (if running the simulations on a remote server), if not done so already. In our simulations, we created directories specifically for each alanine dipeptide system with a `reweight` directory for all our reweighting work. As a reminder, the majority of this post-processing can be done using the `gamd_postprocess.sh`.

5.5.2 Reweighting

All scripts required to perform reweighting and generate free energy surfaces are in the GitHub repository. These scripts execute all steps besides generating the boost distributions and are executed using the `gen_fes_converge.sh` script. Before beginning reweighting, it is beneficial to check the anharmonicity of ΔV to ensure the applied boosts follow a Gaussian, or normal, distribution. To do so, use the `make_dist.py` in the GitHub repository to generate a probability density distributions using the boost in kcal/mol found in the third column of `weights.dat`.

```
$ python make_dist.py -data weights.dat
    -col 2 -out dist_boost.dat
```

Once generated, `dist_boost.dat` can be visualized using XmGrace or any other 2D plotting utility. For each of our systems, we generated the boost distributions, which are shown in Figure 4.

In general, each of the systems produced a distribution of boost potentials that resembles a normal distribution centered around similar boost values. This graph can be complemented by the anharmonicity values, γ, which are also shown in Figure 4. Anharmonicity can be used to characterize the distribution of the applied boost potentials and act as a checkpoint for GaMD simulations. All γ values are close to 0, indicating a Gaussian-shaped distribution in each simulation, irrespective of the force field used. The anharmonicity is an output of reweighting, which will be detailed below.

To begin reweighting, clone the PyReweighting repository into the `reweight` directory. A detailed explanation of the files within the toolkit can be found in Section 4. For this tutorial, we will focus on the files for reweighting in two

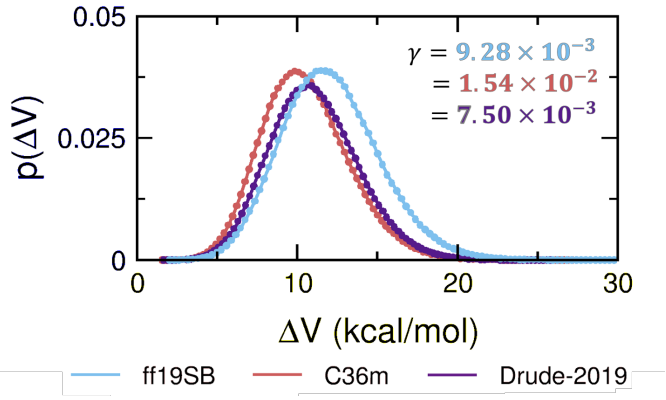


Figure 4. Distribution of boost potential for alanine dipeptide systems.

dimensions: `PyReweighting_2D.py` and `reweight-2D.sh`. In general, the `PyReweighting-2D.py` script will not need to be opened or edited, so it does not need to be in the `reweight` directory. However, `reweight-2D.sh` should be placed in the `reweight` directory in case it needs to be edited for a specific system. The user will also need to specify the path to the `PyReweighting-2D.py` script.

Start by creating a conda environment (see https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html) that is equipped with all Python packages necessary to execute `PyReweighting-2D.py`. At the top of `PyReweighting-2D.py`, there is a section listing the required packages for installation. Specify the Python version to be 3.10 in the conda environment, then install the packages. In our environment, we used SciPy version 1.9.1, NumPy version 1.23.3, and Matplotlib version 3.6.1. We have also provided our environment in the form of a YAML file on GitHub. All versions stated here have been tested for this tutorial but other versions may also work.

Once the environment is set up, activate it to preparing to run the reweighting. The `reweight-2d.sh` script contains a section at the top to set variables required to reweight. These values can either read in via the command line or changed directly in the file, as shown in Listing 8. First, set the path to the `PyReweighting-2D.py` script. This path will depend on where the user decides to save the `pyreweighting` directory. Then, set `Emax` to represent the maximum possible free energy value obtained from the calculations. We set `Emax` to 100 in our script but we tested a range of values for `Emax`, finding that it had little to no impact on the resulting PMF profile. `Emax` essentially acts as a safeguard in the instance that a histogram bin has no occupancy (zero probability), which would otherwise result in an undefined (infinite) energy value. Next, set `cutoff` to 10, which requires that each bin contains at least 10 frames to be considered for reweighting. The x- and y-values will be used for `-Xdim` and `-Ydim` and act as the total possible range of ϕ and ψ values for the dipeptide. Here, we use a range from -180 to 181 as Python will incorporate all values between -180 and 181 but excluding 181. The CV values will be separated into bins depending on the width of each bin, which can be adjusted using `binx` and `biny`. The Python script will then assign these values to the arguments `-discx` and `-discy`. We suggest systematically evaluating the distribution of data points sampled over the range of values along the CVs, such as the ϕ and ψ angles, and setting the bin size accordingly. Here, we chose 15 degrees as the width since it is a factor of 180, a common reporting convention for dihedral angles. In our experience, it is often necessary to test multiple bin sizes to identify which one best represents the data. The CV file, `cv.dat`, is then specified using `data`. Finally, temperature, `T`, is set at 298 K to match our simulation temperature.

Listing 8. Reweight-2d.sh

```
#!/bin/bash
dir_codes=/home/pyreweighting/
Emax=100
cutoff=10
Xmin=-180
Xmax=181
Ymin=-180
Ymax=181
binx=15
biny=15
data="cv.dat"
T=298
```

The only other aspect of the script to check and modify is the name of the weight file. The default file name from the PyReweighting Toolkit is `weights.dat` so if the user's file is named something else, this will need to be edited to read in the correct weight file. Once the `reweight-2d.sh` is edited it can be executed via the command line (e.g., for Bash):

```
$ bash reweight-2d.sh
```

This command will initialize the reweighting process and print a progress report to the terminal and a log file named `reweight_variable.log`. The minimum PMF values for cumulant expansion to the first, second, and third orders will all be printed so one can identify the lowest energy that is then used to normalize all the PMF values such that the lowest value is zero. The total anharmonicity of ΔV is also output in this progress report when `amd_dV` reweighting is used:

```
Anharmonicity of all dV = 0.009284925479752903
```

In this tutorial, we are focused primarily on the outputs of reweighting using cumulant expansion through the `-job amdreweight_CE`. The outputs are all shown in Listing 9.

Listing 9. Reweighting Outputs

```
pmf-2D-c3-cv_500.dat-reweight-discx15-discy15.xvg
pmf-2D-c2-cv_500.dat-reweight-discx15-discy15.xvg
pmf-2D-c1-cv_500.dat-reweight-discx15-discy15.xvg
pmf-2D-cv_500.dat-reweight-CE2-discx15-discy15.png
pmf-2D-cv_500.dat-noweight-discx15-discy15.xvg
pmf-2D-cv_500.dat-noweight-discx15-discy15.png
```

For cumulant expansion to the second order, we will focus on the values in `pmf-2D-c2-cv_500.dat-reweight-discx15-discy15.xvg`. The file is structured with the first CV, ϕ, in the first column and the second CV, ψ, in the second column (Listing 10). The values of ϕ and ψ are in discrete increments of 15 degrees from -180 to 180 as we specified in `reweight-2d.sh`. The corresponding PMF value is in the third column. Put another way, this file shows that for every (ϕ, ψ) pair, there is a corresponding PMF value.

Listing 10. pmf-2D-c2-cv_500.dat-reweight-discx15-discy15.xvg

```
#RC1     RC2      PMF(kcal/mol)

@    xaxis  label "RC1"
@    yaxis  label "RC2"
@TYPE xy
-180.0   -180.0   6.511680581400153
-180.0   -165.0   7.268263041332778
-180.0   -150.0   11.271196453661494
-180.0   -135.0   11.324540393639706
-180.0   -120.0   4.952376113451482
-180.0   -105.0   7.193065613889537
-180.0   -90.0    5.495317166801527
-180.0   -75.0    4.618741411987148
-180.0   -60.0    6.06630454462983
-180.0   -45.0    5.397685481887759
-180.0   -30.0    5.901996954706036
-180.0   -15.0    5.781848121714702
-180.0   0.0      7.741654368167012
...
```

As noted above, the reweighting script normalizes the PMF values such that the global minimum is zero. The lowest free energy value that was printed to the terminal earlier is the offset factor. For example, `pmf_min-c2 = -22.22813527319225` for our ff19SB system, so 22.22813527319225 is added to every PMF value to generate the normalized range of values. Note that these data will be used to produce a Ramachandran plot in the next section, which represents a continuous range of ϕ and ψ values. Plotting the PMF values now will result in a seemingly "missing" row and column corresponding to (ϕ,ψ) = (180, 180) degrees. We included a Python script (`fix_phipsi.py`) in the GitHub repository that will read through the PMF file and copy ϕ and ψ values from -180 to 180. Doing so will generate the symmetry inherent in Ramachandran plots.

```
$ python fix_phipsi.py
    pmf-2D-c2-cv_500.dat-reweight-discx15- \
discy15.xvg fix_pmf_c2.xvg
```

The `fix_phipsi.py` script generates a new file called `fix_pmf_c2.xvg`, which can be used to render the FES and identify the free energy minima.

5.5.3 Plotting the Surface and Identifying Minima

Since we installed the `matplotlib` package into our conda environment, we can continue working within this environment to generate FES. We have included an example Python script, `plot_csv.py`, to generate the FES shown in Figure 5. A complete description of the contents of the script is beyond the scope of this tutorial but we will highlight lines that may be useful to edit depending on the system being analyzed.

zcutoff Conceptually similar to *Emax* from reweighting, the script resets any PMF value above zcutoff to the zcutoff value. In this example, we are using 10.

contour_levels Here you can specify what PMF values you what to outline with contour lines. For this example, we chose PMF values of 0, 2, 4, 6, 8, and 10 kcal/mol.

x and y axes Specifies formatting of the x- and y-axes such as tick marks, label sizes, and label ranges.

The script can be run by specifying the input file, an output image name, and a zcutoff value.

```
$ python plot_csv.py fix_pmf_c2.xvg
    fes_ff19sb_500.png 10
```

This command creates a PNG image file that can be qualitatively analyzed to localize the minima. However, we can also determine the exact x- and y-coordinates on our FES the correspond to these minima. We have provided a `find_minima_2D.py` script that will take user inputs including the PMF file name, output file name, and a PMF threshold, which will be used as search criteria to find any PMF below the specified threshold.

```
$ python find_minima_2d.py
    pmf-2D-c2-cv_500.dat-reweight-discx15- \
discy15.xvg minima.dat 2
```

This command generates a file that designates the minima based on the PMF and the associated ϕ and ψ values to match the FES, as shown in Listing 11.

Listing 11. minima.dat

```
# phi      psi      PMF
-15.0      60.0       0.001988476330325284
30.0       105.0      0.0
75.0       -150.0     0.45605829164178147
165.0      -180.0     0.5940885730567551
```

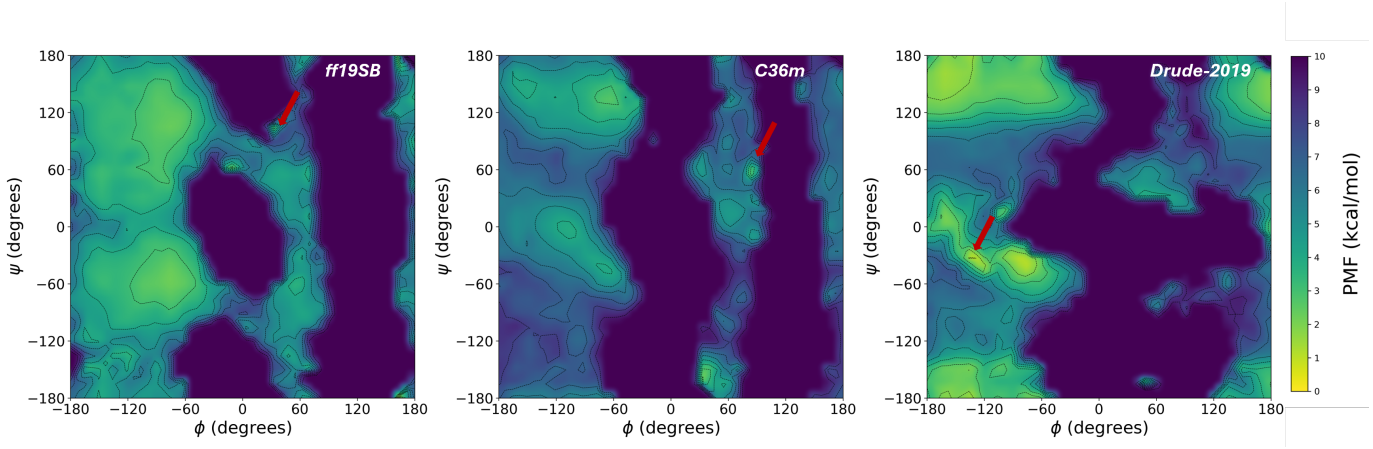


Figure 5. FES for all alanine dipeptide systems. The global minimum in each surface is indicated by a red arrow.

Thus, it is possible to identify the exact location of the global minimum and other local minima, and the associated PMF in kcal/mol. The FES for all three alanine dipeptide systems are shown in Figure 5. The force fields yielded slight differences. For ff19SB and C36m, the global minima are located within the left-handed α-helix region. Typically, this region is not sampled, but due the small methyl side chain of alanine, it experiences fewer steric clashes and ultimately can adopt a more diverse set of dihedral angles, which may explain the location of the global minima in these systems. With the Drude force field, the surface has a global minimum in the α-helical region, which is expected due to the tendency for this force field to sample α_R configurations at slightly higher frequencies than C36m [25].

5.6 Considerations

Throughout the GaMD process, there are a few places of consideration that can potentially alter the system's sampling and FES. We have included a few considerations below that we experienced while setting up our simulations.

5.6.1 Statistical Window size

The `accelMDGStatWindow` keyword is specified in the input scripts for equilibration and production and determines how often statistical averaging for the potential energies is performed and how often the boost is recalculated. Conventionally, this value should be around four times the total number of particles in the system. For the alanine dipeptide systems, this convention would suggest a window between 10000 and 20000 steps would be appropriate. For the tutorials, we used a value of 10000 for each of our systems, but we performed an additional simulation using 20000 to compare the resulting FES. The surfaces for ff19SB with `accelMDGStatWindow` values of 10000 and 20000 steps are shown in Figure 6. The general shape of the surfaces differ only slightly although the surface arising from the simulation with `accelMDGStatWindow` set to 20000 is systematically higher by 3-4 kcal/mol than that of the system with `accelMDGStatWindow` set to 10000. Applying `accelMDGStatWindow` 20000 did have a lower energy minimum prior to normalization during reweighting, which could be the cause of the shift in PMF. Additionally, these differences could be a result of under- or over-sampling of the same state across these two systems and may therefore not be directly linked to the specified value of `accelMDGStatWindow`.

5.6.2 Bin Width

Bin width is specified during the reweighting process and will impact the way the data are binned, thus potentially altering the location of the global minimum on the reweighted FES. We compared the use of different bin sizes by reweighting using `-discx` and `-discy` set to 6 and 15 (Figure 6). Bin size will be based on the amount of CV data collected as well as the range of the data values. Previously published GaMD studies using alanine peptide used a bin size of 6°.[2] In our approach, such a small bin size was not appropriate to visualize the surfaces (Figure 6, top and bottom left), most likely because we saved fewer frames than the previously published studies, implying that the bin width may be directly connected to the number of data points saved during the simulation.

5.6.3 Bin Cutoff

Another consideration that may be dependent upon the number of frames is the bin cutoff. This cutoff is specified during the reweighting process to ensure undersampled bins do not dominate the free energy surfaces. The value specified using `cutoff` sets a minimum number of frames required for that bin to be used during reweighting, any bin

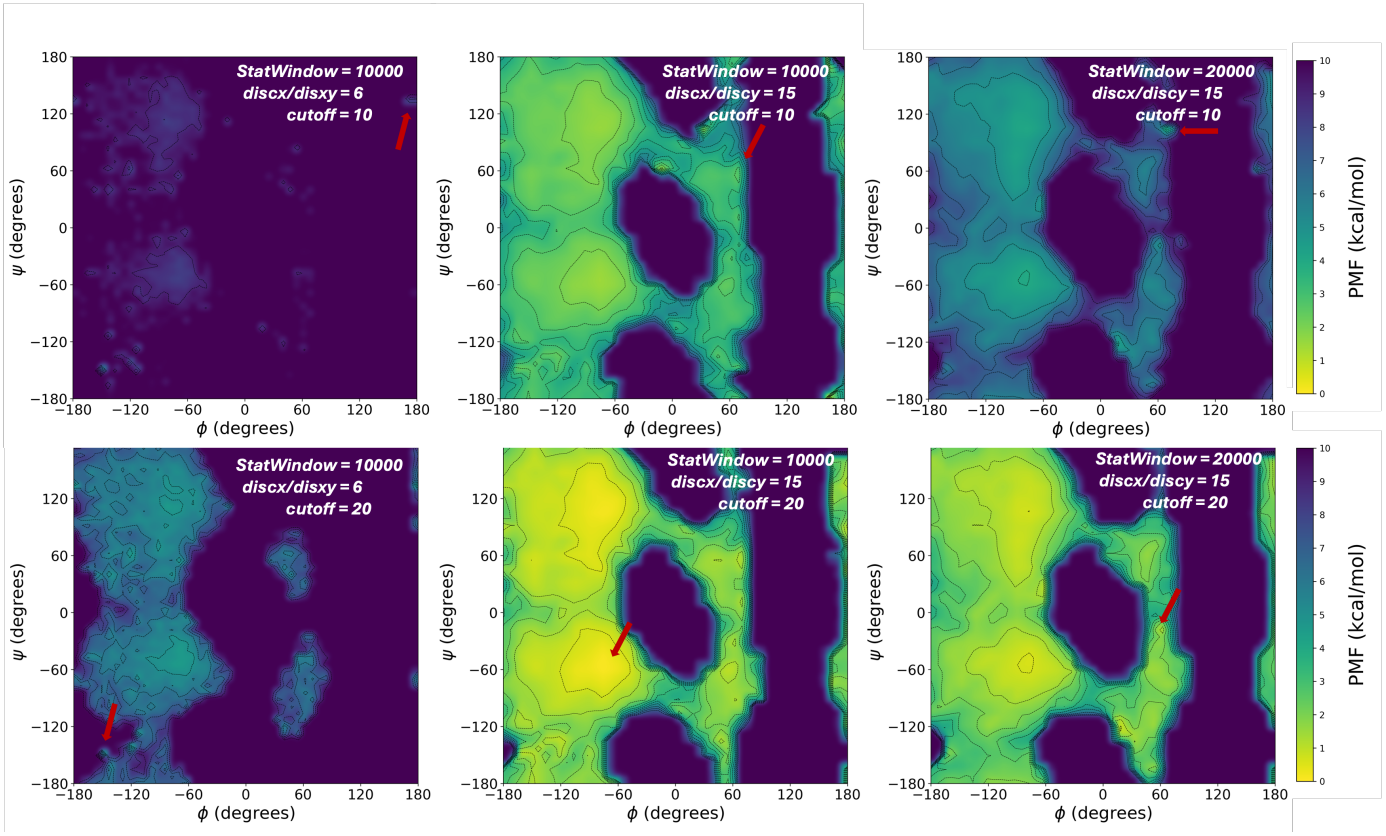


Figure 6. Comparison of the free energy surfaces for ff19SB simulations using `accelMDGStatWindow` values of 10000 and 20000.

containing a number of frames less than the specified value is not considered during reweighting. Thus, different cutoff values can result in FES with different global minima (Figure 6).

5.6.4 Simulation Convergence

An integral component of running MD simulations is ensuring that the system sampled as much of the conformational landscape as theoretical possible. In the case of GaMD, we evaluated the convergence of the data used to produce our FES. We generated surfaces across intervals of 50, 100, 200, 300, 400, and 500 ns to determine if the features of the FES were still changing. These surfaces for ff19SB, C36m, and Drude-2019 are shown in Figures 7 - 9. As mentioned in Section 5.5.2, the `gen_convergence_fes.py` script in our GitHub repository will generate all these surfaces.

5.7 Summary and Review of Objectives

In this tutorial, the user has been guided through the process of completing a GaMD simulation. To review, the objectives for this tutorial were as follows:

1. Identify key elements of the theory underlying GaMD simulations
2. Prepare and run GaMD simulations using NAMD
3. Reweight and analyze the GaMD free energy profiles using the PyReweighting toolkit

The first step towards achieving these objectives was the setup of conventional MD and GaMD equilibration in which the type of boost (total, dihedral, or dual) was specified, along with the number of simulation steps. The output files obtained from GaMD and NAMD were reviewed, with special attention given to `gamd_equil.out`, which contains all the statistical information used to calculate the boost and determine the extent of equilibration convergence, and `gamd_equil.gamd`, which contains the threshold energy to be used for production. The system was then set up for GaMD production, during which the boost for each step was calculated based on the difference between finalized threshold energy from equilibration and the system potential energy at that step. The tutorial concluded with the generation and visualization of free energy profiles using ϕ and ψ dihedrals for the alanine dipeptide system.

6 Author Contributions

HM Michel developed and wrote the tutorial. MD Polêto and JA Lemkul provided guidance for tutorial development.

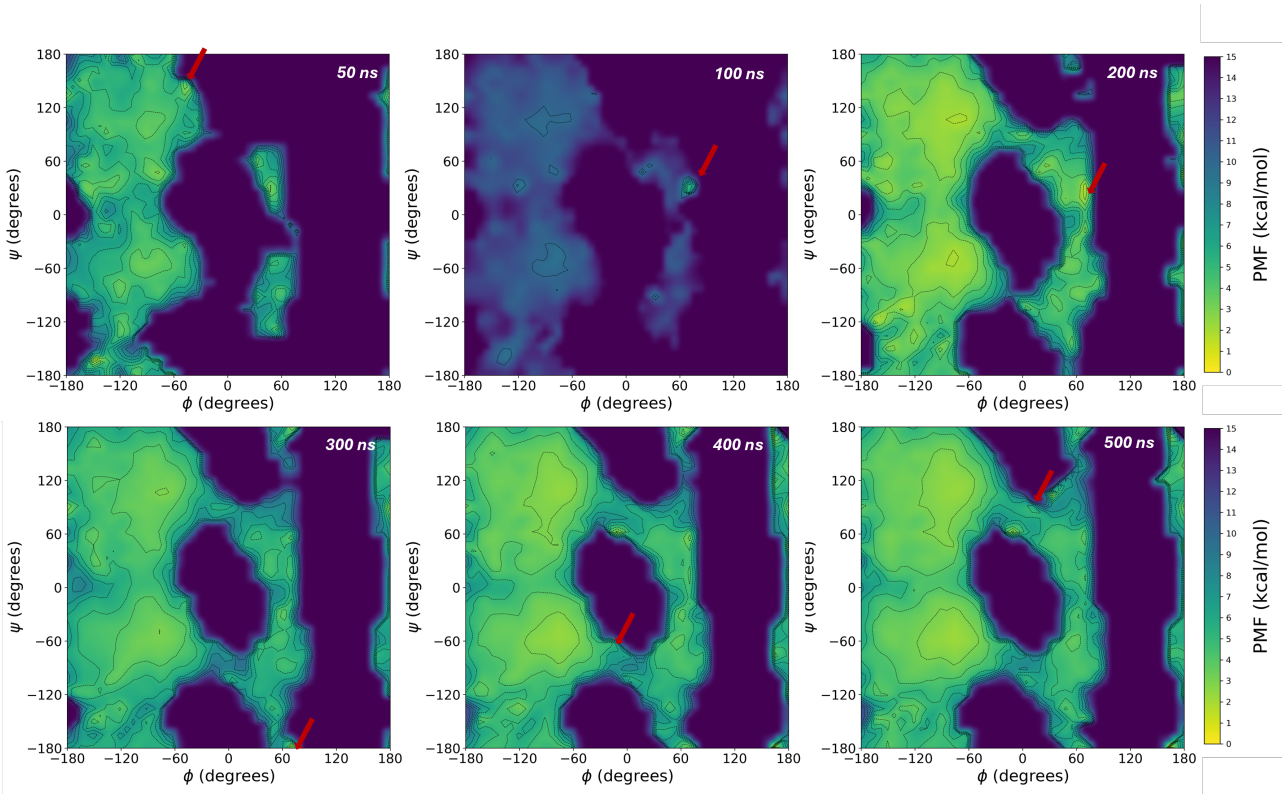


Figure 7. Time evolution of the FES for ff19SB simulations.

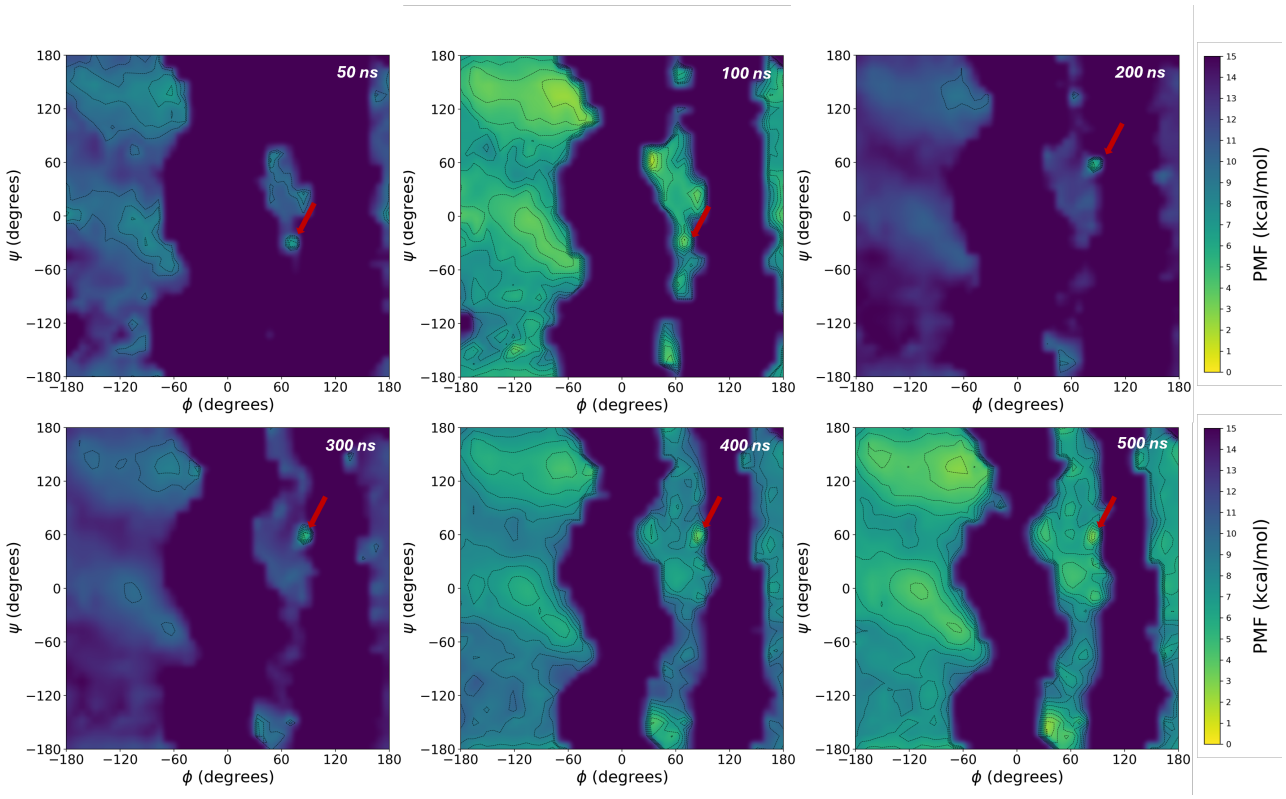


Figure 8. Time evolution of the FES for C36m simulations.

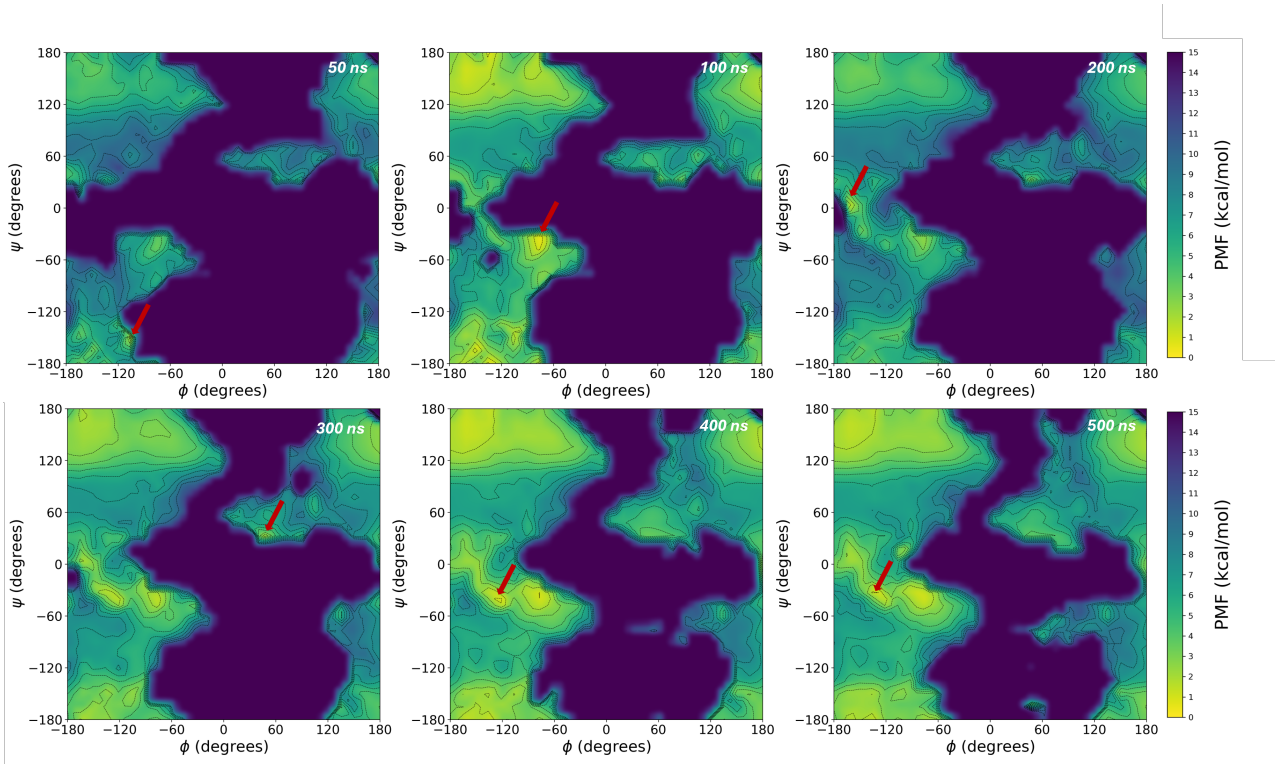


Figure 9. Time evolution of the FES for Drude-2019 simulations.

7 Other Contributions

We thank all members of the Lemkul Lab for helpful discussions and feedback. We specifically thank Laura I. Gil Pineda, Rebekah J. Fogarty, and Patrick C. Gilles for providing detailed feedback and extensive testing throughout the development of the tutorials. Additionally, we would like to thank Aakash Saha and Pablo R. Arantes for insightful discussions about the GaMD process in AMBER.

8 Potentially Conflicting Interests

The authors declare no conflicting interests.

9 Funding Information

This work was supported by the National Institutes of Health (grant R35GM133754 to JAL) and USDA-NIFA (project VA-160211 to JAL).

References

[1] **Salomon-Ferrer R**, Case DA, Walker RC. An overview of the Amber biomolecular simulation package. Wiley Interdiscip Rev Comput Mol Sci. 2013; 3(2):198–210. https://doi.org/10.1002/wcms.1121, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.1121.

[2] **Miao Y**, Feher VA, McCammon JA. Gaussian Accelerated Molecular Dynamics: Unconstrained Enhanced Sampling and Free Energy Calculation. J Chem Theory Comput. 2015; 11(8):3584–3595. https://doi.org/10.1021/acs.jctc.5b00436, pMID: 26300708.

[3] **Copeland MM**, Do HN, Votapka L, Joshi K, Wang J, Amaro RE, Miao Y. Gaussian Accelerated Molecular Dynamics in OpenMM. J Phys Chem B. 2022; 126(31):5810–5820. https://doi.org/10.1021/acs.jpcb.2c03765.

[4] **Célerse F**, Inizan TJ, Lagardère L, Adjoua O, Monmarché P, Miao Y, Derat E, Piquemal JP. An Efficient Gaussian-Accelerated Molecular Dynamics (GaMD) Multilevel Enhanced Sampling Strategy: Application to Polarizable Force Fields Simulations of Large Biological Systems. J Chem Theory Comput. 2022; 18(2):968–977. https://doi.org/10.1021/acs.jctc.1c01024.

[5] **Huang YmM**, McCammon JA, Miao Y. Replica Exchange Gaussian Accelerated Molecular Dynamics: Improved Enhanced Sampling and Free Energy Calculation. J Chem Theory Comput. 2018; 14(4):1853–1864. https://doi.org/10.1021/acs.jctc.7b01226, publisher: American Chemical Society.

[6] **Ahn SH**, Ojha AA, Amaro RE, McCammon JA. Gaussian-Accelerated Molecular Dynamics with the Weighted Ensemble Method: A Hybrid Method Improves Thermodynamic and Kinetic Sampling. J Chem Theory Comput. 2021; 17(12):7938–7951. https://doi.org/10.1021/acs.jctc.1c00770.

[7] **Miao Y**, Bhattarai A, Wang J. Ligand Gaussian Accelerated Molecular Dynamics (LiGaMD): Characterization of Ligand Binding Thermodynamics and Kinetics. J Chem Theory Comput. 2020; 16(9):5526–5547. https://doi.org/10.1021/acs.jctc.0c00395, publisher: American Chemical Society.

[8] **Wang J**, Miao Y. Ligand Gaussian Accelerated Molecular Dynamics 2 (LiGaMD2): Improved Calculations of Ligand Binding Thermodynamics and Kinetics with Closed Protein Pocket. J Chem Theory Comput. 2023; 19(3):733–745. https://doi.org/10.1021/acs.jctc.2c01194, publisher: American Chemical Society.

[9] **Wang J**, Miao Y. Peptide Gaussian accelerated molecular dynamics (Pep-GaMD): Enhanced sampling and free energy and kinetics calculations of peptide binding. J Chem Phys. 2020; 153(15):154109. https://doi.org/10.1063/5.0021399.

[10] **Wang J**, Miao Y. Protein–Protein Interaction-Gaussian Accelerated Molecular Dynamics (PPI-GaMD): Characterization of Protein Binding Thermodynamics and Kinetics. J Chem Theory Comput. 2022; 18(3):1275–1285. https://doi.org/10.1021/acs.jctc.1c00974, publisher: American Chemical Society.

[11] **Harris CR**, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, et al. Array programming with NumPy. Nature. 2020; 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2.

[12] **Virtanen P**, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nat Methods. 2020; 17:261–272. https://doi.org/10.1038/s41592-019-0686-2.

[13] **Case DA**, Aktulga HM, Belfon K, Cerutti DS, Cisneros GA, Cruzeiro VWD, Forouzesh N, Giese TJ, Götz AW, Gohlke H, Izadi S, Kasavajhala K, Kaymak MC, King E, Kurtzman T, Lee TS, Li P, Liu J, Luchko T, Luo R, et al. AmberTools. J Chem Inf Model. 2023; 63(20):6183–6191. https://doi.org/10.1021/acs.jcim.3c01153.

[14] **Roe DR**, Cheatham TEI. PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data. J Chem Theory Comput. 2013; 9(7):3084–3095. https://doi.org/10.1021/ct400341p, publisher: American Chemical Society.

[15] **Hwang W**, Austin SL, Blondel A, Boittier ED, Boresch S, Buck M, Buckner J, Caflisch A, Chang HT, Cheng X, Choi YK, Chu JW, Crowley MF, Cui Q, Damjanovic A, Deng Y, Devereux M, Ding X, Feig MF, Gao J, et al. CHARMM at 45: Enhancements in Accessibility, Functionality, and Speed. J Phys Chem B. 2024; 128(41):9976–10042. https://doi.org/10.1021/acs.jpcb.4c04100, pMID: 39303207.

[16] **Hamelberg D**, de Oliveira CAF, McCammon JA. Sampling of slow diffusive conformational transitions with accelerated molecular dynamics. J Chem Phys. 2007; 127(15):155102. https://doi.org/10.1063/1.2789432.

[17] **Wang Y**, Harrison CB, Schulten K, McCammon JA. Implementation of Accelerated Molecular Dynamics in NAMD. Comput Sci Discov. 2011; 4(1):015002. https://doi.org/10.1088/1749-4699/4/1/015002.

[18] **Shen T**, Hamelberg D. A statistical analysis of the precision of reweighting-based simulations. J Chem Phys. 2008; 129(3):034103. https://doi.org/10.1063/1.2944250.

[19] **Miao Y**, Sinko W, Pierce L, Bucher D, Walker RC, McCammon JA. Improved Reweighting of Accelerated Molecular Dynamics Simulations for Free Energy Calculation. J Chem Theory Comput. 2014; 10(7):2677–2689. https://doi.org/10.1021/ct500090q, publisher: American Chemical Society.

[20] **Wang J**, Arantes PR, Bhattarai A, Hsu RV, Pawnikar S, Huang YmM, Palermo G, Miao Y. Gaussian accelerated molecular dynamics: Principles and applications, vol. 11; 2021. https://doi.org/10.1002/wcms.1521.

[21] **Pang YT**, Miao Y, Wang Y, McCammon JA. Gaussian Accelerated Molecular Dynamics in NAMD. J Chem Theory Comput. 2017; 13(1):9–19. https://doi.org/10.1021/acs.jctc.6b00931.

[22] **Bhakat S**. Collective variable discovery in the age of machine learning: reality, hype and everything in between. RSC Adv. 2022; 12(38):25010–25024. https://doi.org/10.1039/D2RA03660F, publisher: The Royal Society of Chemistry.

[23] **Tian C**, Kasavajhala K, Belfon KAA, Raguette L, Huang H, Migues AN, Bickel J, Wang Y, Pincay J, Wu Q, Simmerling C. ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. J Chem Theory Comput. 2020; 16(1):528–552. https://doi.org/10.1021/acs.jctc.9b00591, publisher: American Chemical Society.

[24] **Huang J**, Rauscher S, Nawrocki G, Ran T, Feig M, De Groot BL, Grubmüller H, MacKerell AD. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. Nat Methods. 2017; 14(1):71–73. https://doi.org/10.1038/nmeth.4067.

[25] **Lin FY**, Huang J, Pandey P, Rupakheti C, Li J, Roux B, MacKerell AD. Further Optimization and Validation of the Classical Drude Polarizable Protein Force Field. J Chem Theory Comput. 2020; 16(5):3221–3239. https://doi.org/10.1021/acs.jctc.0c00057.

[26] **Lemkul JA**. Preparing and Analyzing Polarizable Molecular Dynamics Simulations with the Classical Drude Oscillator Model. In: Moreira IS, Machuqueiro M, Mourão J, editors. *Computational Design of Membrane Proteins* New York, NY: Springer US; 2021.p. 219–240. https://doi.org/10.1007/978-1-0716-1468-6_13.

[27] **Feller SE**, Zhang Y, Pastor RW, Brooks BR. Constant pressure molecular dynamics simulation: The Langevin piston method. J Chem Phys. 1995; 103(11):4613–4621. https://doi.org/10.1063/1.470648.

[28] **Darden T**, York D, Pedersen L. Particle mesh Ewald: An $N \cdot log(N)$ method for Ewald sums in large systems. J Chem Phys. 1993; 98(12):10089–10092. https://doi.org/10.1063/1.464397.

[29] **Lamoureux G**, Harder E, Vorobyov IV, Roux B, MacKerell AD. A polarizable model of water for molecular dynamics simulations of biomolecules. Chem Phys Lett. 2006; 418(1):245–249. https://doi.org/https://doi.org/10.1016/j.cplett.2005.10.135.

[30] **Jorgensen WL**, Chandrasekhar J, Madura JD, Impey RW, Klein ML. Comparison of simple potential functions for simulating liquid water. J Chem Phys. 1983; 79(2):926–935. https://doi.org/10.1063/1.445869, publisher: American Institute of Physics.

[31] **Durell SR**, Brooks BR, Ben-Naim A. Solvent-Induced Forces between Two Hydrophilic Groups. J Phys Chem. 1994; 98(8):2198–2202. https://doi.org/10.1021/j100059a038, publisher: American Chemical Society.

[32] **Neria E**, Fischer S, Karplus M. Simulation of activation free energies in molecular systems. J Chem Phys. 1996; 105(5):1902–1921. https://doi.org/10.1063/1.472061, publisher: American Institute of Physics.