FREE ENERGY SIMULATIONS OF COMPLEX BIOLOGICAL SYSTEMS AT CONSTANT PH

By

JASON M. SWAILS

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

© 2013 Jason M. Swails

I dedicate this dissertation to the late Professor Frederick P. Arnold whose intelligence, excitement, and guidance propelled me into this field.

ACKNOWLEDGMENTS

I would like to thank my parents, Mark and Mindy Swails, for their love, guidance, and constant encouragement.

I thank my siblings, Kerri and Jeffrey Swails, for all the great times and their (vain) attempts to keep me humble.

Thank you to my extended family for the incredible support system I've always had growing up.

The Roitberg Group provided a great deal of support and comaraderie, and Adrian Roitberg provided guidance and instruction during my graduate studies.

I also thank Quantum Theory Project for all the good times.

And finally, I would like to thank my wife, Roxy Lowry Swails, for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure page

LIST OF ABBREVIATIONS

API	Application Programmer Interface
AMBER	Assisted Model Building with Energy Refinement
BOA	Born-Oppenheimer Approximation
CMAP	Correction Map
CpHMD	Constant pH Molecular Dynamics
DNA	Deoxyribonucleic Acid
EAF	Exchange attempt frequency
ESP	Electrostatic Potential
FEP	Free Energy Perturbation
FFT	Fast Fourier Transform
GBSA	Generalized Born with Surface Area
GPU	Graphical Processing Unit
GRF	Generalized Reaction Field
H-REMD	Hamiltonian Replica Exchange Molecular Dynamics
HEWL	Hen egg-white lysozyme
IPS	Isotropic Periodic Sum
LIE	Linear Interaction Energy
LJ	Lennard-Jones
MC	Monte Carlo
MD	Molecular Dynamics
MM	Molecular Mechanics

MPI	Message Passing Interface
MO	Molecular Orbital
MTP	Multiple Trajectory Protocol
PBC	Periodic Boundary Conditions
PBSA	Poisson-Boltzmann with Surface Area
pH-REMD	pH Replica Exchange Molecular Dynamics
PMF	Potential of Mean Force
QM	Quantum Mechanics
REFEP	Replica Exchange Free Energy Perturbation
REMD	Replica Exchange Molecular Dynamics
RESP	Restrained Electrostatic Potential
RMSD	Root mean squared deviation
RNA	Ribonucleic Acid
STP	Single Trajectory Protocol
T-REMD	Temperature Replica Exchange Molecular Dynamics
TI	Thermodynamic Integration

LIST OF CONSTANTS AND OPERATORS

$erf(x)$	$\frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt$	Error function
$erfc(x)$	$1 - erf(x)$	Complementary Error Function
h	$6.626068 \times 10^{-34} m^2 kg/sec$	Planck's Constant [1]
i	$\sqrt{-1}$	Imaginary unit
$\vec{\bigtriangledown}$	$(\frac{\partial}{\partial \vec{x}}, \frac{\partial}{\partial \vec{y}}, \frac{\partial}{\partial \vec{z}})$	Gradient Operator
$\bigtriangledown^2$	$\bigtriangledown \cdot \bigtriangledown = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$	Laplace Operator
k_B	1.380 658(12) $\times 10^{-23}$ J K^{-1}	Boltzmann's constant [1]

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

FREE ENERGY SIMULATIONS OF COMPLEX BIOLOGICAL SYSTEMS AT CONSTANT PH

By

Jason M. Swails

August 2013

Chair: Adrian E. Roitberg
Major: Chemistry

Solution pH has profound effects on the structure, function, and activity of many complex biomolecules that catalyze the chemical reactions responsible for sustaining life. Even focusing on the human body, the various physiological environments span a wide pH range—as low as 1 in the stomach to values as high as 8.1 in pancreatic secretions. Small changes from the normal pH of a biomolecule's environment can be catastrophically disruptive to its activity. For example, a change in pH of as little as $\pm$0.1 pH units in the human bloodstream is enough to cause life-threatening alkalosis or acidosis.

Due to the importance of pH in biology and the profound effect it can have on biomolecules, it is important to incorporate pH effects in computational models designed to treat these biomolecules. The solution pH controls protonation state equilibria of specific functional groups prevalent in biomolecules, such as carboxylates, amines, and imidazoles. These protonation states in turn affect the charge distribution in the biomolecule which can have a significant impact on both its 3-dimensional structure as well as interactions with the surrounding environment. In many cases, pH can also impact whether or not a proton donor or acceptor will be available for catalysis during the course of the biocatalytic mechanism.

The aim of my work is to develop accurate and efficient computational models to probe the pH-dependent behavior of proteins and nucleic acids. The models must

be carefully designed to obey the laws of thermodynamics under the constraint of an externally applied pH. Only then can the results be directly compared to experimental measurements.

In this dissertation, I present my work on the development of pH-based models for biomolecules and other work performed in the area of molecular modelling. In the first chapter, I introduce the fundamental concepts computational biomolecular modeling that lay the foundation for the presented work. This is followed by chapters on free energy and sampling, constant pH simulations, replica exchange, and some useful tools I developed to aid in conducting computational research with the Amber simulation package.

CHAPTER 1
INTRODUCTION

1.1 Origins of Computational Chemistry

The seeds of computational chemistry were sown in the mid-1800s with Ludwig Eduard Boltzmann's formulation of *statistical mechanics*. In an era when the existence of atoms and molecules was hotly disputed within the physics community, Boltzmann fathered a theory in which the behavior and interaction of individual atoms or molecules on the microscopic scale could be used to describe and predict macroscopic phenomena. Using his theorems and equations, it became possible to reduce the problem of simulating $\sim 10^{23}$ molecules to simulating ~ 1 molecule. Boltzmann used this to great effect in describing and deriving previously-known, phenomenological equations for ideal gases, such as the widely known equation of state, $PV = nRT$. All that remains to provide the foundation for using molecular simulations is the proper description of atoms and molecules on the microscopic scale.

The theories required to accurately model the behavior of individual atoms interacting with each other and their surroundings would not be developed until the first half of the 20^{th} century with the advent of *quantum mechanics*. The limits of classical mechanics became apparent when considering the Rayleigh-Jeans formula for calculating the spectral emission of a radiating black body. The Rayleigh-Jeans law, given by Eq. 1–1, is completely derived using the laws of classical mechanics. By looking at Eq. 1–1, we see that the emission spectrum diverges at high frequencies (ν)—a clear violation of the well-established law of conservation of energy.

$$B_\nu(T) = \frac{2\nu^2 kT}{c^2} \tag{1–1}$$

To address this apparent disparity, Max Planck suggested that the error in the classical mechanical approach was to assume a continuous emission spectrum. Instead, Planck suggested that the emission spectra was quantized, leading to an equation that agreed

much closer with experiment. This idea of quantized energy emissions, while developed to reconcile the mathematics of black-body radiation with experimental measurements, would forever change our understanding of the microscopic world.

As quantum mechanics matured, our ability to explain and predict behavior at the atomic scale dramatically improved. In 1929, Paul Dirac proclaimed, “The fundamental laws necessary for the mathematical treatment of a large part of physics and the whole of chemistry are thus completely known, and the difficulty lies only in the fact that the application of these laws leads to equations that are too complex to be solved.” Even approximations designed to simplify the equations of quantum mechanics in molecular systems resulted in computations too complex to apply to all but the simplest systems. With the fundamental theory necessary to describe single molecules and the machinery required to extend that description to experimental measurements at our disposal, computers provided the catalyst that thrusted theoretical chemistry into a prominent role in chemical research.

The next sections will describe the theory of quantum mechanics and the approximations typically employed to simplify its equations, followed by a description of statistical mechanics.

1.1.1 Quantum Mechanics

Twenty years after Planck introduced the idea of quantized oscillators to explain black-body radiation, Erwin Schrödinger introduced a wave equation formulation of quantum mechanics (QM). [2] Schrödinger’s equation (Eq. 1–2) bears a strong resemblance to Hamilton’s formulation of classical mechanics by employing an analogous *Hamiltonian* operator comprised of a kinetic energy term (related to the momentum operator) and a potential energy term.

$$\begin{aligned} E\Psi(\vec{x},t) &= \hat{H}\Psi(\vec{x}) \\ &= \left(-\frac{\hbar^2}{2m}\bigtriangledown^2 + V(\vec{x})\right)\Psi(\vec{x}) \end{aligned} \tag{1–2}$$

In Eq. 1–2, E is the total energy, $\hat{H}$ is the Hamiltonian operator, and $\Psi(\vec{x}, t)$ is the wavefunction—the central object of Schrödinger's equation containing all of the information and properties inherent to the system.

Eq. 1–2 is a special form of Schrödinger's equation corresponding to a stationary state (*i.e.,* the potential function is time-independent, so the energy for that state is constant). In chemistry when we wish to calculate observable properties of a system composed of atoms, the kinetic energy is the sum of the kinetic energies of the atomic particles in the system, and the potential energy is calculated as the interaction of all charged particles—protons and electrons—in the electric field they create (plus any external field that may be present).

The wavefunction contains all of the information about each of the particles in the system. As the number of particles in the system increases, so too does the complexity of the wavefunction and the effort required to solve Eq. 1–2. Therefore, we turn to a number of approximations developed to simplify computing a solution to Schrödinger's equation.

1.1.1.1 Born-Oppenheimer Approximation

The Born-Oppenheimer approximation (BOA) is pervasive in the field of computational chemistry. Electrons can move far more rapidly than nucleons since electrons are roughly 1000 times lighter. This implies that electrons can reorganize around moving nuclei so quickly that nuclear protons are always subject to the potential from the average electric field generated by the electrons.

Using the BOA, the wavefunction of a molecular system can be separated into two parts: an electronic part where the nuclei are treated as fixed point charges, and a nuclear part where the nuclei move through the average electric field generated by the electrons. [3] So critical is the BOA to computational chemistry that it appears at the heart of nearly every computational molecular model.

1.1.1.2 Computational Quantum Mechanics

The main goal of most QM calculations in chemistry and molecular physics is to determine atomic and molecular properties of the system by estimating the electronic part of the wavefunction from the BOA. These calculations have provided valuable assistance to experimental investigations. QM calculations can provide reliable measurements of molecular geometries, [4] potential and free energy barriers of chemical reactions, [5] ionization energies, [6] proton affinities and gas-phase basicities [7], and many other chemical and molecular properties. [8]

These calculations are becoming routine as more and more experimental studies employ some form of calculation to help interpret results or strengthen conclusions. Despite all their successes and the rapid increase of computational power over recent years, however, the computational demands of QM methods often remain prohibitively high for systems with more than 100 – 200 atoms. Furthermore for researchers interested in these large systems, calculations on a single arrangement of atomic nuclei becomes increasingly insufficient to quantify the behavior of those systems.

For such applications, we turn our attention back to statistical mechanics with the aim of ultimately applying those principles to molecular mechanical simulations of large molecules that often contain thousands—even hundreds of thousands—of atoms.

1.1.2 Statistical Mechanics

Macroscopic chemical systems are composed of a vast number of atoms—on the order of Avogadro's number, or 6.022×10^{23}. How, then, can our calculations of a single molecule or a small cluster of molecules be used to predict the behavior of a collection of 10^{23} molecules? For that we turn to statistical mechanics and the idea of an *ensemble*.

In a system with N particles (where N is typically on the order of Avagadro's number in magnitude), there are $6N$ total degrees of freedom in the system corresponding to the position and momentum of each particle in all three dimensions. This ultra-high,

$6N$-dimensional space is referred to as *phase space*, and the collection of all points that conform to a small set of thermodynamic constraints—*e.g.,* constant volume or energy—represents an ensemble. [9] The connection between this imaginary ensemble of systems and experimental measurements of real systems was provided by Josiah Gibbs. The experimental value of any system in the lab is postulated to be equal to the value of that mechanical observable averaged over every member of the ensemble. [9] By knowing the probability of finding a member of an ensemble with a given set of properties, this ensemble average can be calculated according to Eq. 1–3.

$$\begin{aligned}\langle A\rangle &= \frac{\sum_a W(a)A(a)}{\sum_a W(a)} \\ &= \sum_a P(a)A(a)\end{aligned} \tag{1–3}$$

$W(a)$ in Eq. 1–3 can be thought of as the number of states in the ensemble with the same value of A. $P(a)$ is the normalized probability for that state, where $\sum_a W(a)$ is the normalization factor.

Given that there are on the order of 10^{23} particles in the typical system, the number of ensemble members from which the average is calculated appears at first glance to be intractable. However, it turns out that the mean square fluctuations of measurable properties within the ensemble scale as roughly $1/\sqrt{N}$ where N is the number of particles in the system. Because N is on the order of 10^{23}, the fluctuations around the most probable value in the ensemble vanish and the ensemble average and most probable value become identical. The problem of calculating the ensemble average of a desired property is reduced to the far more tractable task of calculating its most probable value.

The most commonly used ensemble, called the *canonical* ensemble, is constrained such that each member has the same number of particles, volume, and temperature (NVT). Other common ensembles include the microcanonical ensemble (NVE), the

grand canonical ensemble (μVT), and the isobaric-isothermal ensemble (NpT), where E, μ, and p stand for constant energy, chemical potential, and pressure, respectively. At typical temperatures and particle densities, the fluctuations of mechanical properties in each of these ensembles becomes negligible. Therefore, these ensembles are effectively equivalent to one another, allowing us to choose the one that is most convenient to work with mathematically.

The link between these ensembles and thermodynamics is the natural logarithm of the *partition function*, which happens to be the normalization constant from Eq. 1–3 for each of the ensembles. The partition functions of the common ensembles are shown in Eqs. 1–4 to 1–7.

The logarithm of the partition function for each ensemble is directly proportional to the thermodynamic function that has the same set of 'natural' variables. These connections are summarized in Eqs. 1–8 to 1–11. [9]

$$\text{Microcanonical} \qquad \Omega(N, V, E) = \omega(E) \tag{1–4}$$

$$\text{Canonical} \qquad Q(N, V, T) = \sum_{E} \Omega(N, V, E) \exp(-\beta E) \tag{1–5}$$

$$\text{Grand Canonical} \qquad \Xi(\mu, V, T) = \sum_{N} Q(N, V, T) \exp(\beta \mu N) \tag{1–6}$$

$$\text{Isobaric-Isothermal} \qquad \Delta(N, p, T) = \sum_{V} Q(N, V, T) \exp(-\beta p V) \tag{1–7}$$

In Eqs. 1–4 to 1–7, ω is the total number of states with a given energy and β is $1/k_B T$. According to the principle of equal *a priori* probabilities, all states in the microcanonical ensemble are considered equally probable simply because there is no reason to assume otherwise.

Microcanonical	$S = k \ln(\Omega(N, V, E))$	(1–8)
Canonical	$A = -kT \ln(Q(N, V, T))$	(1–9)
Grand Canonical	$pV = kT \ln(\Xi(\mu, V, T))$	(1–10)
Isobaric Isothermal	$G = -kT \ln(\Delta(N, p, T))$	(1–11)

With the link to classical thermodynamics now firmly established through the partition function, statistical mechanics can now explain the whole of thermodynamics from the microscopic behavior of individual atoms and molecules. One of the principle challenges of computational chemistry becomes how to efficiently estimate the partition function.

Significant effort in computational chemistry centers on estimating the canonical partition function $Q(N, V, T)$. The naïve approach to calculate the sum in 1–5 would be to calculate the degeneracy of each energy level ($\Omega(N, V, E)$) and scale it with the exponential weighting factor ($\exp(-\beta E)$) called the *Boltzmann factor*. Due to the immeasurable size of $\Omega(N, V, E)$, however, this approach is highly inefficient in practice. It turns out that most of the effort put into computing $Q(N, V, T)$ results in terms that contribute very little to the partition function since there are more available states at higher energies (which carry little weight with the Boltzmann factor).

Because it is infeasible to calculate the full sums in Eqs. 1–4 to 1–7, partition functions are estimated using a representative subsample of the available points to construct the needed distributions. The strategies of generating these subsamples are collectively referred to as *sampling*. The two most common approaches—Monte Carlo and Molecular Dynamics—are discussed in the following sections.

1.1.2.1 Monte Carlo

One approach to approximating Eq. 1–5, called *Monte Carlo* (MC) sampling, is to select new configurations of atomic positions at random in the molecular system,

evaluate the energy of that structure, and add its contribution—weighted by the Boltzmann factor—to the sum of the partition function. This is equivalent to reorganizing the sum in Eq. 1–5 to sum over individual states rather than energy levels. Eq. 1–5 is then estimated as

$$Q(N, V, T) \approx \sum_{i=1}^{N_{samples}} \exp(-\beta E_i) \tag{1–12}$$

Because we assume no prior knowledge of phase space beforehand, using random configurations in MC sampling is critical to avoid introducing bias into the subsample. The MC approach to approximating the partition function proves to be highly inefficient, however, as most random atomic configurations in a chemical system correspond to species that are unphysical and contribute $\sim$0 to the partition function.

For example, consider characterizing the phase space of an ethane molecule using MC. A random configuration is generated by placing both carbon atoms and all six hydrogen atoms at a random point in space, evaluating the energy of that configuration using a QM calculation, and adding that term to the summation in Eq. 1–12. Figure 1-1 depicts two conformations of ethane with an equal probability of being chosen, only one of which will have an energy low enough to contribute significantly to $Q(N, V, T)$. It should be easy to see that there are far more unphysical arrangements of the atoms in ethane than chemically reasonable ones.

While MC suffers severe limitations, Metropolis et al. proposed a modification to the traditional MC approach that helped alleviate many of the problems described above. [10] This variant, described below, is called *Metropolis Monte Carlo* after the method's architect.

Metropolis Monte Carlo

Metropolis's breakthrough in MC methods is a subtle change to the standard approach. Instead of generating random structures and adding them all to the ensemble with a weight equal to the Boltzmann factor, random structures are generated and accepted as *full* members of the ensemble with a probability proportional to the Boltzmann

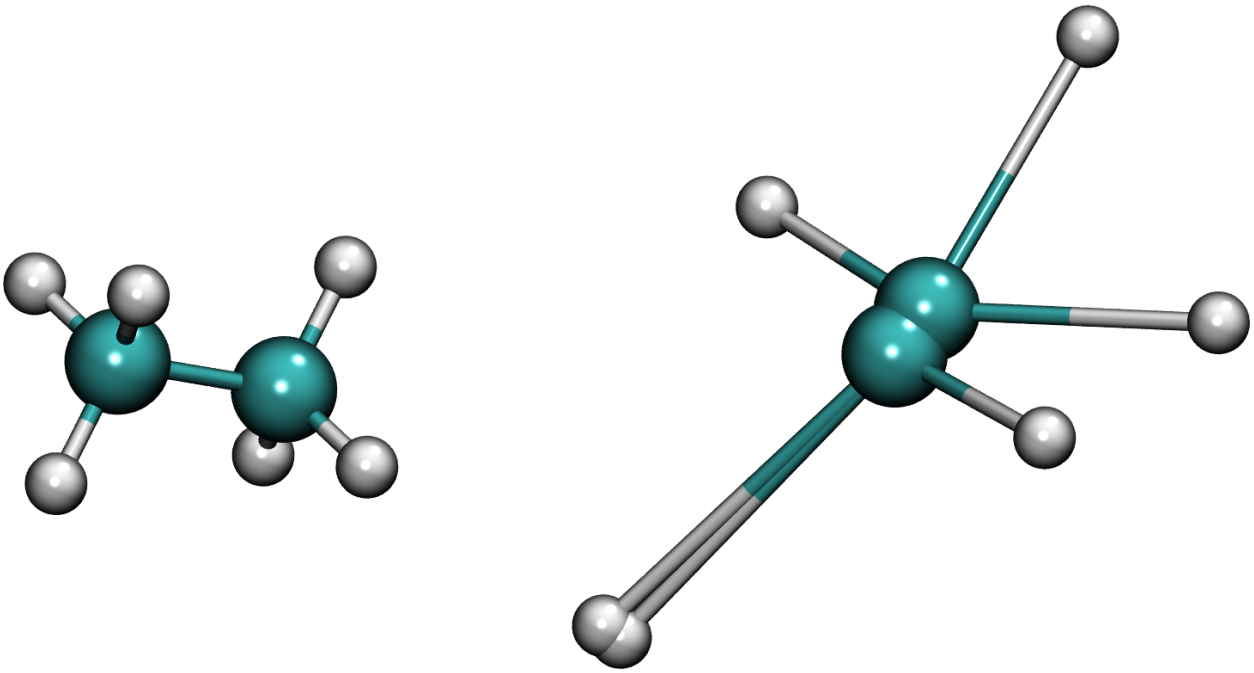

Figure 1-1. Two conformations of an ethane molecule. The conformation on the left is the typical 'staggered' conformation known to be the lowest-energy structure. The structure on the right is an absurd conformation that is never considered experimentally. While the structure on the right contributes negligibly to the partition function, it is an equally likely structure to be proposed by Monte Carlo as the one on the left.

factor. Therefore, lower-energy structures are more likely to be added to the ensemble since the probability of accepting the former is significantly greater.

In practice, an ensemble is built using Metropolis MC by constructing a chain of states beginning with some initial structure. The 'next' structure is generated randomly and accepted with a probability that ensures the constructed ensemble reproduces the correct probability distributions for each state. The process of generating a random conformation and evaluating its acceptance into the ensemble is called a *trial move*.

The resulting chain of states generated by Metropolis MC is called a *Markov chain*, and it has two important qualities. First, trial moves are selected from a finite set of available, predetermined moves that cannot change as the Markov chain grows. Second, a Markov chain is said to be memoryless—that is, the probability of accepting a proposed structure depends *only* on the current state and not on any other state that

has come before. Because thermodynamics deals with chemical equilibria, an ensemble built from a Markov chain of states needs an additional property—reversibility.

A reversible Markov chain needs to satisfy the additional condition of *detailed balance*, a relationship shown in Eq. 1–13.

$$P_i \pi_{i \to j} = P_j \pi_{i \to j} \tag{1–13}$$

where P_i is the probability of being in state i and $\pi_{i \to j}$ is the probability of accepting the proposed change of going to state j from state i (called the *transition probability*). The detailed balance condition in a Markov chain asserts an equilibrium between all states in the chain. Eq. 1–13 is nothing more than a common equilibrium expression encountered in general chemistry where P_i is the 'concentration' of state i in the Markov chain and the transition probability is the 'rate' of changing from state i to state j.

The last remaining detail of Metropolis MC is to define a transition probability equation that satisfies detailed balance. For the canonical ensemble, where the probability of being in state i is proportional to the Boltzmann factor, Eq. 1–14 satisfies detailed balance.

$$\pi_{i \to j} = \min \left\{ 1, \frac{\exp(-\beta E_i)}{\exp(-\beta E_j)} \right\} \tag{1–14}$$

Eq. 1–14 can be inserted into Eq. 1–13 to verify that this choice for the transition probability satisfies detailed balance and therefore results in a reversible Markov chain. Models using the Metropolis MC approach instead of traditional MC are far more efficient—so much so that the term *Monte Carlo* often implies Metropolis Monte Carlo, [11, 12] and that convention will be adopted for the rest of this dissertation.

One concern that Metropolis MC does not address, however, is the propensity for random choices to result in meaningless structures. This is alleviated by starting from a chemically reasonable structure and limiting the magnitude of the structure differences allowed in each trial move—a technique referred to as *importance sampling*. The step size becomes a tunable parameter of the method. If it is too small, then it will take a long

time to fill the ensemble with different structures. However, if it is too large, the likelihood of proposing reasonable structures will drop off and the acceptance rate will suffer.

1.1.2.2 Molecular Dynamics and the Ergodic Hypothesis

An alternative method for constructing a statistical ensemble of states, called *molecular dynamics* (MD), corresponds to generating structures by integrating the equations of motion for molecular systems and building ensembles from the resulting trajectories. The idea that a time-average over a trajectory is equal to an ensemble average is called the *ergodic hypothesis*, and is the cornerstone of MD methods.

The most common equations of motion used in MD simulations are those from classical mechanics. The force on each atomic nucleus is calculated as the gradient of the potential energy function $U(\vec{x})$ at the nuclear centers and then integrated numerically according to Newton's laws. A discussion of computational MD and numerical integration of the classical equations of motion is presented in Appendix A.

Molecular dynamics simulations have several advantages compared to Monte Carlo-based methods. First, MD can be used to calculate temporal properties, such as diffusion. Second, every structure that is generated during a molecular dynamics trajectory is a full member of the resulting ensemble. In contrast, MC-based techniques discard some fraction of the structures they generate. Finally, trajectories generated by MD simulations can inform about the nature of how a molecule moves within a particular environment, which may provide insight into the behavior of molecular systems. For these reasons, MD techniques have become very popular in the field since the first reported use on proteins in 1977. [13] Molecular dynamics does have several weaknesses, however, which must be overcome in order to use MD simulations as predictive instruments in chemistry.

Standard molecular dynamics—simple integration of Newton's laws—samples strictly from the microcanonical ensemble since energy is conserved. While the various thermodynamic ensembles are equivalent in the thermodynamic (macroscopic)

limit, it is often more convenient to work with other ensembles, like the canonical and isobaric-isothermal ensembles. The desire to simulate systems with different thermodynamic constraints led to the development of numerous ways to control temperature and pressure. [12] These techniques are referred to as thermostats and barostats, respectively.

The naïve approach to maintaining a constant temperature is to scale all velocities at each time step such that each point along the trajectory has the same kinetic energy (and therefore temperature). [14] For large systems, however, the resulting perturbation on the system is too large. To address this problem, Berendsen et al. proposed a method in which the factor by which velocities are scaled is reduced so that thermalization occurs on a finite time scale (rather than instantaneously). [15] Similar approaches exist for maintaining constant pressure. [15] Analogous to scaling the velocities to maintain a constant temperature, the system volume is scaled to maintain a constant pressure.

Another major challenge in MD simulations is choosing the integration time step. The time step must be chosen short enough to avoid accumulating integration errors, but long enough that slow structural changes may be sampled in a reasonable amount of simulation time. While the slow motions with small frequencies are often the most interesting since they correspond with global conformational changes in macromolecules, the time step is dictated by the high frequency motions—see Fig. 1-2 for a graphical illustration clarifying this phenomena.

As an example, bonds between hydrogen and 'heavy' atoms (*e.g.,* carbon, oxygen, and nitrogen) often give rise to the highest frequency motions in typical macromolecules. These degrees of freedom cut the maximum time step that can be used for MD simulations in half. As a result, constraints are often applied to these high-frequency degrees of freedom to permanently fix them to their equilibrium bond lengths using any of a number of algorithms. [16–20]

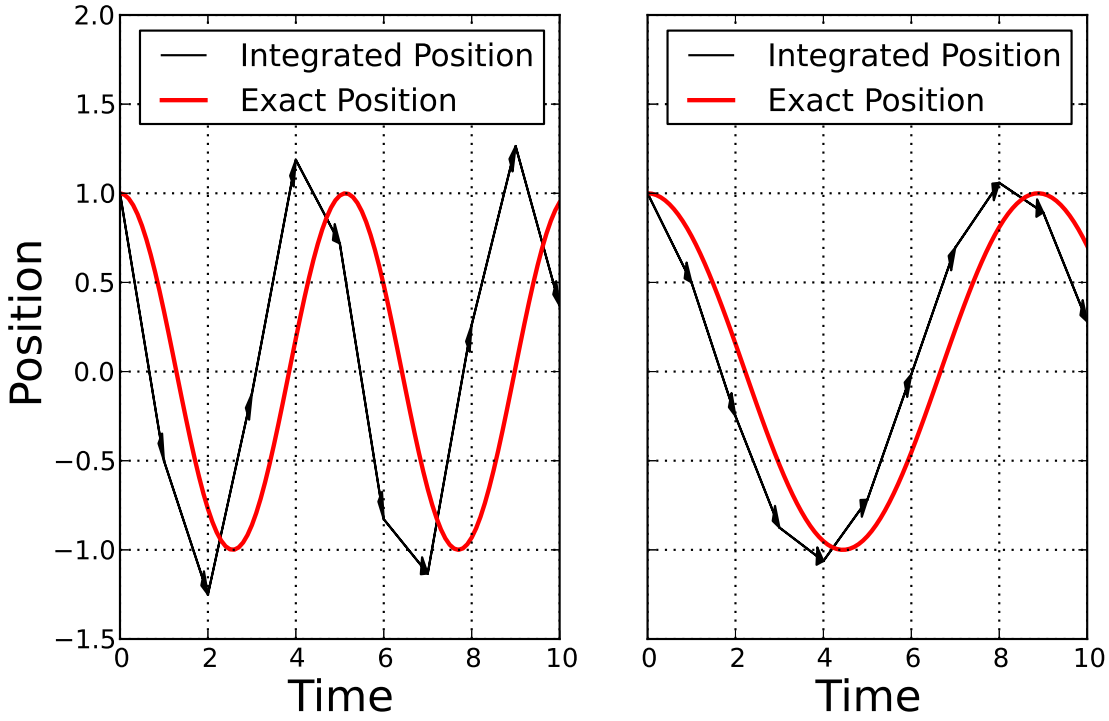

Figure 1-2. The curves represent the trajectory of simple harmonic oscillators with a high frequency (left) and low frequency (right). The black arrows are the trajectory traced out integrating Newton's laws numerically using a time step of 1 time units in the plot. The red line is the analytical trajectory to simple harmonic oscillation.

While atomic forces can be derived from QM calculations on molecular systems and MD can be performed using this potential, the massive computational expense of QM models hinders their utility for large biomolecules. It is necessary, therefore, to develop a model that can accurately describe large molecules while being simple enough to solve with reasonable computational effort. For that, we turn to *molecular mechanics*.

1.2 Molecular Mechanics

We saw from Sec. 1.1.1.2 that computational chemists use quantum mechanics to solve the electronic Schrödinger equation in order to calculate the energy as a function of nuclear coordinates. For small molecules containing 20 – 30 atoms, there are typically a small number of conformations that the molecule can reasonably adopt at typical

temperatures, and partition functions can be reasonably approximated using only a handful of different structures.

For larger systems, however, it becomes increasingly difficult to use QM methods for two reasons. First, the computational demand for obtaining the energy of a single structure rapidly increases. Second, phase space becomes so massive that calculating the potential energy of a small number of snapshots is no longer a reasonable approximation to the partition function. For these reasons, we seek to develop a model with which we can efficiently calculate interatomic potentials in molecular systems without solving the electronic Schrödinger equation. This model will fit a simple functional form to the potential of the molecule, describing the interaction between every atom in the system. These functions typically have analytic derivatives that can be rapidly evaluated to facilitate their use in MD simulations. Because the analytic gradients of these potentials are the forces that act on the atomic centers, these molecular mechanical models are called *force fields*. I will now discuss how these force fields are designed, with special attention paid to the *Amber* family of force fields.

1.2.1 Force Fields

In this section, I will discuss the various parameters found in common force fields, including bonds, angles, torsions, and non-bonded interactions.

1.2.1.1 Bonds

I begin with modeling the chemical bond. Using the Born-Oppenheimer approximation, we can calculate the potential energy surface for a chemical bond by calculating the potential energy at different nuclear separations using an appropriate QM methodology. A common choice of function to reproduce the 'correct' potential energy surface is a Taylor series expansion centered around the equilibrium bond length. This series can be truncated at any order to achieve the desired accuracy and precision. An example for the Hydrogen molecule is shown in Fig. 1-3, where the 'exact' potential energy surface is taken from Ref. 21. When the deviation from the minimum bond length is small, the

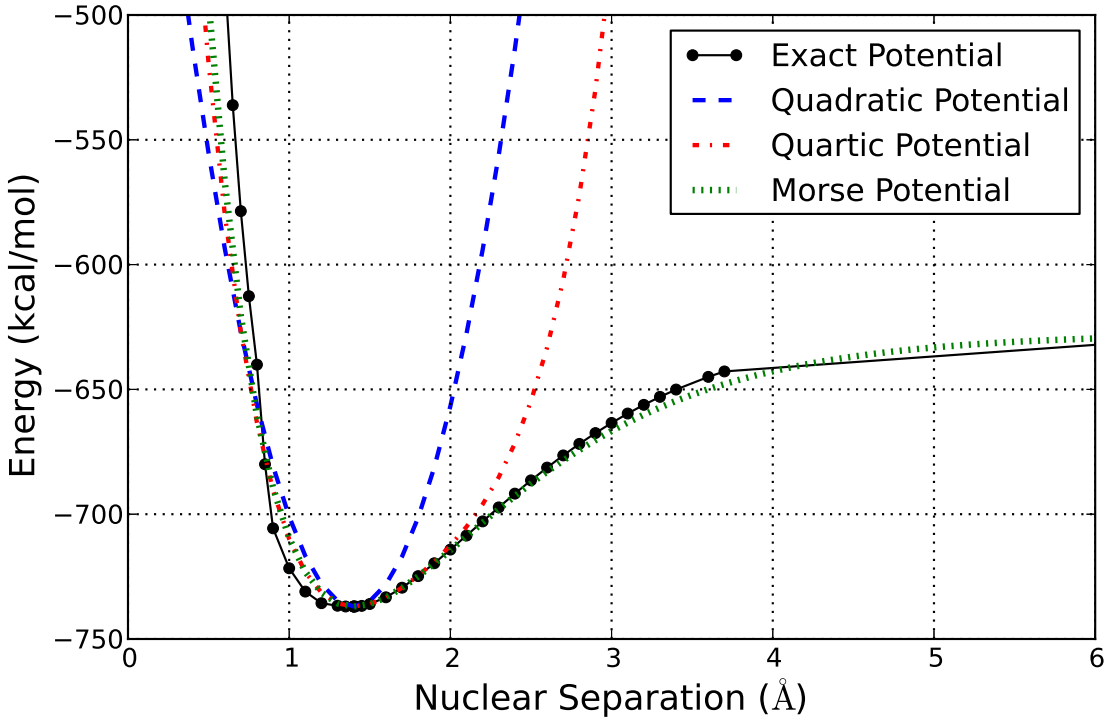

Figure 1-3. The exact potential energy surface for H_2 [21] plotted with the best-fitting quadratic and quartic polynomials and the best-fitting Morse potential (Eq. 1–16).

potential behaves like a simple harmonic oscillator obeying the potential

$$U(\vec{x}) = \frac{1}{2}k(\vec{x} - \vec{x}_{eq})^2 \tag{1–15}$$

where $\vec{x}_{eq}$ is the equilibrium bond length.

Another function commonly used to model chemical bonds, called the Morse potential, is shown in Eq. 1–16. The Morse potential has the benefit that it can model bond dissociation ($D_{\vec{x}}$ in Eq. 1–16)—an effect that cannot be captured with a low-order, truncated Taylor series expansion. It is used less frequently than a second- to fourth-order truncated Taylor series, however, because it is costlier to compute and most simulations employing force fields study conformations in which bonds remain close to their equilibrium values. When bond lengths deviate little from equilibrium, the difference between the Morse potential and a quadratic (or quartic) polynomial is small. [22]

$$U(\vec{x}) = D_{\vec{x}}\left[1 - \exp\left(\alpha_{\vec{x}}(\vec{x} - \vec{x_{eq}})\right)\right]^2 \tag{1–16}$$

Bond parameters can be derived from either high-level quantum calculations—such as those shown in Fig. 1-3—or from experimental measurements. Vibrational force constants (k in Eq. 1–15) and dissociation energies ($D_{\vec{x}}$ in 1–16) can be determined spectroscopically and subsequently used to define the bond parameters.

1.2.1.2 Angles

A *valence angle* is defined as the angle (θ) between atoms separated by two consecutive bonds—see Fig. 1-5B. Like bonds, they behave like simple harmonic oscillators when they are sufficiently close to the equilibrium value. As a result, they are typically treated with the simple quadratic potential function $U(\vec{\theta}) = 1/2k(\vec{\theta} - \vec{\theta_{eq}})^2$.

Angle parameters, too, can be derived from either high-level QM calculations or from spectroscopic measurements. Infrared spectroscopy is particularly well-suited for deriving these parameters, since vibrational frequencies correspond with harmonic force constants.

1.2.1.3 Torsions

A *torsion* is defined between four atoms connected by three sequential bonds—for simplicity I will reference the atom numbers from the labels in Fig. 1-5C. The torsion angle (ϕ), then, is the angle between the bonds 1–2 and 3–4 when projected onto a plane whose normal vector is the 2–3 bond. This projection is easily visualized for the Newman projection of a torsion, shown in Fig. 1-5D. It should be apparent that torsion potentials should repeat with a maximum period of 360° since torsion angles separated by 360° are identical.

The functional form used for torsions is different than that used for bonds and angles. While a periodic function can be represented by a Taylor series polynomial of infinite order, a Fourier series is far more suited to fitting torsion potentials than Taylor

series since the basis functions of a Fourier series are, themselves, periodic. A common functional form for torsion potentials is given in Eq. 1–17. [22]

$$U(\phi) = \sum_{i}^{N} k_i \left[1 + \cos\left(n_i\phi + \psi_i\right)\right] \tag{1–17}$$

where the torsion potential is represented as a sum of N terms with barrier heights k_i, periodicities of n_i, and phase shifts of ψ_i.

Torsion potentials are easily the most important of all bonded parameters in force fields. Bonds and angles are relatively rigid, since they are often modeled by quadratic potentials with modestly large force constants. Even making the force constant for bonds and angles two times larger than they *should* be will result in only a small change in conformational sampling. Torsion potentials, on the other hand, typically have much smaller barriers and give rise to far more significant conformational changes.

Consider the ethane molecule in which torsions are defined between H–C–C–H. At room temperature, neither the individual bonds or angles will deviate much from their equilibrium values, but the torsion angle will readily sample every value due to the low energy barriers between staggered and eclipsed conformations. In order to accurately calculate the partition function, then, a force field must properly reproduce the energy barriers along the torsion coordinate to provide a reasonable estimate of the thermodynamic properties of ethane.

Unlike bond and angle parameters, there are no spectroscopic techniques that can be used to extract torsion parameters. Furthermore, force field parameters are not orthogonal with one another—for example, different choices for non-bonded potential terms (described in Sections 1.2.1.4 and 1.2.1.5) will impact torsion profiles. Therefore, torsion terms are typically the last values fitted when designing a force field, and are used as correctional terms to 'fix' the deficiency of the other force field parameters in describing conformational equilibria. Force fields are often systematically improved just by changing some torsion terms. [23–25]

1.2.1.4 Electrostatic Interactions

The three potentials that I just discussed are called *bonded interactions* since they occur between atoms connected by bonds. Potentials between *all* atoms are called *non-bonded interactions*. The first of the non-bonded interactions I will discuss arise due to charge-charge interactions, typically referred to as *electrostatic* interactions.

Atoms treated in a force field are assigned partial charges that roughly correspond to atom electronegativities, although each force field has a precise recipe for deriving partial atomic charges. A common strategy to assign partial charges is to fit to an electrostatic potential (ESP) calculated using a QM method. It is common practice to apply constraints to the fit to ensure that rotationally degenerate atoms (*e.g.,* the three hydrogen atoms in a freely rotating methyl group) have the same charge—a technique referred to as restrained electrostatic potential (RESP). [26–28]

There are two principle charge-charge interaction models utilized in modern force fields: so-called polarizable and fixed-charge force fields. The polarizable force fields allow the partial atomic charge of each atom to change in response to its surroundings, providing additional flexibility to force field parametrization. Due to the added computational expense of computing polarizable potentials and the difficulty this imposes on deriving other aspects of the force field, fixed-charge force fields (*i.e.,* force fields where partial atomic charges never change) are more commonly used. All future discussion in this dissertation of electrostatic interactions in the MM framework will focus on fixed-charge, monopole-monopole interactions.

The electrostatic potential is calculated according to

$$U(r_{i,j}) = k\frac{q_i q_j}{r_{i,j}} \tag{1–18}$$

In Eq. 1–18, k is the electrostatic constant, q_i is the partial charge on atom i, and $r_{i,j}$ is the distance between atoms i and j. One thing to note about Eq. 1–18 is the long-ranged nature of the interaction. While the electrostatic energy of two charged particles

falls to 0 as the distance between them becomes infinite, $1/i$ decays so slowly that $\sum_{i=1}^{\infty} 1/i = \infty$. Therefore, electrostatic interactions typically have to be evaluated over a very long distance (or calculated completely).

1.2.1.5 van der Waals Interactions

In addition to electrostatic interactions, force fields also employ another non-bonded potential that accounts for *van der Waals* interactions. The van der Waals potential is composed of two parts—a strongly repulsive term that models steric clashes and an attractive term accounting for dispersion interactions. The attactive term of the van der Waals potential is derived mostly from the London dispersion forces shown for an ideal gas dimer in Eq. 1–19. [3]

$$U(r_{i,j}) = -\frac{3}{2}\frac{\alpha I}{r^6} \tag{1–19}$$

where I is the first ionization energy and α is the polarizability. This attractive interaction arises even in noble gases due to instantaneous atomic polarization caused by correlated movements of the electrons.

The most common functional form used to model van der Waals interactions is called the *Lennard-Jones* (LJ) potential, shown in Eq. 1–20.

$$\begin{aligned} U_{LJ}(r_{i,j}) &= 4\varepsilon_{i,j}\left[\left(\frac{\sigma_{i,j}}{r_{i,j}}\right)^{12} - \left(\frac{\sigma_{i,j}}{r_{i,j}}\right)^{6}\right] \\ &= 4\varepsilon_{i,j}\left[\frac{1}{4}\left(\frac{R_{min,i,j}}{r_{i,j}}\right)^{12} - \frac{1}{2}\left(\frac{R_{min,i,j}}{r_{i,j}}\right)^{6}\right] \\ &= \frac{a_{i,j}}{r_{i,j}^{12}} - \frac{b_{i,j}}{r_{i,j}^{6}} \end{aligned} \tag{1–20}$$

where $r_{i,j}$ is the distance between atoms i and j, and the remaining terms are labeled in a schematic diagram showing the nature of the LJ potential in Fig. 1-4. The three forms

of Eq. 1–20 are equivalent if

$$\begin{aligned} R_{min,i,j} &= 2^{1/6}\sigma_{i,j} \\ a_{i,j} &= \varepsilon_{i,j}R_{min,i,j}^{12} \\ b_{i,j} &= 2\varepsilon_{i,j}R_{min,i,j}^{6} \end{aligned}$$

Due to its computational efficiency, the third form of Eq. 1–20 is typically used in molecular simulations. To use Eq. 1–20 in these simulations, the $a_{i,j}$ and $b_{i,j}$ values must be computed for every pair of atoms in the system. For transferable force fields (*i.e.,* force fields whose parameters can be used for many different, but related, systems), each type of atom defined in the force field is typically assigned an individual ε and σ parameter which must be combined with every other atom type to yield $a_{i,j}$ and $b_{i,j}$. The way in which these individual atomic parameters are mixed is referred to as the *combining rules*.

1.2.1.6 Other Force Field Terms

The parameters presented in the previous sections make up the bulk of all parameters found in typical force fields. In this section I will describe some of the less-commonly used types of parameters.

Improper Torsions. Improper torsions are typically used as correction terms to control out-of-plane motion. There are numerous instances where four or more atoms should be predominantly coplanar—such as aromatic five- and six-membered rings. The existing parameters I have already mentioned do not necessarily ensure that the proper planarity of these systems will be maintained. As a result, *improper torsion* terms are added to the force field in key locations to suppress unwanted out-of-plane motion. A diagrammatic depiction of an improper torsion is shown in Fig. 1-5E.

Correction Map. Torsion potentials are so important to ensuring that MM simulations generate a sensible conformational ensemble that some force fields parametrize coupled torsion parameters to improve the accuracy. The most common implementation

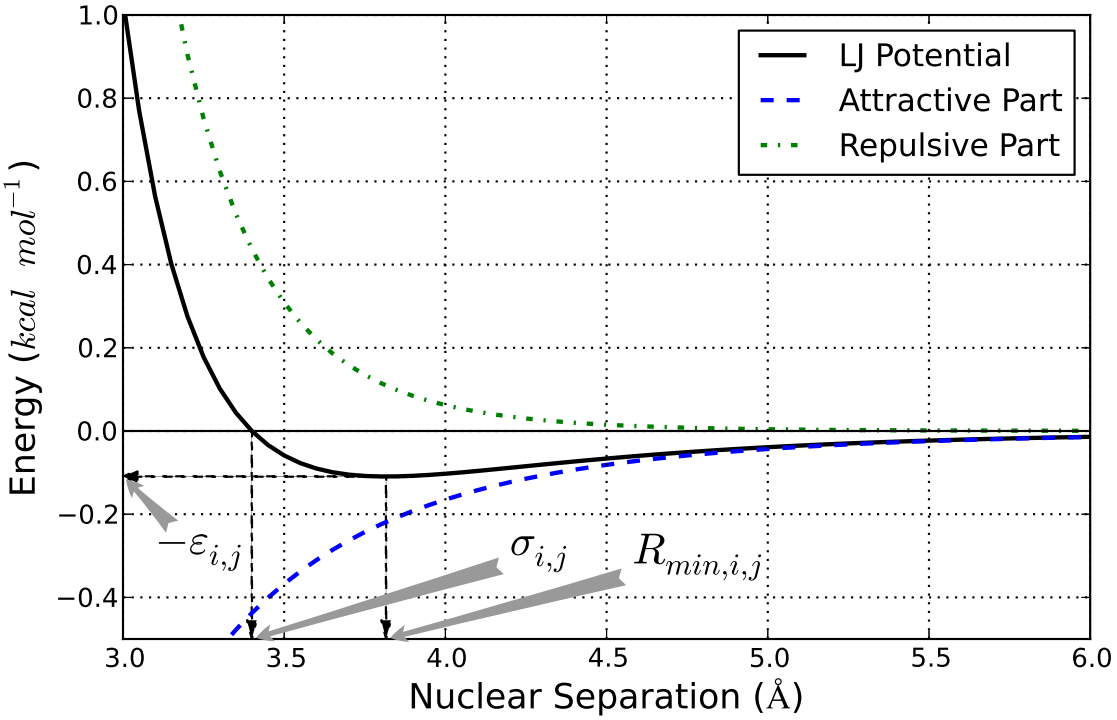

Figure 1-4. The Lennard Jones potential between two atoms with a $R_{min,i,j}$ of 3.816 Å and ε of 0.1094 kcal mol^{-1}. The various parameters are indicated on the graph, and the full LJ potential is shown alongside its repulsive and attractive terms.

of these coupled-torsion corrections is done in the form of a *correction map*, or CMAP term. [29] The CMAP is generated by mapping the potential energy surface of two torsions in a small sample system without the CMAP correction and subtracting that from the 'true' potential energy surface calculated with some high-level QM method.

The CMAP is then laid out on a grid, using some type of interpolating spline (*e.g.,* bicubic splines) to calculate potential energies and forces during MD simulations. A schematic of the coupled-torsions commonly parametrized via CMAPs is shown in Fig. 1-5F.

Urey-Bradley. Another parameter commonly used in CHARMM force fields [30] is called the *Urey-Bradley* potential. The functional form of the Urey-Bradley term is identical to the bond term in Sec. 1.2.1.1 (Eq. 1–15), but exists between atoms

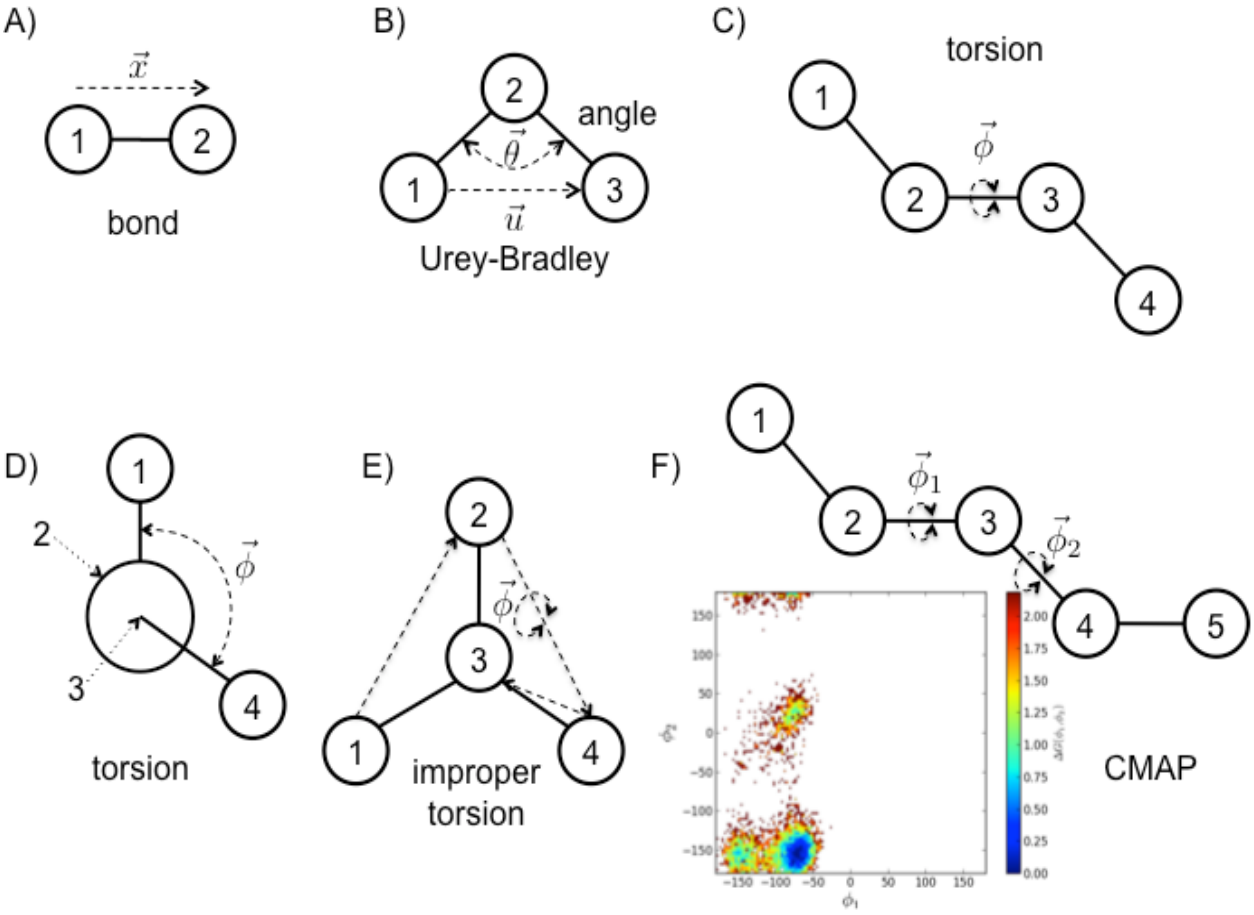

Figure 1-5. Schematics shown for various parameters present in typical force fields. A) is a bond parameter, B) shows the valence angle parameter and the Urey-Bradley parameter where $\vec{u}$ is the shown distance, C) depicts a torsion, D) depicts the same torsion using a Newman projection, E) depicts an improper torsion, and F) depicts two coupled torsions alongside a typical free energy map of two torsions that CMAP parameters attempt to fit to.

separated by two bonds (*i.e.,* forming a valence angle). The Urey-Bradley term is shown in Fig. 1-5B alongside the valence angle.

1.2.2 The Amber Force Field

The Amber force field is a popular family of force fields designed to treat large biomolecules such as proteins, DNA, and RNA. This section will focus on the functional form and implementation of the Amber force fields [23, 31, 32] in the supporting Amber programs. [33]

1.2.2.1 Functional Form

The functional form of the Amber force fields is typically presented as in Eq. 1–21, [31] although this is an incomplete specification. A more rigorous definition is presented in Eq. 1–22, taking into account the proper exclusion of non-bonded terms between bonded atoms.

$$U(q) = \sum_{\text{bonds}} K_r(r - r_{eq})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{eq})^2 + \sum_{\text{torsions}} \frac{V_n}{2}\left[1 + \cos(n\phi - \gamma)\right] + \frac{1}{2}\sum_i \sum_j \left[\frac{A_{i,j}}{R_{i,j}^{12}} - \frac{B_{i,j}}{R_{i,j}^{6}} + k_{elec}\frac{q_i q_j}{\epsilon R_{i,j}}\right] \quad (1\text{–}21)$$

$$U(q) = \sum_{\text{bonds}} K_r(r - r_{eq})^2 + \sum_{\text{angles}} + K_\theta(\theta - \theta_{eq})^2 + \sum_{\text{torsions}} \frac{V_n}{2}\left[1 + \cos(n\phi - \gamma)\right] + \frac{1}{2}\sum_i \sum_{j \in l_{1-4,i}} \left[\frac{A_{i,j}}{2.0R_{i,j}^{12}} - \frac{B_{i,j}}{2.0R_{i,j}^{6}} + \frac{q_i q_j}{1.2\epsilon R_{i,j}}\right] + \frac{1}{2}\sum_i \sum_{j \notin l_{excl,i}} \left[\frac{A_{i,j}}{R_{i,j}^{12}} - \frac{B_{i,j}}{R_{i,j}^{6}} + k_{elec}\frac{q_i q_j}{\epsilon R_{i,j}}\right] \quad (1\text{–}22)$$

Amber employs a simple harmonic potential to model angles and bonds to completely describe the interactions between atoms separated by one and two bonds—*i.e.,* no electrostatic or Lennard-Jones potentials are calculated between pairs of atoms connected by a bond or angle. Torsions are treated with a truncated Fourier series expansion, typically using integral values for the periodicity (n in Eqs. 1–21 and 1–22). Therefore, the sum over torsions in Eqs. 1–21 and 1–22 is a sum over all individual torsion terms for each distinct torsion. Improper torsions are modeled the same way as 'proper' torsions with only a single term designed to maintain their planar geometry.

The non-bonded interactions are composed of a Lennard-Jones term (the third form of Eq. 1–20) and electrostatic term calculated between all atom pairs that are not excluded from the computation. The non-bonded exclusion list—$l_{excl,i}$ for atom i in Eq.

1–22—is composed all atoms separated by one, two, and three bonds (*i.e.,* that form bonds, angles, or torsions with atom i). Finally, the LJ interactions between all atoms separated by three bonds are scaled by $1/2$ and the electrostatic interactions between those atom pairs are scaled by $1/1.2$. In Eq. 1–22, $I_{1-4,i}$ represents the list of atoms related to atom i like atoms 1 and 4 in Fig. 1-5C.

1.2.2.2 Implementation

This section will describe how the Amber family of force fields is implemented in the Amber program suite. It is important to note that the Amber force field is not unique—there are many variants, each with a different name. [23, 31, 32, 34–36] All information necessary to fully describe a molecular system with the Amber force field is contained in two files—the parameter-topology file (*prmtop*) and the coordinate file. The prmtop file—fully described in Appendix B—contains all of the information regarding the bonded network and the necessary parameters for evaluating Eq. 1–22. The coordinate file contains the Cartesian coordinates and velocities for each atom in the system described by the prmtop file.

The prmtop file is generated by the *tleap* program by matching the parameters from a database to the assigned 'atom types' of the input structure. Atom types are descriptors of individual atoms that specifies the properties and typical chemical structure of bonds involving that atom. Each atom type, i, has a predetermined set of atom parameters—an atomic mass, a LJ radius r_i, and a LJ well-depth ε_i. The pairwise $R_{min,i,j}$ in Eq. 1–20 of atom types i and j is the sum $r_i + r_j$. The combined well depth $\varepsilon_{i,j}$ is the geometric mean of the individual well depths ($\sqrt{\varepsilon_i \varepsilon_j}$). These are the so-called *combining rules* employed by *tleap* when parametrizing a molecule with an Amber force field.

The Amber parameter databases store a list of all recognized atom types as well as the bond, angle, and torsion parameters between the various bonded arrangements of the available atom types. For instance, each pair of atom types that could form a bond

(*e.g.,* two aromatic carbons or an aromatic carbon and an aromatic hydrogen) has an equilibrium bond length and bond force constant associated with it. Also stored in these parameter databases are the equilibrium angle displacements with corresponding force constants and torsion parameters (periodicities, barrier heights, and phase shifts for each term of every torsion).

CHAPTER 2
BIOMOLECULAR SIMULATION: SAMPLING AND FREE ENERGY

All of the simulation models discussed in Chapter 1 use an electrostatic equation dealing with charges interacting in a vacuum. However, biological chemistry occurs almost exclusively in an aqueous environment, necessitating the development of models capable of simulating these systems in solution. In this chapter, I will describe the various methods by which solvent effects are introduced into simulation, followed by the ensemble sampling techniques that will be used for the principle studies in this dissertation.

2.1 Simulations in the Condensed Phase

The techniques by which solvation effects can be incorporated into various computational models can be separated into two groups. The most obvious way is to include the solvent atoms and molecules directly into the simulation alongside the system of interest—referred to collectively as *explicit solvent* methods. While explicit solvent models are the most accurate approach in principle, they drastically increase the size of the system—simultaneously increasing the cost of the simulation and amount of sampling required to obtain converged results.

An alternative way to include solvent effects is by modifying the electrostatic interactions in a system to account for the natural screening that a particular solvent provides. These approaches are called *implicit solvent* methods because solvent effects are included in an average way without including the actual solvent atoms or molecules in the simulation. Simulations employing implicit solvent models result in smaller systems in which comformational sampling converges more rapidly because the solvent degrees of freedom are already included in a mean field way. However, individual solvent molecules often play a critical role in the structure and function of biological molecules and behave very differently from molecules in bulk solvent—an effect implicit solvent models are ill-equipped to handle.

The following sections describe the various implicit and explicit solvent models commonly used in biomolecular simulations.

2.1.1 Implicit Solvent

One of the most important qualities of a solvent—especially an aqueous solvent—is its ability to polarize in response to an electric field, thereby reducing the magnitude of electrostatic interactions across a given distance. While the naïve approach of simply applying the solvent dielectric everywhere is attractive in its simplicity, solvent-excluded regions should obviously not be subject to the screening effects of the solvent. For large biomolecules, the solvent-excluded regions can be quite large, so it becomes important to deal with these regions effectively.

2.1.1.1 Distance-dependent Dielectric

Among the earliest approaches to account for the different dielectric environments of the interior of a bimolecule and bulk solvent introduced a dielectric constant that changed as a function of the distance between two charged particles. As the separation between two particles increased, so too did the likelihood that they were separated by solvent, and were therefore subject to dielectric screening effects.

This approach is attractive in its simplicity—it adds little to the computational cost of the model while retaining the simple, pairwise-decomposable nature of the electrostatic potential term. A common equation modeling the dielectric constant is given below in Eq. 2–1. [12]

$$\varepsilon_{eff}(r) = \frac{\varepsilon_{bulk} - 1}{2}\left[(rS)^2 + 2rS + 2\right]\exp(-rS) \tag{2–1}$$

where r is the distance between the two particles, ε_{bulk} is the dielectric constant of the bulk, ε_{eff} is the effective dielectric constant at a given particle separation, and S is a free parameter. Fig. 2-1 plots the resulting curve for ε_{eff} from Eq. 2–1 for different values of the free parameter.

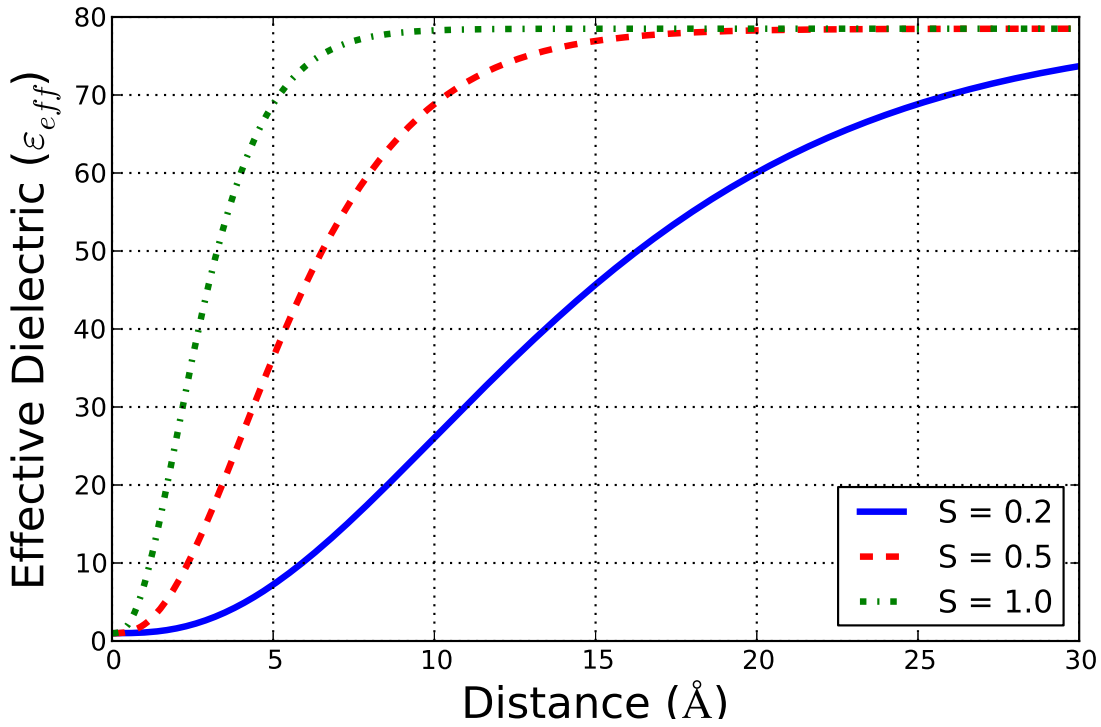

Figure 2-1. Distance-dependent dielectric for different values of the free parameter S in Eq. 2–1.

This effective dielectric constant is then incorporated as ϵ in Eq. 1–22, and influences the calculated forces due to its dependence on $r_{i,j}$. One of the biggest weaknesses of distance-dependent dielectrics is that it treats every atom in the biomolecule as though they are in the same environment, whereas the shapes of biomolecules—and their solvent-excluded volumes—are often highly irregular. That is, two atoms buried inside the solvent-excluded volume separated by d Å are treated exactly the same way as two different atoms d Å apart whose interstitial region is solvent-accessible. Furthermore, because the shapes of biomolecules can vary greatly from system to system, the 'optimal' value for S in Eq. 2–1 is highly system-dependent. Finally, while the true dielectric regions are either the value of the bulk solvent or the molecule interior, a distance-dependent dielectric has a large region corresponding to unphysical, intermediate values of the dielectric.

For these reasons, the distance-dependent dielectric model is rarely used in modern simulations, having given way to the more accurate methods like the Poisson-Boltzmann and Generalized Born equations.

2.1.1.2 Poisson-Boltzmann

At the heart of most modern implicit solvent models lies the Poisson equation

$$\bigtriangledown\epsilon(\vec{r}) \cdot \bigtriangledown\phi(\vec{r}) = -4\pi\rho(\vec{r})$$

where ϕ is the electrostatic potential distribution function, ρ is the charge distribution function, and ϵ is the dielectric constant at a given point in space. The dielectric constant is often divided into two regions—a region of low dielectric in the solvent-excluded volume and that of the bulk solvent 'outside' the system of interest. [22]

The Poisson equation is only valid, however, at zero ionic strength. When mobile ions are present—as is the case *in vivo* with all biomolecules—the Poisson equation must be augmented with an appropriate distribution of counterions. The probability of finding an ion in a particular region of space is related to its Boltzmann factor $\exp(-\beta q\phi(\vec{r}))$, where $q\phi(\vec{r})$ is the energy of a point charge in a given electrostatic potential. Because ions come in pairs with both positive and negative charges, the Boltzmann probability of finding both types of ions must be included. The equation for calculating the electrostatic potential in a biomolecular system with a given solution ionic strength, termed the *Poisson-Boltzmann* (PB) equation, is shown in Eq. 2–2. [22]

$$\begin{aligned} \bigtriangledown\epsilon(\vec{r}) \cdot \bigtriangledown\phi(\vec{r}) - \epsilon(\vec{r})\lambda(\vec{r})\kappa^2\frac{k_B T}{2q}\exp\left(-\beta q\phi(\vec{r})\right) + \epsilon(\vec{r})\lambda(\vec{r})\kappa^2\frac{k_B T}{2q}\exp\left(\beta q\phi(\vec{r})\right) &= -4\pi\rho(\vec{r}) \\ \bigtriangledown\epsilon(\vec{r}) \cdot \bigtriangledown\phi(\vec{r}) - \epsilon(\vec{r})\lambda(\vec{r})\kappa^2\frac{k_B T}{q}\sinh\left(\frac{q\phi(\vec{r})}{k_B T}\right) &= -4\pi\rho(\vec{r}) \end{aligned} \tag{2–2}$$

In Eq. 2–2, q is the charge of the ions (both positive and negative ions are present), $\lambda(r)$ is a simple switching function that is 0 in solvent-excluded regions and 1 in solvent-accessible regions, and κ^2 is related to the ionic strength as

$$\kappa^2 = \frac{8\pi q^2 I}{\epsilon k_B T}$$

Eq. 2–2 is a non-linear, second-order differential equation in the electrostatic potential that must be solved iteratively until the desired level of self-consistency in the electrostatic potential is achieved. The sinh term in Eq. 2–2 may be expanded using its Taylor series expansion. If the ionic strength is low and the solute is not highly charged (so $\phi(\vec{r})$ is relatively small), the Taylor series expansion for sinh can be truncated after the first term to yield the much simpler Eq. 2–3 with little loss of accuracy. Eq. 2–3 is called the *linearized* Poisson-Boltzmann equation because the Taylor series expansion for sinh is truncated after its linear term.

$$\bigtriangledown \epsilon(\vec{r}) \bigtriangledown \phi(\vec{r}) - \epsilon(\vec{r})\lambda(\vec{r})\kappa^2\phi(\vec{r}) = -4\pi\rho(\vec{r}) \tag{2–3}$$

The Poisson Equation can be solved exactly for only the simplest systems, like solvating a point-charge or a conducting sphere with a uniform charge distribution on its surface. Eq. 2–2 or 2–3 must be solved numerically for complex biomolecules with irregular shapes. A common approach is to set up a three-dimensional grid surrounding the solute and calculate the charge distribution on the grid from the partial charges of each solute atom. The dielectric boundary can be calculated from the solvent accessible surface [37], so each grid point has an associated charge and dielectric value. The differential equations can then be solved via finite differences within the defined grid. [38]

After the electrostatic potential is calculated via Eq. 2–2, the free energy is calculated by integrating the product of the charge distribution and the calculated electrostatic

potential according to

$$G = \frac{1}{2}\int \rho(\vec{r})\phi(\vec{r})\,d\vec{r}$$

where the $1/2$ factor corrects for double-counting the interactions. The free energy of solvation due to solvent polarization is calculated from the difference in the electrostatic potentials in vacuum and solvent ($\phi_{solv} - \phi_{vac}$)—a quantity referred to as the *reaction field*. [12] The charge-dependent portion of the solvation free energy then becomes

$$\Delta G_{pol} = \frac{1}{2}\int \rho(\vec{r})\left(\phi_{solv}(\vec{r}) - \phi_{vac}(\vec{r})\right) d\vec{r} \tag{2–4}$$

Models employing implicit solvent via the PB equation have proven effective in many cases. [38–41] However, due to requirements of a fairly dense grid and the iterative, self-consistent nature of solving the PB equation, the computational cost of this model is too high for many applications. Furthermore, the dielectric function is discontinuous at the boundaries of the solvent-excluded and solvent-accessible regions, making stable gradients (and therefore forces) difficult to calculate. [42] Therefore, I will now consider a common approximation to the PB equation called the *Generalized Born* model that seeks to provide an efficient, analytical alternative to solving the PB equation.

2.1.1.3 Generalized Born

While the electrostatic potential generated by most charged species cannot be solved analytically using the Poisson equation, I will consider two simple, ideal systems that can. The first is a perfect conducting sphere of radius r with a uniform charge distribution. Given a total charge q and using the Poisson equation to calculate the electrostatic potential induced by the charged sphere, the polar contribution to the free energy of solvation can be calculated from Eq. 2–4, giving the familiar *Born* equation, shown below. [22]

$$\Delta G_{pol} = -\frac{1}{2}\left(\frac{1}{\epsilon_{vac}} - \frac{1}{\epsilon_{bulk}}\right)\frac{q^2}{r} \tag{2–5}$$

In Eq. 2–5, ϵ_{vac} is the dielectric constant of a vacuum, which is unity. It is shown explicitly here to demonstrate that the dielectric constant of the solvent-excluded volume in the Poisson equation does not appear in the Born equation.

If instead of being a perfect conducting sphere with a uniform charge distribution, the sphere had a perfect dipolar charge distribution, the free energy of solvation using the Poisson equation would result in the *Kirkwood-Onsager* equation, shown below. [22]

$$\Delta G_{pol} = -\frac{1}{2}\left(\frac{2(\epsilon-1)}{2\epsilon+1}\right)\frac{\mu^2}{r^3} \tag{2–6}$$

where the ϵ is the dielectric constant of the bulk solvent and the dielectric constant of vacuum has simply been replaced by 1.

The *Generalized Born* (GB) formalism for calculating the polar contribution to the solvation free energy is, as its name would suggest, an extension of the *Born* solution shown in Eq. 2–5 to complex molecules with an arbitrary size and shape. [43–47] Still et al. were the first to propose the method, adjusting the Born equation (Eq. 2–5) as shown below. [43]

$$\Delta G_{pol} = -\frac{1}{2}\left(1-\frac{1}{\epsilon}\right)\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{q_i q_j}{f_{GB}} \tag{2–7}$$

where q_i is the charge of atom i, ϵ is the dielectric constant of the solvent, and f_{GB} is an arbitrary, analytic function of atom positions designed to calibrate Eq. 2–7 to experiment. The most common form of f_{GB} devised by Still et al., and still used predominantly today, is shown in Eq. 2–8.

$$f_{GB} = \sqrt{r_{i,j}^2 + \alpha_i\alpha_j \exp\left(-\frac{r_{i,j}^2}{4\alpha_i\alpha_j}\right)} \tag{2–8}$$

where $r_{i,j}$ is the distance between atoms i and j and α_i is called the effective Born radius of atom i for reasons that will soon be apparent.

Eq. 2–8 does not represent a theoretically 'correct' choice for f_{GB}, nor has it been shown to be the best choice—in fact it probably is not. [48] However, it is a good choice

for several reasons. First, Eq. 2–8 is a simple formula with analytical gradients—assuming α_i is an analytic function of the nuclear positions—and can be computed rapidly. More importantly, however, Eq. 2–8 has the appropriate limiting behavior. [43] For a single particle—or two identical point charges separated by a distance of 0—f_{GB} reduces to α and Eq. 2–7 reduces to the Born equation (Eq. 2–5) in which the radius of the 'sphere' is α_i. It is for this reason that the α_i values can be thought of as an 'effective' radius. Furthermore, for two point charges separated by a small distance (*i.e.,* smaller than the effective radii of the two particles) the result agrees with the Kirkwood-Onsager solution (Eq. 2–6) to within 10% of the true value. [43]

The next major challenge in solving Eq. 2–7 is calculating the effective Born radii, α_i, for each atom. The effective radius of an atom reflects the spherically average distance of that atom from the solvent excluded surface. Calculating the effective radii is particularly challenging because it must be done rapidly, accurately, and so gradients may be easily computed. Because GB was developed as an efficient alternative to solving the PB equation, computationally intensive approaches to calculating the effective radii offer little advantage over using the more precise PB equation. Furthermore, Onufriev et al. has demonstrated the importance of computing effective Born radii accurately, [47] showing that so-called 'perfect radii' reproduce PB results very closely. Finally, gradients are necessary to perform either geometry optimization or molecular dynamics, and an expression that lends itself to rapid computation of an accurate gradient is an attractive feature.

The most common approach to computing the effective radius is called the *coulomb field approximation*, shown below in Eq. 2–9. [22]

$$I_i = \int_{\Omega_i} \frac{d^3 r}{4\pi r^4} \tag{2–9}$$

I_i in Eq. 2–9 is the coulomb field integral for atom i and Ω_i signifies the integral is over all space centered on atom i. The effective radius is then computed from this integral using Eq. 2–10.

$$\alpha_i = \left(\rho_i^{-1} - I_i\right)^{-1} \tag{2–10}$$

where ρ_i is the intrinsic van der Waals radius of atom i and I_i is the integral from Eq. 2–9.

As computational power increased and simulations reached longer time scales and larger systems, however, deficiencies in these equations began to surface, leading to efforts to improve the calculation of the effective Born radii. [47, 49–51] The two approaches that have been implemented in the Amber suite of programs are briefly described below.

Onufriev et al. noticed that Eqs. 2–9 and 2–10 tended to underestimate the effective radii of buried atoms because it assumed that interstitial regions of space between atoms were solvent-filled, despite the fact that they were too small to contain a full water molecule. [49] As a result, they modified Eq. 2–10 into the following form:

$$\alpha_i = \left[\rho_i^{-1} - \rho_i^{-1} \tanh\left(a\Psi - \beta\Psi^2 + \gamma\Psi^3\right)\right]^{-1} \tag{2–11}$$

where $\Psi = I\rho_i$ (I_i is taken from Eq. 2–9), and a, β, and γ are fitting parameters. The $\tanh$ function was chosen because it is infinitely differentiable (analytically) and increases the effective radii of more deeply-buried atoms while leaving the effective radii of atoms closer to the surface unchanged. In this way, Eq. 2–11 maintains the success Eq. 2–10 displayed for small compounds while improving the behavior of deeply-buried residues. [49] This GB variant is referred to as $\mathrm{GB}^{\mathrm{OBC}}$ (where OBC comes from the authors Onufriev, Bashford, and Case).

Mongan et al. took a different approach. While Eq. 2–11 provided uniform scaling for all atoms with a given degree of burial (as measured by the value of I_i in Eq. 2–9),

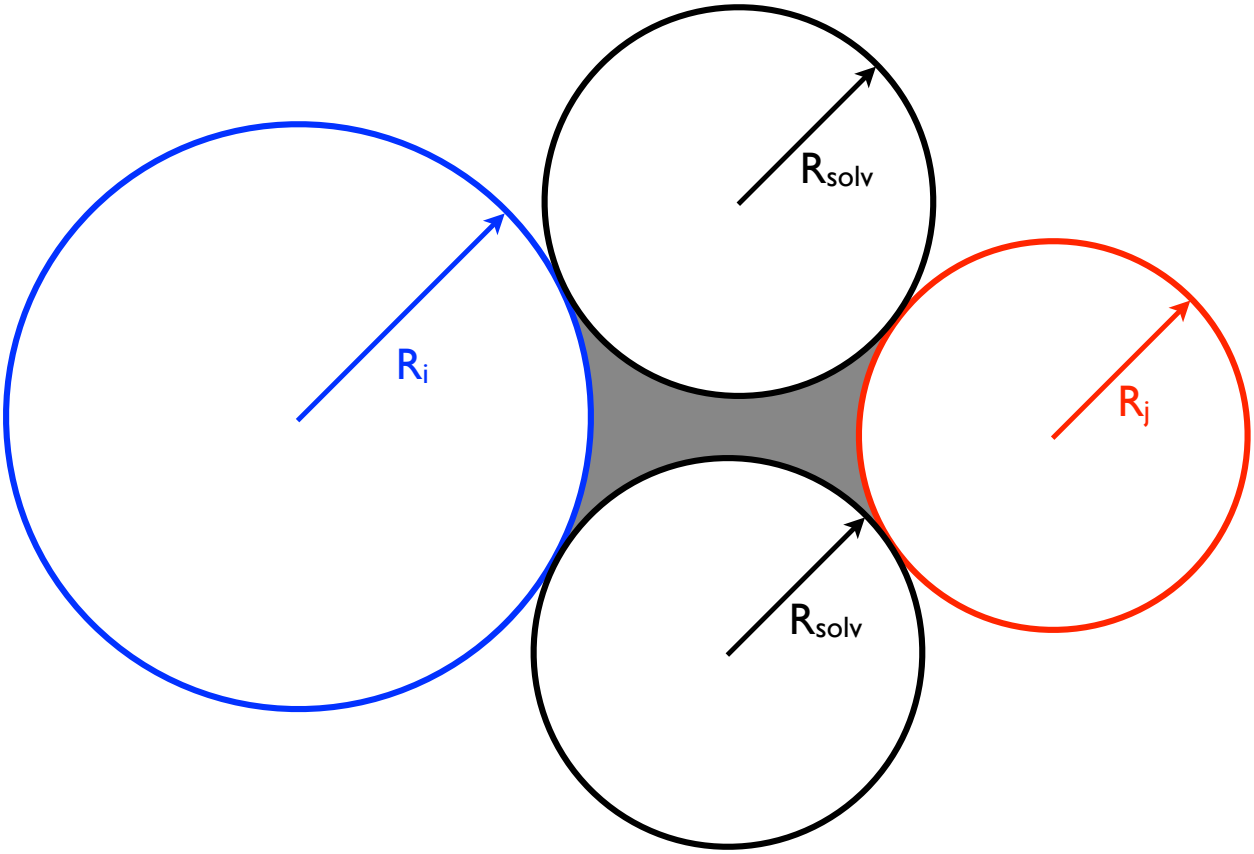

Figure 2-2. Region of space between two atoms i and j of radius R_i and R_j that is inaccessible to a spherical solvent molecule of radius R_{solv}. This inaccessible region is called the *neck* and is shaded gray.

Mongan et al. adopted an approach based on geometry. By treating each atom and each solvent molecule as a sphere—a good approximation for a water molecule—the interstitial space between two solute atoms that is inaccessible to solvent can be quantified. Because this interstitial region resembles a neck—as seen in Fig. 2-2—this model is referred to as $\mathrm{GB}^{\mathrm{neck}}$. [50]

The most recent approach by Nguyen et al. involves a re-parametrization of the intrinsic atomic radii (ρ_i in Eq. 2–10) for atoms commonly involved in salt bridges and a combination of the ideas presented in the $\mathrm{GB}^{\mathrm{OBC}}$ and $\mathrm{GB}^{\mathrm{neck}}$ models described above. [51]

2.1.1.4 Non-polar Solvation

The process of solvation can be broken down into two fictitious steps—a cavitation step in which the solvent is excluded from a region of space equal to the solute's solvent excluded volume, and a charging step where the solvent-polarized charge distribution of the solute is inserted into that cavity. Because the free energy is a state function, this gedanken decomposition will yield an identical free energy to the true experimental free energy—assuming of course that each step can be calculated exactly. The free energies of these two steps are referred to as the non-polar and polar solvation free energy, respectively.

The Poisson-Boltzmann and Generalized Born equations shown in Eqs. 2–2 and 2–7 are used to compute the polar solvation free energy (*i.e.,* the portion of the free energy derived from the reaction field). There are several methods for calculating the non-polar solvation free energy.

Methods for calculating the non-polar contribution to solvation are often parametrized by assuming that the solvation free energy for extended and branched alkanes is non-polar in nature. The most common way to calculate non-polar solvation is to fit a surface tension value to the experimental solvation free energies of the alkanes. [22] This approach can be rationalized using the idea that the presence of a non-polar solute immersed in solvent disrupts the solvent-solvent interactions, thereby restricting solvent structure in the solvation shell surrounding the solute. This effect imposes an entropic penalty to solvation (that is offset by the polar solvation term for soluble compounds). If this was the only source of the 'non-polar' solvation free energy, then its magnitude would vary with the size of the molecule, which is directly related to its surface area.

Combining this surface-area non-polar solvation term with either the Poisson-Boltzmann or Generalized Born equations for the polar solvation term results in the so-called *PBSA* and *GBSA* methods, respectively. One of the most common methods for calculating the surface area in GBSA molecular dynamics simulations is called the *linear*

combination of pairwise overlaps (LCPO) method—so-called because it is parametrized by fitting five parameters to the spherical overlaps of individual atoms. [52] The chief advantage of LCPO is that it provides an efficient way to calculate surface areas using an analytical formula whose derivatives can be easily calculated for use in molecular dynamics.

2.1.2 Explicit Solvent

While the implicit solvent methods described above are useful ways of incorporating solvation effects in molecular simulations, all solvent effects are accounted for in an average way. Therefore, individual water molecules that play structurally important roles in biomolecules are not treated well by either PB or GB methodologies. In such cases, it is advantageous to include the solvent molecules explicitly in the simulation. Explicit solvent significantly increases the cost of the simulation by adding a large number of atoms to the system, but should improve the accuracy by creating a model closer to reality.

A large drawback when adding explicit solvent, however, is the fact that modern simulations at an atomic resolution (*i.e.,* where all atoms are treated explicitly) are limited to, at most, 10^8 atoms, [53] although simulations between 10^5 to 10^6 atoms are more reasonable. Macroscopic systems, on the other hand, contain on the order of 10^{23} atoms—an intractable number for modern hardware, so our simulations must be scaled down to a microscopic size.

As a demonstration, the hairpin ribozyme is a biomolecule that contains roughly 2100 atoms. Adding only 22,000 water molecules—enough to create a 20 Å spherical solvent buffer around the ribozyme—increases the simulation size to roughly 90,000 atoms. It is quite clear, therefore, that even on the largest supercomputers, we can only model a microscopic droplet in explicit solvent. At such small sizes, the ratio of surface area to volume for these minuscule droplets is astronomically large, and water

molecules at the solvent-air interface behave quite differently from those molecules in bulk solvent.

While early approaches of applying a 'cap' potential—an artificial biasing potential penalizing solvent that diffuses too far away from the solute—helped overcome some surface effects like evaporation, it made direct comparison to experiment dubious. A major breakthrough in explicit solvent calculations came with the introduction of periodic boundary conditions. [54]

2.1.2.1 Periodic Boundary Conditions

To emulate bulk solution behavior using a system composed of a tractable number of atoms, we impose periodic boundary conditions (PBC) on the system, replicating it infinitely in every dimension. In such a system, each atom interacts with all other atoms in all other simulation cells—including its own periodic images. [54] A two-dimensional illustration of PBC is shown in Fig. 2-3 for a rectangular unit cell illustrating these ideas.

A practice commonly adopted in PBC simulations in which each atom interacts directly with only a single image of every other atom—specifically the nearest image—is called the *minimum image convention*. The minimum image convention is employed to simplify the problem, but also imposes a limit to the range of calculated interactions. Specifically, the non-bonded interactions do not extend beyond half the length of the shortest side of the unit cell. Employing the minimum image convention, the energy calculated for a system with PBC is the energy of a single unit cell in the field generated by every periodic cell. The challenge is how to calculate the non-bonded interactions for every atom in the system. The common approaches employed in biomolecular simulation are explored in the next three sections.

2.1.2.2 Cutoff Methods

The simplest approach to calculating non-bonded interactions is to employ a simple cutoff that is smaller than half the length of the shortest side of the unit cell—*i.e.,* all non-bonded interactions between atoms closer than the cutoff are included and all

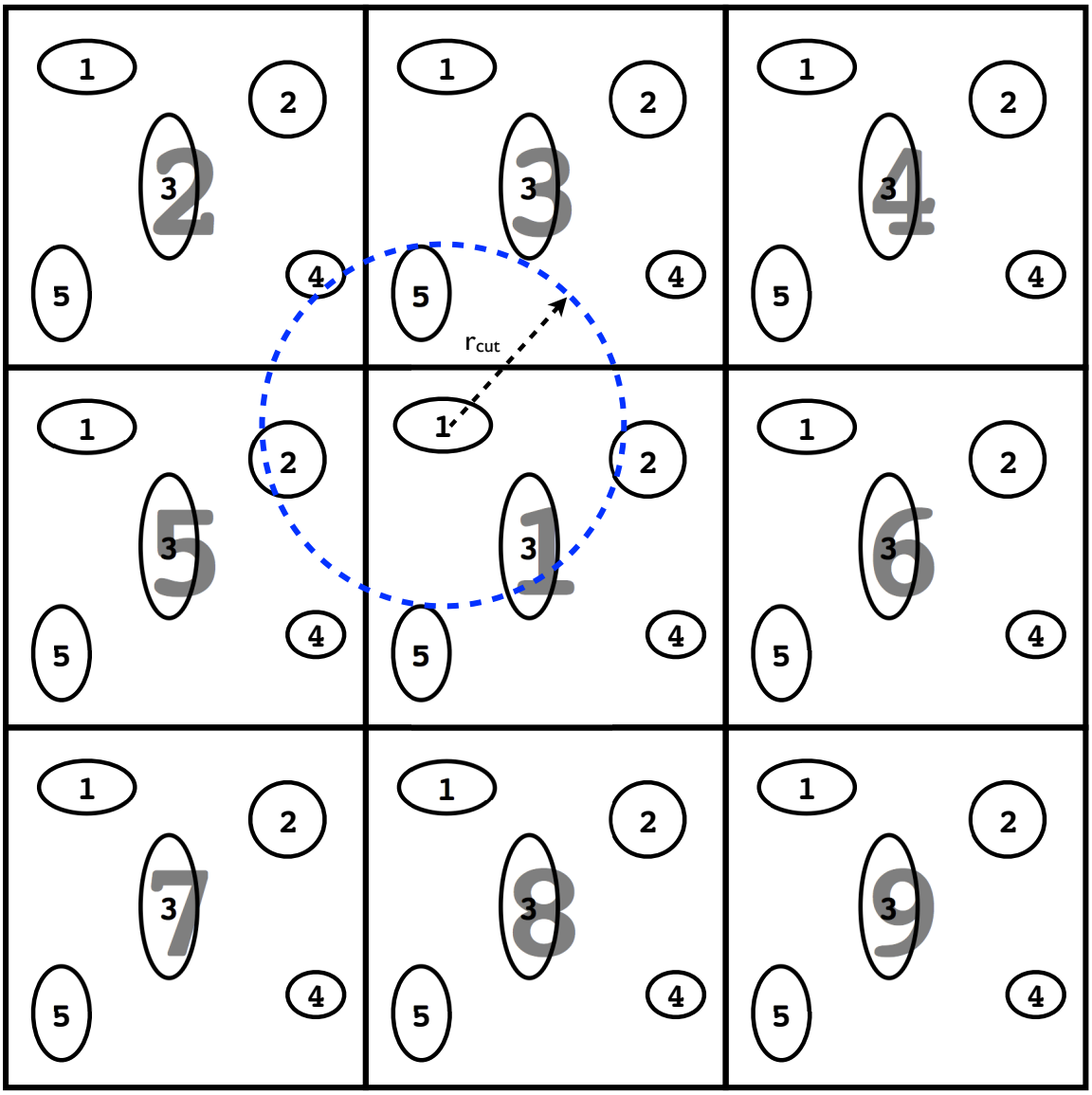

Figure 2-3. Periodic simulation in two dimensions with a rectangular unit cell. The maximum permissible cutoff (r_{cut}) for the minimum image convention is shown with a blue dotted circle centered on particle 1 in the first box.

those between atoms greater than the cutoff are neglected. Because the common forms of the non-bonded potential decay as the distance between atoms increases (see Eqs. 1–18 and 1–20), interactions between distant atoms are significantly smaller than interactions between nearby atoms. The non-bonded interactions are modeled as the simple piecewise function shown in Eq. 2–12.

$$U'(\vec{x}_{i,j}) = \begin{cases} U(\vec{x}_{i,j}) & : |\vec{x}_{i,j}| < x_{cut} \\ 0 & : |\vec{x}_{i,j}| > x_{cut} \end{cases} \tag{2–12}$$

While conceptually simple and computationally efficient, simple cutoffs suffer from a severe limitation. The potential, and therefore the force, encounters a discontinuity at the cutoff distance, shown in Fig. 2-4A. This discontinuity results in simulations that do not conserve energy and leads to numerous, non-physical artifacts. [55–61] This effect is particularly pronounced for electrostatic interactions—a very long-range potential of the form $1/r$. As mentioned before, this function decays so slowly that $\sum_{i=1}^{\infty} 1/i = \infty$. Two monovalent ions must be separated by $\sim$332 Å before their interaction energy drops to 1 kcal/mol. Such a cutoff would require a unit cell size at least 664 Å on each edge containing $\sim 10^7$ water molecules.

Given the need to improve the behavior of the non-bonded potentials near the cutoff distance, two popular modifications to the simple cutoff approach were introduced—a smooth switching function and a shifting function. The switching function approach applies a smooth function at a given distance that satisfies the following criteria: a) the potential and its gradient is continuous everywhere, b) the short-ranged form of the potential is unchanged, and c) the potential approaches 0 at the cutoff. Eq. 2–12 is an example of a very simple switching function in which the original potential is multiplied by 1 when the interparticle distance is less than the cutoff and 0 otherwise. Of course, this switching function does not obey either the a) or c) conditions listed above. An example of a smooth switching function is shown in Fig. 2-4B. [62]

The second family of methods commonly employed are so-called shifting functions since the potential is modified by 'shifting' the potential up such that the value of the potential becomes zero at the cutoff distance. [54, 62] Simply shifting the potential, though, is not enough for MD simulations, since the force will remain unchanged and still faces a discontinuity at the cutoff. Therefore, the shifting function often contains a

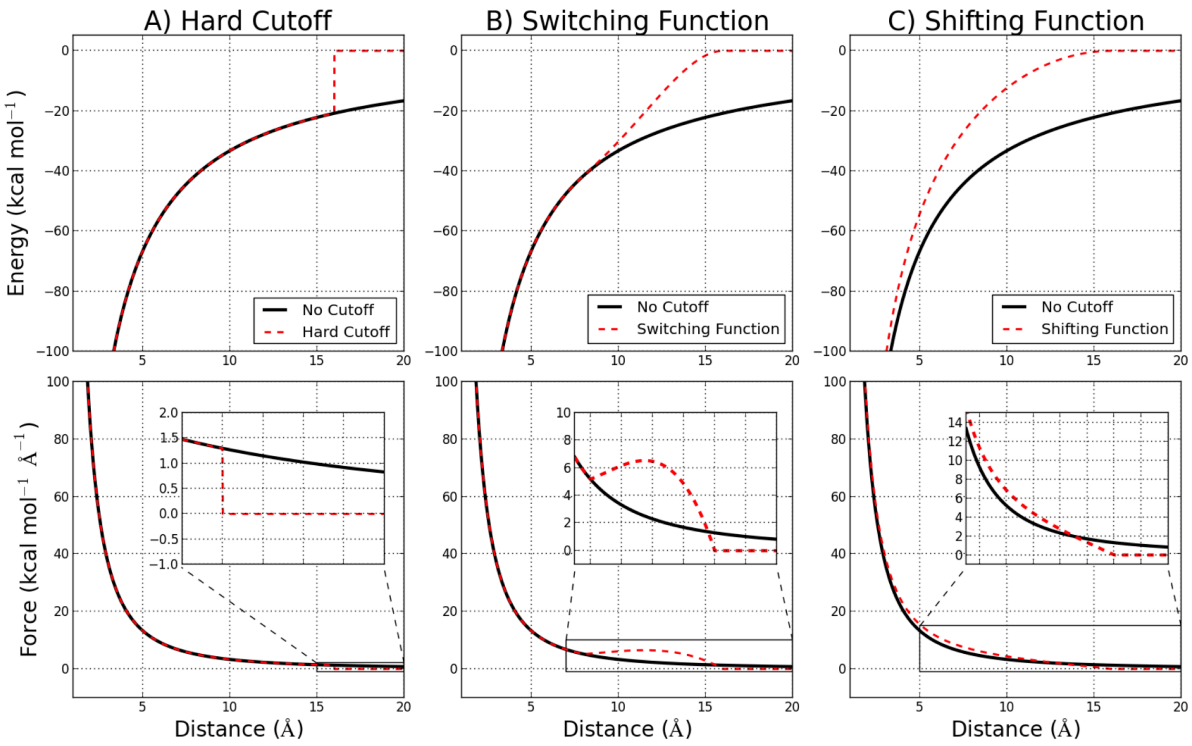

Figure 2-4. Effects of various 16 Å cutoff schemes on the electrostatic interaction of two monovalent ions with opposite charges. A) shows the effect of imposing a hard cutoff. B) shows a typical switching function starting at 8 Å. C) shows a typical shifting function for the electrostatic potential. The energies as a function of distance are shown in the top 3 plots and the forces as a function of distance are the bottom 3 plots.

force-shifting component, as shown in Eq. 2–13. [54] The effect of the shifting function is shown in Fig. 2-4C.

$$U_s(\vec{r}_{i,j}) = \begin{cases} U(\vec{r}_{i,j} - U(\vec{r}_{cut}) - \left(\frac{dU(\vec{r}_{i,j})}{d\vec{r}_{i,j}}\right)_{\vec{r}_{i,j}=\vec{r}_{cut}} (\vec{r}_{i,j} - \vec{r}_{cut}) & \vec{r}_{i,j} < \vec{r}_{cut} \\ 0 & \vec{r}_{i,j} \geq \vec{r}_{cut} \end{cases} \tag{2–13}$$

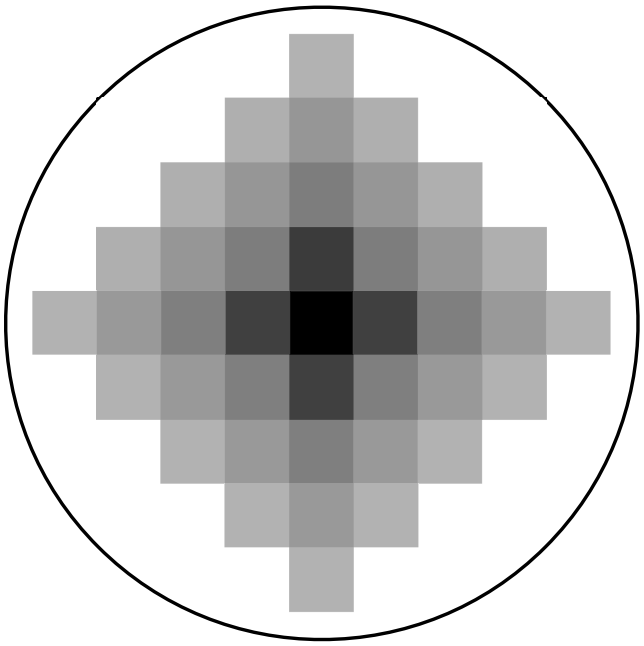

Figure 2-5. Periodic cells added in a spherical shape radially from the central unit cell. The progression from darker to lighter cells shows the order in which interactions are accumulated in the sum of the electrostatic interactions (with the darker cells being added before the lighter ones). The example, adapted from Allen and Tildesley, is shown in two dimensions, but can be trivially extended to three dimensions. [54]

2.1.2.3 Ewald Summation

Ideally, simulations in the condensed phases would be performed *without* truncating electrostatic interactions at all. The full electrostatic interaction for a net neutral unit cell takes the functional form $\sum_{i=1}^{N}(-1)^i/i$, since there are an equal number of charges of both signs. This sum is *conditionally convergent*, meaning that, while it converges to a finite value, that value depends on the order in which the terms are summed. [54] A natural choice for ordering the summation of the infinite number of electrostatic interactions with particle i is by summing all of the electrostatic interactions with each particle j in every unit cell extending radially from the unit cell containing i. This approach is shown diagrammatically in Fig. 2-5. [54]

In 1921, Ewald devised a method whereby the electrostatic interactions between an ion and all of its periodic images in a crystal lattice could be computed according to the technique presented in Fig. 2-5. [63] The same approach can be used for simulations in the condensed phase when PBC are used. The technique, called the Ewald sum, utilizes a trick to cause the electrostatic interactions between particles to decay arbitrarily rapidly, allowing the interactions to be truncated at a distance where the interactions themselves are negligible. To do this, a Gaussian charge distribution is centered at each point charge with the opposite sign of the point charge, as shown in Fig. 2-6. Given a width of the Gaussian distribution α, the functional form of the neutralizing charge distribution is shown in Eq. 2–14.

$$\rho_i(r) = \frac{q_i \alpha^3}{\pi^{3/2}} \exp\left(-\alpha^2 r^2\right) \tag{2–14}$$

where ρ_i is the charge distribution due to particle i and its neutralizing Gaussian and α is the tunable parameter controlling how diffuse the Gaussian is.

The electrostatic interaction of two charged particles i and j with their neutralizing charge distribution is

$$E_{i,j} = q_i q_j \frac{erfc(\alpha r_{i,j})}{r_{i,j}}$$

where *erfc* is the complementary error function. The complementary error function decays rapidly—more rapidly for narrower neutralizing distributions. The narrower the neutralizing distributions are, the smaller the cutoff that may be used without compromising accuracy. In fact, at the limit where the Gaussian width is zero, the neutralizing charge distribution becomes a delta function that exactly cancels the original point charge, allowing a cutoff of zero!

However, while adding the neutralizing charge distributions has allowed us to compute the direct electrostatic energies between particles rapidly by imposing a relatively short cutoff, we have changed our system. The effect of the neutralizing charge distributions must be canceled by inverting all of the neutralizing charge distributions and

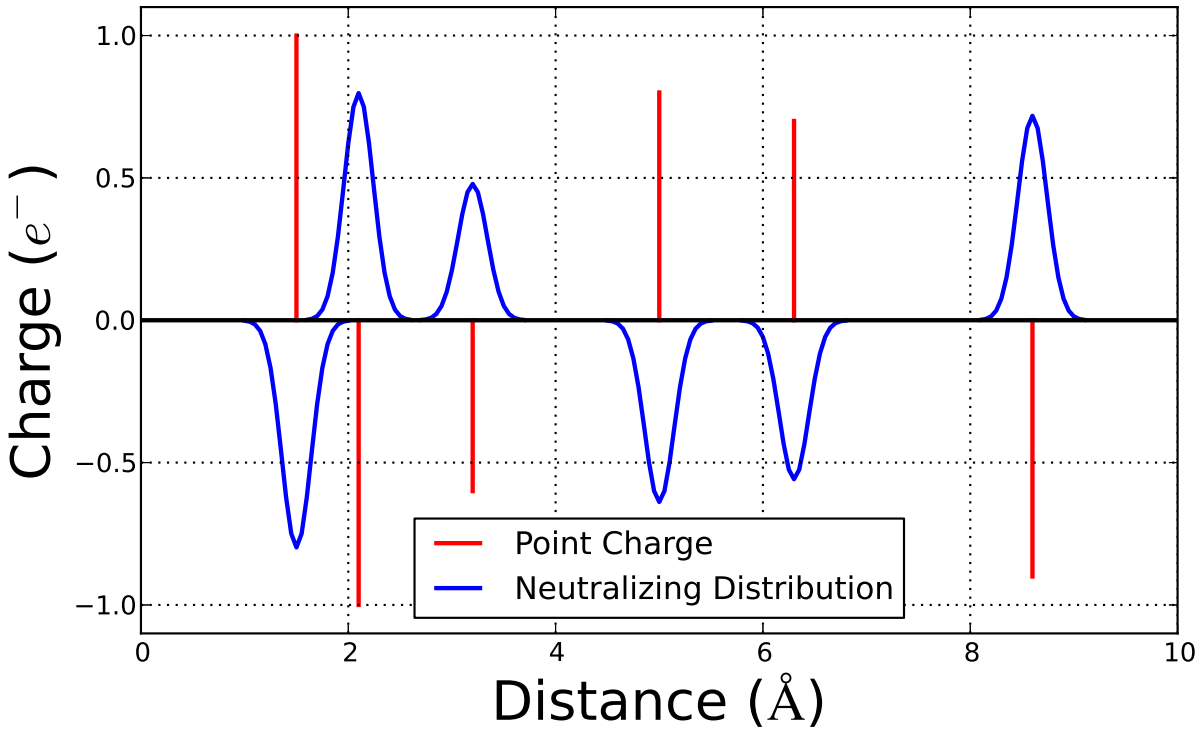

Figure 2-6. A one-dimensional example of particles with a given charge (red) with a neutralizing Gaussian charge distribution (blue) shown.

adding their interaction back to the original sum. By adding these so-called canceling charge distributions back to the electrostatic sum, the original interaction of just the point charges is recovered. The interactions between these neutralizing charge distributions represent a number of convolution integrals which may be computed very rapidly by taking the Fourier transform of the distributions and summing the contributions in reciprocal space. The result is then reverse-Fourier transformed to obtain the electrostatic potential at each of the particles. [54]

Particle-Mesh Ewald. A weakness of Ewald's summation is that the Fourier transform is a slow operation—on the order of $O(N^2)$ where N is the number of particles. To address this shortcoming, the charge density due to the canceling charge distributions can be discretized on a 3-dimensional mesh with a given grid spacing. This allows us to use the fast Fourier transform algorithm (FFT) to perform both the Fourier transform

and reverse Fourier transform to calculate the electrostatic potential at each of the mesh points. Unlike the standard Fourier transform, the FFT scales as $O(N\log(N))$, resulting in a substantial increase in computational efficiency. The potential at each of the particles—and its gradient—can then be interpolated from the adjacent grid points on the mesh using cardinal B-splines. [64] This approach is termed *Particle-Mesh Ewald* (PME) due to the way in which the particles interact with the mesh to determine the long-range electrostatic interactions.

2.1.2.4 Other Approaches

Ewald-based methods employing the discrete fast Fourier transform have been very popular over the past two decades. As the rapid increase in computational power allowed simulations to run increasingly longer, the deficiency of typical cutoff methods for simulating highly charged systems—such as DNA or RNA—became readily apparent. [65, 66] Properly accounting for long-range electrostatic effects using PME resulted in stable simulations of not only proteins, but also highly charged systems like DNA and RNA. [59] Furthermore, by employing the FFT, PME allowed calculations to be done more rapidly by reducing the computational cost of the non-bonded interactions.

However, there are two principle drawbacks of Ewald-based methods. First, the use of periodic boundaries may introduce artifacts into the system caused by the correlated motions of each periodic image. [67] For instance, if periodic boundary conditions was imposed on a gas of monovalent ions such that each cell had a single particle, the particle distribution would necessarily be uniform since periodic symmetry reduces dimensionality of the system to a single degree of freedom. While this effect does not seem to induce measurable artifacts for most simulations, [67] a more serious limitation of Ewald-based methods has to do with the changing architecture of modern computers.

For many years, the efficiency of the central processing unit (CPU), typically measured in the speed with which it executes each operation (*i.e.,* clock speed), improved as engineers were able to shrink the size of the transistors and place increasingly more

of these transistors onto each CPU die. Recently, however, the power requirements to increase the clock speed caused chips to melt since the the heat generated could not be dissipated quickly enough. This drove chip manufacturers to increase the computational power of these CPUs by adding additional cores. To take advantage of this form of improved CPU efficiency, computational algorithms must be designed to run in parallel. It turns out that due to the non-local nature of the FFT and the algorithmic details of its efficient implementation, calculations employing such methods are limited in their ability to take advantage of the increasing parallelism of modern processors.

To alleviate the limited scalability of standard PME, Cerutti and Case devised an approach, termed *Multi-level Ewald*, to divide the system into smaller charge grids so that the reciprocal-space sum can be performed in parallel in multiple, independent 'chunks.' [68] These independent grids can then be 'stitched' together using a much coarser global grid that can be computed far more rapidly.

To combat both shortcomings mentioned for Ewald-based methods, many researchers have investigated alternatives to the PME treatment of long-range electrostatic interactions in bimolecular simulations. One such method, the *isotropic periodic sum* (IPS), assumes an isotropic distribution of particles by replicating the surrounding region around each particle within a cutoff infinitely in all directions. [69] While this method necessitates using a larger cutoff to more fully characterize each particle's surroundings, it avoids needing a charge grid populated from every atom in the system, thereby reducing the communication overhead. As a result, IPS can be implemented in such a way that is more scalable on modern hardware than PME.

The generalized reaction field (GRF) method employs yet another approach to treating long-range electrostatics based on the PB equation. A sphere is constructed around each particle whose radius is equal to the non-bonded cutoff distance, inside which all interactions are computed directly. The surroundings are modeled as a bulk dielectric environment, and the reaction field potential is calculated on the sphere

analytically according to the linearized PB equation. The force exerted by this electric field can be calculated on the atom at the center of the constructed sphere. [70] This approach has the same cost as typical cutoff methods, but models interactions outside the cutoff as though it were bulk solvent. Such treatment necessitates the use of a larger cutoff value than that required by Ewald methods.

While the list of methods here is not comprehensive, the general aim of PME-replacements is to either lessen the likelihood of observing periodicity artifacts in simulations and/or to present an algorithm that is more amenable to parallelization. Despite the challenges in computational scaling and efficiency associated with parallelizing the reciprocal-space Ewald sum, Ewald-based methods are still widely used today, even on highly tuned, specialized hardware designed specifically to accelerate MD simulations. [71]

2.2 Sampling

Sampling is the principle problem in most condensed phase simulations—especially involving biomolecules. For such large systems, the size of phase space—a $6N$-dimensional hyperspace composed of positions and momenta for N particles in all 3 spatial dimensions—is unconscionably large. Although any chemical system can be characterized completely if the density of states is known at an arbitrary energy ($\Omega(E)$), this number is so vast that it cannot be directly computed.

Luckily, the partition functions of most thermodynamic ensembles—in particular that for the canonical ensemble (Eq. 1–5)—are almost entirely comprised of low-energy structures due to the exponential weighting in the Boltzmann factor. Despite this fortuitous simplification, no simulation is capable of truly exhaustive sampling for typical biomolecular simulations, and it is unlikely that exhaustive sampling will ever be attainable.

The most naïve approach to sampling—running pure molecular dynamics or Monte Carlo simulations—is frequently insufficient to characterize rare events that happen on

the millisecond or even second timescale. Even with highly specialized (and expensive) hardware, pure MD simulations are currently confined to the millisecond timescale. [72] In this section, I will discuss three approaches to enhance sampling compared to traditional MD simulations—umbrella sampling, steered molecular dynamics (SMD), and expanded ensemble techniques (and the special case of replica exchange).

2.2.1 Umbrella Sampling

Umbrella sampling is a biased sampling technique that acts on a specific reaction coordinate. In complex systems, there are often free energy barriers separating different states that are far larger than the average available thermal energy, $k_B T$. An example is shown as a black line in Fig. 2-7 in which the $6N$-dimensional free energy surface (reduced to $3N$-dimensional when the momentum integral is separated from the canonical partition function) is projected onto a 1-dimensional reaction coordinate. This reduced-dimension free energy surface, called a *potential of mean force* (PMF) shows a free energy barrier of roughly $6\, k_B T$ in Fig. 2-7. In standard dynamics simulations, it would take a very long unbiased simulation to cross that barrier.

The trick involved in umbrella sampling is to modify the underlying potential with a harmonic biasing potential to encourage the simulation to sample higher energy structures more often. Fig. 2-7 shows how a quadratic umbrella potential changes the shape of the underlying PMF such that higher-energy structures are sampled more frequently. Clearly, the two biasing potentials shown in Fig. 2-7 tend to favor sampling near the two transition states separating different minima, since that portion of the reaction coordinate is lowest in energy.

The resulting ensemble of the modified potential, shown in Eq. 2–15, contains more snapshots around the areas that are traditionally sampled poorly by MD simulations of finite duration. However, all properties calculated based on these statistics refer to a fictitious system, and will not translate into experimental observables. In other words, the statistics collected from an umbrella sampling simulation correspond to the

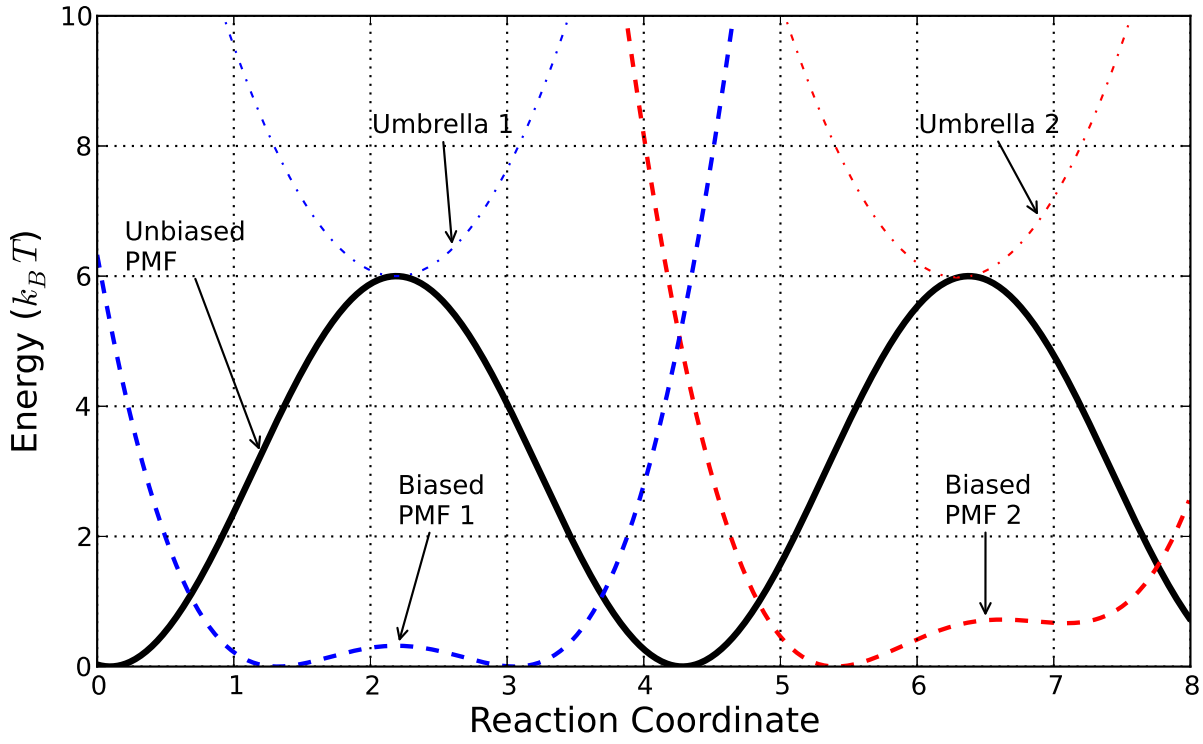

Figure 2-7. An example 1-dimensional PMF (shown in black). Two biasing umbrella potentials are shown alongside the resulting, biased PMF. All PMF curves have been translated so that the 'minimum' free energy is 0. Because only energy differences are significant, vertical translations of the PMF have no effect on calculated properties.

H_{bias} Hamiltonian in Eq. 2–15, whereas the physical system actually obeys the H_{orig} Hamiltonian. [12]

$$H_{bias}(\vec{x}) = H_{orig}(\vec{x}) + \frac{1}{2} k_{umb}(f(\vec{x}) - s)^2 \tag{2–15}$$

H_{orig} is the original, unbiased Hamiltonian in Eq. 2–15, k_{umb} is the force constant on the harmonic umbrella potential, $f(\vec{x})$ is the reaction coordinate, and s is the center of the umbrella potential along that reaction coordinate.

Since the exact shape of the biasing potential is known, and the sampling provides information about the shape of the total biased potential, we can use that information to deduce the underlying shape of the original Hamiltonian along the chunk of the PMF

that our simulation has effectively characterized through sampling. However, because the umbrella potential is monotonically increasing on either side of the umbrella center, configurations far away from that center will be sampled very poorly, leading to poor convergence in those regions. To alleviate this issue, a series of umbrella sampling simulations are performed in intervals along the reaction coordinate—called windows—which are used to construct 'pieces' of the PMF near the center of the respective umbrellas. These pieces are then stitched together to approximate the total, unbiased PMF.

The free energy of the biased potential along the PMF is related to the probability density function at that point according to

$$\pi_{bias}(\vec{x}_0) = \frac{\int \exp\left(-\beta H_{bias}(\vec{x})\right) \delta(\vec{x} - \vec{x}_0) d\vec{x}}{\exp\left(-\beta A_{bias}(\vec{x}_0)\right)}$$

where π is the probability distribution function, δ is the Dirac delta function that serves to extract only those ensemble members that correspond to the specific point $\vec{x}_0$ on the PMF, and A is the free energy along the PMF at that value. The unbiased probability distribution, which is directly related to the unbiased free energy up to an arbitrary constant, can be estimated according to

$$\pi_{unbias}(\vec{x}) = \exp\left(-\beta(A_{bias} - A_{unbias})\right) \exp\left[\beta\left(\frac{1}{2}k_{umb}(f(\vec{x}) - s)^2\right)\right] \pi_{bias}(\vec{x})$$

where A is the Helmholtz free energy along the PMF. The unbiased probability distribution function is estimated for each window, and must be recombined to calculate the full PMF. [11]

While the weighted histogram analysis method has been arguably the most popular method for determining the additive constants necessary at each window to construct the 'best' complete PMF, [73] more recent methods have been shown to be better estimators of the unbiased PMF. Such examples include the multistate Bennett acceptance ratio (MBAR) [74] and variational free energy profile, [75] which have demonstrated

superior performance in computing not only the PMF more efficiently with less data, [75] but also reasonable estimations of the statistical errors. [74]

2.2.2 Steered Molecular Dynamics

The idea of steered molecular dynamics (SMD) is very similar to that of umbrella sampling. A harmonic biasing potential is added to the underlying potential along a reaction coordinate to drive the sampling along that coordinate. Unlike umbrella sampling in which the harmonic potentials are fixed at a given position along the reaction coordinate, the potential is moved along the reaction coordinate at some speed in SMD simulations.

While SMD appears similar to umbrella sampling, the fact that the umbrella potential moves marks a significant fundamental difference between the two techniques. Umbrella sampling performs equilibrium sampling with the biased Hamiltonian, whereas the finite speed of the moving umbrella in SMD simulations is inherently non-equilibrium. [11] The non-equilibrium work done by moving umbrella is tabulated, and effectively represents an upper-bound estimate on the free energy according to Eq. 2–16. [11]

$$\langle W_{1,2}(\vec{x})\rangle \geq \Delta A_{1,2} \tag{2–16}$$

where W is the work along the path given by $\vec{x}$ between states 1 and 2 and ΔA is the free energy change between those two states. Clearly, the utility of the work profile calculated using SMD simulations is severely limited since Eq. 2–16 is simply an inequality.

The link between equilibrium free energies and computed work profiles from SMD simulations was supplied by Jarzynski in 1997. [76] The so-called *Jarzynski equality*, shown in Eq. 2–17, states that equilibrium free energies can be calculated from a complete ensemble of work profiles along the reaction coordinate between an ensemble of starting points at state $\vec{x}_1$ and driving the center of the umbrella to state $\vec{x}_2$.

$$\exp\left(-\beta \Delta A_{1,2}\right) = \left\langle \exp\left(-\beta W_{1,2}(\vec{x}_0)\right)\right\rangle \tag{2–17}$$

A caveat to Eq. 2–17 is that an infinite number of work profiles between states 1 and 2 are necessary for the equality to hold. Because simulating an infinite number of trajectories is impossible, we must be content to estimate the total free energy using a finite number of simulations. Fortunately, the exponential average converges very rapidly with a small number of 'good' work profiles (*i.e.,* low-energy work profiles that follow the true PMF closely), since high-energy profiles contribute little to the average.

Optimizing the computational performance of SMD simulations is a balancing act. Pull too quickly and all computed work profiles will most likely be much higher than the true PMF, giving you a poor estimate of the actual free energy. Pull too slowly and the simulations will take too long to traverse the full reaction coordinate. The *optimal* pulling speed will generate a wide distribution of work profiles that gives a good estimate of the total PMF.

2.2.3 Expanded Ensemble

A common class of techniques used to enhance sampling compared to standard molecular dynamics are so-called *expanded ensemble* techniques. The canonical ensemble, for instance, is limited by the thermodynamic constraints imposed by requiring all members of the ensemble to have the same number of particles, volume, and temperature (*NVT*). These variables are referred to as *state* parameters, since they define each state present in the ensemble.

The way expanded ensemble techniques enhance sampling is to generate a larger ensemble in which many, smaller thermodynamic ensembles are brought into equilibrium. True to its name, enhanced sampling is obtained by sampling from an *expanded* ensemble of numerous standard thermodynamic ensembles. The first example examined in the literature involved expanding the canonical ensemble to multiple temperatures. [77] This new ensemble is a combination of multiple canonical ensembles each

at a different temperature. By allowing a simulation to migrate through temperature-space as it is sampling new conformations, expanded ensemble simulations can take advantage of the flatter free energy surfaces present at higher temperatures to enhance conformational sampling while still collecting statistics at the target temperature of interest.

The total partition function for this new, expanded ensemble is shown in Eq. 2–18. [77]

$$Q = \sum_{m=0}^{M} Q_m \exp \eta_m \tag{2–18}$$

where Q_m is the canonical partition function at a given temperature m and η_m is a carefully chosen set of tuning parameters designed to bias the simulation toward spending more time near the temperatures of interest. [77] Either periodically or at random intervals throughout the MD or MC simulation, a Monte Carlo attempt to change the temperature of the 'current' conformation is performed. Successful attempts between temperatures k and m are evaluated according to the Monte Carlo criteria shown in Eq. 2–19.

$$P_{k \to m} = \min \left\{ (\beta_k - \beta_m) \, H(\vec{x}) + \eta_m - \eta_k \right\} \tag{2–19}$$

where $P_{k \to m}$ is the probability of changing from temperature k to temperature m and η is the constant tuned to control the residence time of the simulation at each temperature (see Eq. 2–18).

If statistics are desired for a specific temperature, an ensemble can be generated from all snapshots with the target temperature. By allowing the simulation to visit higher temperatures, new pathways around and over barriers are opened up by traversing temperature-space and configuration (conformation) space simultaneously. The available kinetic energy at higher temperatures makes it more likely that high barriers will be crossed than at lower temperatures, while the samples taken at lower temperatures provide the resolution necessary to characterize the thermodynamic properties

at biologically relevant temperatures. However, since the simulation is allowed to visit multiple temperatures, a significant portion of the simulation is 'wasted' sampling higher temperatures that contribute little to the low-temperature ensemble. The amount of time that the simulation is permitted to spend at each temperature must be carefully balanced to enhance sampling with enough simulation done at higher temperatures and maintain a desired level of resolution of the low temperature ensemble. This distribution is controlled by the η parameter at each temperature (Eq. 2–18), whose optimal values are determined by running short simulations at each temperature to estimate the shape of phase space. [77]

2.2.4 Replica Exchange Molecular Dynamics

Replica exchange molecular dynamics (REMD) simulations are a special case of expanded ensemble simulations that are designed to be scalable to modern, parallel computers. In these simulations, a finite number of independent simulations, or replicas are run, each with a different state parameter (*e.g.,* different temperatures). These replicas periodically attempt to exchange information between each other—either configurations or state parameters—in such a way that maintains the validity of the 'subensemble' of each replica. A diagrammatic representation of REMD simulations is shown in Fig. 2-8. [78]

To ensure that each replica is in a state of equilibrium with all other replicas in the REMD simulation, a reversible Markov chain of moves along the state parameter dimension is necessary (see Fig. 2-8). Trial moves are typically done between a single pair of replicas to simplify the expression for calculating the exchange probability. As we saw in Section 1.1.2.1, applying the Metropolis criteria to a randomly proposed MC move satisfies the requirement of detailed balance. Therefore, Metropolis MC is used to enable replicas to sample along the state space coordinate in REMD calculations.

There are several different choices one can make for the state space parameter when setting up a REMD calculation. Common choices include temperature [78],

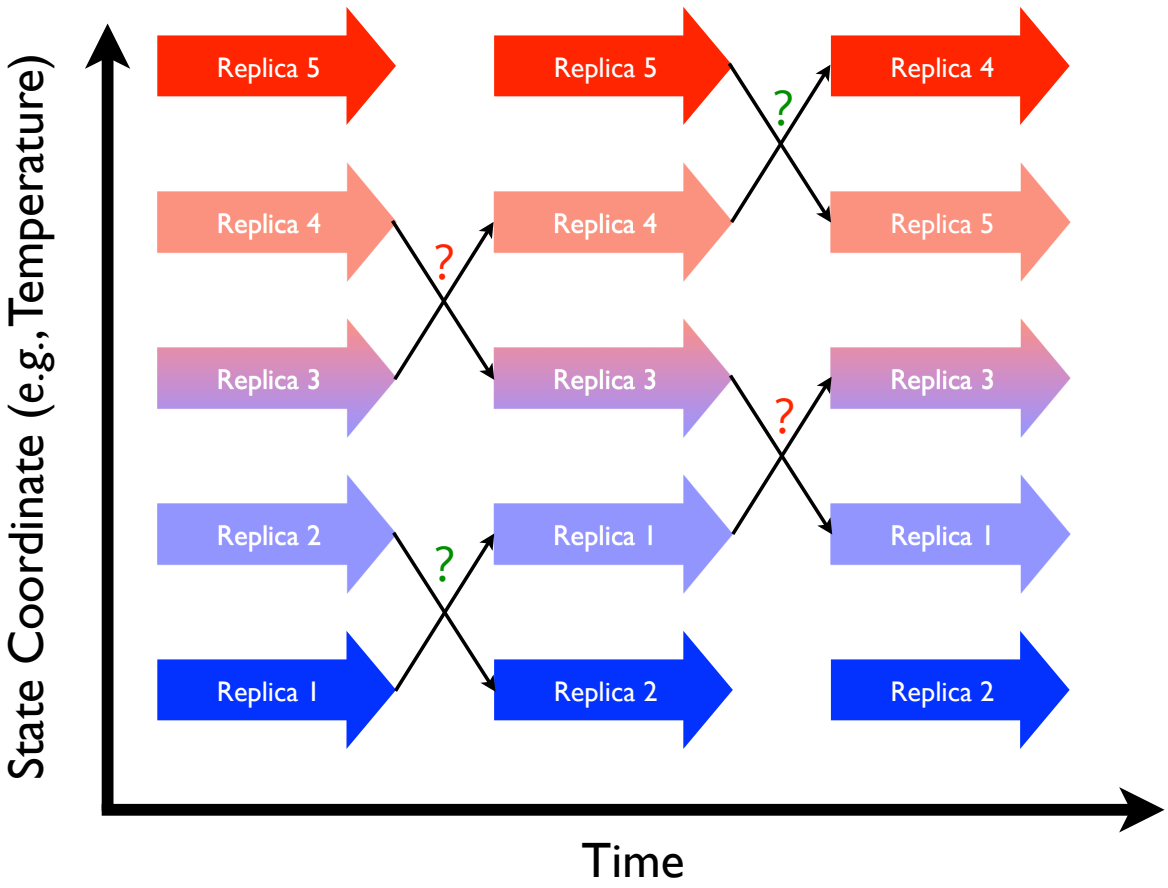

Figure 2-8. Diagrammatic sketch of REMD simulations. Replicas are represented as thick arrows and exchange attempts are shown between adjacent replicas connected by thin black arrows. The question-mark indicates that a MC move is accepted with the probability calculated according to the Metropolis criteria. Successful and unsuccessful exchange attempts are shown with a green or red question mark, respectively.

umbrella potentials (for umbrella sampling simulations) [79, 80], Hamiltonians, [81–85] and solution pH, [86–89] among others. [90, 91] These methods are discussed in detail in later chapters.

2.3 Free Energy Calculations

Calculating the 'free energy' is the Holy Grail of computational chemistry, as it furnishes the ultimate comparison with experimental observables. As a result, significant effort has been spent searching for computationally efficient ways to accurately calculate

free energy changes of various processes, including conformational rearrangement, [92, 93] protein folding, [94–96] solvation, [97, 98] protein-ligand binding, [99, 100] and protein-protein binding [101, 102] among others.

Because the free energy is a state function, the free energy differences between two distinct states are independent of the path taken from the starting state to the other. In fact, this principle holds even if that pathway is completely fictitious! This gives simulation a significant advantage in computing free energies, since the easiest path along which to compute this value may be used—even if that pathway is chemically nonsensical. Despite this advantage, however, free energies remain exceedingly difficult to compute directly. [103] In this section, I will briefly outline several methods commonly used to compute free energy differences between two states—Thermodynamic Integration, Free Energy Perturbation, and end-state free energy methods.

2.3.1 Thermodynamic Integration

Thermodynamic Integration (TI) is a so-called *alchemical* free energy calculation method, since it contains an interpolating parameter λ that 'morphs' one system into another. [11, 12] Assuming the two states 0 and 1 obey the potential energy functions, or Hamiltonians, H_0 and H_1, respectively, the Hamiltonian of the perturbed system is shown in Eq. 2–20.

$$H(q,p) = f(\lambda)H_0(q,p) + g(\lambda)H_1(q,p) \tag{2–20}$$

where λ is a switching parameter with the continuous domain between 0 and 1 and the functions $f(\lambda)$ and $g(\lambda)$ obey the relationships

$$\begin{aligned} f(0) &= 1, \quad f(1) &= 0 \\ g(0) &= 0, \quad g(1) &= 1 \end{aligned}$$

such that the Hamiltonian at either end point is a pure function of one of the two states. A linear switching function, with

$$\begin{aligned} f(\lambda) &= \lambda \\ g(\lambda) &= 1 - \lambda \end{aligned}$$

is commonly used due to its simplicity. Because the scaling parameter λ is continuous and can be made to vary infinitely slowly, sampling done at the intermediate states (*i.e.,* $0 < \lambda < 1$) are always at equilibrium. The total free energy, then, can be calculated via the integral shown in Eq. 2–21. [12]

$$\Delta G_{0\to 1} = \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_\lambda d\lambda \tag{2–21}$$

where the average is taken over the ensemble generated at each λ. Because doing "true" TI would require an infinite number of simulations for λ equal to all real numbers between 0 and 1, Eq. 2–21 is approximated using a Riemann sum, shown below.

$$\Delta G_{0\to 1} \approx \sum_{\lambda=0}^{1} \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_\lambda \Delta\lambda$$

Therefore, TI calculations require the selection of a set of *windows* (*i.e.,* λ selections between 0 and 1) at which an ensemble must be generated to evaluate the gradient of the coupled Hamiltonian with respect to the coupling parameter λ. A sufficient number of λ values must be chosen to obtain an accurate and converged free energy—the number of required windows varies from system to system.

For the simple linear switching function described previously, the gradients required by Eq. 2–21 can be computed analytically based on the functional form of the underlying Hamiltonians. The only terms that contribute to $\partial G/\partial\lambda$ are those terms that include interactions with one of the atoms that differ in some way between the two end states. Therefore, as one would expect, the TI calculations converge more rapidly when the perturbation between states 0 and 1 are small. Indeed, TI calculations have been

successfully employed to calculate many free-energy based properties, such as protein pK_as, [104] and solvation free energies. [105]

Traditional TI calculations suffer from a severe limitation when applied to typical MM force fields, however. Using the functional form of the Amber force field (Eq. 1–21) as an example, there is a significant problem with converging TI calculations at windows where λ approaches either 0 or 1 when atoms are 'appearing' or 'disappearing' (*i.e.,* when those atoms exist only in one end point). The problem arises in the Lennard Jones term which has a very strong repulsive force at close intermolecular distances with a singularity at the origin. This singularity exists as long as the Hamiltonian containing this atom has non-zero weight according to the chosen λ value, which is true for all values except 0 or 1. Therefore, even when λ is arbitrarily close to either 0 or 1, there is a region of space around the center of all disappearing atoms in which no atom can enter due to the repulsive r^{-12} term of the almost-vanished atom. This phenomena, referred to as a *hard core*, hurts convergence of TI calculations by preventing configurations in which molecules enter the space partially occupied by a disappearing atom. [106] Figure 2-9 demonstrates this effect by plotting the Lennard Jones potential between two carbon atoms when one of them vanishes at $\lambda = 1$.

To address this limitation, an additional λ-dependent term is added to disappearing atoms to soften the core near the end points and eliminate the singularity that prevents particles from entering the space occupied by a partially-vanished atom. This approach, described below, is referred to as *soft-core thermodynamic integration*.

Soft-core TI. To avoid the singularity in the Lennard Jones potential term of a vanishing atom in TI calculations, the functional form of this potential is adjusted by Eq. 2–22. A good choice for the functional form of the soft-core potential should satisfy several conditions. First, the potential should be either 0 for a vanished atom or the original Lennard Jones potential for an atom that is 'fully' present. Second, the potential must not diverge between a partially vanished atom and an unperturbed atom when

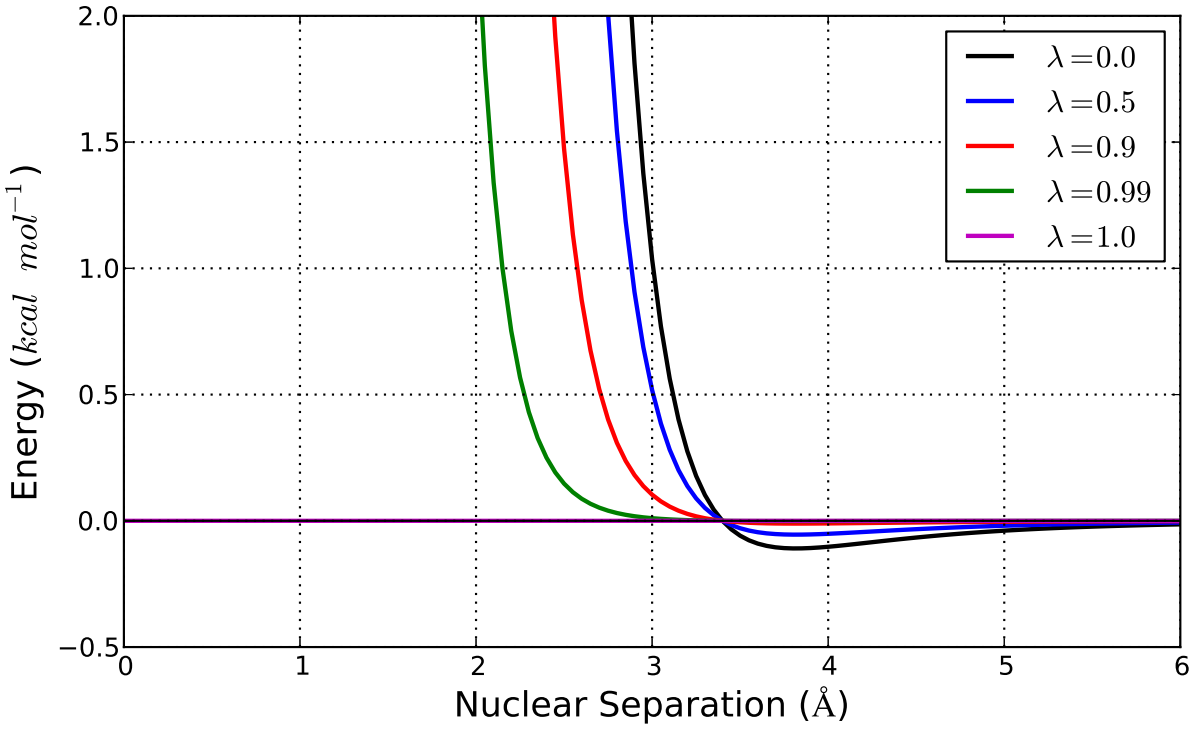

Figure 2-9. Hard core of disappearing atom caused by the Lennard Jones terms. The $\lambda = 1$ state is the one in which a carbon atom has vanished.

their separation approaches zero. Finally, the force must remain conservative (*i.e.,* the energy difference between any two points must be independent of the path taken between them). Eq. 2–22 satisfies all of these requirements, making it a good candidate to replace the standard Lennard-Jones potential in vanishing atoms.

$$U_{ij}^{LJ}(r_{ij},\lambda) = \lambda^n 4\varepsilon_{ij}\left(\frac{1}{\left[\alpha_{LJ}(1-\lambda)^2+\left(\frac{r_{ij}}{\sigma_{ij}}\right)^6\right]^2}-\frac{1}{\alpha(1-\lambda)^2+\left(\frac{r_{ij}}{\sigma_{ij}}\right)^6}\right)$$
$$U_{ij}^{LJ}(r_{ij},\lambda) = (1-\lambda)^n 4\varepsilon_{ij}\left(\frac{1}{\left[\alpha_{LJ}\lambda^2+\left(\frac{r_{ij}}{\sigma_{ij}}\right)^6\right]^2}-\frac{1}{\alpha\lambda^2+\left(\frac{r_{ij}}{\sigma_{ij}}\right)^6}\right) \quad (2\text{–}22)$$

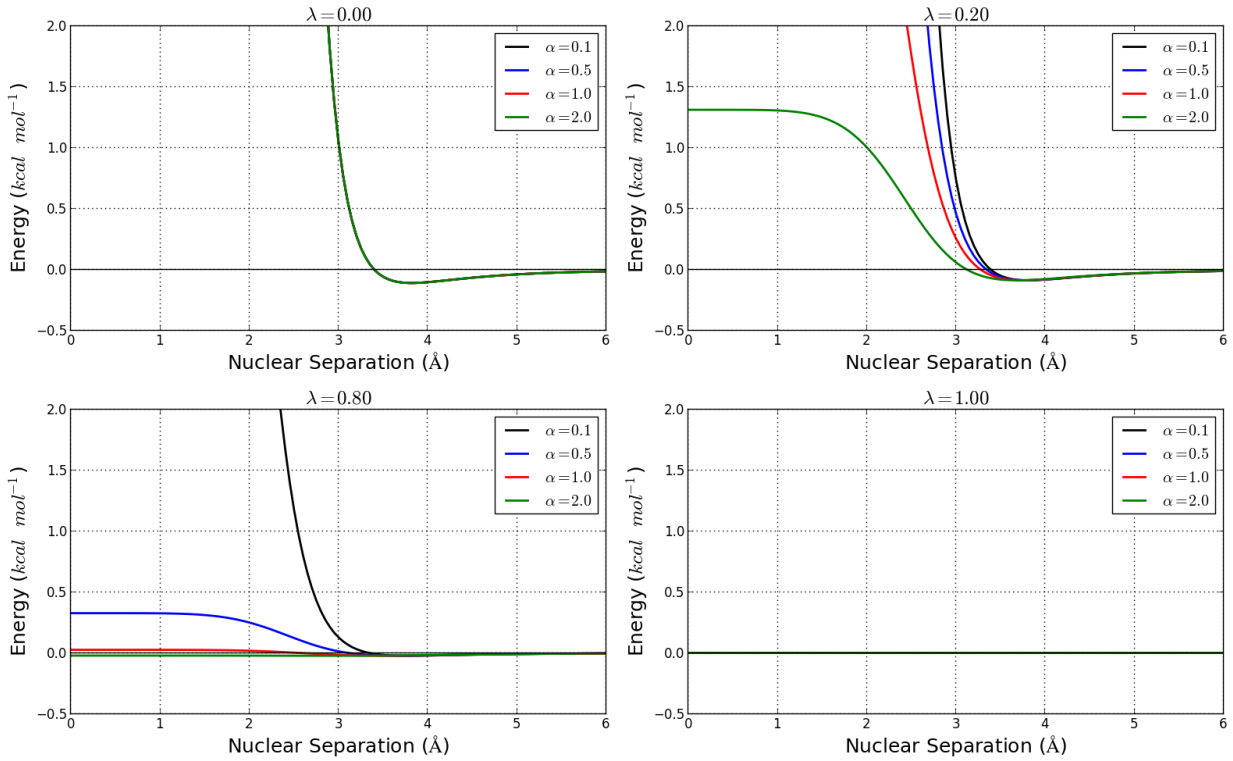

Figure 2-10. Functional form of soft-core Lennard Jones interactions with different values of α from Eq. 2–22. The $\lambda = 1$ state is the one in which an atom has vanished. See Fig. 2-9 to see how soft cores enable sampling close to the center of the vanishing atom when $\lambda \approx 1$.

The top equation of Eq. 2–22 corresponds to the functional form when one of the atoms vanishes when $\lambda = 0$ and the bottom equation corresponds to the case where one of the atoms vanishes when $\lambda = 1$. The denominators in Eq. 2–22 do not contain a singularity when $r_{i,j} = 0$ when one of the atoms has vanished. The parameter α controls how 'soft' the core of the vanishing atoms are, as shown in Fig. 2-10. [107]

TI simulations using soft-core potentials for the Lennard Jones terms of vanishing atoms show significantly better convergence of free energies. [105, 107]

2.3.2 Free Energy Perturbation

An alternative approach to calculate the free energy difference between two states is the free energy perturbation (FEP) method proposed by Zwanzig. [108] The free energy between two states A and B can be calculated according to Eq. 2–23.

$$\Delta G_{A\to B} = -k_B T \ln \langle \exp\left(-\beta(E_B - E_A)\right)\rangle_A \tag{2–23}$$

where the averages are taken over the ensemble generated in state A and E_B are the energies of the structures in ensemble A evaluated with the Hamiltonian governing the behavior of ensemble B. This is called *forward* sampling, since the ensemble we used to estimate the free energy came from the original state. [12] The reverse, or *backward* sampling, represents the reverse process (*i.e.,* by swapping the indices A and B in Eq. 2–23). Because free energy is a state function, the forward and backward free energies should sum exactly to 0.

This balance between the forward and reverse sampling rarely balances completely for complex transformations, however, indicating a shortcoming in the naïve FEP approach. If the ensembles generated by the two states are significantly different, the forward and reverse free energies will be systematically different. For instance, if we are simulating the free energy change of transforming benzene into phenol to calculate their difference in solvation free energies, the solvent arrangement around the two systems will be significantly different due to the added bulk of the hydroxyl group in phenol as well as the difference in the dipole moment caused by that hydroxyl.

To address this shortcoming, two end-states are often interpolated using a coupling parameter λ similar in spirit to TI. By perturbing the system slowly from state A to B, the differences between adjacent states are reduced, leading to similar ensembles that generate more consistent forward and reverse free energies when used in Eq. 2–23. [12]

2.3.3 End-state Calculations

The final family of free energy methods I will discuss here are so-called *end-state* calculations since they involve calculating the free energy change $\Delta G_{A\to B}$ from simulations performed only on the two physical end states A and B. These methods are

often used to estimate binding free energies of a non-covalently bound protein-ligand or protein-protein complex. [99, 101, 102, 109, 110]

I will discuss two methods within this family that are routinely used in binding free energy calculations—the so-called Molecular Mechanics Poisson-Boltzmann surface area (MM-PBSA) method [111, 112] and the linear interaction energy (LIE) method. [113, 114]

2.3.3.1 MM-PBSA

MM-PBSA, and its closely-related counterparts MM-GBSA (GB implicit solvent), MM-3DRISM (3D-RISM implicit solvent), and QM/MM-GBSA, are commonly used to calculate binding free energies of noncovalently bound complexes. These methods compute binding free energies via the thermodynamic cycle shown in Fig. 2-11, where the solvation free energy terms are computed using an implicit solvent model (*e.g.,* Poisson-Boltzmann, Generalized Born, or 3D-RISM). The total free energy computed along the cycle, shown in Eq. 2–24, is taken from ensemble averages over a simulated trajectory. The ensembles are typically generated by running either a MD or MC simulation for each of the three states—the bound complex, unbound receptor, and unbound ligand. [110]

$$\Delta G_{binding} = \langle \Delta H_{solv,bound} \rangle + \langle \Delta H_{binding,gas} \rangle - \langle \Delta H_{solv,unbound} \rangle \tag{2–24}$$

The averages in Eq. 2–24 are taken from the ensembles of each system.

The principal computational cost of an MM-PBSA calculation is due to the initial simulations required to construct each ensemble. To reduce the cost of computing binding free energies with MM-PBSA, all three ensembles mentioned above can be extracted from a single simulation of the bound complex, a technique referred to as the *single trajectory protocol*. [110] This approach will always underestimate the binding free energy (predicting overly-stable binding), since the bound states of the receptor and ligand will always be less stable in the bound conformation than they are when

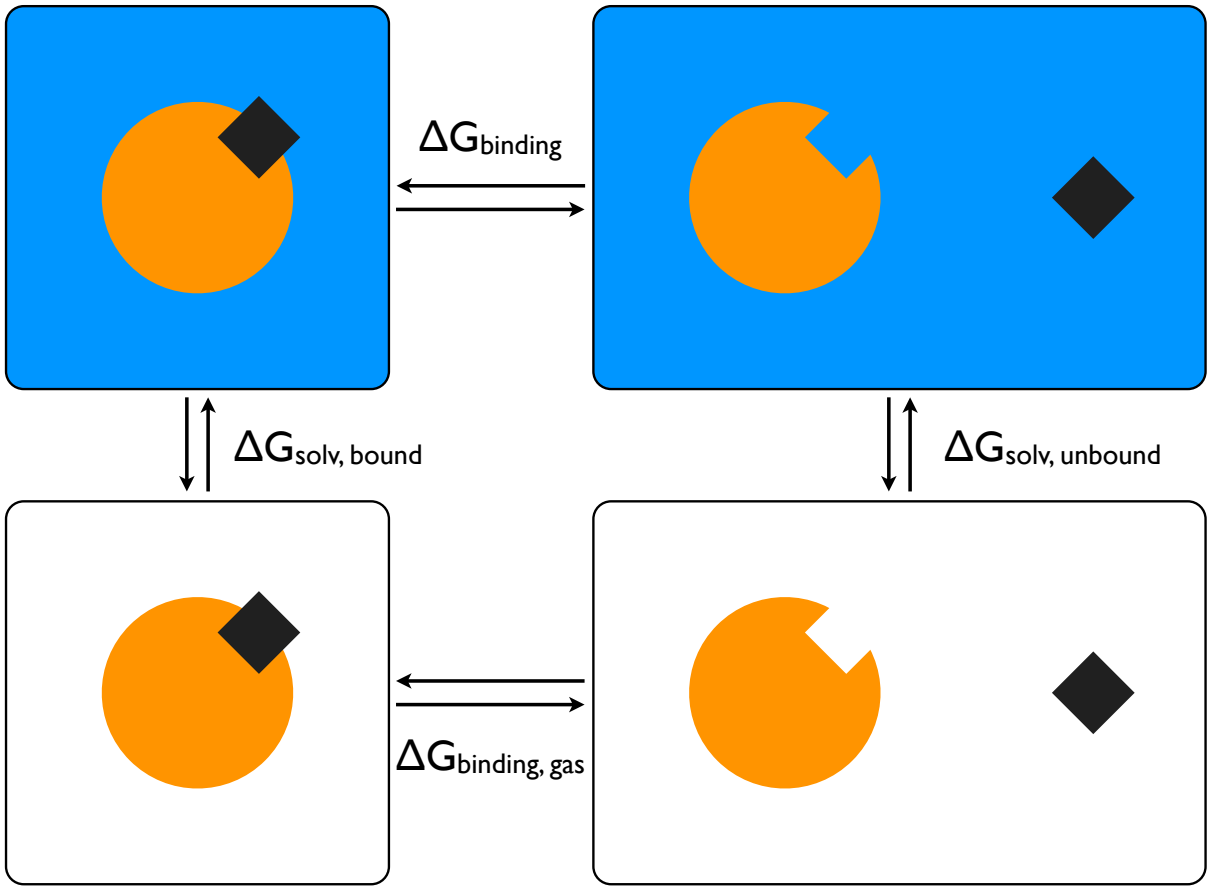

Figure 2-11. Thermodynamic cycle for MM/PBSA calculations. Implicit solvent is represented with a blue background, while a white background represents systems in the gas phase.

free in solution. However, when making the same approximation for a family of related receptors and ligands, the systematic errors in each end-state calculation will be similar. Therefore, MM-PBSA methods can be useful for tasks like rank-ordering a handful of proposed inhibitors for a specific enzyme by calculating accurate relative binding free energies. [109]

2.3.3.2 LIE

The linear interaction energy method (LIE) is another end-state method widely used to calculate noncovalent binding free energies of small ligands to proteins. LIE is based on assuming that the binding free energy is a result of the energetic differences of the

ligand in the two environments—bound in the active site of the protein versus hydrated in solution—and obeys linear response theory. [113] The electrostatic contribution to LIE can be simplified to the concept of charging the atoms of the ligand inside a cavity that has the same shape as the ligand. Following the ideas Marcus theory, [115] the reorganization energy of the surrounding solvent λ can be expressed as [113]

$$\lambda = \langle V_B - V_A \rangle_A - \Delta G_{A \to B} = \langle V_A - V_B \rangle + \Delta G_{A \to B} \tag{2–25}$$

Solving for $\Delta G_{A \to B}$ in Eq. 2–25 yields the following expression for the free energy of the change from the ligand bound in environment A to the ligand bound in environment B:

$$\Delta G_{A \to B} = \frac{1}{2} \left(\langle \Delta V \rangle_A + \langle \Delta V \rangle_B \right) \tag{2–26}$$

Eq. 2–26 can be readily applied to the electrostatic contribution of the binding free energy, but the non-polar non-bonded interactions—namely the van der Waals interactions—are not known to obey Marcus theory accurately. As a result, the non-polar interaction energy is scaled in Eq. 2–27 by a parameter that is adjusted to fit a database of known binding affinities. [113]

$$\Delta G_{bind} = \frac{1}{2} \left\langle \Delta V_{w \to p}^{elec} \right\rangle + \alpha \left\langle \Delta V_{w \to p}^{vdW} \right\rangle \tag{2–27}$$

where the ‘w’ subscript indicates the ligand free in water and ‘p’ indicates the ligand bound in the protein. The van der Waals interactions are scaled by the parameter α, which was adjusted to give good agreement with experimental binding affinities. LIE calculations require two ensembles to be generated—one of the ligand free in explicit solvent and the other of the ligand bound in the protein. A diagrammatic representation of the LIE calculation is shown in Fig. 2-12.

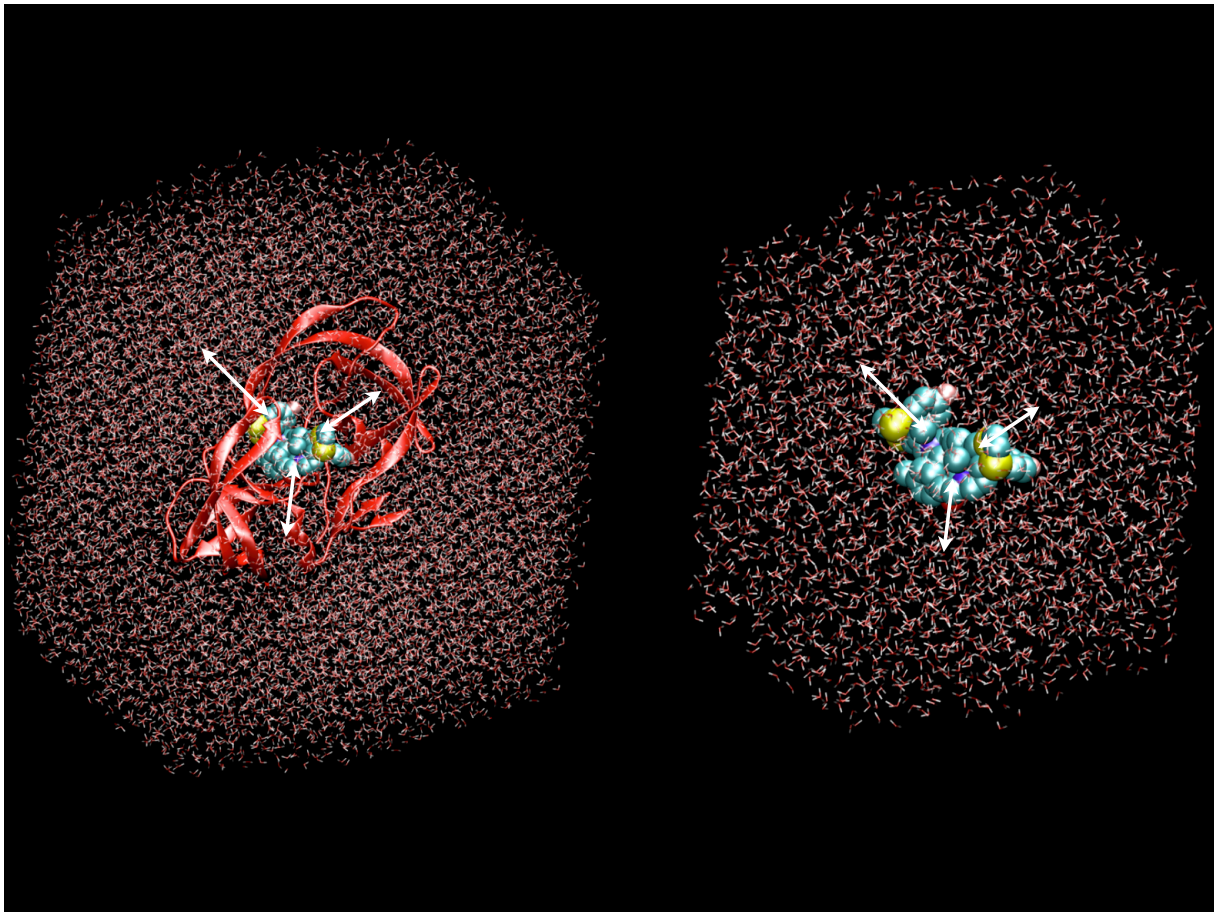

Figure 2-12. Schematic showing interactions necessary to compute the LIE free energy of noncovalent binding for a ligand in a protein using white arrows.

CHAPTER 3
CONSTANT PH REPLICA EXCHANGE MOLECULAR DYNAMICS

In this chapter, I will discuss my work with REMD simulations in which the state parameter—the property exchanged between replicas—is the solution pH. This work is reprinted with permission from Swails, and Roitberg, *J. Chem. Theory Comput* **2012**, *8* 4393–4404. Copyright 2012 American Chemical Society. [88]

3.1 Constant pH and pK_a Calculations

Solution pH is often critical to the proper functioning of biological catalysts. [116, 117] The pH environment of biological systems influences the ionization equilibria present in the system, thereby affecting the protonation state of various *titratable residues* in the system. A titratable residue is any residue that has a pK_a value within 1 or 2 units of the biological pH range (which is roughly 1 – 9). The protonation states of these residues can have a profound effect on the stability of the system, the system's interactions with its surroundings, and any catalytic mechanism that relies on a specific set of protonation states to carry out general acid-base catalysis or nucleophilic attack. [118]

Simulations aimed at modeling proteins or nucleic acids must have some method for assigning protonation states for each titratable residue. Because bond breaking and bond formation are impossible in classical force fields, each residue is typically assigned one protonation state and the entire simulation is run using this set of states. This approach has two drawbacks. First, the choice of protonation state is often based on the behavior of each titratable residue when free in solution. This may not be a valid assumption, however, because the protein or nucleic acid environment can modulate a residue's protonation state equilibrium. Second, a single protonation state may not accurately represent the true ensemble of states at the desired pH. If the pH is close to the pK_a of a given residue, or if the system populates conformations in which

the dominant protonation state changes, then the true ensemble is represented by conformations with different protonation states.

The first drawback can be addressed by using tools such as *PROPKA* [119] and *H++*, [120] which provide a means to assign protonation states to titratable residues by calculating the pK_a of the starting structure. However, this does not address the possibility that multiple protonation states may be necessary to build the desired ensemble.

While it may seem that both drawbacks can be addressed by simply running simulations with every possible set of protonation states, this approach quickly becomes unwieldy. Given N titratable residues, there are at least 2^N distinct protonation states assuming each residue is either protonated or deprotonated. With only 10 titratable residues this amounts to a minimum of 1024 distinct simulations! While most of these states may not be found in the given ensemble, there is no way to know which ones to exclude *a priori*. It is important, then, to develop a method capable of directly probing protonation state equilibria in biological molecules.

In order to probe protonation state equilibria in a thermodynamically meaningful way, simulations must be run at constant pH. The first approaches for constant pH simulations used *continuum electrostatics* methods to calculate the perturbing effect of the system environment on protonation state equilibria using an implicit solvent model (*e.g.,* the Poisson-Boltzmann equation) on a single structure. [121–123] These methods, while sometimes useful for calculating pK_a values in biological systems, assume that the full protonation state equilibria can be characterized with a single structure. In particular, using a single structure neglects the response of the system relaxing to accommodate the new protonation state. While the effects of system relaxation has been addressed to some degree by treating the protein interior with a large dielectric constant, [123] this approach assumes an unphysical homogeneity in the system's dielectric response to protonation state changes.

A more sophisticated approach to incorporating the system response involves simultaneous sampling of both protonation states and side-chain rotamers. [124] This approach dramatically improves pK_a prediction with respect to experiment, but may be insufficient for systems with large scale conformational changes that cannot be attributed only to side chain mobility.

To capture the coupled nature of conformational flexibility with protonation state sampling, several constant pH molecular dynamics (CpHMD) methods have been proposed. [125–131] These methods have proven to be powerful tools for pK_a calculation and prediction, although there is still room for improvement. [132] For systems in which some titratable residues experience large pK_a shifts, predicted pK_a values are often in error by more than 1 pH unit even in the studies that reproduce experimental values the closest. [132] This is usually a direct result of insufficient sampling of protonation and conformational states or a limitation of the underlying model. Machuqueiro and Baptista have shown that correcting some of the limitations of the underlying model, such as improving the definition of the reference compound (whose role is described below in the *Theory* section) and improving the underlying force field improves results. [133] Other work has coupled enhanced sampling techniques, such as accelerated molecular dynamics [134], with CpHMD to show that improved conformational sampling also improves predicted pK_as with respect to experiment. [135] Webb et al. recently published a systematic study showing that the errors inherent to experimental measurements are often larger than those reported, which has important implications for assessing the accuracy of theoretical predictions. [136]

Replica exchange molecular dynamics (REMD) is a family of extended ensemble techniques that have been shown to dramatically improve sampling. [78, 85, 137–140] In REMD simulations, a series of independent replicas (single MD trajectories of a system) periodically attempt to exchange information, such as temperature [78, 137] and, more recently, pH [86, 87] to sample from an expanded ensemble covering multiple states.

In this study, I implemented the pH-REMD method described by Itoh et al. [87] in the *sander* module of the Amber [141] software package. I show how this method significantly improves sampling compared to CpHMD in hen egg-white lysozyme (HEWL), a system commonly used as a benchmark for pK_a calculations. Titration curves generated using pH-REMD contain significantly less noise and converge more rapidly than CpHMD, suggesting pH-REMD is a powerful tool for carrying out pK_a predictions.

Our group has previously shown that temperature REMD simulations converge significantly faster with increasing exchange attempt frequency (EAF). [142, 143] Here, I show that increasing the EAF in pH-REMD simulations causes pH-dependent observable properties to converge faster as well.

In the next sections, I will describe the foundation of the constant pH method developed by Mongan et al. [130] and the corresponding pH-REMD method. [86, 87] I will then describe the details of my study on HEWL followed by the results and conclusions drawn from that study.

3.2 Theory

Here I will describe the theory behind constant pH simulations, beginning with a description of the statistical ensemble corresponding to this family of simulations and following up with an overview of the methods used in this study.

3.2.1 The Semi-Grand Ensemble

At conditions of constant pH, systems no longer obey the constraints of the typical canonical ensemble presented in the opening chapter. Instead, the chemical potential of hydronium—related directly to the solution pH by Eq. 3–1—is held constant, thereby allowing the H^+ count to fluctuate.

$$\begin{aligned}\mu^{H^+} &= \frac{\partial G}{\partial N^{H^+}} \\ &= -kT\ln[H^+] \\ &= -kT\ln(10)\log[H^+] \\ &= kT\ln(10)\,pH\end{aligned} \tag{3–1}$$

where μ_{H^+} is the chemical potential of hydronium and the activity of the hydronium ion has been replaced by the concentration due to the very low concentrations in which it is typically present in biological systems. Looking back at Eq. 1–6, we can calculate the partition function of the semi-grand canonical ensemble using Eq. 3–2, introducing a pH-dependence in our sampling scheme.

$$\begin{aligned}\Xi(\mu^{H^+}, V, T) &= \sum_{N^{H^+}} Q(N, V, T)\exp(\beta RTN^{H^+}\ln(10)\,pH) \\ &= \sum_{N^{H^+}} Q(N, V, T)\exp(N^{H^+}\ln(10)\,pH)\end{aligned} \tag{3–2}$$

3.2.2 CpHMD

I used the constant pH molecular dynamics (CpHMD) method developed by Mongan et al. [130] that employs Monte Carlo transitions between discrete protonation states at periodic intervals during a MD simulation to probe protonation state equilibria. In this CpHMD implementation, both the dynamics and the MC protonation state sampling are performed in Generalized Born implicit solvent. After a predetermined number of steps, the MD is halted and a protonation state change is attempted by evaluating the energetic cost of that proposed change, calculated according to Eq. 3–3. [130]

$$\Delta G = k_B T\,(pH - pK_{a,ref})\ln 10 + \Delta G_{elec} - \Delta G_{elec,ref} \tag{3–3}$$

Table 3-1. Reference pK_a values for the acidic residues treated in this study. Values are the same as those used in the original Amber CpHMD implementation.[130]

Residue	Reference pK_a
Aspartate	4.0
Glutamate	4.4
Histidine (H^{δ})	7.1
Histidine (H^{ϵ})	6.5

Eq. 3–3 represents a free energy change of protonating or deprotonating a titratable residue embedded in a biological system with respect to a predefined reference compound. The reference compound is a monomer of the titratable residue capped with small, neutral functional groups. In Eq. 3–3, ΔG_{elec} is calculated by taking the difference of the electrostatic energy between the proposed and existing protonation states. [130]

Directly calculating the free energy change associated with protonation or deprotonation is difficult because evaluating the energetic cost of desolvating a free proton and making and breaking chemical bonds is impossible in a classical mechanical framework. Therefore, we calculate the free energy cost of this protonation state change by comparing the free energy of the protonation state change to $\Delta G_{elec,ref}$ in Eq. 3–3, a precomputed free energy for the reference compound that is adjusted to reproduce experimental pK_a values. Eq. 3–3, then, represents a shift in the pK_a of a titratable residue in a biological system from its value free in solution. The reference compound pK_a values used in the Amber CpHMD implementation [130] are shown in Table 3-1.

Running a CpHMD simulation, we obtain an ensemble consisting of multiple protonation states properly weighted for the semi-grand canonical ensemble, the thermodynamic ensemble corresponding to constant temperature, volume (or pressure) and chemical potential of hydronium (*i.e.,* constant pH). [126] Because the simulation is assumed to be ergodic, the deprotonation fraction can be calculated by simply counting the fraction of ensemble members in which the residue is deprotonated. Multiple CpHMD simulations must be run with a range of pHs to calculate pK_a values for titratable residues in biological systems by fitting a titration curve to the data.

Running a simulation with an expanded ensemble so each CpHMD simulation is in equilibrium with simulations at different pHs can further enhance sampling from the desired semi-grand canonical ensemble. For this, we turn to the pH-REMD method.

3.2.3 pH-REMD

Replica exchange simulations at constant pH (pH-REMD) is a variant of replica exchange in which each replica is simulated at a separate pH. The full pH-REMD simulation represents an expanded ensemble in which each replica samples conformations with a fixed pH and samples different pH values at a fixed conformation.

In this study, I implemented the pH-REMD method introduced by Itoh et al. [87] in the *sander* module of Amber. [141] In pH-REMD, adjacent replicas in the pH ladder swap pH with the Monte Carlo exchange probability

$$P_{i\rightarrow j} = \min\left\{1, \exp\left[\ln 10\left(N_i - N_j\right)\left(pH_i - pH_j\right)\right]\right\} \tag{3–4}$$

for replicas i and j where N_i is the number of titratable protons present in replica i and pH_i is the pH of replica i prior to the exchange attempt.

Our group recently developed a different pH-REMD method in which replica exchanges are attempted via Hamiltonian exchange where only atomic coordinates are swapped. [89] In contrast, the currently proposed method only swaps the solution pH between replicas. For large systems with more than 3 – 5 titratable residues, the proposed method of swapping solution pH between replicas achieves more efficient replica exchanges than the variant employing Hamiltonian exchange. For HEWL, specifically, the Hamiltonian REMD variant experienced an exchange success rate of $< 0.01\%$, which is effectively indistinguishable from CpHMD simulations.

3.3 Methods

3.3.1 Starting Structure

I chose to study hen egg white lysozyme because it is well-characterized both experimentally [136, 144, 145] and computationally. [86, 130, 146] I chose the structure

from the protein data bank (PDB) with the code 1AKI [147] because it was the focus of Mongan's original study. [130]

The topology file was prepared in the *tleap* module of AmberTools 12 using the Amber *ff10* force field, which is equivalent to *ff99SB* [23] for proteins. Crystallographic water molecules were removed from the starting structures, and *tleap* added all hydrogen atoms. Finally, the *mbondi2* intrinsic radii for implicit solvent calculations were selected in *tleap* to be consistent with the initial implementation of CpHMD. [130]

3.3.2 Molecular Dynamics

To be consistent with the original implementation, the Generalized Born model described by Onufriev et al. [49] (corresponding to the input parameter *igb=2* for Amber programs) was used with the salt concentration, modeled as a Debye screening parameter, set to 0.1 M in every simulation. [130] Due to the long-range nature of the electrostatic forces, I always used an infinite cutoff for non-bonded interactions.

Each starting structure was minimized using 50 steps of steepest descent followed by 950 steps of conjugate gradient with 10 kcal mol^{-1} $Å^{-2}$ restraints on the backbone atoms to relieve bad contacts. Then, the minimized structure was heated by varying the target temperature linearly from 10 K to 300 K for 667 ps, keeping weak restraints—1 kcal mol^{-1} $Å^{-2}$–on the backbone. I used the Langevin thermostat with a collision frequency of 5 ps^{-1} to control the temperature. These simulations were performed using the *pmemd* module of the Amber 12 program suite. [141]

After heating, each structure was further run at 300 K for 1 ns with 0.1 kcal mol^{-1} $Å^{-2}$ restraints on the backbone. Each titratable carboxylate was deprotonated and the histidine was protonated, and no protonation state changes were attempted during the simulation. Next, the resulting structure was used to start 16 ns of CpHMD at pH values spanning 2 to 7 with an interval of 0.5. Only the 10 acidic residues—the aspartates, glutamates, and histidines—were titrated because HEWL is catalytically active at low pH [148] and 1AKI was solved in these conditions. I used a 2 fs time step and attempted

protonation state changes every 5 steps for all simulations in which protonation state changes were attempted. The Langevin thermostat with a collision frequency of 10 ps^{-1} was used to control the temperature, and simulations were begun with a different random seed to avoid synchronization artifacts. [149] I used the *sander* module of Amber 12 for each of these simulations.

3.3.3 Replica Exchange

All pH-REMD simulations were run with 12 equally spaced replicas at pH values spanning 2 to 7.5—identical to the pH values used for the CpHMD simulations with the addition of a replica at pH 7.5. The additional replica is necessary because the REMD implementation in *sander* requires an even number of replicas so that each replica has a partner for each exchange attempt. The structures obtained after 1 ns of CpHMD simulation for each pH were used as the starting structure for the replica exchange simulations (the structure from CpHMD run at pH 7 was used for the replica run at pH 7.5 as well).

I ran pH-REMD simulations with exchange attempt frequencies (EAFs) (*i.e.,* the frequency with which replicas attempt to swap pH values) equal to 50 ps^{-1}, 10 ps^{-1}, 5 ps^{-1}, and 0.5 ps^{-1} to assess the effect of EAF on the convergence of observable properties. This corresponds to attempting exchanges every 10, 50, 100, and 1000 steps, respectively. All pH-REMD simulations were run for 15 ns. I note here that the CpHMD simulations are equivalent to a REMD simulation with an EAF equal to 0.

I used an in-house, modified version of *sander* in which I implemented pH-REMD for these simulations and wrote in-house scripts to extract pH-based titration data from the ensemble of replica-based files. The replica at pH 7.5 was ignored in all pH-REMD analyses so the simulations could be compared fairly.

3.4 Results and Discussion

CpHMD methods must sample both protonation states and conformation states to build a thermodynamically meaningful ensemble. Here I will discuss how well CpHMD,

as implemented in Amber, [130] samples from the desired ensemble. I then analyze how pH-REMD affects protonation and conformational state sampling compared to CpHMD, and how effective these tools are for pK_a prediction.

3.4.1 Simulation Stability

CpHMD involves instantaneous changes in the charge distribution of the protein as the protonation states are changed. Therefore, it is important to verify that trajectories generated from CpHMD and pH-REMD remain stable with respect to secondary and tertiary structure during the course of the simulation.

Mongan et al. [130] showed that temperature and energy fluctuations greater than those obtained with standard MD (*i.e.,* MD simulations with static protonation states) were minimal during the course of a 1 ns simulation. Most of the energy fluctuations arise from the intrinsic response of the force field to the new charge state.

To analyze structural stability, I plotted the root mean squared deviation (RMSD) of every α-carbon in the protein with respect to the minimized crystal structure vs. time. The results from CpHMD and pH-REMD simulations are shown in Fig. 3-1 for the lowest pH, 2; the highest pH, 7; and an intermediate pH, 4.5. The RMSDs are bounded below 4 Å, suggesting that the trajectories remain stable for both CpHMD and pH-REMD during the entire simulation.

3.4.2 Accuracy of Predicted pK_as

One of the goals of any constant pH simulation method is to accurately predict pK_a values of titratable residues. The pK_a of each residue was calculated by using the *Levenberg-Marquardt* non-linear optimization method to fit the titration data at each pH to the standard Hill equation shown below:

$$f_d = \frac{1}{10^{n(pK_a - pH)} + 1} \tag{3–5}$$

where f_d is the fraction of the total simulation that the titratable residue spent in a deprotonated state.

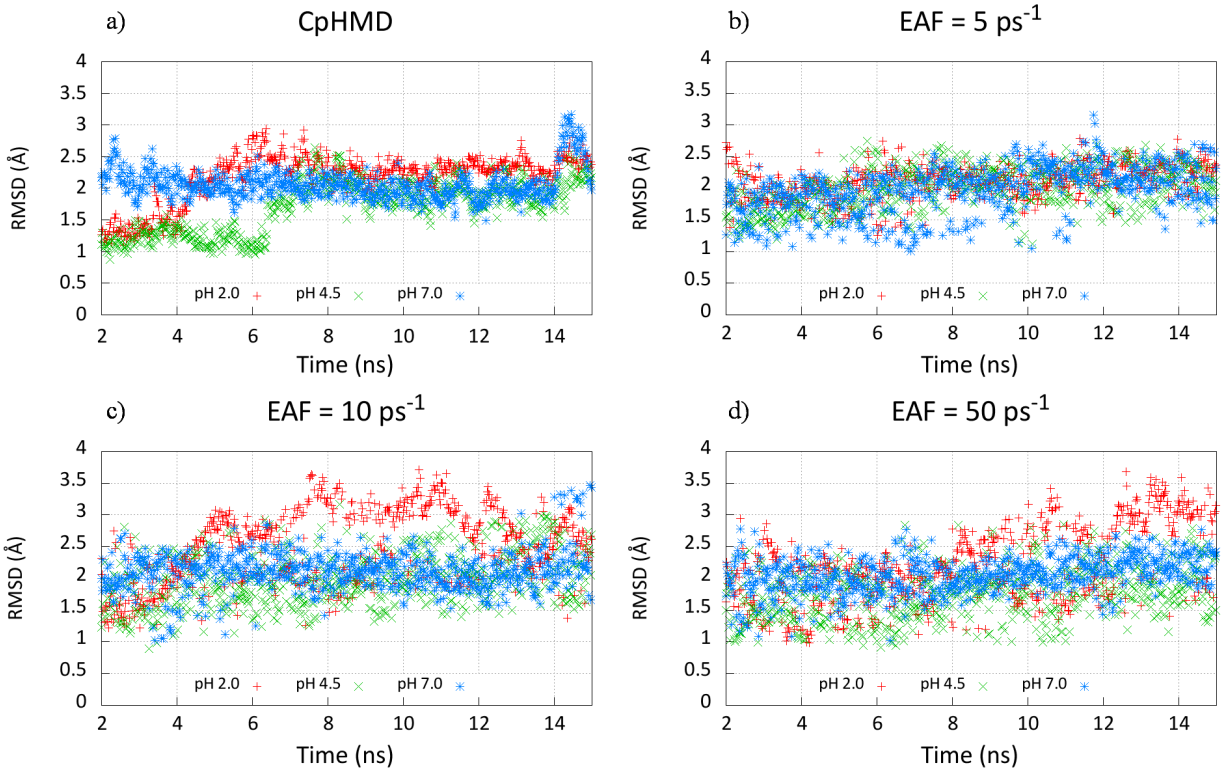

Figure 3-1. RMSD plots for CpHMD simulations (a) and pH-REMD simulations at different exchange attempt frequencies (b-d) as a function of time throughout the simulation. The first nanosecond is excluded, as described in *Methods*. The RMSD is plotted with respect to the minimized crystal structure 1AKI and are shown for one low, one medium, and one high pH simulation (2, 4.5, and 7, respectively).

Table 3-2 shows the calculated pK_a values for each titratable residue calculated from Eq. 3–5 for select exchange attempt frequencies. Hill coefficients that differ significantly from 1 imply either that the pK_a of that residue displays significant non-Henderson-Hasselbalch (non-HH) behavior, or that protonation space is poorly sampled at some pH values, depending on how well Eq. 3–5 fits the data. If Eq. 3–5 fits the data poorly, then poor protonation state sampling is at least partially responsible for the deviation of the Hill coefficient from 1. Because only the CpHMD simulations show several residues whose Hill coefficient deviates substantially from 1, we conclude that pH-REMD improves sampling from the desired ensemble.

Table 3-2. pK_a and Hill coefficients for each residue taken from each set of simulations. The pK_as and Hill coefficients (*n*) are shown for each EAF. pK_a root mean square errors (RMSEs) from the ^{13}C-NMR experimental values published by Webb et al. [136] are shown in the last row. Asp 66 was problematic because it is positioned between several Arginine residues, causing it to resist protonation. When it significantly impacts the RMSE, the RMSE for all residues *except* Asp 66 is shown in parentheses next to the total RMSE.

	CpHMD		EAF=0.5 ps^{-1}		EAF=50.0 ps^{-1}		
Residue	pK_a	*n*	pK_a	*n*	pK_a	*n*	Expt. pK_a
GLU 7	3.62	1.16	3.60	0.88	3.84	0.96	2.6 $\pm$ 0.2
HIS 15	5.96	1.05	5.74	1.09	5.90	0.97	5.5 $\pm$ 0.2
ASP 18	2.26	1.11	2.01	0.94	2.01	0.89	2.8 $\pm$ 0.3
GLU 35	5.67	3.82	5.44	1.16	4.98	0.98	6.1 $\pm$ 0.4
ASP 48	1.22	0.43	1.11	0.71	1.99	0.83	1.4 $\pm$ 0.2
ASP 52	2.69	1.16	2.37	0.96	2.30	0.75	3.6 $\pm$ 0.3
ASP 66	-18.13	0.11	-3.17	0.49	1.53	1.19	1.2 $\pm$ 0.2
ASP 87	2.52	0.81	2.61	0.81	2.66	0.91	2.2 $\pm$ 0.1
ASP 101	3.69	2.10	3.66	1.00	3.57	0.87	4.5 $\pm$ 0.1
ASP 119	2.26	0.98	2.73	1.04	2.43	0.96	3.5 $\pm$ 0.3
RMSE	6.15 (0.74)		1.56 (0.76)		0.89		—

3.4.3 Enhancing Protonation State Sampling with pH-REMD

Replica exchange methodologies are well-known to improve sampling in the desired ensemble [78, 138] as long as replicas traverse the state-space ladder regularly. If replica exchange attempts always fail, the simulation does not benefit from those attempts. In our pH-REMD simulations, exchange attempts between replicas with neighboring pH values (*i.e.,* replicas with solution pHs separated by 0.5 pH units) succeeded between 40% to 98% of the time, displaying very efficient traversal of the pH-space replica ladder. Here I will discuss the extent to which pH-REMD, with different exchange attempt frequencies (EAFs), improves protonation state sampling compared to CpHMD.

I show sample titration curves for simulations with no exchange attempts and simulations in which replica exchanges were attempted with a frequency of 50 ps^{-1}. Residues for which 'good' titration curves are obtained with CpHMD are shown in Fig.

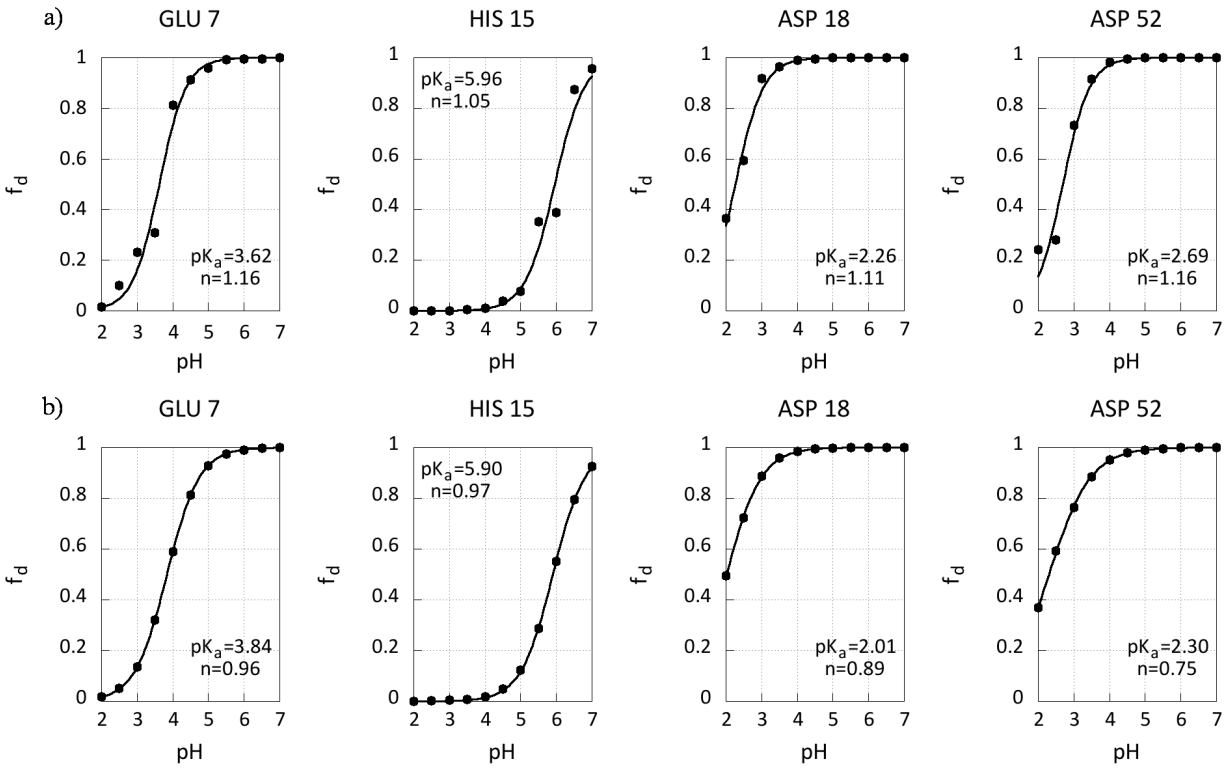

Figure 3-2. Titration curves obtained with (a) EAF=0 ps^{-1}, and (b) EAF=50 ps^{-1}. The data for these residues show the best fit to Eq. 3–5 for the CpHMD simulations.

3-2. I characterize ‘good‘ titration curves by small deviations of each point from the fitted titration curve and Hill coefficients between 0.5 and 1.5. Residues that show poor titration curves for CpHMD—characterized by large deviations of points from the fitted titration curve and/or Hill coefficients significantly shifted from 1—are shown in Fig. 3-3.

Figure 3-2 shows that even when CpHMD generates data that closely fit Eq. 3–5, using pH-REMD still improves the fit. More drastic improvement is shown in Fig. 3-3 where CpHMD performs poorly because some residues become conformationally trapped at several pHs, impacting protonation state sampling and skewing the points away from the titration curve.

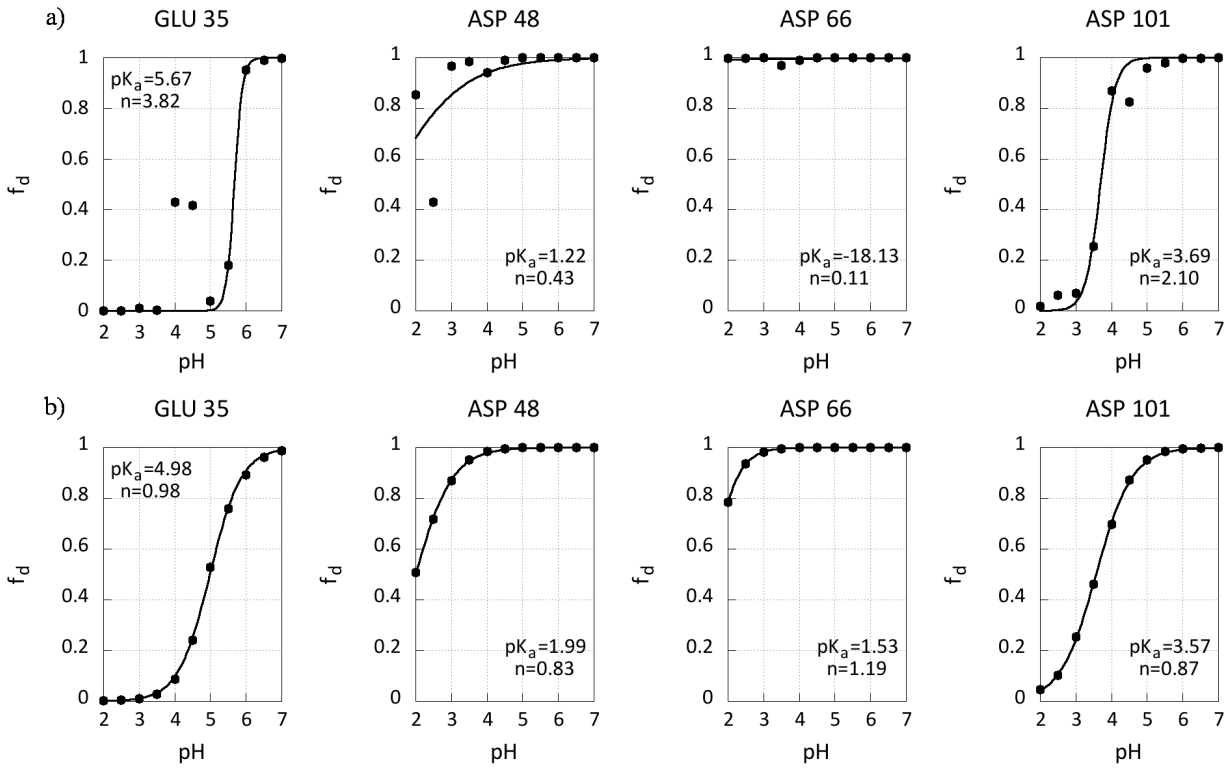

Figure 3-3. Titration curves obtained with (a) EAF=0 ps^{-1} and (b) EAF=50 ps^{-1}. The data for these residues have the poorest fit to Eq. 3–5 for the CpHMD simulations.

The quality of the fits can be quantified by measuring the deviation of each point from the fitted equation according to Eq. 3–6

$$RSS = \sum_{points} (O(x) - E(x))^2 \tag{3–6}$$

where RSS is the residual sum of squares, $O(x)$ is the actual data point, and $E(x)$ is the value of the fitted equation at that value of x. Eq. 3–6 provides an easy way to quantitatively evaluate how well the titration data from the pH-REMD simulations fit Eq. 3–5 compared to the CpHMD simulations. The results for the 8 residues plotted in Figs. 3-2 and 3-3 are shown in Table 3-3.

The improvement by using pH-REMD over conventional CpHMD, already apparent by viewing Figs. 3-2 and 3-3, is striking as measured in Table 3-3. Even when CpHMD performs well, pH-REMD results in an improvement of 2 – 3 orders of magnitude in the

Table 3-3. Value of *RSS* according to Eq. 3–6 for the 8 residues shown in Figs. 3-2 and 3-3. Larger values represent more deviation from the fitted curve, whereas a value of 0 represents a perfect fit. The 'good' titratable residues (Fig. 3-2) are the first 4 entries and the 'bad' titratable residues (Fig. 3-3) are the last 4 entries.

Residue	*RSS* (CpHMD)	*RSS* (EAF=50 ps^{-1})
GLU 7	2.7×10^{-2}	7.9×10^{-5}
HIS 15	3.8×10^{-2}	2.7×10^{-5}
ASP 18	6.3×10^{-3}	3.3×10^{-5}
ASP 52	2.2×10^{-2}	1.3×10^{-3}
GLU 35	3.6×10^{-1}	4.2×10^{-4}
ASP 48	1.7×10^{-1}	2.4×10^{-5}
ASP 66	8.2×10^{-4}	2.9×10^{-4}
ASP 101	3.4×10^{-2}	2.7×10^{-4}

RSS metric, and results in at least another order of magnitude of improvement in the cases where CpHMD performs poorly (except for Asp 66, for which CpHMD has already proven to perform poorly in this study).

By exchanging structures with other replicas, ensembles generated at each pH in pH-REMD simulations are able to escape from local minima that prevent titratable residues from accurately sampling protonation states. Because each replica is run independently, snapshots in each replica are not correlated with one another, so ensembles generated at each pH contain more uncorrelated members in simulations with more rapid EAFs (as long as replica exchange attempts succeed regularly). Therefore, pH-REMD's ability to cross free energy barriers more efficiently reduces to an entropy argument—ensembles at each pH are given more opportunities to sample different conformations.

Another way of thinking about pH-REMD simulations is to consider the entire expanded ensemble, in which the simulations sample in both conformational-space and pH-space. CpHMD simulations, on the other hand, do not sample in pH-space, as the pH remains constant throughout the entire simulation. The protonation state of each titratable residue strongly depends on both the solution pH and the protein conformation, and is coupled to other titratable residues in complicated ways. Therefore, pH-REMD

simulations can move through a much larger free energy space extended to another dimension relative to CpHMD simulations—pH-space. CpHMD simulations are unable to take advantage of lower free energy barriers in this expanded ensemble, causing them to become more easily trapped in conformations that skew predicted pK_as compared to pH-REMD simulations.

In an extreme case—Asp 66—the CpHMD simulations at low pH never visited conformations favorable to protonating. By allowing exchanges between pH replicas, ensembles at lower pH crossed into regions of phase space favorable to Asp 66 protonation.

The case of Asp 66 further demonstrates that increasing the EAF improves protonation state sampling. The pK_a prediction systematically improves as EAF is increased, and the Hill coefficient improves from 0.11 to 1.19.

3.5 Exchange Attempt Frequency and Protonation State Sampling

To analyze the effect EAF has on pK_a convergence, I divided each simulation into sections of 0.25 ns and calculated the standard deviations of the pK_a and Hill coefficient, as well as the mean Hill coefficient, by fitting Eq. 3–5 to the data obtained from each pH. The results are summarized in Table 3-4.

The average fluctuation in pK_a systematically decreases as EAF increases, in large part due to the improvement of residues that titrate poorly, namely Asp 48 and Asp 66. The large standard deviation of these two residues is evidence that the protonation state sampling does not converge on the 0.25 ns intervals that were used to generate the statistics. However, increasing the EAF leads to a systematic decrease in the fluctuations of the calculated pK_a for Asp 48 and Asp 66, because a higher EAF decreases the simulation time required to achieve pK_a convergence.

The trend of the Hill coefficient shown in Table 3-4 also shows a radical improvement in protonation state sampling with pH-REMD. While a Hill coefficient that deviates

Table 3-4. Standard deviations of pK_a (σ_{pK_a}) and Hill coefficient (σ_n) and average Hill coefficient ($\bar{n}$) calculated by dividing each simulation into sections of 0.25 ns. The pK_a and Hill coefficients are calculated for each section of the simulation by fitting fitting data from all pH replicas to Eq. 3–5 and calculating the statistics from the 60 resulting data points.

	CpHMD			EAF=0.5 ps^{-1}			EAF=50.0 ps^{-1}		
Residue	σ_{pK_a}	$\bar{n}$	σ_n	σ_{pK_a}	$\bar{n}$	σ_n	σ_{pK_a}	$\bar{n}$	σ_n
GLU 7	0.18	3.1	3.3	0.29	1.0	0.2	0.15	1.0	0.1
HIS 15	0.24	2.6	1.7	0.21	1.2	0.4	0.20	1.0	0.1
ASP 18	0.27	1.4	0.7	0.29	1.0	0.3	0.28	1.0	0.3
GLU 35	0.42	8.1	6.7	0.30	1.6	0.6	0.41	1.2	0.3
ASP 48	6.61	1.8	3.5	5.1	1.0	1.2	3.3	1.1	0.8
ASP 52	1.04	1.2	0.8	0.40	1.2	0.5	0.64	1.0	0.7
ASP 66	15	0.8	0.9	15	1.0	0.3	4.9	1.4	0.9
ASP 87	0.42	2.0	1.8	0.37	1.0	0.4	0.43	1.1	0.3
ASP 101	0.20	3.8	2.1	0.19	1.1	0.2	0.30	0.7	0.2
ASP 119	0.33	1.4	0.8	0.22	1.2	0.3	0.24	1.1	0.3
Average	2.4	3.0	2.7	2.2	1.1	0.4	1.1	1.1	0.4

significantly from 1 may indicate cooperativity between titrating residues, previous evidence suggests these residues mostly titrate independently. [130] Furthermore, because CpHMD and pH-REMD simulations converge to the same limiting ensemble, Hill coefficients that diverge significantly from 1 in CpHMD simulations but remain close to 1 in the pH-REMD simulations most likely indicate poor protonation state sampling in the CpHMD simulations.

The CpHMD simulations show an average Hill coefficient of at least 2 for half of the titratable residues, and its standard deviations are nearly as large as the Hill coefficient itself. In this case, even low EAFs result in Hill coefficients closer to 1, and their average relative standard deviation drops from 100% to 33%. Therefore, the Hill coefficients from the CpHMD simulations symbolize poor protonation state sampling rather than strong cooperativity between titrating residues.

A final metric for analyzing protonation state sampling of a particular residue is to count the number of times the protonation state changes over a specified period of time. I call these protonation state changes *transitions*, and I only consider a transition

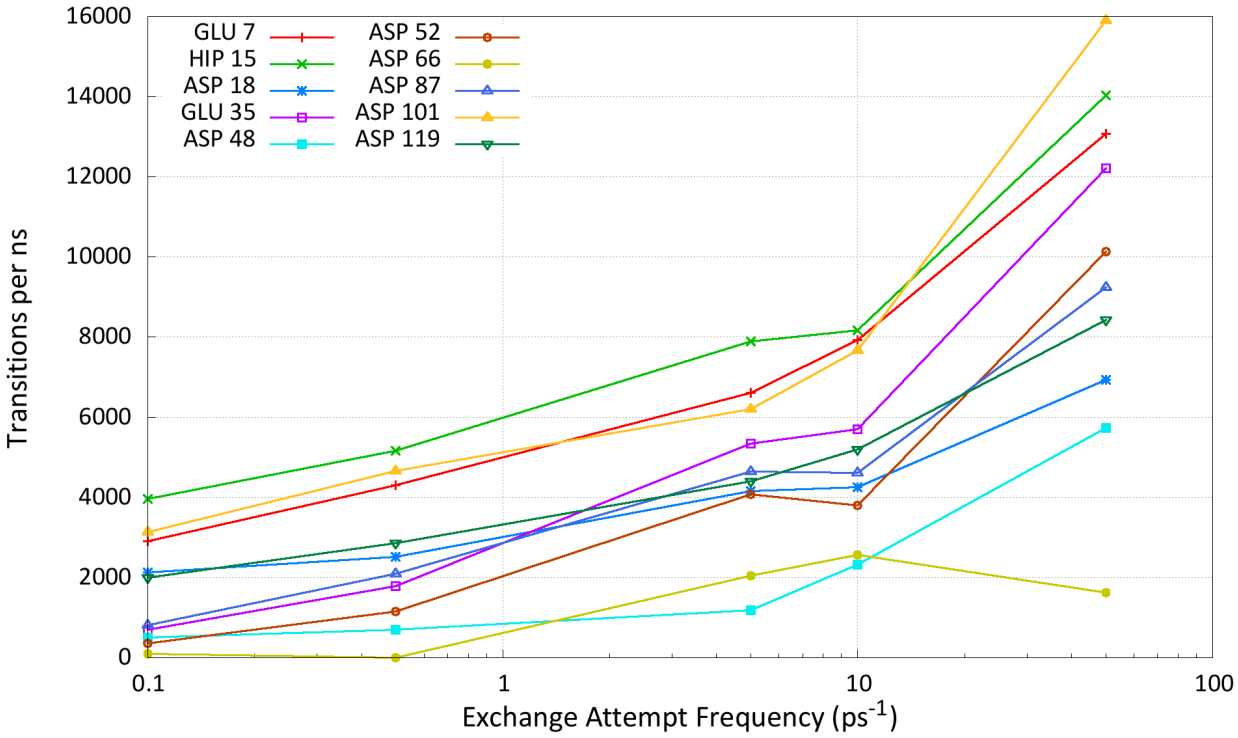

Figure 3-4. Number of protonation state transitions per ns of simulation time. A transition is counted if consecutive snapshots in the ensemble have a different number of protons for that residue. The CpHMD results are labeled with EAF=0.1 ps^{-1} to fit on the log-scale.

to have occurred if the number of protons on the titratable side-chain changed from one snapshot to the next in a given ensemble. In particular, a tautomeric change, such as a proton changing from one oxygen in a carboxylate to the other oxygen, is not counted. Fig. 3-4 shows how the number of protonation state transitions per ns of simulation, summed over every replica from pH 2 to 7, changes with EAF. In every simulation, protonation state changes are attempted every 5 steps.

Simulations that result in more transitions demonstrate enhanced protonation state sampling, since more protonation state changes occur in the same amount of time. All simulations were carried out with the same set of parameters so every ensemble generated at a given pH will converge to the same result given enough simulation time. Therefore, simulations with more transitions will obtain converged pK_a values faster.

Fig. 3-4 shows that increasing EAF dramatically increases the number of transitions despite the fact that the frequency of attempted protonation state changes is constant. This is due to the nature of the probability of accepting a *replica* exchange attempt, which is governed by Eq. 3–4. Because the success of an exchange attempt depends only on the *net* difference of titrating protons between the two replicas, it is possible for this net difference to be small, therefore the probability of accepting the exchange attempt large, even when several residues have different protonation states. Therefore, numerous protonation state changes for individual residues often accompany a successful exchange of replicas.

3.5.1 Enhancing Conformational State Sampling with pH-REMD

Because conformations and protonation states are coupled, enhanced conformational sampling from pH-REMD naturally accompanies enhanced protonation state sampling. In well-designed pH-REMD simulations (*i.e.,* pH-REMD simulations in which efficient mixing occurs in pH-space), each replica contributes structures to the ensemble at each pH, which serves to increase the number of conformations visited at each pH.

RMSD is a metric that reflects how different the sampled conformations are from a reference—in this case the original, minimized crystal structure. The histogrammed RMSD data from Fig. 3-1 is shown in Fig. 3-5 to allow easier comparison between the different simulations.

Simulations with higher EAFs traverse the replica ladder more rapidly, allowing trajectories to break out of local minima that are tied to a particular protonation state. The widening bin-widths in Fig. 3-5 show that RMSD-space is explored more thoroughly within the 15 ns timescale sampled in each simulation as EAF increases to 10 ps^{-1} (there is no noticeable difference between the 10 ps^{-1} and 50 ps^{-1} EAF).

Because each simulation is subjected to the same set of external constraints (*e.g.,* temperature, pH, solvation model, etc.), each generated ensemble should represent a subset of the theoretically complete ensemble under these external constraints.

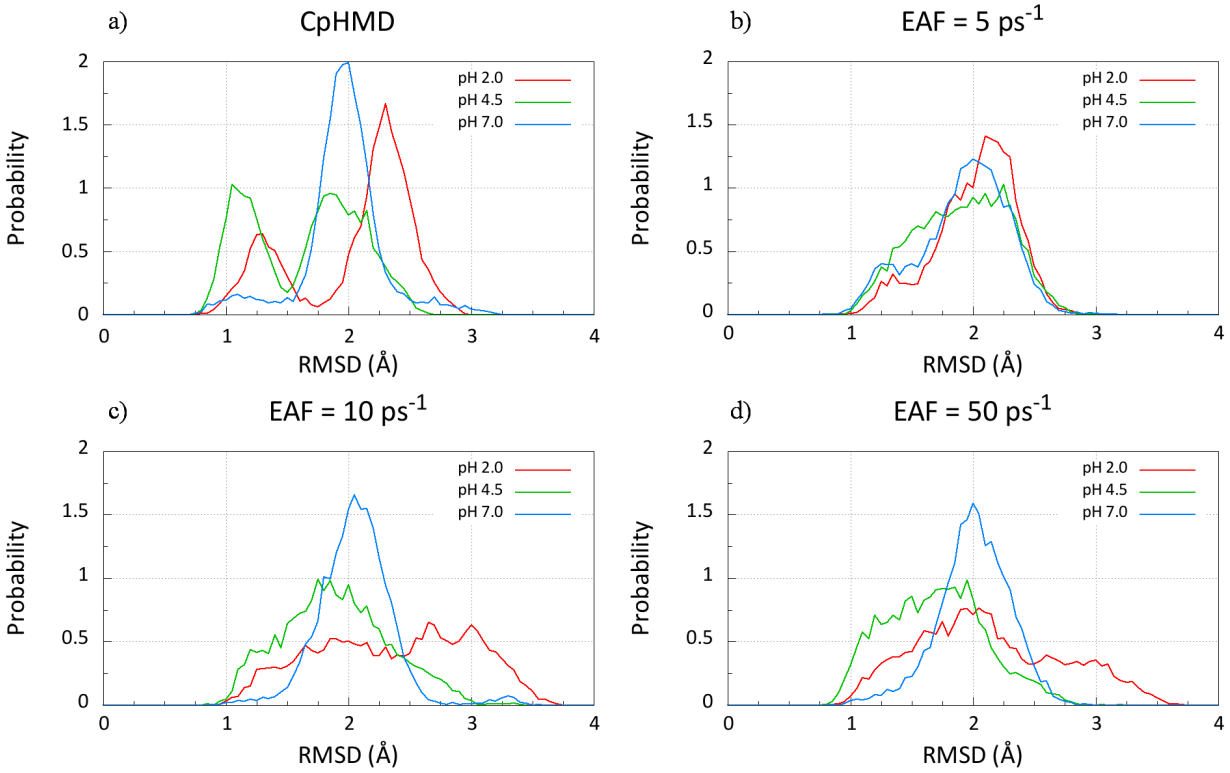

Figure 3-5. Histogrammed RMSD data for pH 2, pH 4.5, and pH 7 taken from simulations run with different EAFs.

Because these wider RMSD distribution reflects sampling of more conformations further from the starting structure, it is almost certain that these wider distributions at high EAF are thermodynamically 'better' (*i.e.,* the ensembles approximate the theoretically complete ensembles better) than their narrower counterparts in the CpHMD and 5 ps^{-1} EAF simulations. The original RMSD data, plotted in Fig. 3-1, also suggests that pH-REMD simulations converge more rapidly because those simulations display many transitions between conformations with different RMSDs.

In addition to sampling more RMSD space than CpHMD simulations, the pH-REMD simulations also converge to their final RMSD distributions much more rapidly. To quantify this measure, I used the Kullback-Leibler divergence [150, 151] (D_{KL}). The Kullback-Leibler divergence, calculated via Eq. 3–7, quantifies the similarity between two distinct probability distributions $P(i)$ and $Q(i)$.

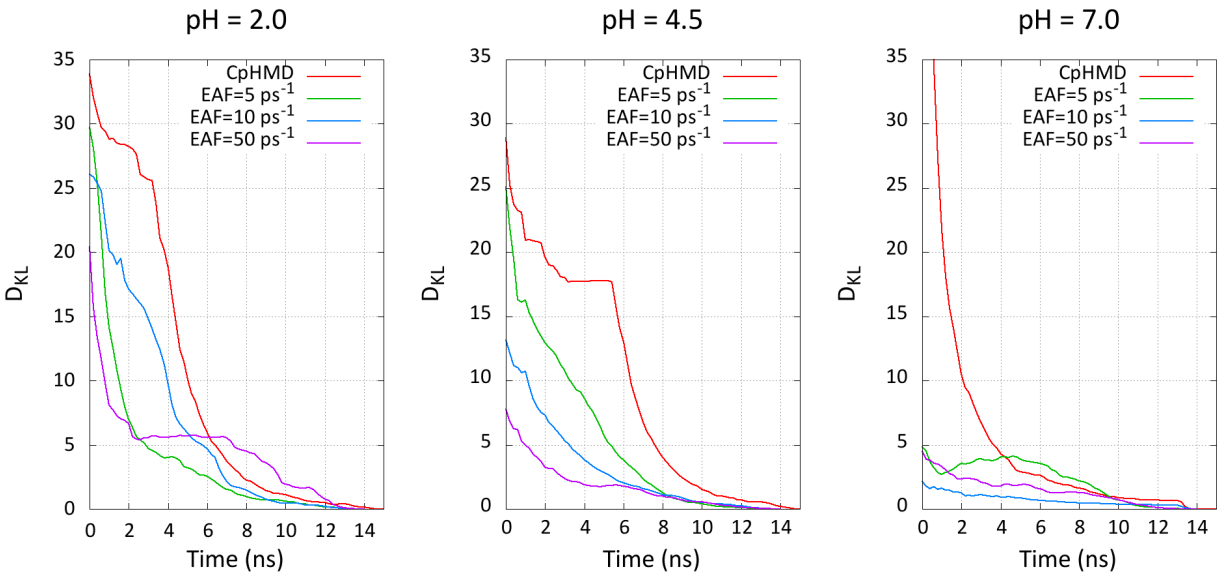

Figure 3-6. Kullback-Leibler divergence for each simulation calculated via Eq. 3–7. $P(i)$ is the RMSD histogram of the indicated simulation at time t and $Q(i)$ is the RMSD histogram of the entire indicated simulation. Values closer to zero indicate distributions of higher similarity.

$$D_{KL} = \sum_{i=1}^{N} P(i) \ln \left(\frac{P(i)}{Q(i)} \right) \tag{3–7}$$

where D_{KL} is the Kullback-Leibler divergence metric, i is the property of interest (RMSD in this case), and $P(i)$ and $Q(i)$ are two probability distribution functions on i-space. For discrete spaces (such as those obtained by histogramming data), Eq. 3–7 is represented as a sum (as shown), but becomes an integral over all of i-space for continuous probability distribution functions $P(i)$ and $Q(i)$.

Fig. 3-6 plots D_{KL} calculated from Eq. 3–7 where $P(i)$ is the RMSD distribution of each simulation at time t and $Q(i)$ is the final RMSD distribution of each simulation. As $P(i)$ and $Q(i)$ become more similar, D_{KL} tends toward zero. Therefore, the curves that approach zero more rapidly approach their final RMSD distribution in a shorter amount of time.

The pH-REMD simulations not only explore more RMSD space than the corresponding CpHMD simulations, but they characterize this larger space more rapidly as

well, since they converge to their final distribution faster than CpHMD. Furthermore, the simulations with an EAF of 10 ps^{-1} and 50 ps^{-1} typically converge faster than the one with an EAF of 5 ps^{-1}. The only exception in this case, including the pHs not shown in Fig. 3-6, is the simulation at pH 2. In general, simulations with EAF 10 ps^{-1} and 50 ps^{-1} are indistinguishable with respect to RMSD.

At pH 2, the RMSD distribution of the EAF=5 ps^{-1} simulation is much narrower than the corresponding distributions at EAF 10 ps^{-1} and 50 ps^{-1}. Therefore, it's not surprising that the D_{KL} of the 5 ps^{-1} EAF simulation converges more rapidly than the higher EAFs. In general, however, pH-REMD simulations with high EAFs sample more RMSD space more efficiently than CpHMD and simulations with low EAFs.

To probe the nature of the conformational flexibility of HEWL at different EAFs, I calculated the average atomic fluctuations for each residue from the average structure. These fluctuations provide insight into the flexible regions of the protein, giving a more fine-grained, structural analysis than RMSD does. The results, shown in Fig. 3-7, show that the same parts of HEWL are generally flexible for each simulation, but the pH-REMD simulations tend to display enhanced flexibility compared to the CpHMD simulations. Again, because each simulation samples from the same ensemble subject to the same thermodynamic constraints, this increased flexibility suggests that the simulations at high EAF converge to the true ensemble more rapidly than the CpHMD simulations and the pH-REMD simulations with a low EAF.

While the overall flexibility in pH-REMD simulations is increased with respect to the CpHMD simulations, the dynamics still reveal different behavior at different pH. In particular, the region between residues 100 and 120 shows drastically increased flexibility at pH values higher than 4 for the pH-REMD compared to the CpHMD simulations.

Furthermore, the region around residue 70, which shows heightened flexibility at pH 2 (Fig. 3-7), contains the problematic titratable residue Asp 66. This increased flexibility

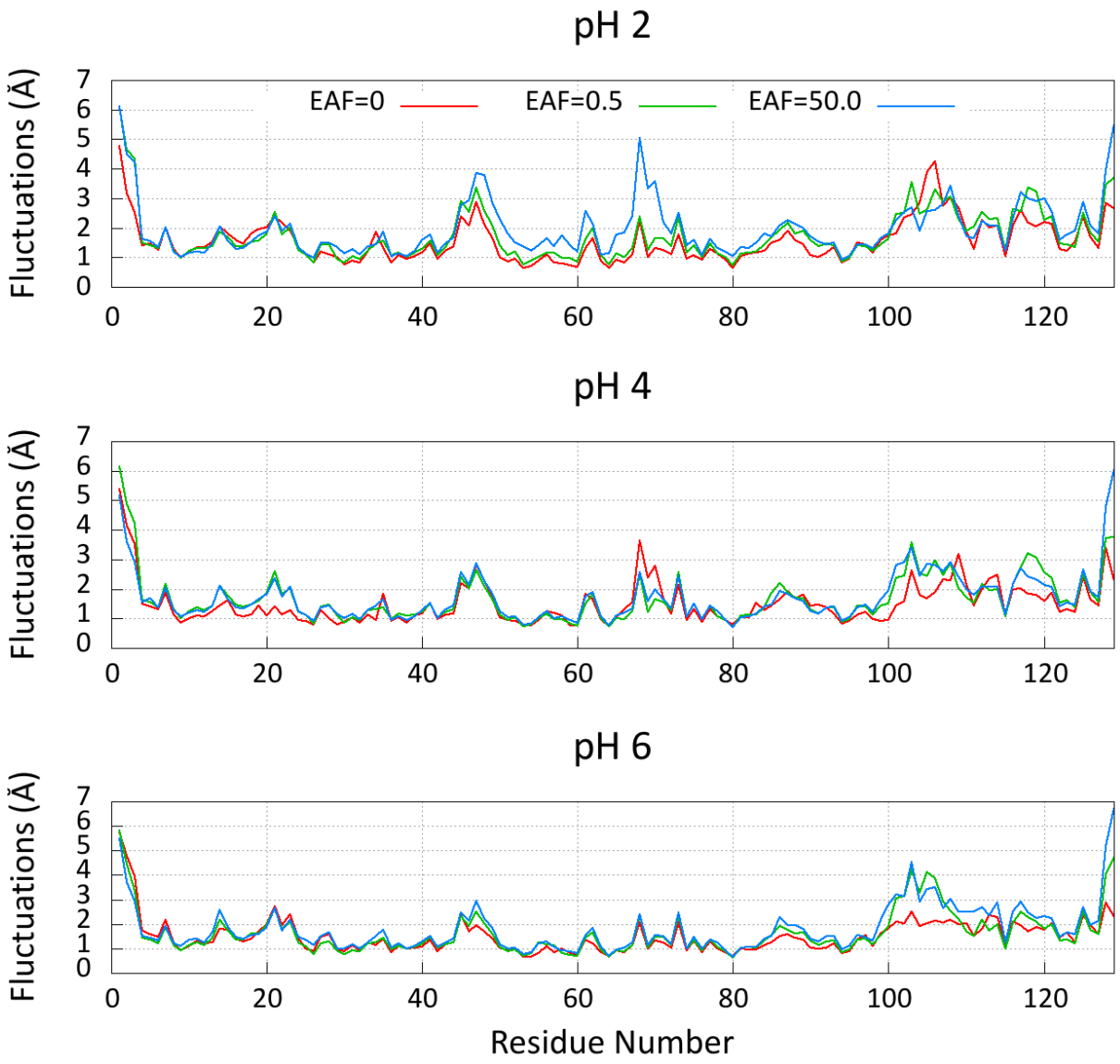

Figure 3-7. Average atomic fluctuations for each residue relative to the average structure of the ensemble. Data are shown for CpHMD, low EAF (0.5 ps^{-1}) and high EAF (50 ps^{-1}).

at high EAF is the likely explanation why Asp 66 protonates at low pH in the pH-REMD simulations but not in the CpHMD simulation where this loop is substantially less flexible.

To probe the pH-dependence of HEWL dynamics further, I plot the distributions of the distance between the carboxylate carbons of the catalytic residues Asp 52 and Glu 35 at each pH. Because Asp 52 and GLU 35 are the catalytic residues in HEWL, this behavior may have important implications in the HEWL catalytic activity profile as a function of pH.

Fig. 3-8 shows that only the simulation run at pH 5 samples conformations in which Asp 52 and Glu 35 are closely interacting for the CpHMD simulations. Furthermore, the simulation at pH 5 spends roughly 75% of its time 'stuck' in this close interaction. It is highly unlikely that this interaction is so strong at pH 5, yet is almost non-existent at pH 4.5 and 5.5. More likely, Fig. 3-8 suggests that the CpHMD simulation run at pH 5 became trapped while trajectories at other pH values were unable to enter this conformational bin within the 16 ns of simulation.

pH-REMD simulations with a high EAF easily overcome this barrier within the simulation time scale. The distributions from pH-REMD simulations with a 50 ps^{-1} EAF display more expected behavior, given that the calculated pK_as of Asp 52 and Glu 35 are 2.30 and 4.98, respectively (Table 3-2). This interaction is likely strongest when one of the carboxylates is protonated and the other is deprotonated, and it is likely weakest when both are deprotonated. Therefore, this interaction should be strongest at a pH between 2.30 and 4.98.

The Asp 52—Glu 35 interaction is the strongest at pH 2.5 and decays as the pH either increases or decreases. At pH 2.5, Asp 52 is most likely deprotonated while Glu 35 is most likely protonated. At pH 2.0, both residues are likely to be protonated, resulting in a slightly weaker interaction. However, this is still more favorable than when both residues are deprotonated, so the interaction becomes significantly weaker as the pH increases.

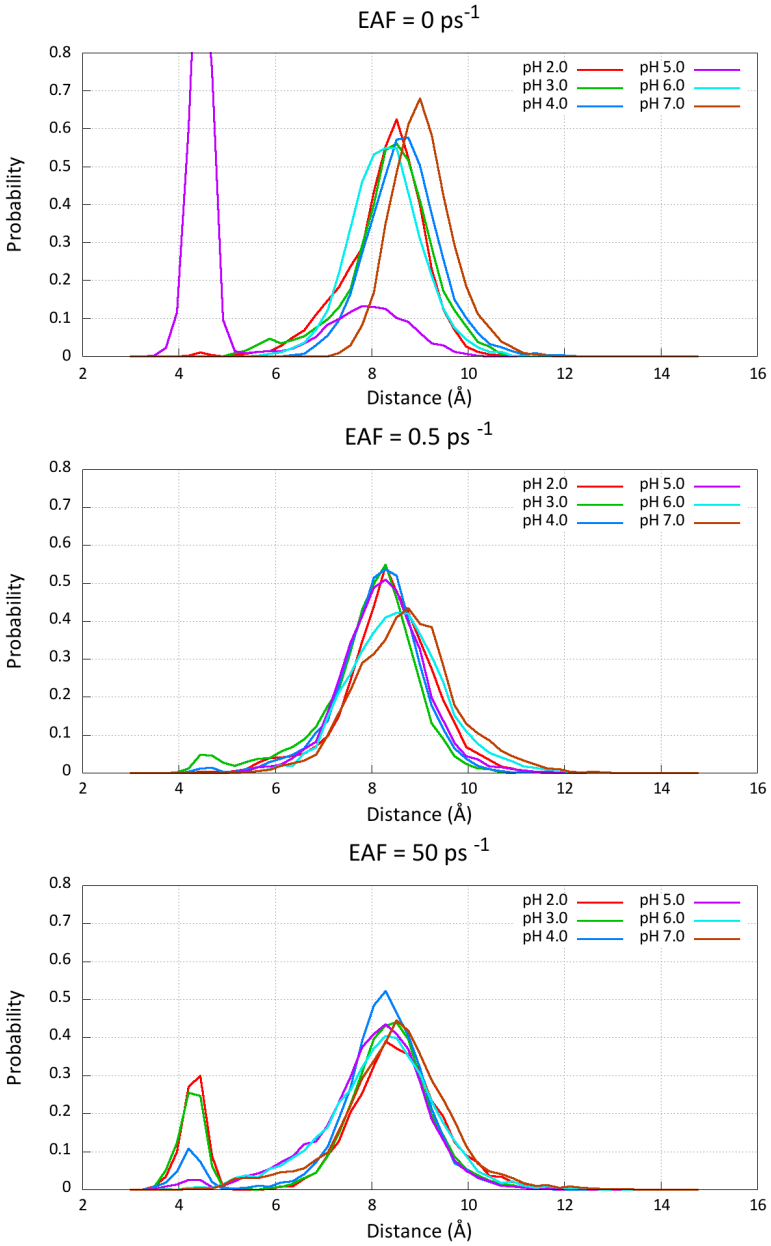

Figure 3-8. Distributions of the Asp52-C^{γ}–Glu35-C^{δ} carboxylate carbons. Asp 52 and Glu 35 are the catalytic residues of HEWL.

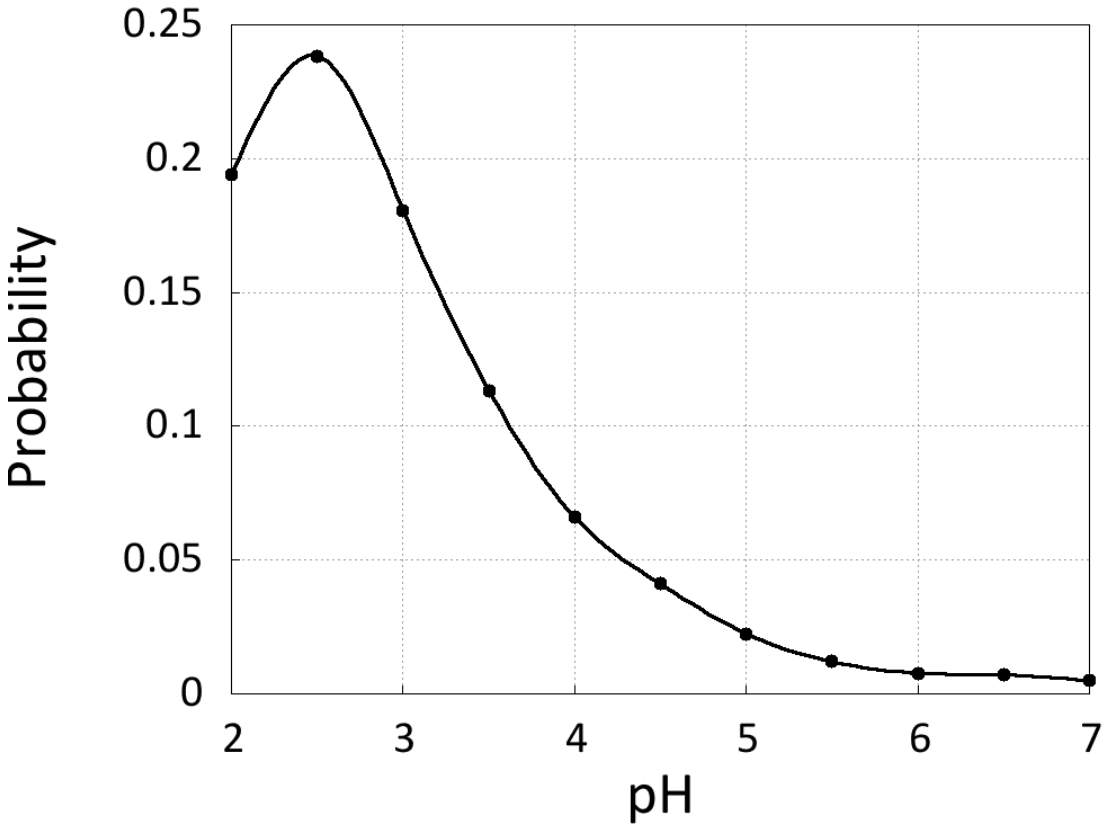

Figure 3-9. Fraction of simulation with the Glu 35 – Asp 52 distance shorter than 5 Å vs. pH.

Furthermore, tight coupling between Asp 52 and Glu 35 likely induces non-HH behavior as these residues no longer titrate independently. This explains why the Hill coefficient for Asp 52, reported in Table 3-2, is more significantly shifted away from 1—to 0.75—for the EAF of 50 ps^{-1}. Over the pH range that contains the Asp 52 inflection point, Fig. 3-8 shows that the interaction between the two active site residues is strong. Over the pH range that contains the Glu 35 inflection point, however, the interaction is weak, causing the Glu 35 titration to display nearly ideal Henderson-Hasselbalch behavior.

To better illustrate the pH-dependence of the Glu 35 – Asp 52 interaction depicted in Fig. 3-8, (at 50 ps^{-1} EAF) I integrated each of the distributions from 0 Å to 5 Å and plotted the result against pH, shown in Fig. 3-9.

Table 3-5. Average timings for CpHMD and pH-REMD simulations. CpHMD simulations used 24 processors, whereas pH-REMD simulations used 288 processors—24 per replica. All simulations were performed on NICS Keeneland. [152]

EAF (ps^{-1})	Efficiency (ns/day)
0.0	7.56
0.5	6.81
5.0	6.55
10.0	6.72
50.0	6.70

3.5.2 Scalability With Increasing Exchange Attempt Frequency

It is important when selecting a simulation protocol to consider the performance implications of each of the choices, since there is often a trade-off between computational expense and theoretical rigor. As Mongan et al. demonstrated in their work, the CpHMD method implemented in Amber is only marginally more expensive than traditional MD with constant protonation states. [130] Here, I will discuss the performance implications of increasing the EAF of pH-REMD simulations.

The computational cost of the pH-REMD simulations is the sum of the cost of the underlying CpHMD method [130] and the cost of the replica exchange attempts. The exchange success probability in pH-REMD simulations is governed by Eq. 3–4 and can be implemented so that the computational cost of each exchange attempt is negligible. The replicas of every simulation were carried out on 24 processors on NICS Keeneland [152] so the simulation efficiency, measured in terms of ns of simulation per day, can be directly compared. The average results obtained for each EAF is summarized in Table 3-5.

The decreased performance from the CpHMD simulation (EAF=0.0 in Table 3-5) to the EAF=5.0 ps^{-1} simulations arises from the fact that REMD simulations in Amber currently require each replica to perform the same number of MD steps between exchange attempts. This synchronization causes each replica to run only as fast as the slowest replica.

The pH-REMD simulations are run with 288 processors (12 replicas with 24 processors each), whereas the CpHMD simulations are run each replica independently—with only 24 processors. Therefore, the synchronization of the replicas in REMD is responsible for the 10% performance reduction between the CpHMD simulation and the pH-REMD simulation with an EAF of 0.5 ps^{-1} (attempting exchanges every 1000 integration steps).

Increasing the EAF of pH-REMD simulations from 0.5 ps^{-1} to 50.0 ps^{-1} (attempting exchanges every 10 steps) results in a 1% reduction in average simulation efficiency—a value that falls well within the fluctuations between two different simulations with the same EAF run on the same machine. Given the lack of performance degradation as EAF increases and the improved performance of simulations as EAF increases, the best EAF to use with the presented pH-REMD implementation is one where exchanges are attempted every 10 to 50 integration steps (10.0 ps^{-1} and 50.0 ps^{-1} in this study, respectively).

3.6 Conclusion

In this study, I have shown that pH-REMD effectively enhances sampling from the semi-grand canonical ensemble compared to CpHMD in the case of hen egg-white lysozyme. The titration curves generated from pH-REMD simulations are considerably less noisy than the analogous titration curves generated from CpHMD simulations, and they fit to the Hill equation much better. Furthermore, pK_as calculated from pH-REMD simulations converge faster and achieve better precision than CpHMD.

In some cases, pH-REMD can effectively cross potential energy barriers that trap residues in CpHMD simulations. In the case of the Asp 66 residue in HEWL, CpHMD simulations were unable to obtain noticeable protonation fractions even at a pH as low as 2 when started from the 1AKI crystal structure. Utilizing pH-REMD simulations with a rapid EAF, I obtained a titration curve with a Hill coefficient close to 1 and a calculated pK_a that compares well to experiment.

I have demonstrated that increasing the EAF improves sampling and convergence of several observables in this study. Asp 66 titrates more efficiently with a high EAF due to enhanced the mobility of flexible regions of the protein. Furthermore, analysis of the distance between the catalytic residues Asp 52 and Glu 35 show that increasing the EAF can provide valuable chemical insight into biologically significant pH-dependent behavior of proteins. Similarly to past work with temperature REMD, [142, 143] high EAFs give rise to more rapid convergence.

Replica exchange methodologies can be implemented efficiently to reduce the cost of each exchange attempt. In pH-REMD, the exchange success probability, governed by Eq. 3–4, involves only trivial mathematics so the cost of evaluating Eq. 3–4 is negligible. For efficient REMD implementations, like the one presented in this work, I recommend setting the EAF to at least 10 ps^{-1}, although some improvement is still seen with higher EAFs.

Chodera and Shirts [138] provide an explanation for the improved efficiency of high EAFs by relating it to Gibbs' sampling and the effect high EAF has on 'state space' sampling (pH-space in this study). In their paper, Chodera and Shirts propose enhancements to the exchange process in REMD simulations, such as exchanges between non-adjacent neighbors in 'state space' as well as attempting multiple exchanges before resuming dynamics. [138]

pH-REMD is likely to benefit by attempting exchanges between non-adjacent neighbors, since the difference in pH between replicas is small. Given the simplicity of the exchange probability equation (Eq. 3–4), the exchange success rate can be calculated between any two replicas over the course of the pH-REMD simulation. The calculated success rates show a non-negligible probability of accepting exchange attempts between replicas up to 2 pH units away from each other (*i.e.,* separated by 3 replicas).

CHAPTER 4
CONSTANT PH MOLECULAR DYNAMICS IN EXPLICIT SOLVENT

In this chapter, I present a new method for performing CpHMD simulations in explicit solvent building on the discrete protonation state model developed and explained in Ch. 3.

4.1 Introduction

Recently, Machuqueiro and Baptista raised concerns about pK_a predictions inheriting problems related to the model compound definition and inaccuracies in the underlying force field. [133] In particular, force field deficiencies have been shown to result in incorrect—even unphysical—global minima. [23, 36, 153–155] Machuqueiro and Baptista's work suggests that pK_a predictions are improved when the sampled conformations more closely resemble the true, experimental ensemble. Therefore, the role of GB in evaluating the dynamics of biomolecules may be problematic in certain situations where GB is known to fail, such as the over-stabilization of salt bridges. [156, 157] Indeed, for highly charged systems, like nucleic acids, a more accurate treatment of electrostatic interactions is required to build a sensible ensemble.

While most of the physics-based methods designed to describe a biomolecular system at constant pH use an implicit solvent representation of the solvent, several CpHMD methods have been extended to sample, at least conformations, in explicit solvent with both the discrete [126] and continuous [86, 158, 159] protonation models. The methods proposed by Baptista et al. [126] and Wallace and Shen [86] use an implicit solvent potential to sample protonation states while the methods developed by Goh et al. [159] and Donnini et al. [158] perform λ-dynamics on the titration coordinate directly in explicit solvent. A more recent approach by Wallace and Shen uses a λ-dynamics approach in pure explicit solvent, but adds a counter-ion whose charge is changed simultaneously with a titratable residue in order to maintain charge neutrality in the unit cell. [160]

Discrete protonation methods use molecular dynamics to propagate the spatial coordinates, while occasionally interrupting the dynamics to attempt change(s) to the protonation states of the titratable residues using a Metropolis Monte Carlo criteria. The CpHMD method implemented in Amber [130] (and later implemented in CHARMM [87]) performs MD in GB solvent, periodically attempting to change the protonation state of one or two interacting residues roughly every 10 fs. [130] In the stochastic titration method described by Baptista et al., dynamics is run in explicit solvent for 2 ps [161], after which a cycle of protonation state change attempts are evaluated using the Poisson-Boltzmann (PB) equation to treat solvation effects for every titratable residue and interacting titratable residue pair. About 40,000 full cycles are attempted each time protonation state changes are attempted. [162] Afterwards, the solute is held fixed while MD is propagated on the solvent to reorganize the solvent distribution to the new set of protonation states.

Implicit solvent models—in this case GB and PB—average over all solvent degrees of freedom, allowing such approaches to instantly incorporate the solvent relaxation to discrete protonation state changes. Therefore, MC moves in which a protonation state change is attempted have a reasonable probability of succeeding when the solution pH is set close to the intrinsic pK_a of the titratable group. When explicit solvent molecules are present, however, the solvent orientation around any solvent-exposed, titratable residue will oppose every proposed protonation state change. On average, the solvent distribution tends to resist protonation state changes by imposing a barrier on the order of 100 kcal/mol as estimated by measurements in our lab and in others', [86] making titration with discrete protonation states difficult directly in explicit solvent.

In this study, I present a new method of performing CpHMD simulations in explicit solvent using discrete protonation states. This method is similar in some respects to that proposed by Baptista et al., [126] and I evaluate its performance on the model compounds, a pentapeptide, and two proteins: Ribonuclease A (RNase A) and the hen

egg white lysozyme (HEWL). To enhance the sampling capabilities of this new CpHMD method, I used replica exchange in the pH-dimension (pH-REMD), whose theory and performance were discussed previously in the context of implicit solvent calculations. [87, 88] This chapter is organized as follows: I will first describe the method and its implementation in the Theory and Methods section, followed by a description of the calculations I performed in the Calculation Details section. Afterwards, I will evaluate its performance as well as sensitivity to the method's tunable parameters in the Results and Discussion section.

4.2 Theory and Methods

In this section, I will discuss the details of our proposed method and highlight how it differs from the approach used by Baptista et al. [126] The theoretical foundation of our CpHMD method is described in detail, as well as the pH-REMD method I used in our simulations.

4.2.1 Conformational and Protonation State Sampling

In CpHMD, structures are sampled from the semi-grand canonical ensemble, whose probability distribution function is given by

$$\rho(q, p, \mathbf{n}) = \frac{\exp\left(\beta\mu^{H^+} n - \beta\hat{H}(q, p, \mathbf{n})\right)}{\sum_{\mathbf{n}'} \int dp' dq' \exp\left(\beta\mu_{H^+} n' - \beta\hat{H}(p', q', \mathbf{n}')\right)} \tag{4–1}$$

where $\beta = 1/k_B T$, μ_{H^+} is the chemical potential of hydronium (directly related to the solution pH), q is the generalized coordinates of the system particles, p is the conjugate momenta, and n is the total number of titratable protons present in that state. When bold, $\mathbf{n}$ refers to the protonation state vector, specifying not only the total number of protons present, but on which titratable sites those protons are located. The denominator in Eq. 4–1 is the *partition function* of the semi-grand canonical ensemble.

To sample from the probability function ρ in Eq. 4–1, discrete protonation state methods use MD with a fixed set of protonation states to sample coordinates and momenta coupled with a MC-based protonation state sampling at fixed conformations

throughout the trajectory. This is equivalent to separating ρ into conditional probabilities. In Eq. 4–2, $\rho(q, p|\mathbf{n})$ is sampled via MD and $\rho(\mathbf{n}|q, p)$ is sampled via the MC protonation state changes.

$$\int\int\int \rho(q, p, \mathbf{n}) dq\ dp\ d\mathbf{n} = \int\int \rho(q, p|\mathbf{n}) dq\ dp \int \rho(\mathbf{n}|q, p) d\mathbf{n} \tag{4–2}$$

In explicit solvent, $\rho(\mathbf{n}|q, p)$ is difficult to sample directly, since the solvent orientation is set according to the current protonation state vector. Following the arguments of Baptista et al., the system coordinates (and momenta) can be separated into solute and solvent degrees of freedom. [126] The protonation state sampling is then performed according to the conditional probability

$$\rho' = \rho(p_{solvent}, q_{solvent}, \mathbf{n}|p_{solute}, q_{solute}) \tag{4–3}$$

where $q_{solvent}$ and $p_{solvent}$ are relaxed solvent distributions of positions and momenta around the protonation state vector, **n**. [126] Implicit solvent models account for solvent reorganization instantantly, so the distribution function ρ' in Eq. 4–3 can be approximated using continuum models, such as the PB or GB equations, thereby avoiding the otherwise costly solvent relaxation calculation associated with each attempted protonation state change.

Contrary to the stochastic titration method that calculated solvation free energies using the PB equation to evaluate protonation state changes, [126] I chose to use the GB implicit solvent model for three main reasons. First, *sander* has numerous GB models readily available, [49, 50, 163–165] allowing us to use the existing code to evaluate protonation state change attempts. Second, results from the original GB-based CpHMD implementation by Mongan et al., and from a number of previous studies using the method, have been promising. [88, 130, 135, 166] Furthermore, GB was shown to be effective when used in a hybrid solvent method with continuous protonation states [86] and is less computationally expensive than PB, and its calculation is more easily

split up among many processors, allowing longer simulations to be performed in the same amount of time.

4.2.2 Explicit Solvent CpHMD Workflow

The process of the CpHMD method presented here can be divided into three repeating steps, summarized in the workflow diagram in Fig. 4-1. This workflow is very similar to the one presented in Baptista et al. (Fig. 2), [126] although the nature of the MC protonation state move is different.

In the proposed method standard MD in explicit solvent is carried out using a constant set of protonation states (an initial set must be provided at the start of the simulation). At some point the MD is stopped, the solvent (including any non-structural ions) are stripped, the potential is switched to an available GB model, and a set of N protonation state changes are attempted where N is the number of titratable residues. While in principle the MD can be stopped randomly with a predetermined probability at any step, in this iteration of our proposed method MD is run for a set time interval, τ_{MD}, similar to the stochastic titration method. [126]

After the MD is halted and the solvent stripped, protonation state changes are proposed for each titratable residue once, in random order, choosing from the available protonation states of that residue excluding the currently occupied state. The electrostatic energy difference between the proposed and current protonation states, as well as the MC decision regarding whether or not to accept the proposed state, are calculated the same way as in the original GB implementation. [130] If the protonation state change is accepted, the 'current' state is appropriately updated, and the next residue, chosen at random without replacement, is titrated with this new state.

For each residue that is titrated, there is a 25% chance that a so-called multi-site titration will occur with a neighboring residue—that is, the proposed change will involve changes to the protonation state of both neighbors. Two titratable residues are considered 'neighbors' if their two titrating hydrogen atoms are within 2 Å from

each other. If either residue has more than one titrating proton, the two residues are neighbors if the minimum distance between any pair of titrating hydrogens meets the cutoff. Like the single-residue change attempts, if this protonation state change attempt fails, the system remains in its original protonation states for both residues.

Including multi-site protonation state jumps is important for systems that have closely-interacting titratable residues. Without these multi-site moves, proton transfers between adjacent titratable residues involved in a hydrogen bond would never occur due to the high penalty of disrupting the interaction by adding another proton or removing the proton involved in the hydrogen bond. This feature was actually present in the initial implementation, and while no mention of it was made in the original paper, a small note was made in the Users' manual. [130]

If any of the protonation state change attempts were accepted, the solute is frozen while MD is performed on the solvent (and any ions) to relax the solvent distribution around the new protonation states. The length of this relaxation is a tunable parameter of the method, which I will call τ_{rlx}. When the relaxation time is infinitely long, this process becomes exact. After the relaxation is complete, the velocities of the solute atoms are restored to their values prior to the relaxation and the standard dynamics is continued.

4.2.3 pH-based Replica Exchange

The underlying theory behind replica exchange in pH-space with MD run in explicit solvent is unchanged from the version I implemented in implicit solvent, as described in Ch. 3. [87, 88] Replicas are ordered by their solution pH parameter, and adjacent replicas attempt to exchange their pH periodically throughout the MD simulations.

The probability of accepting these replica exchange attempts, given by Eq. 3–4, depends only on the difference in the number of titrating protons present in each replica and their respective difference in pH. [88] As a result, the number of replicas necessary to obtain efficient mixing in pH-space does not increase as explicit solvent is added.

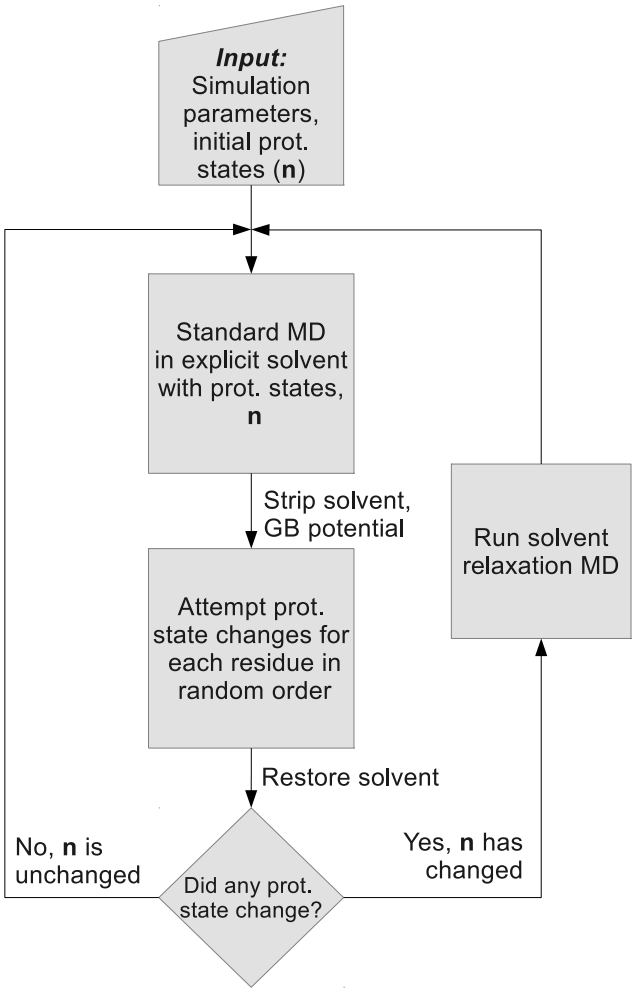

Figure 4-1. Workflow of the proposed discrete protonation CpHMD method in explicit solvent. Following the standard MD, the solvent, including all non-structural ions (as determined by user-input), are stripped and the protonation state changes are evaluated in a GB potential. After that, the solvent and the original settings are restored for the remaining steps.

This, coupled with the improved sampling found with implicit solvent simulations, [88] makes pH-REMD an effective tool for explicit solvent CpHMD.

4.3 Calculation Details

To evaluate the performance of the proposed method, I applied it to the amino acid model compounds, a small pentapeptide (ACFCA), and two proteins commonly used in pK_a calculation studies—ribonuclease A (RNase A) and the hen egg-white lysozyme (HEWL).

4.3.1 Model Compounds

Absolute pK_as are very difficult to calculate in solution—they are impossible using classical force fields. As a result, every physics-based CpHMD method uses the idea of a *model compound* whose experimental pK_a is easy to measure with a high level of accuracy. An empirical parameter—the *reference energy*—is then added so that CpHMD reproduces the experimental pK_as of these model compounds. In this way, CpHMD computes the pK_a *shift* of a titratable residue in a biomolecule with respect to the isolated model compound in solution via the thermodynamic cycle shown in Fig. 4-2

The model compounds have the sequence *ACE-X-NME*, where *ACE* is a neutral acetyl capping residue, *X* is a titratable residue, and *NME* is a neutral methyl amine capping residue. [130] The available titratable residues in Amber are aspartate (AS4), glutamate (GL4), histidine (HIP), lysine (LYS), tyrosine (TYR), and cysteine (CYS), which are all defined as described by Mongan et al. [130] A 10 Å TIP3P [167] solvent buffer was added in a truncated octahedron around the model compound. The aspartate model compound was also simulated with larger box sizes—15 Å and 20 Å buffers—to determine if it had any effect on the calculated pK_a.

After the system topologies were generated, each system was minimized using 100 steps of steepest-descent minimization followed by 900 steps of conjugate gradient

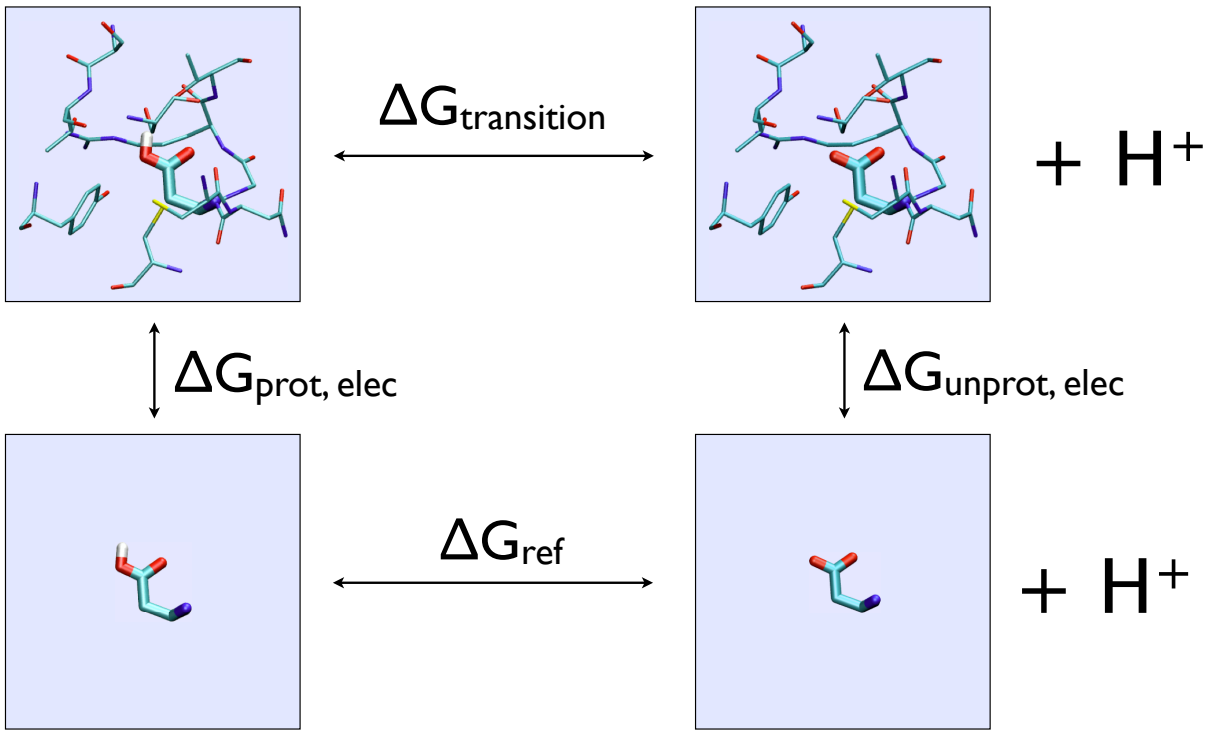

Figure 4-2. Thermodynamic cycle used to evaluate protonation state changes in CpHMD simulations.

minimization. They were then heated at constant pressure, varying the target temperature linearly from 50 K to 300 K over 200 ps. The solvated model compounds were then simulated, free of restraints, for 2 ns at constant temperature and pressure.

Each model compound system was simulated at constant pH and volume for 2 ns, setting $\tau_{rlx} = 200\ fs$. Each system was simulated with pH-REMD using six replicas with the solution pH set to $pK_a \pm 0.1$, $pK_a \pm 0.2$, and $pK_a \pm 1.2$. To evaluate the effect of the solvent relaxation time, the cysteine and aspartate model compounds were run with τ_{rlx} set to 10 fs, 40 fs, 100 fs, 200 fs, and 2 ps.

Table 4-1. Model compound pK_a values and reference energies. Because only differences in reference energies are used in the MC calculation, one state is arbitrarily assigned a reference energy of 0 (the deprotonated states of AS4 and GL4, the protonated state of CYS, and the double-protonated state of HIP). The listed reference energies are calculated with respect to the arbitrary zero value. pK_a^{calc} are pK_as calculated with the proposed method using the GB reference energy (ΔG_{ref}^{GB}). Adjusted reference energies for explicit solvent CpHMD are labeled as ΔG_{ref}^{*}. All energies are in *kcal/mol*

Residue	Reference pK_a	ΔG_{ref}^{GB}	pK_a^{calc}	ΔG_{ref}^{*}
Aspartate	4.0	32.38803	4.20	32.11310
Glutamate	4.4	14.45421	4.75	13.97287
Histidine-δ	6.5	-16.34790	6.55	-16.47559
Histidine-ϵ	7.1	-11.77701	7.19	-11.71159
Cysteine	8.5	89.15114	8.40	89.28861

The results from these simulations were used to adjust the original reference energies to reproduce the correct model compound pK_as in explicit solvent. This adjustment can be calculated directly from the pK_a shift relative to experiment when using the original reference energy. To calculate the required adjustment, the reference energy is broken into two components—a TI-based component which is equal to the free energy difference between the two states and a pK_a-based component that offsets the energies of the protonation states to the necessary value required to obtain the correct pK_a for the model compound. This is shown in Eq. 4–4. A summary of the required changes is given in Table 4-1.

$$\Delta G_{ref} = \Delta G_{TI} + kT \ln 10 \, \Delta N \, pK_{a,model} \tag{4–4}$$

4.3.2 ACFCA

A pentapeptide with the sequence *Ala-Cys-Phe-Cys-Ala* (ACFCA) was solvated with a 15 Å buffer of TIP3P molecules around the solute in a truncated octahedron.

The system was minimized using 100 steps of steepest descent minimization followed by 900 steps of conjugate gradient. The minimized structure was heated by varying the target temperature linearly from 50 K to 300 K over 200 ps at constant

pressure. The resulting structure was then simulated at 300 K at constant temperature and pressure to stabilize the system density and equilibrate the solvent distribution around the small peptide.

Simulations at six different pH values—7.1, 8.1, 8.3, 8.7, 8.9, and 9.9—were performed beginning from the resulting 'equilibrated' struture. These pH values were chosen because the pK_a of the cysteine model compound is 8.5, so the two cysteines of ACFCA were expected to titrate in this pH range. To demonstrate the effect that pH-REMD had on the titration of ACFCA, two sets of simulations were run—CpHMD with no exchanges and pH-REMD—with each replica being run for 2 ns. The relaxation dynamics following successful exchange attempts were run for 100 fs.

4.3.3 Proteins: HEWL and RNase A

Two different starting structures were selected from the PDB for both HEWL and RNase A. The structures solved in PDB codes 1AKI [147] and 4LYT [168] were used as starting structures for the HEWL calculations, while those from PDB codes 1KF5 [169] and 7RSA [170] were used for the RNase A calculations.

All PDB files were prepared by removing all solvent and keeping only the first conformation present for each residue if more than one was present. All aspartate residues were renamed AS4, all glutamate residues were renamed GL4, and all histidine residues were renamed HIP in preparation for CpHMD and pH-REMD simulations. By default, aspartate and glutamate are in their deprotonated state while histidine is in its double-protonated state.

All disulfide bonds were added manually in *tleap* and each structure was solvated with a 10 Å TIP3P water buffer surrounding the protein in a truncated octahedron. To test the effect of ions in the explicit solvent titrations, a second set of systems was set up for each starting structure by adding several ions randomly distributed around the unit cell. I added 14 chloride ions and 6 sodium ions to the RNase A starting structures, and 15 chloride ions and 6 sodium ions to the HEWL starting structures, which neutralized

all systems in their initial protonation states. The added counter-ions resulted in salt concentrations ranging from 0.17 M to 0.18 M for all four simulations following the constant pressure equilibration.

All structures were minimized using 1000 steps of steepest descent minimization followed by 4000 steps of conjugate gradient, with 10 kcal/mol positional restraints applied to the backbone. The structures were then heated at constant volume, varying the target temperature linearly from 10 K to 300 K over 400 ps. The heated structures were then equilibrated for 2 ns at constant temperature and pressure.

Following the setup stages of the simulations, each structure was simulated using pH-REMD simulations for 20 ns, with 8 replicas spanning integer pH values from 1 to 8 to characterize the acidic-range titration behavior of the systems.

4.3.4 Simulation Details

All systems were parametrized using the Amber ff10 force field, which is equivalent to the Amber ff99SB force field for proteins. [23] The *tleap* program of the AmberTools 12 program suite was used to build the model compound and ACFCA molecules, to add hydrogen atoms to RNase A and HEWL, and to solvate each system.

All simulations were performed using the *sander* module of a development version of Amber 12. [141] Langevin dynamics was used in every simulation to maintain constant temperature with collision frequencies varying from 1 ps^{-1} to 5 ps^{-1}, and the random seed was set from the computer clock to avoid synchronization artifacts. [149, 171] The Berendsen barostat was used to maintain constant pressure for the equilibration dynamics with a coupling constant of 1 ps^{-1}.

All molecular dynamics, including the solvent relaxation dynamics, are run with a 2 fs time step, constraining bonds containing hydrogen using SHAKE. [16, 18] Replica exchange attempts between adjacent replicas were made every 200 fs for all pH-REMD simulations. Protonation state changes were attempted every 200 fs for all constant pH simulations.

Long-range electrostatic interactions were treated with the particle-mesh Ewald method [172, 173] using a direct-space and van der Waals cutoff of 8 Å. Defaults were used for the remaining Ewald parameters. The GB model proposed by Onufriev et al., specified by the parameter *igb=2* in *sander*, [49] was used to evaluate the protonation state change attempts to be consistent with the original implementation in implicit solvent. [130]

4.4 Results and Discussion

Here I will analyze the performance of our proposed CpHMD and pH-REMD methods as well as ways to optimize its overall performance. I will start by discussing the behavior of the model compounds when the size of the unit cell and the length of the relaxation dynamics (τ_{rlx}) is varied. I will follow this discussion with a similar analysis on a slightly larger system—ACFCA—before discussing the application of our proposed method to real proteins.

4.4.1 Box Size Effects

To study the effect that the unit cell size has on titrations in our proposed method, I prepared three systems of the aspartate model compound with different TIP3P solvent buffers surrounding it. I prepared systems with a 10 Å, 15 Å, and 20 Å TIP3P solvent buffer around the model aspartate.

Because protonation state sampling takes place in GB solvent without periodic boundary conditions, any effect of the box size on calculated pK_as will arise due to alterations of the structural ensembles induced by artifacts from the box size. The calculated pK_as of the three systems were 4.02 ± 0.07, 4.05 ± 0.08, and 4.12 ± 0.07 for the 10 Å, 15 Å, and 20 Å solvent buffer systems, respectively. To estimate the uncertainties I divided each simulation into 100 ps chunks and took the standard deviation of the set of 20 pK_as calculated from those segments.

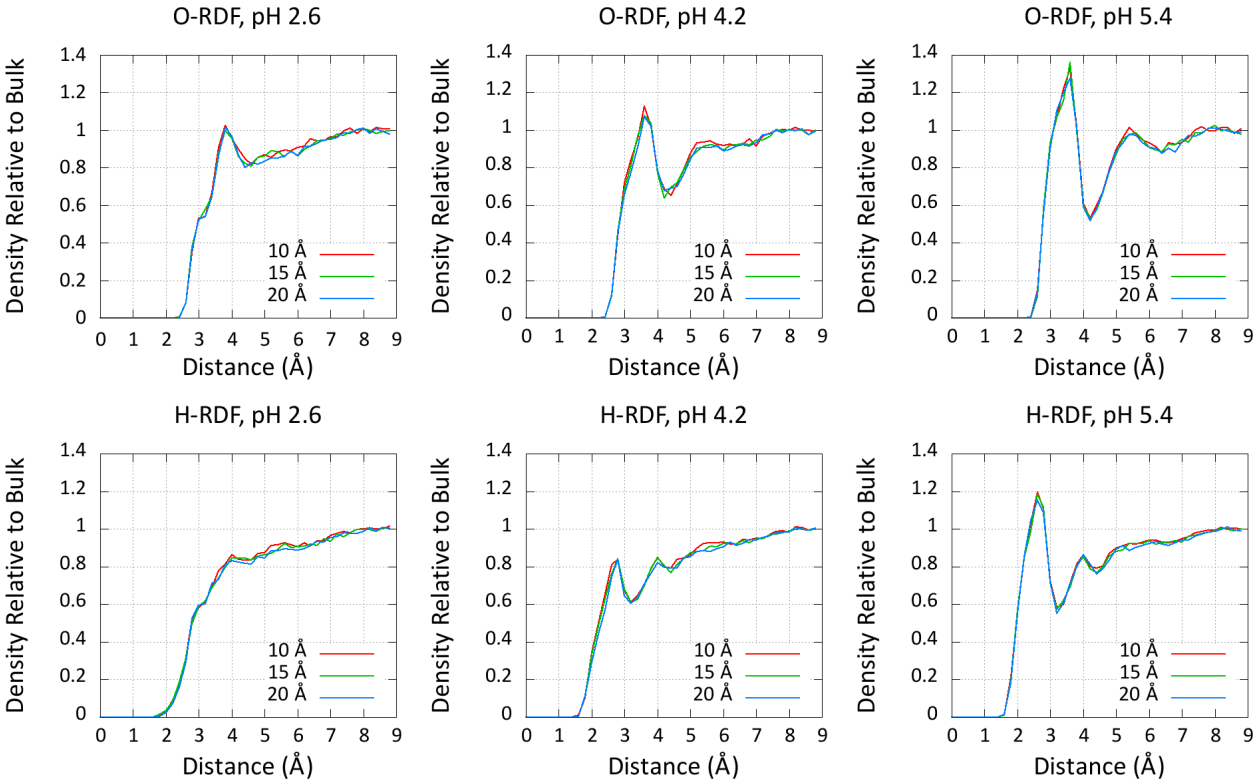

Figure 4-3. Radial distribution functions (RDFs) of solvent oxygen atoms (O) and hydrogen atoms (H) with different unit cell sizes. The shown measurements—10, 15, and 20 Å—represent the size of the solvent buffer surrounding the solute. RDF plots for three different pHs are shown, highlighting the pH dependence of the solvent structure around the carboxylate of the aspartate model compound and its invariance to box size.

To further demonstrate the insensitivity of box size to pH-REMD titrations, I plotted the solvent radial distribution functions (RDFs) around the center of mass of the carboxylate functional group in three different solution pH environments, shown in Fig. 4-3. The insensitivity of the pK_a and solvent structure with respect to the model compound provides strong evidence that no undue care is necessary when choosing the size of the solvent buffer for these types of simulations.

4.4.2 τ_{rlx} Effects

An important approximation in the proposed method is that the protonation state sampling ρ' from Eq. 4–3 can be replaced using an implicit solvent model followed by

relaxation MD to generate the relaxed solvent positions and momenta. The question then becomes how long this relaxation dynamics should be run.

To address this, 2 ns of constant protonation molecular dynamics simulations were run on the model cysteine compound in both protonation states—protonated and deprotonated—after the same minimization and heating protocols were used as for the other model compound simulations. The protonation state was then swapped for the final structures of both simulations, and MD was performed while constraining the solute position for 20 ns, equivalent to the relaxation dynamics protocol in our proposed method.

The optimum value for τ_{rlx} is the time after which the energy of the relaxation trajectory stabilizes and the simulation loses all memory of its initial configuration. To be truly equivalent to having been chosen at random, the final, relaxed solvent distribution must be completely uncorrelated from the initial distribution at the time the protonation state was changed.

To probe the necessary time scales for these relaxation dynamics, the energy of each snapshot in the relaxation trajectory is plotted alongside the autocorrelation function of that energy in Fig. 4-4 to clearly demonstrate the 'appropriate' value of τ_{rlx} for this model system.

I chose the cysteine model compound for this test for two reasons. First, the model compounds are fully solvent-exposed due to their small size, which results in a worst-case scenario in terms of the number of water molecules that must be reorganized during the relaxation dynamics. The optimum τ_{rlx} value for model compounds is expected to be an upper-bound on the values required for larger systems. Secondly, cysteine is the smallest and simplest of the titratable amino acids, eliminating potential complications from tautomeric states compared to aspartate, glutamate, and histidine.

The relaxation energies plotted in Fig. 4-4 begin to stabilize after 4 to 6 ps of relaxation dynamics, and the autocorrelation function indicates that the relaxation

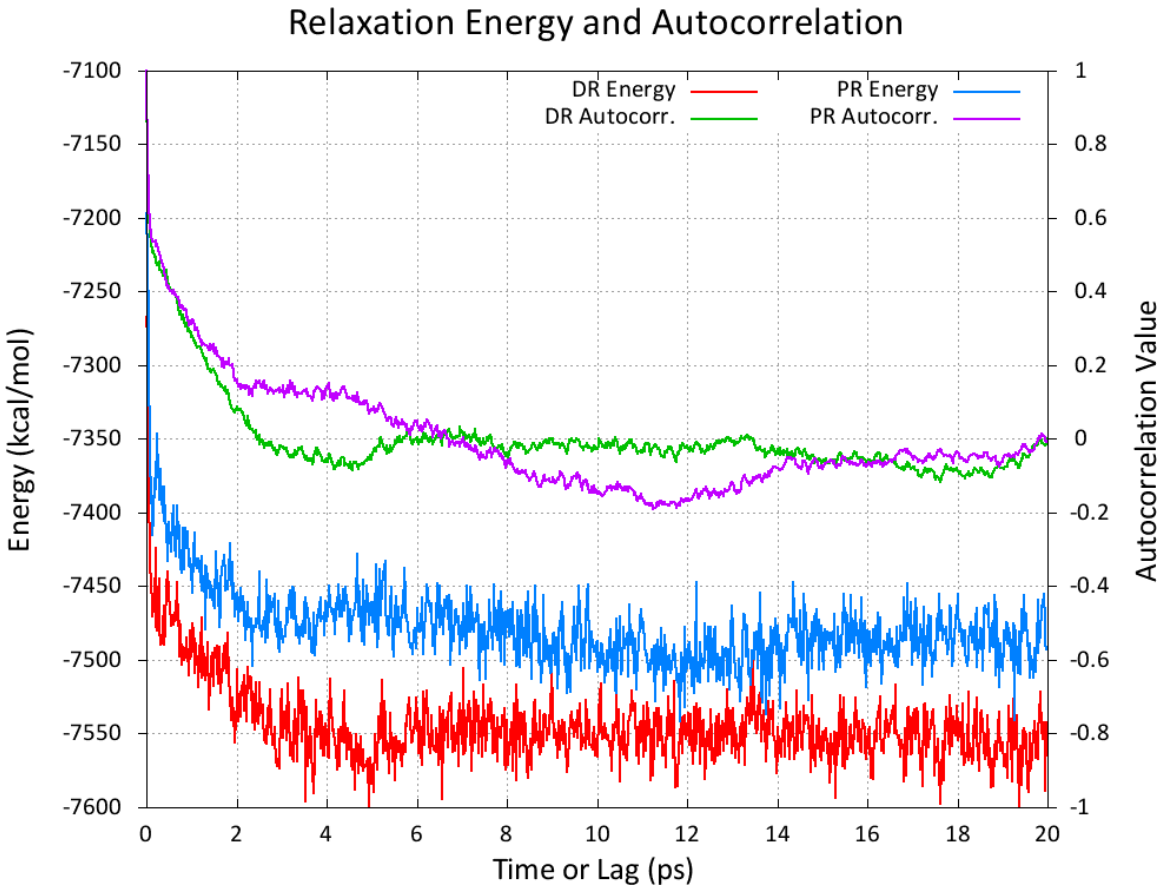

Figure 4-4. The relaxation of the protonated state—starting from the protonated trajectory—is shown in blue with its autocorrelation function shown in purple. The relaxation of the deprotonated state from an equilibrated snapshot from the protonated ensemble is shown in red with its autocorrelation function shown in green. Here, *PR* and *DR* stand for *Protonated-Relaxation* and *Deprotonated-Relaxation*, respectively.

energies are uncorrelated from the point of the protonation state change. However, because 4 ps of MD—corresponding to 2000 steps of dynamics with a 2 fs time step—adds dramatically to the cost of CpHMD simulations in explicit solvent, I explored the approximation of using a significantly smaller value for τ_{rlx}.

Both the relaxation energies and autocorrelations drop very sharply at the start of the relaxation dynamics, so the majority of the benefit gained by relaxing the solvent is realized within the first few steps.

For example, the energies from the relaxation of the deprotonated structure in the protonated state drops from -7197 kcal/mol to -7377 kcal/mol during the first 200 fs. The average energy of the final 10 ns of that trajectory is -7490 kcal/mol. Likewise, the energies from the other relaxation dynamics drops from -7267 kcal/mol to -7465 kcal/mol over the first 200 fs, finally settling into an average of -7552 kcal/mol over the final 10 ns. In both cases, 70% of the total relaxation energy was realized during the first 200 fs of relaxation dynamics. The autocorrelation function of the relaxation energy decays similarly, so the assumption that the relaxed solvent distribution is uncorrelated from its starting point is a reasonable approximation.

To validate the use of a shorter τ_{rlx}, I titrated the aspartate model compound using pH-REMD with five different values for τ_{rlx}—10 fs, 40 fs, 100 fs, 200 fs, and 2 ps. The calculated pK_as were 4.08 $\pm$ 0.02, shown in Fig. 4-5. Furthermore, comparing the solvent radial distribution functions of the different solvent relaxation times (Fig. 4-6) shows little dependence of the solvent distribution on the value of τ_{rlx}.

4.4.3 ACFCA: CpHMD vs. pH-REMD

The small peptide chain ACFCA, described in Sec. 4.3.2, was chosen as a test due to its small size and predictable titration behavior. The simplicity of the system makes it an ideal test—its small size mitigates the conformational sampling problem, and the simple titrating behavior of cysteine further simplifies protonation state sampling. Unlike aspartate and glutamate, which have the four defined tautomeric states defined by anti- and syn-protonation on each of two carboxylate oxygens, and histidine which has two tautomeric states on the imidazole, cysteine has only one protonated and one deprotonated state, presenting fewer degrees of freedom that must be exhaustively sampled.

Each cysteine is in a slightly different micro-environment due to the different charges of the N- and C-termini. Because Cys 2 is typically closer to the N-terminus, it is expected to experience a negative pK_a shift with respect to the model compound due to

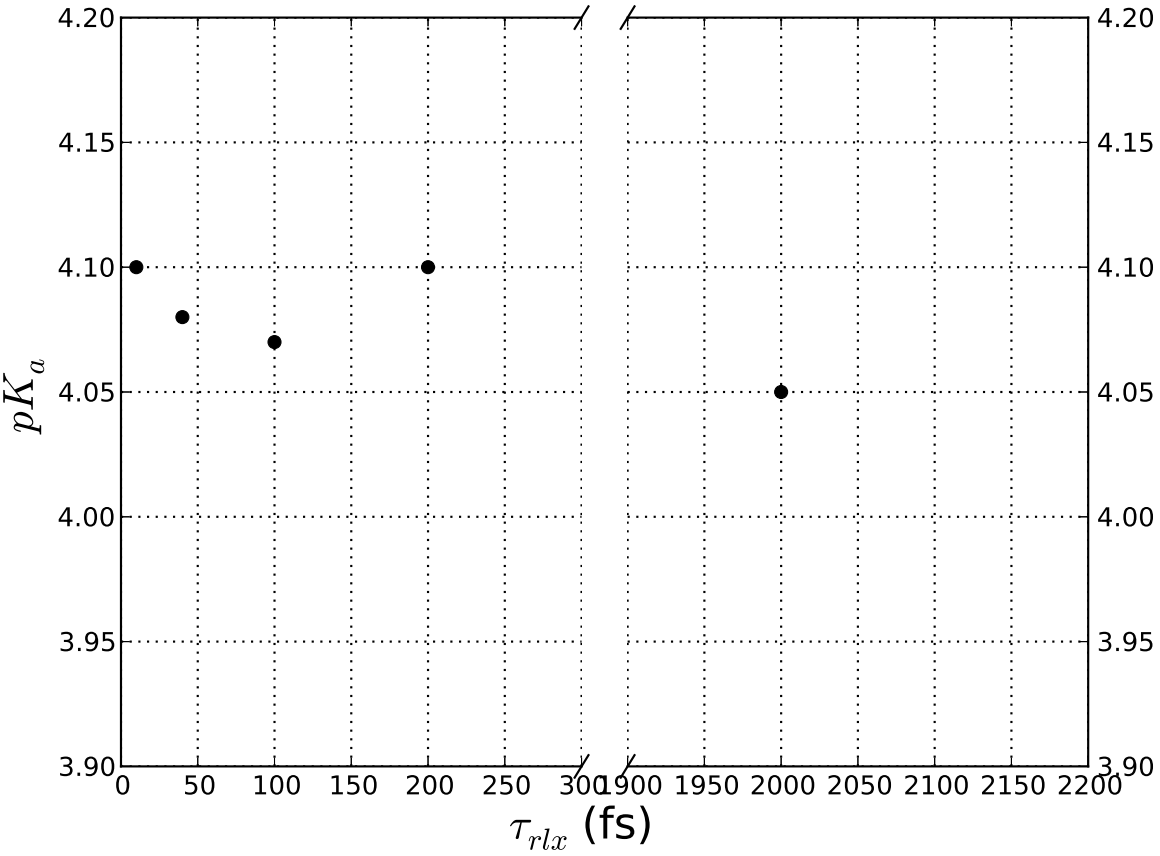

Figure 4-5. Computed pK_as for the Aspartate model compound using different relaxation times (τ_{rlx}).

the electrostatic influence of the positively-charged terminus. Cys 4, on the other hand, is expected to experience a pK_a shift in the opposite direction due to the electrostatic pressure of the negatively-charged C-terminus.

I ran simulations at pH 7.1, 8.1, 8.3, 8.7, 8.9, and 9.9 to sufficiently characterize the titration behavior of both cysteine residues around their pK_as. One set of replicas was run with pH-REMD while the other set was run using CpHMD (i.e., without attempting exchanges between the replicas). The titration curves for both sets of simulations, shown in Fig. 4-7, demonstrate the importance of using pH-REMD in constant pH simulations in explicit solvent. The pK_a of Cys 2 and Cys 4 were 8.2 and 9.4, respectively. As expected, these pK_as represent shifts of -0.2 pK units for Cys 2 and +0.9 pK units

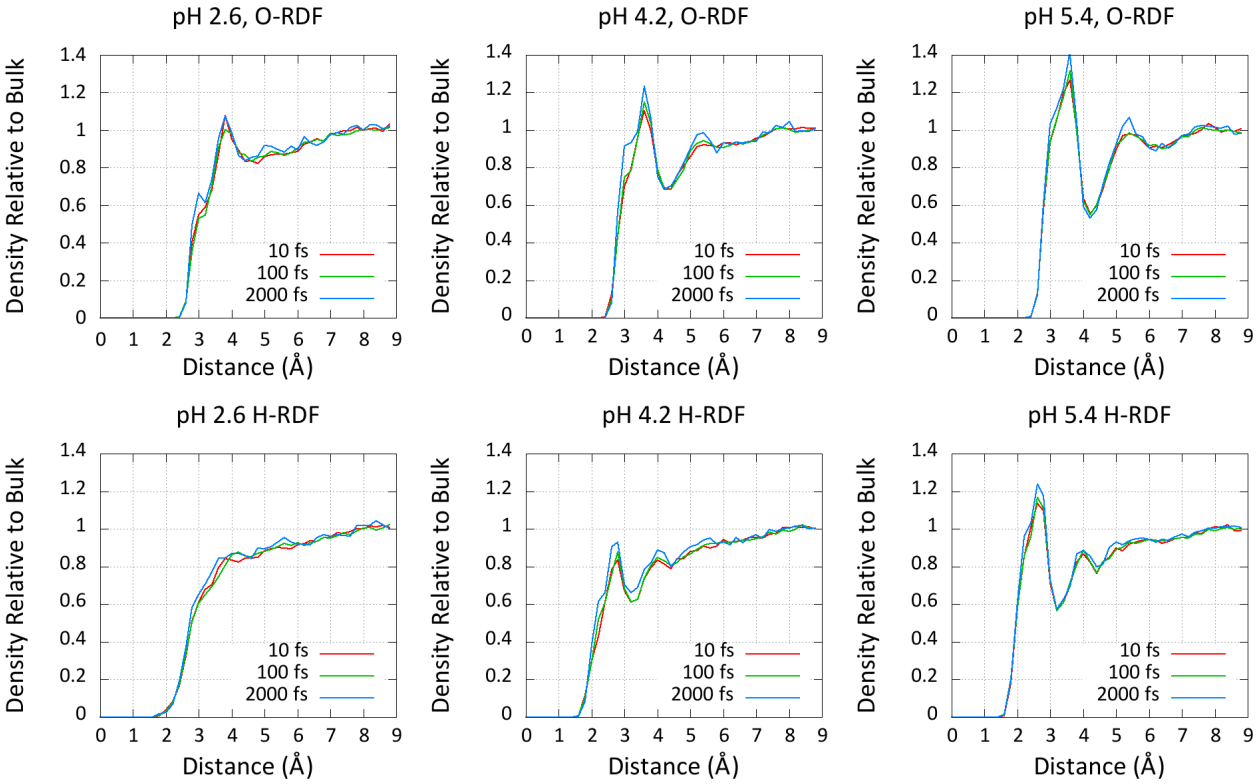

Figure 4-6. RDFs of water oxygen atoms (O) and hydrogen atoms (H) around the center-of-mass of the carboxylate group of the model aspartate molecule at different solution pHs.

for Cys 4 with respect to the model Cys compound. As a test, this compound was run using the original CpHMD implementation in implicit solvent [130] to ensure that we obtained the same results. Because the available phase space in simulations ACFCA is so small due to the small size of the molecule, the sampled ensembles in implicit and explicit solvent are expected to be very similar. When run in implicit solvent, the two cysteine residues have almost identical pK_as to those obtained by the simulations in explicit solvent—8.1 and 9.4, for Cys 2 and Cys 4, respectively.

Even for a simple system such as ACFCA, using pH-REMD on top of standard CpHMD simulations results in a drastic improvement in titration curve fit—a result of improved protonation state sampling. The residual sum of squares (RSS), a quantity that measures how well an equation fits a data set whose equation is shown in 3–6, shows drastic improvement using pH-REMD. The RSS for Cys 2 and Cys 4 using

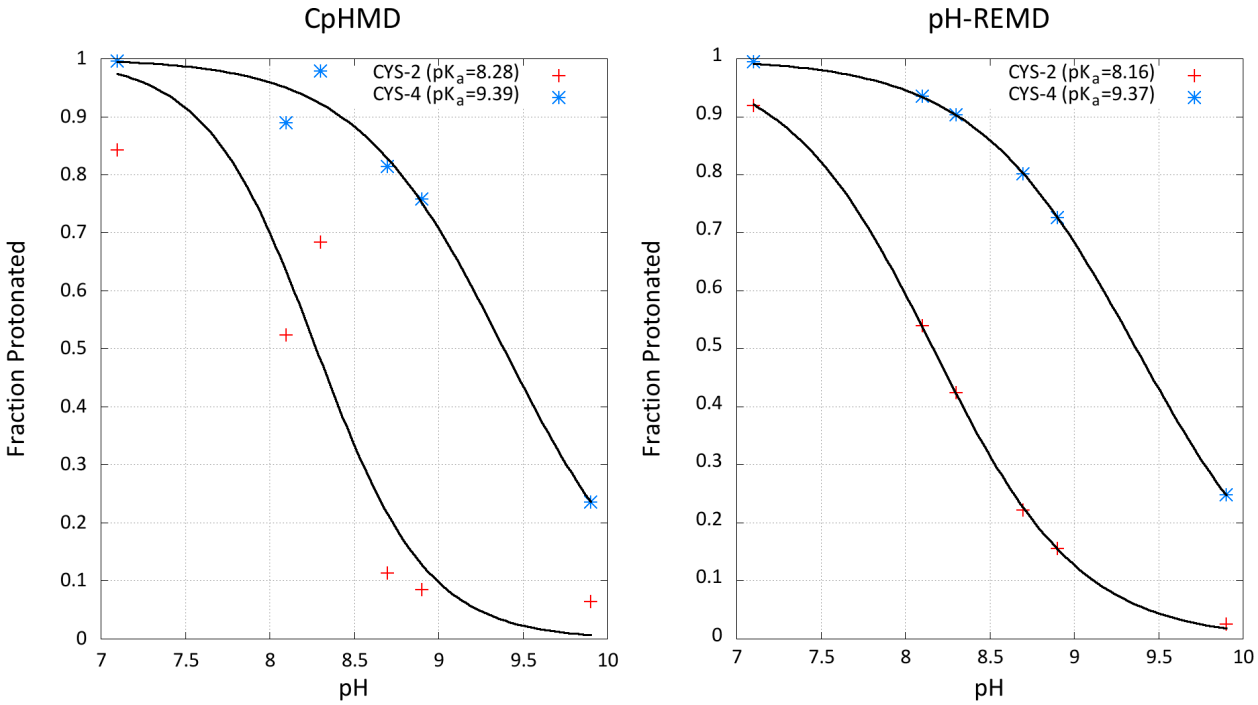

Figure 4-7. Titration curves of Cys 2 and Cys 4 in the *ACFCA* pentapeptide. Results from CpHMD (no replica exchange attempts) and pH-REMD are shown in the plots on the left and right, respectively.

CpHMD was 9×10^{-2} and 7×10^{-3}, respectively. For the pH-REMD simulations, on the other hand, the RSS was reduced by several orders of magnitude to 7×10^{-5} and 9×10^{-6} for Cys 2 and Cys 4, respectively.

4.4.4 Hen Egg White Lysozyme

HEWL is a common benchmark for pK$_a$ calculations because it has been studied extensively both experimentally [136, 144, 145] and theoretically, [86, 88, 130, 135, 146, 161] and it has a large number of titratable residues—some with a marked pK$_a$ shift compared to the isolated model compound.

The 1AKI and 4LYT crystal structures were prepared initially without any ions, resulting in unit cells with a net charge of +9 electrons when the carboxylate residues are negatively charged and the histidine residue is positively charged. The calculated pK$_a$ for all 10 residues that titrate in the acidic range are summarized in Table 4-2 for both starting structures.

Table 4-2. Calculated pK_as for acid-range titratable residues in HEWL using the proposed method for both starting structures—PDBs 1AKI and 4LYT—without ions. The root mean square error (RMSE) and mean unsigned error (MUE) with respect to the experimental values are shown in the last two rows. Experimental values are taken from Webb et al. [136]

Residue	PDB 1AKI	PDB 4LYT	Experiment [136]
Glu 7	1.6	1.8	2.6
His 15	7.1	6.5	5.5
Asp 18	1.8	1.8	2.8
Glu 35	5.0	4.9	6.1
Asp 48	-0.2	-0.3	1.4
Asp 52	-0.3	-1.2	3.6
Asp 66	-1.8	-1.0	1.2
Asp 87	0.5	0.5	2.2
Asp 101	3.7	3.8	4.5
Asp 119	0.0	0.3	3.5
RMSE	2.19	2.20	—
MUE	1.91	1.83	—

The pK_as predicted here agree worse than our results presented in Ref. 88, due mainly to the poor treatment of aspartate residues 48, 52, 66, and 119. The disparity between the implicit and explicit solvent results probably stems from the enhanced conformational sampling attainable with implicit solvent simulations. Dynamical events occur much slower in explicit solvent simulations due to the friction and viscosity of the solvent. However, the conformations sampled in implicit solvent are frequently artifacts of the inaccuracies in the underlying solvent model [156, 157] that may hinder the performance of the CpHMD simulations. [133]

The difference in the conformational sampling ability of the proposed method and the original, implicit solvent-based method is pronounced enough that a simple comparison of the root mean squared deviation (RMSD) is a sufficient illustration. The RMSD of the trajectories in the current study (Fig. 4-8) is 2 to 3 times smaller than the RMSDs shown in Fig. 3-5 from Ch. 3, despite the extra 4 ns of production MD performed for each replica in explicit solvent. Furthermore, the dynamics displays none

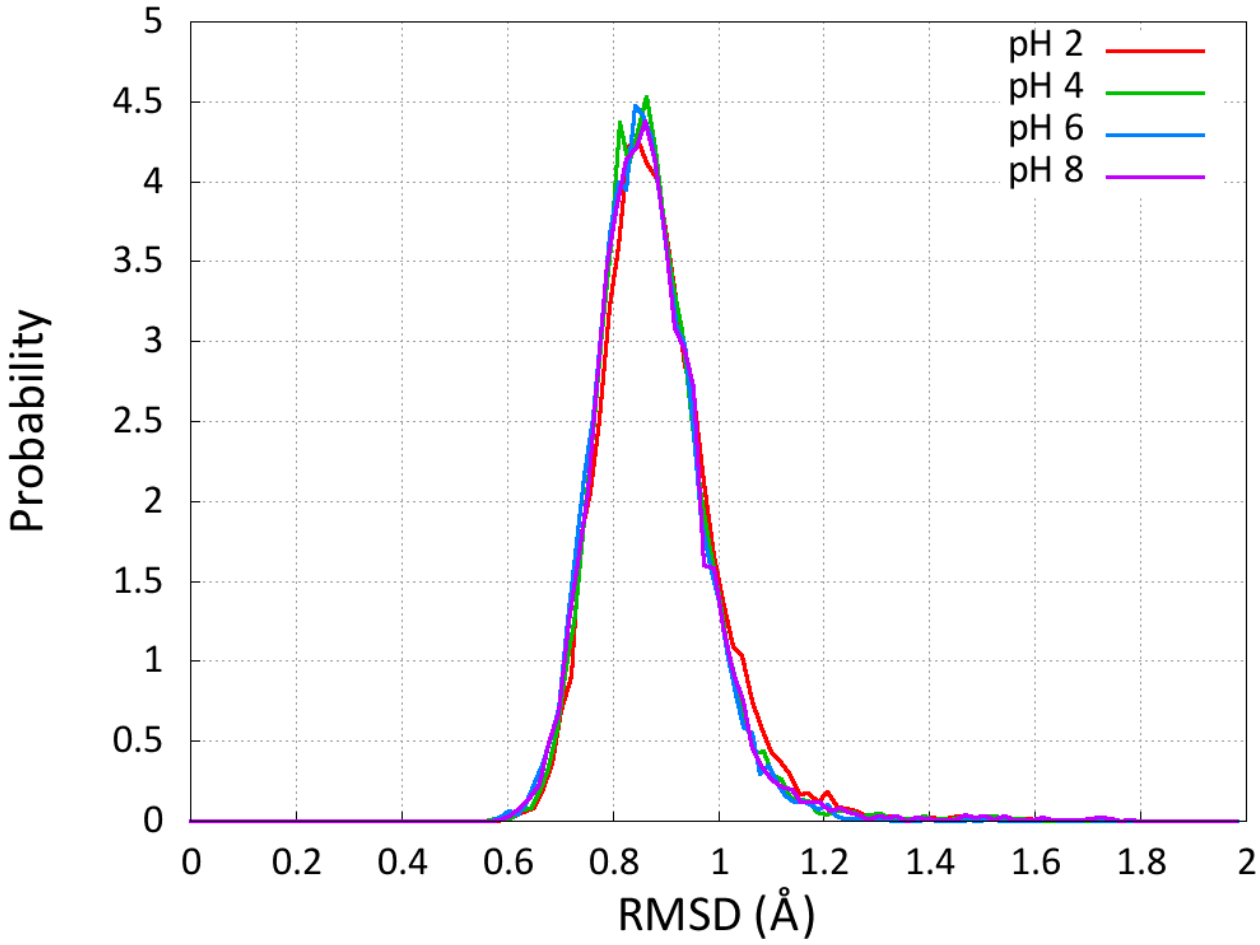

Figure 4-8. RMSD plots over 20 ns of pH-REMD simulation for HEWL at pH 2, 4, 6 and 8 with respect to the starting crystal structure 1AKI. The distributions are very similar for the starting structure 4LYT as well.

of the regions of flexibility noted in implicit solvent that were correlated with the improved titration of aspartate 66. [88]

Ions

When all aspartate and glutamate residues are deprotonated and the histidine is protonated at both the δ and ϵ positions, HEWL has a net charge of +9 electrons. Even though the PME implementation in Amber applies a net neutralizing plasma for such systems, the lack of counterions in the unit cell may lead to unusual behavior by any of the 30 charged residues in the system.

Therefore, I added 21 ions—15 chloride and 6 sodium—to add ionic strength and to provide the ions necessary to neutralize the initial unit cell. Like the effects of the unit cell size and τ_{rlx} value, ions will only affect the calculated pK_as by modifying the sampled conformations since the protonation state changes are performed using implicit solvent.

The predicted pK_as, shown in Table 4-3, show a marked improvement for several residues whose calculated pK_a was too low without ions. The simulations with explicit ions did not exhibit heightened sampling of large-scale conformational changes—the RMSD plots of these simulations are shown in Fig. 4-9—but is rather a result of changes to the microenvironment around the relevant titratable residues significant enough to effect a noticeable change to the titrating behavior.

The residue that experienced the largest pK_a shift as a result of the added ions was Asp 66. As I observed in our previous work, Asp 66 is surrounded by proton donors, such as the Arg 68 and the hydroxyl groups of Thr 69 and Ser 60. [88] The arginine residue, carrying a net positive charge, is the strongest driving force favoring the deprotonated state. To compare how Arg 68 may affect Asp 66 differently when ions are present, I show in Fig. 4-10 that the distribution of Asp 66–Arg 68 distances is shifted to larger values when ions are present. Because Arg 68 can occasionally interact with chloride ions in the bulk solvent when they are present instead of Asp 66, Asp 66 is more likely to accept proposed protonation moves when these ions are present.

It is surprising that the presence of ions can have such a large impact on predicted pK_as without inducing global conformation changes. That the ions are not included in the GB-based, protonation state changes only increases the peculiarity of this result. Explicit ions can modify the local environment around titratable residues enough to induce large pK_a shifts, making them important to include in the proposed method.

4.4.5 Ribonuclease A

Like HEWL, RNase A is a common benchmark for constant pH studies due to its large number of titrating residues. Furthermore, the currently proposed mechanism

Table 4-3. Calculated pK_as for acid-range titratable residues in HEWL using the proposed method for both starting structures—PDBs 1AKI and 4LYT—with 21 ions. The root mean square error (RMSE) and mean unsigned error (MUE) with respect to the experimental values are shown in the last two rows. Experimental values are taken from Webb et al. [136]

Residue	PDB 1AKI	PDB 4LYT	Experiment [136]
Glu 7	1.9	1.8	2.6
His 15	6.4	6.3	5.5
Asp 18	1.9	1.7	2.8
Glu 35	4.6	4.9	6.1
Asp 48	-1.4	0.8	1.4
Asp 52	0.5	0.1	3.6
Asp 66	0.2	-0.4	1.2
Asp 87	0.4	0.5	2.2
Asp 101	3.5	3.5	4.5
Asp 119	0.6	-0.3	3.5
RMSE	1.89	1.91	—
MUE	1.68	1.56	—

requires one catalytic histidine to be a proton donor (general acid) and another to be a proton acceptor (general base)—His 119 and His 12, respectively. Because a specific combination of protonation states are necessary for catalysis, the proposed method is a useful tool for probing the pH-dependence of RNase A.

The predicted pK_as for RNase A in an acidic-range titration—summarized in Table 4-4—are in better agreement with experiment than those from HEWL. This is expected, however, since the average magnitude of the pK_a shifts with respect to the model compounds is smaller in RNase A.

While most of the residues have a calculated pK_a close to the experimental value, several are trapped in environments that resist changing their protonation state across the entire range of simulated pHs. Glu 2 is adjacent to the N-terminal lysine residue with a +2 charge and interacts closely with the positively-charged lysine 7 and arginine 10 residues much of the time, pushing the predicted pK_a very low. If the time scale of the simulation is insufficient to escape this local conformational basin, the predicted pK_a of Glu 2 will be unphysically low, as seen in Table 4-4 for the 1KF5 structure.

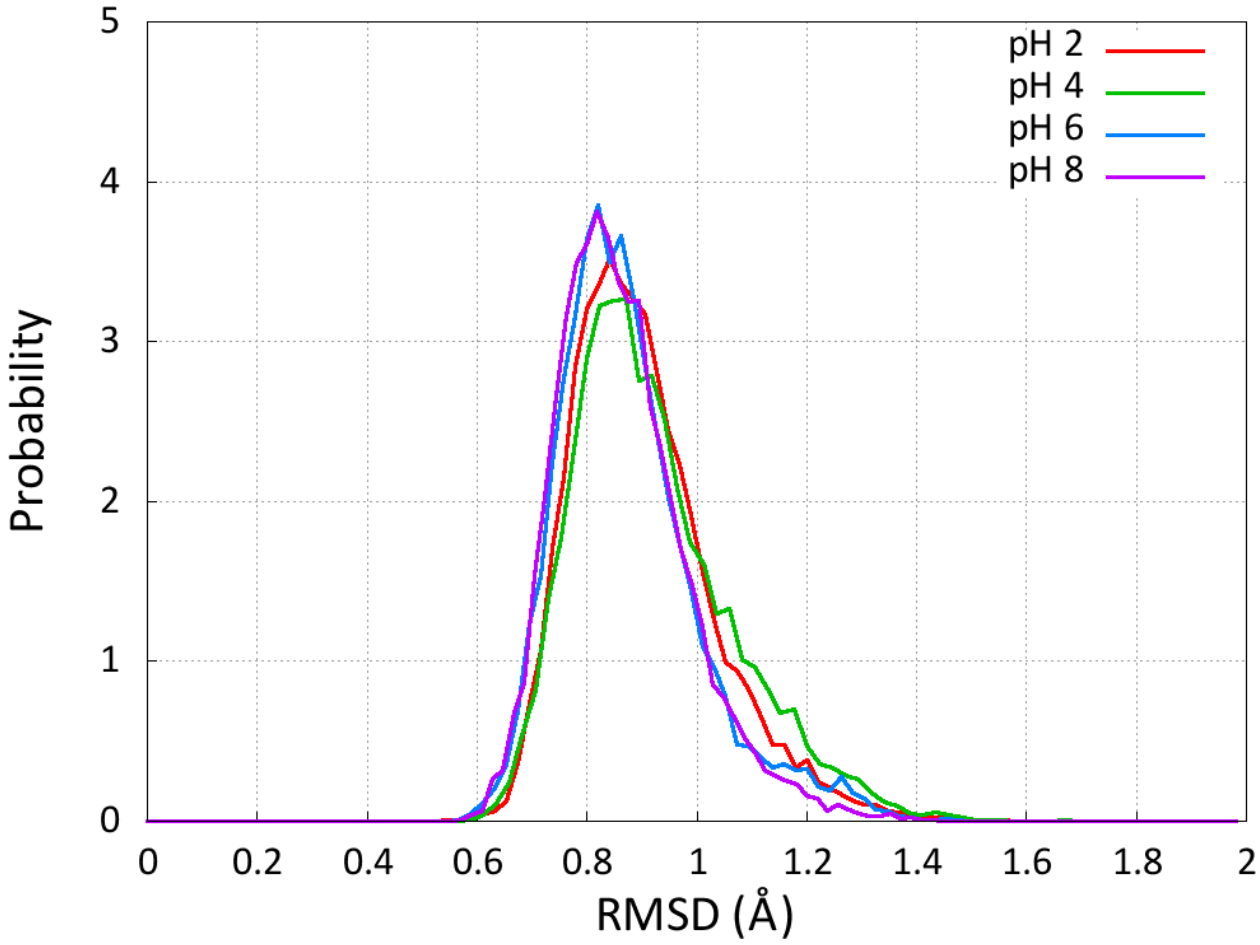

Figure 4-9. RMSD plots over 20 ns of pH-REMD simulation with explicit counterions for HEWL at pH 2, 4, 6 and 8 with respect to the starting crystal structure 1AKI.

Similar traps are seen around Asp 14, which is surrounded by hydrogen bond donors. The His 48 residue, on the other hand, interacts closely with numerous backbone carbonyl atoms in a configuration that resists deprotonating either N^{δ} or N^{ϵ} in the 1KF5 starting structure.

Like with the HEWL simulations, I ran a second set of 20 ns simulations in which I added 14 chloride and 6 sodium ions to generate a net ion concentration around 0.18 M. More chloride was added because the net charge of the initial protonation states was +8 electrons. The predicted pK_as for the simulations with explicit counterions resulted in marked improvement in most titratable residues that proved problematic in

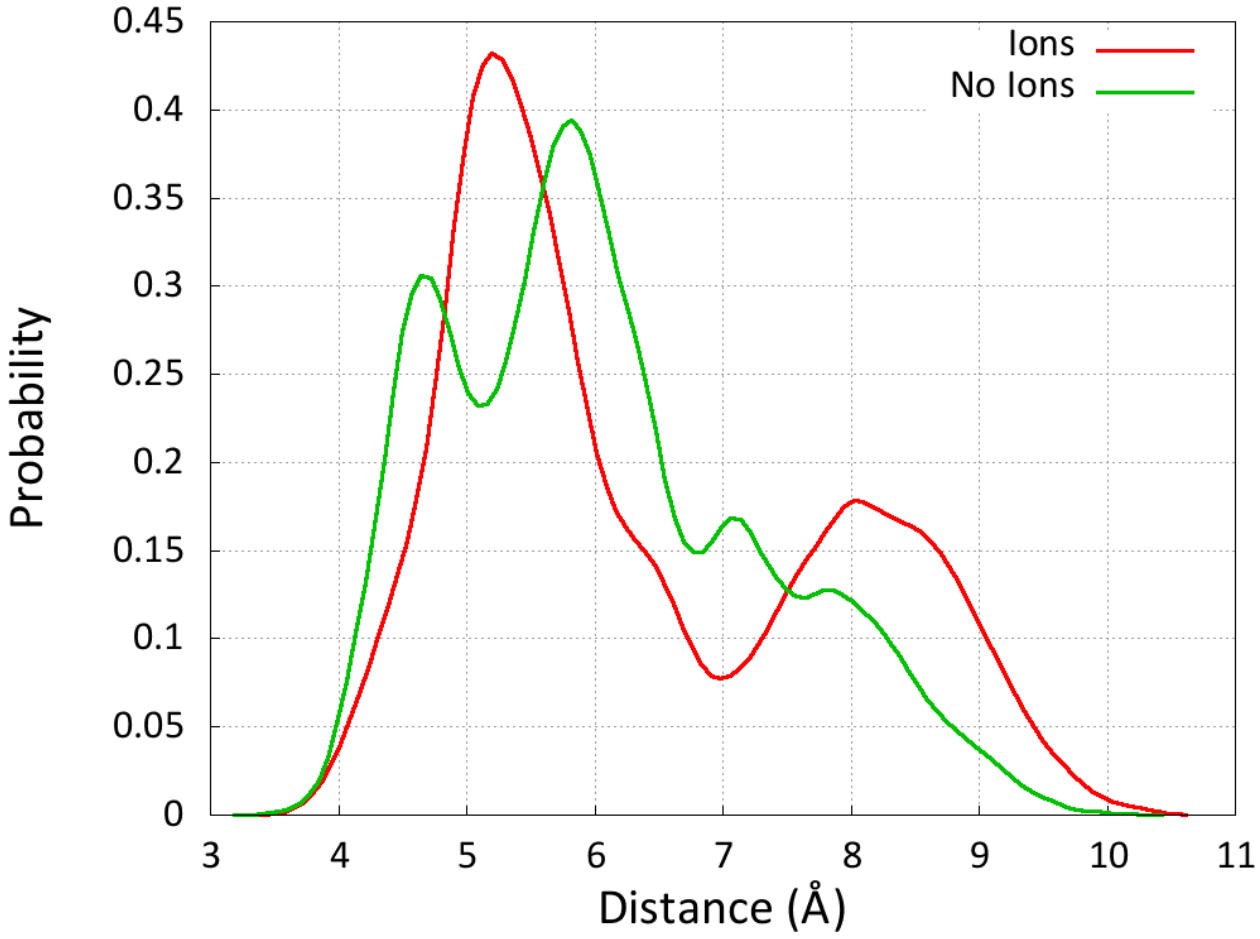

Figure 4-10. Distance distribution functions calculated from HEWL simulations begun with crystal structure 1AKI for all snapshots in the ensemble at pH 1.0. The probability distributions were calculated using 10,000 snapshots and smoothed using a gaussian kernel density estimate with a bandwidth of 0.1

the simulations without the ions, following the trend seen in the HEWL calculations. The full summary of calculated pK_as is shown in Table 4-5.

4.5 Conclusion

I have extended the constant pH molecular dynamics method developed by Mongan et al. [130] so that the dynamics can be run in explicit solvent. I tested a wide range of parameters in our proposed method for their effect on the conformational and protonation state sampling of small test systems. Because these test systems are small and their titratable sites are completely solvent-exposed, they likely represent the highest level of sensitivity to these various parameters.

Table 4-4. Calculated pK_as for RNase A using simulations begun from crystal structures 1KF5 and 7RSA. Experimental values shown are taken from Ref. 174. Root mean squared error (RMSE) and mean unsigned error (MUE) does not include His 48 for the 1KF5 structure or Asp 14 for the 7RSA structure.

Residue	PDB 1KF5	PDB 7RSA	Experiment [174]
Glu 2	-5.7	-0.2	2.5
Glu 9	3.6	3.6	3.9
His 12	6.1	5.8	6.0
Asp 14	-1.2	-8.8	1.8
Asp 38	2.1	2.3	2.1
His 48	∞	5.8	6.1
Glu 49	3.4	3.3	4.3
Asp 53	3.6	3.7	3.7
Asp 83	1.6	1.8	3.3
Glu 86	4.3	4.3	4.0
His 105	7.3	7.8	6.5
Glu 111	3.6	3.4	3.8
His 119	6.0	6.0	6.5
Asp 121	0.6	-0.8	3.0
RMSE*	1.83	2.20	—
MUE*	1.18	1.26	—

Table 4-5. Calculated pK_as for RNase A simulations run with explicit counterions present. All calculated pK_as are included in the calculated root mean squared error (RMSE) and mean unsigned error (MUE).

Residue	PDB 1KF5	PDB 7RSA	Experiment [174]
Glu 2	0.6	-1.8	2.5
Glu 9	3.7	3.6	3.9
His 12	6.2	6.9	6.0
Asp 14	-1.4	-0.3	1.8
Asp 38	1.8	2.3	2.1
His 48	7.9	6.7	6.1
Glu 49	4.2	5.2	4.3
Asp 53	3.3	2.5	3.7
Asp 83	1.5	1.3	3.3
Glu 86	3.8	3.8	4.0
His 105	7.1	6.9	6.5
Glu 111	3.7	3.6	3.8
His 119	5.7	5.7	6.5
Asp 121	-0.2	0.0	3.0
RMSE	1.53	1.70	—
MUE	1.07	1.20	—

In particular, I found that the box size of the unit cell had no discernible effect on the titration behavior of the aspartate model compound, given cell sizes that ranged from 20 Åin diameter—one of the smallest sizes permissible when using the minimum image convention with an 8 Å cutoff—to 40 Å in diameter.

Another key aspect of the current method is the necessity to relax the solvent around any new protonation state selected by the MC moves carried out in GB. By analyzing the decay of the potential energy in the solvent relaxation dynamics, I determined that 4 ps of MD was sufficient to stabilize the energy of the solvent distributions and generate relaxed solvent conformations that are uncorrelated from the initial arrangements. However, given the expense of such a long relaxation period, I investigated using fewer relaxation steps to increase the simulation efficiency and found shorter times—down to 0.2 ps—had no measurable effect on the calculated pK_a and very little effect on the solvent distribution around the model cysteine compound.

Further tests on a small pentapeptide test system with two titratable sites (ACFCA) showed the importance of using pH-REMD over conventional CpHMD with the proposed method. While I showed in Ch. 3 that the enhanced protonation state sampling of pH-REMD results in smoother titration curves for complex proteins in implicit solvent, [88] even the simplest systems in explicit solvent require pH-REMD to obtain a smooth titration curve.

I tested the proposed method on two protein systems, hen egg white lysozyme (HEWL) and ribonuclease A (RNase A). While calculated pK_as were in good agreement with experiment for numerous titratable residues, others appeared stuck in conformational traps resistant to changing their protonation states for the duration of the 20 ns simulation. Many of the residues whose calculated pK_as differed by more than 1 to 2 pK_a units from experiment were surrounded by charged residues that strongly favored a specific protonation state. Unlike our previous study on HEWL, [88] the large conformational

changes seen in implicit solvent occur on a much longer timescale in explicit solvent due to the friction and viscosity of the water molecules.

The larger the pK_a shift a titratable residue experiences inside the protein environment compared to the model compound, the less that environment resembles bulk solution. It is for these residues, therefore, that accurate, extensive, conformational sampling is required to reproduce experimental pK_a measurements. Given the limited mobility of HEWL and RNase A in our simulations, it is therefore not surprising that the most problematic residues were those whose experimental pK_as were several pK units lower than their model compounds.

When I added explicit ions to the simulation cell, the calculated pK_as of the most problematic residues shifted toward their experimental values—in some cases by more than 1 full pK_a unit—despite the limited sampling of global conformational changes on the 20 ns timescale. Therefore, while it is important to include explicit ions to provide a more accurate microenvironment around the titratable residues, the method would probably benefit strongly from attempts to improve conformational sampling, either by longer simulations or some type of enhanced sampling technique. For example, accelerated MD was used in conjunction with the original CpHMD implementation in Amber with promising results. [135]

To summarize, our proposed extension to Amber's CpHMD method allows dynamics to be carried out at constant pH even for systems that cannot yield sensible results when treated with an implicit solvent model, such as DNA and ribozymes. As an example, I tested GB simulations of the hepatitis delta virus (HDV) ribozyme—where nucleobases are thought to act as general acids and bases—and even after careful preparation, the secondary and tertiary structures began breaking down almost immediately and had completely fallen apart within 5 ns. While it may seem that such poor behavior in the MD simulations would preclude GB from being effective for sampling protonation states, the protonation state sampling benefits from better cancellation of errors. The

MC protonation state move is evaluated based on a difference of energy differences—the energy difference between the two charge states from the model compound is subtracted from the energy difference between the two charge states in the biomolecule. Many of the errors inherent to GB should cancel after the second difference is taken so that sensible results may be extracted from these simulations.

In future work I will explore the use of enhanced sampling techniques in conjunction with pH-REMD in an attempt to improve the efficiency of the conformational sampling in explicit solvent, as well as apply our method to systems requiring an explicit solvent representation, such as HDV.

CHAPTER 5
REMD: GPU ACCELERATION AND EXCHANGES IN MULTIPLE DIMENSIONS

This chapter contains a description of my work implementing replica exchange molecular dynamics (REMD) in the *pmemd* program of the Amber program suite. [141] The first sections describe the general theory of the state exchanges supported by Amber, followed by details of their implementation. I'll then finish with a description of my design of multiple-dimension REMD in Amber that I implemented in both the *sander* and *pmemd* programs.

5.1 Temperature REMD

The most common variant of REMD simulations involves assigning replicas with different temperatures (T-REMD) [78] between which the Monte Carlo-based replica exchange attempts occur. The exchange success probability—calculated in a way that satisfies detailed balance to preserve valid thermodynamics—is solved for the proposed change of two replicas swapping temperatures, as shown in Eq. 5–1. When $2N$ replicas are present, N independent exchange attempts can be made simultaneously between different pairs of replicas. If no replica is involved in multiple exchange attempts, these moves can be evaluated independently. While this may not be the most efficient way to perform replica exchange attempts, it is the most common approach due to its simplicity and efficiency.

To calculate the exchange probability in T-REMD exchange attempts, we start with the detailed balance equation (Eq. 1–13) in which replicas m and n have temperatures T_m and T_n, respectively in our initial state i. The temperatures swap in our proposed state such that replicas m and n have temperatures T_n and T_m, respectively. Because the potential energy function of each replica is the same—only the temperature differs between replicas—the probability of a replica having a specific temperature is directly proportional to the Boltzmann factor (in the canonical ensemble). The derivation of the exchange probability equation in T-REMD simulations is shown in Eq. 5–1.

$$
\begin{aligned}
P_i \pi_{i \to j} &= P_j \pi_{j \to i} \\
\frac{\exp\left[-\beta_m E_m\right] \exp\left[-\beta_n E_n\right]}{Q_m Q_n} \pi_{i \to j} &= \frac{\exp\left[-\beta_n E_m\right] \exp\left[-\beta_m E_n\right]}{Q_n Q_m} \pi_{j \to i} \\
\frac{\pi_{i \to j}}{\pi_{j \to i}} &= \min\left\{1, \exp\left[(\beta_n - \beta_m)(E_n - E_m)\right]\right\} \qquad (5\text{–}1)
\end{aligned}
$$

where β_m is $1/k_B T_m$ for replica m and E_m is the potential energy of the structure in replica m.

Because the temperature of the system uniquely defines its kinetic energy, the potential energy can be used in lieu of the total energy in Eq. 5–1 as long as the total temperature remains consistent after the exchange attempt completes. Therefore, the momenta of replica m are typically scaled by $\sqrt{T_n / T_m}$ after successfully exchanging with replica n. [78] By scaling the velocities in this way, snapshots following a successful exchange attempt are immediately 'equilibrated' members of the new temperature's ensemble, thereby eliminating the need to relax the structure to its 'new' temperature. This allows REMD simulations to be carried out more efficiently by permitting exchange attempts very frequently. [142, 143]

An important consideration for T-REMD simulations is how many temperature replicas you should use as well as what temperatures those replicas should have. As the temperature of a system increases, the number of low-energy structures that are sampled during the simulation decreases. In fact, at infinite temperatures, MD is effectively equivalent to random sampling, whose consequences were illustrated in Fig. 1-1. The temperature ladder (*i.e.,* the selection of temperatures at which to run each replica) should be chosen so as to optimize the simulation efficiency. If the temperature difference between adjacent replicas is too great, then the average potential energy difference between adjacent replicas will be large and the exchange probability in Eq. 5–1 will be very small. As a result, the low temperature ensembles will not benefit from the enhanced sampling achievable at the higher temperatures. On the other hand, if the

temperature difference between adjacent replicas is too small, then computational effort will be wasted by simulating unnecessary replicas that do not enhance sampling from the generalized ensemble.

By analyzing Eq. 5–1, it is clear that in order to have a high exchange acceptance probability, either the temperature difference or the potential energy difference between exchanging replicas must be small—in the extreme case, if a higher temperature replica has a conformation whose potential energy is less than or equal to the lower-temperature replica, that exchange attempt is always accepted. By plotting the potential energy distributions obtained from a short simulation at each temperature, the exchange rate between any two replicas can be estimated based on the degree by which their potential energy distributions overlap, shown in Fig. 5-1. A good choice of temperatures for each replica can be made *a priori* based simply on the number of degrees of freedom present in the system. [175]

One challenge with T-REMD is its scalability for large systems. It is well-known that thermodynamic fluctuations scale as $1/\sqrt{N}$ in statistical ensembles where N is the total particle count. Therefore, the larger a system gets, the narrower its potential energy distribution becomes. Consequently, as the potential energy distributions narrow, replicas must be spaced closer and closer together to achieve sufficient mixing along the temperature-space parameter. For this reason, T-REMD simulations on systems that are explicitly solvated are rare. While some approaches, like the one proposed by Okur et al., use a hybrid solvation scheme whereby exchange attempts are carried out in implicit solvent, the first two solvation layers are often represented poorly by implicit solvent, requiring their inclusion even in the hybrid approach. [176]

Furthermore, the snapshots generated at higher temperatures in the generalized ensemble are typically discarded from analyses for two reasons. First, we are typically interested in the thermodynamics of room temperature, so the high-temperature dynamics are not of general interest. Second, our force fields are parametrized for

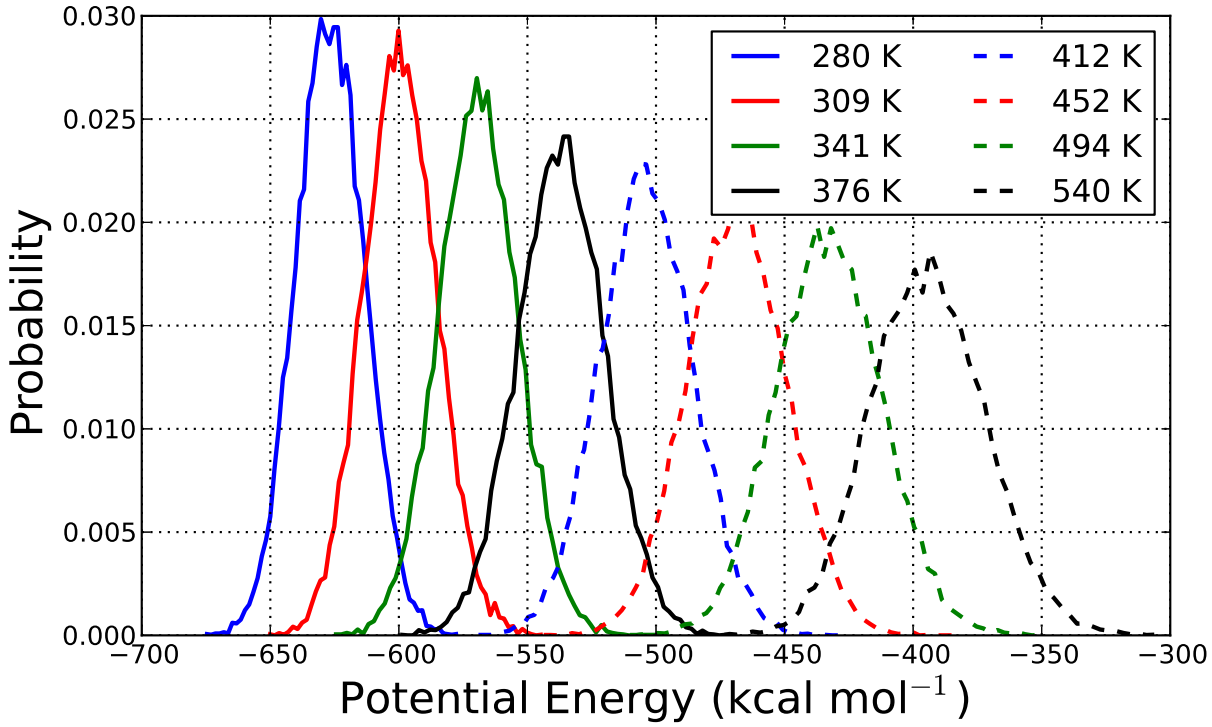

Figure 5-1. Potential energy distributions of TrpCage—a 20-residue peptide—at various temperatures in a T-REMD simulation.

use at temperatures near 300 K, and higher temperatures may break the applicability of harmonic functions for several bonded potentials. While the high-temperature data may be reweighted for inclusion in low-temperature ensembles, [177] higher temperature replicas contribute increasingly little information to the temperatures of interest.

5.2 Hamiltonian REMD

Another common variant of REMD simulations involves swapping Hamiltonians between replicas (H-REMD). Because the nature of the exchange in H-REMD simulations is fundamentally different from those in T-REMD, Eq. 5–1 cannot be used to calculate the exchange probability for H-REMD simulations. The proper exchange probability for H-REMD simulations, generalized for running replicas at different temperatures, is derived in Eq. 5–2. Eq. 5–3 is the special case of Eq. 5–2 when the temperatures of exchanging replicas are the same. The easiest and most general way of implementing

H-REMD is to swap coordinates between exchanging replicas. This approach, as implemented in Amber, can be used for umbrella sampling REMD, [79, 80] accelerated REMD with different boost parameters, [82, 83] and alchemical changes between two end states. [85] As a result, Eq. 5–2 is derived subject to exchanging only coordinates.

$$
\begin{aligned}
&P_i \pi_{i\to j} = P_j \pi_{j\to i} \\
&\frac{\exp\left[-\beta_m H_m(\vec{x}_m)\right] \exp\left[-\beta_n H_n(\vec{x}_n)\right]}{Q_m Q_n} \pi_{i\to j} = \frac{\exp\left[-\beta_m H_m(\vec{x}_n)\right] \exp\left[-\beta_n H_n(\vec{x}_m)\right]}{Q_n Q_m} \pi_{j\to i} \\
&\frac{\pi_{i\to j}}{\pi_{j\to i}} = \min\left\{1, \exp\left[-\beta_m \left(H_m(\vec{x}_n) - H_m(\vec{x}_m)\right) - \beta_n \left(H_n(\vec{x}_m) - H_m(\vec{x}_n)\right)\right]\right\} \qquad (5\text{–}2) \\
&\frac{\pi_{i\to j}}{\pi_{j\to i}} = \min\left\{1, \exp\left[-\beta \left(H_m(\vec{x}_n) - H_m(\vec{x}_m) + H_n(\vec{x}_m) - H_m(\vec{x}_n)\right)\right]\right\} \qquad (5\text{–}3)
\end{aligned}
$$

Looking at Eqs. 5–2 and 5–3, it is readily apparent that exchange attempts in H-REMD simulations are far more expensive than exchange attempts in T-REMD (Eq. 5–1) or pH-REMD (Eq. 3–4) simulations. To calculate the probability of accepting an exchange in H-REMD simulations, each replica must calculate the potential energy of the coordinates of its exchange partner. T-REMD and pH-REMD exchange probabilities, on the other hand, are calculated via a single exponential of quantities known *before* the exchange attempt occurs.

When performing replica exchange on an umbrella coordinate, however, the exchange attempt can be modified to significantly reduce its cost. Since the underlying Hamiltonian is the same for each replica, the energy differences $H_m(\vec{x}_n) - H_m(\vec{x}_m)$ are equal to the difference in their umbrella potentials (Eq. 2–15), which can be calculated very rapidly. This approach reduces the cost of the exchange attempts in two ways. First, the umbrella potentials can be swapped between adjacent replicas rather than the coordinates and momenta, thereby significantly reducing the communication overhead and eliminating the need to reconstruct a new pairlist immediately. Second, computing

the potential due to an umbrella restraint requires a small number of geometric measurements, which is negligible compared to evaluating the energy of the entire system (including the restraint potential).

Despite the apparently high cost of evaluating Eq. 5–2, attempting exchanges every 100 MD steps incurs, at most, a 1% performance hit due to performing one extra energy evaluation every 100 steps (each of which requires a full force evaluation for standard dynamics). Therefore, there has not been enough incentive for writing an optimized exchange routine specifically for umbrella sampling simulations in Amber. Such an exchange routine would be useful in future studies if Gibbs' sampling exchange attempts were implemented, [138] or in situations where swapping only an umbrella potential simplifies calculating Eq. 5–2.

Replica Exchange Free Energy Perturbation. Here I will refocus on free energy perturbation—Eq. 2–23—and its relationship with the H-REMD exchange probability shown in Eq. 5–3. By comparing these two equations, we see that the energy differences required in Eq. 2–23 are calculated every time the exchange probability is calculated in Eq. 5–3! Therefore, the term $\langle \exp\left(-\beta(E_B - E_A)\right)\rangle$ can be accumulated in both the forward and reverse directions during the course of the H-REMD simulation. This approach of computing FEP-based energy differences between two states during a H-REMD simulation is referred to as *Replica Exchange Free Energy Perturbation* (REFEP). [84, 85]

5.3 Multi-Dimensional REMD

As the architecture of modern computers continues its push into massive parallelization, highly scalable techniques such as REMD become increasingly cost-efficient methods in the field of computational chemistry. While we have seen that REMD simulations, in general, enhance sampling by expanding our original ensemble through state space (*e.g.,* temperature space, Hamiltonian space, etc.), different variants of REMD

bestow different advantages on the simulation. For instance, T-REMD enhances conformational sampling by flattening the free energy surface, pH-REMD enhances sampling by allowing simulations to dodge free energy barriers through pH-space, and H-REMD enhances conformational sampling by coupling different energy functions.

As availability to large numbers of processing cores increases, it becomes feasible to combine multiple types of replica exchanges into a single, super-expanded ensemble. In this new, larger ensemble, replicas are defined by a series of state parameters, such as a specific temperature, Hamiltonian, umbrella potential, or solution pH. Exchange attempts between replicas must now take into account changes in multiple state parameters, which may lead to complex equations for the exchange probability. To simplify the exchange process, the replicas can be separated into different *dimensions* in which only a single state variable changes along that dimension.

By adopting this approach, the exchange routines described in previous chapters and sections can be reused in this new, multi-dimensional REMD method. To visualize which exchanges are performed, consider a 2-dimensional square matrix in which the rows and columns represent two different state parameters. In single rows or columns, only a single state parameter changes, so the exchange probability equations that have already been derived apply to these exchange attempts. Fig. 5-2 displays the arrangement of replicas—and the allowed exchange attempts—in a simple diagram. While these ideas can be trivially extended to an arbitrary number of dimensions, the number of replicas required increases exponentially with each additional dimension.

5.4 Implementation

In this section, I will describe how REMD is implemented in Amber, with focus paid to how exchange attempts are carried out as well as the programmatic details of how information is traded between exchanging replicas.

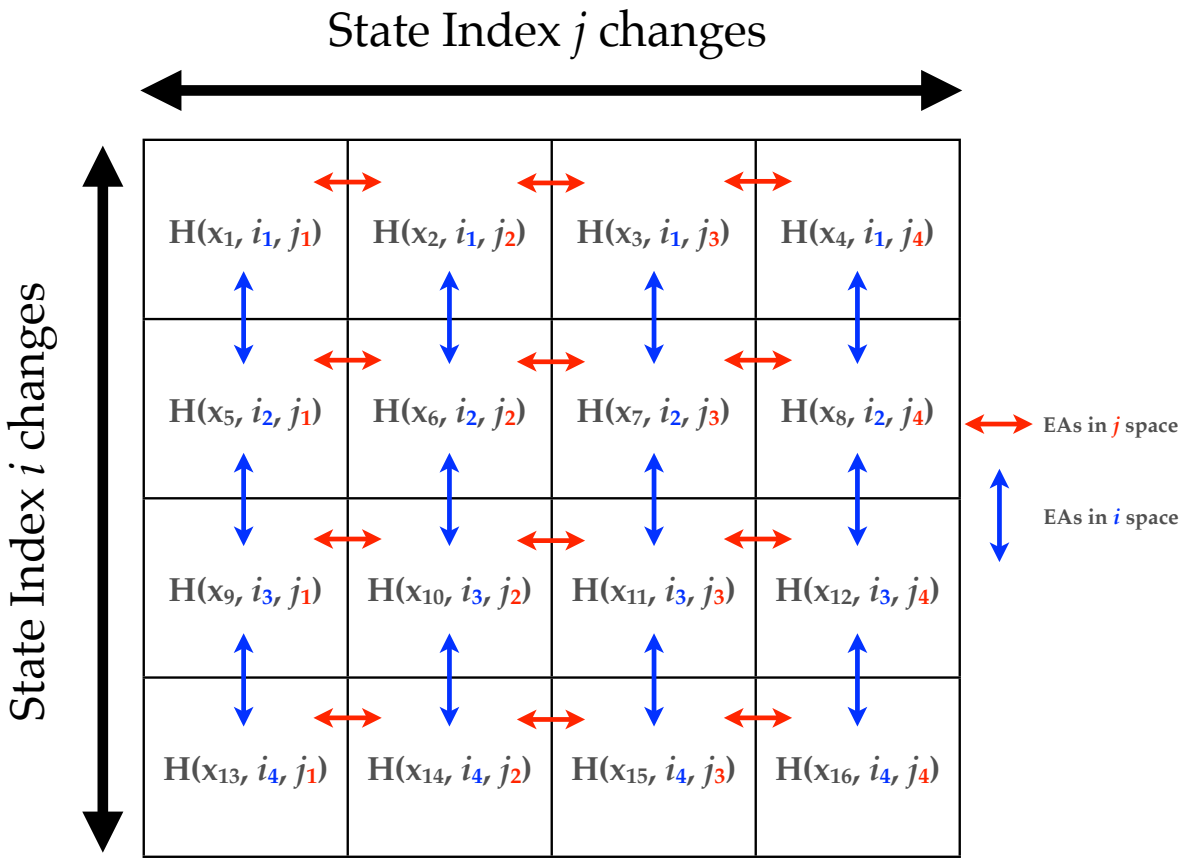

Figure 5-2. Schematic showing exchange attempts in multi-dimensional REMD simulations. Exchange attempts are indicated by the colored arrows, where red arrows indicate exchange attempts between the *j* state parameters and blue arrows indicate exchange attempts between the *i* state parameters

5.4.1 Exchange Attempts

Specific details of how and when exchanges are attempted between replicas is very important to not only the efficiency of our overall simulations, but also the theoretical rigor of its correctness. As I have already mentioned, exchange attempts are restricted to a single pair of replicas in which only a single state parameter differs between them. The question of *which* replicas attempt to exchange information also has a strong impact on how quickly observable properties converge. The easiest and most naïve approach is to choose a single partner and attempt an exchange. To maximize the likelihood that the

exchange attempt is successful, exchanges are attempted between nearest-neighbors in the state parameter that is being swapped. Due to its simplicity, this is the approach that was implemented in Amber by Cheng et al.. [178] Recent evidence suggests, however, that such an approach limits sampling in the state space coordinate. [138] Sampling along the state space coordinate can be enhanced by employing ideas from *Gibbs' sampling*, [138] or simply increasing the frequency of exchange attempts. [142, 143]

Another important consideration in REMD simulations is *when* to suspend the MD in each dimension and attempt to exchange information. Strict adherence to the condition of detailed balance and the principle of reversibility in the resulting chain of states requires these exchange attempts be done stochastically. [138] However, while deterministic exchange attempts violate detailed balance, they satisfy the less restrictive condition of general balance, so the thermodynamical rigor of such an approach is preserved. [138] Amber employs a deterministic, synchronous approach to deciding when exchange attempts should be performed by attempting exchanges between adjacent replicas every n steps, where n is a tunable input parameter.

Also important is the nature of the exchange itself. The two approaches currently used in Amber—exchanging state parameters or exchanging coordinates—are described below.

Exchanging State Parameters. The most efficient way to carry out replica exchanges is to swap state parameters—an approach used in Amber for both T-REMD and pH-REMD. In this case, replicas typically differ by a term that modifies a potential energy function that is otherwise the same for each replica. In this instance, simulations are subject to a different thermodynamic constraint after exchanges are successful. Following successful exchanges, the position of each replica in the ordered list of state parameters changes. As a result, the nearest neighbors between whom exchanges are attempted changes after each exchange attempt. Prior to each exchange attempt, each

replica must figure out where every replica resides in state space so they know how to carry out exchanges.

When exchanging state parameters, replicas typically need to exchange a minimal amount of information—their state parameter and a related conjugate property. In the case of T-REMD, replicas exchange temperatures and potential energies, and individual replicas adopt different temperatures as a function of time. For pH-REMD, the solution pH and total number of 'active' titratable protons are swapped between adjacent replicas.

When an exchange attempt can be completed simply by swapping states, the resulting output files from the simulations follow the course of a single trajectory as it passes through both phase space and state space. As a result, the trajectory file must be modified so that the state parameter of each frame can be identified. This is necessary for reconstructing the sub-ensemble of interest (*e.g.,* the ensemble at 300 K, or pH 7). While this approach adds the complexity of the bookkeeping required to post-process the data, the communication required scales as $O(1)$ with system size, improving the scalability of these REMD simulations. [88] Because the cost of exchange attempts in this family of REMD methods is negligible, there is no practical limit to the frequency with which replicas attempt to exchange, allowing us to take advantage of the faster convergence accessible via rapid exchange attempts [142, 143] or Gibbs' sampling. [138]

Exchanging Coordinates and Momenta. The alternative to swapping state parameters between replicas is to swap coordinates and their conjugate momenta, which is logically equivalent to swapping full potential energy functions. It is significantly simpler—and requires far less communication between exchanging replicas to be fully general—than swapping the full potential energy function (which potentially includes particle charges, masses, pairwise Lennard-Jones parameters, restraints, etc.). It is for the added simplicity and reduced communication overhead that H-REMD is

implemented in Amber by swapping coordinates and velocities (scaling the velocities if exchanging pairs have different temperatures). [85]

Adding to the computational expense, however, is the need to either recompute or exchange the full pairlist of each replica. Either choice is quite expensive since the pairlist is a very large array that requires evaluating all pairwise distances in the system to build. Unlike approaches that exchange state parameters, the cost of exchange attempts that require coordinate exchanges and extra energy evaluations (and multiple additional pairlist builds) imposes a very real upper limit on the practical efficiency of employing rapid exchange attempts or Gibbs' sampling ideas to these simulations.

The most efficient way of performing REMD using umbrella potentials would be to swap the umbrella potentials—essentially a state parameter—and track a replica's trajectory through umbrella space.

5.4.2 Message Passing: Data Exchange in REMD Simulations

While REMD simulations can be carried out 'in serial' by simulating chunks of each replica sequentially by a single process, such an approach defeats the purpose of proposing REMD simulations as a scalable protocol for enhanced sampling. REMD is most efficient when each replica can be simulated simultaneously using different processes, or *threads*. The main simulation engines in Amber use the *Message Passing Interface* (MPI) to enable distributed memory parallelization (*i.e.,* each working thread contains its own memory that is inaccessible by other threads). MPI—described in more detail in Appendix C—is ideally suited for large-scale parallelization since it allows workers to be spread across multiple processing cores that do not share a common memory bank. The most powerful supercomputers in the world that we typically use to carry out our simulations are so-called *distributed* supercomputers since they are constructed from many individual computers with dedicated memory that are networked together.

MPI enables parallelism by allowing groups of threads to exchange information by sending and receiving data through a series of predefined functions and subroutines (typically called an application programmer interface, or API). Data, or *messages*, can be sent and received between two threads in an MPI program that are grouped together in the same *communicator*. Because communicators provide a simple and efficient way of programmatically separating threads into different groups, we take advantage of this feature when organizing the workload in MPI programs. Intra-replica communication—which allows a single replica to be run using multiple processors—is handled by a dedicated replica communicator. An arbitrarily designated *master* thread of each replica is assigned to separate communicators for communicating all data pertinent to carrying out replica exchange attempts.

In typical REMD simulations involving only a single state parameter, the REMD communicator is simply a communicator that links all replica masters. In multi-dimensional REMD, however, exchanges are only permitted between replicas that differ in only one state parameter. Therefore, communicators are defined between only those replicas between which exchanges are permitted. Using Fig. 5-2 as a guide, communicators are defined between the masters of the replicas in a single row or column. These communicators have to be set up and destroyed after each exchange attempt because successful exchange attempts in a dimension that implements state parameter swaps will change the REMD communicator that the replica belongs to in the other dimensions. This is illustrated in Fig. 5-3.

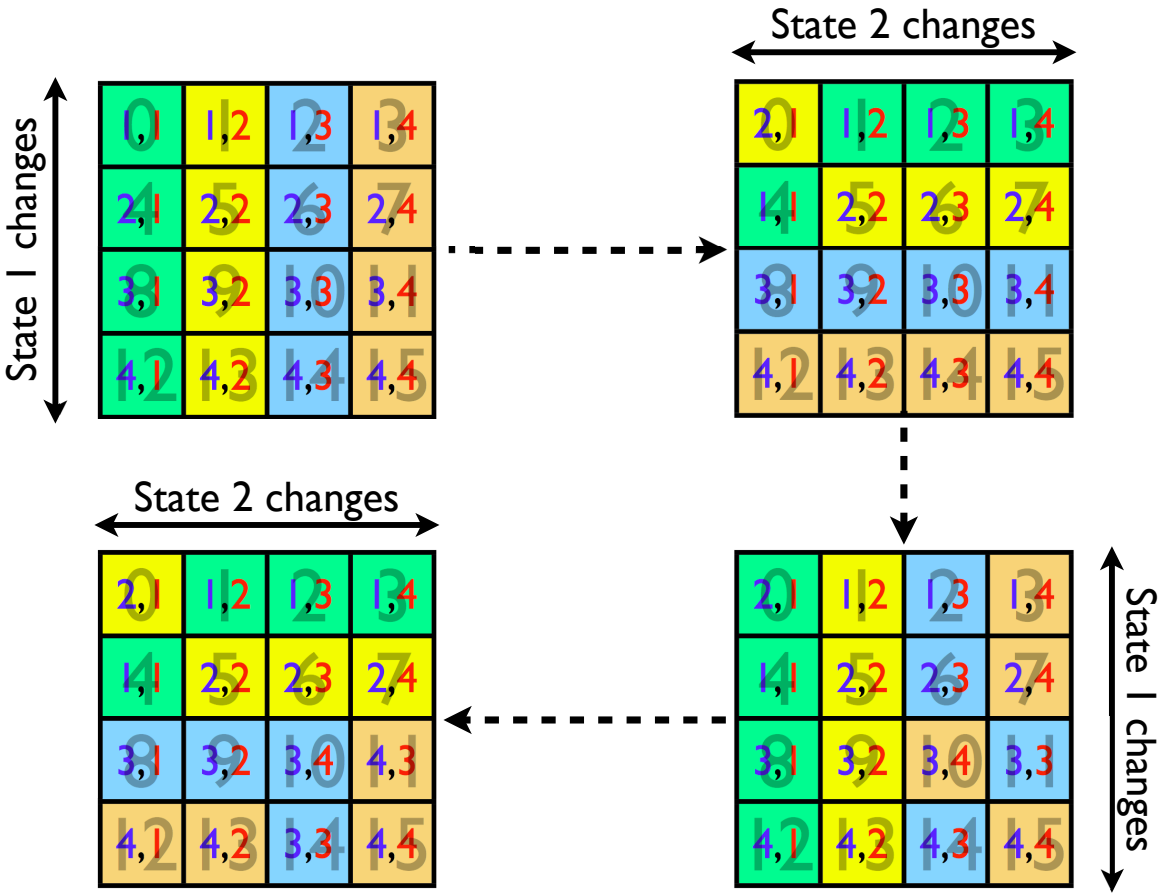

Figure 5-3. Communicator arrangement in multi-dimensional REMD simulations at multiple exchange steps following some successful state parameter exchanges. The large numbers in the background are the (unchanging) thread numbers in the communicator linking the 'master' threads of each replica. The blue and red numbers are the indexes in the first and second state parameter tables, respectively. Every cell with the same background color is a member of the same REMD communicator.

CHAPTER 6
FLEXIBLE TOOLS FOR AMBER SIMULATIONS

In this chapter, I will describe the motivation behind creating two tools to aid users in carrying out biomolecular simulations with the Amber programming package as well as some details regarding their functionality and implementation. During the course of my graduate studies, I wrote several scripts and programs to aid in my work—several of which I polished and released with the Amber suite of programs. The two I will describe in this chapter are *MMPBSA.py* [110] and *ParmEd*.

6.1 MMPBSA.py

Portions of this section are reprinted with permission from Miller III, McGee Jr., Swails, Homeyer, Gohlke, and Roitberg, "MMPBSA.py: An Efficient Program for End-State Free Energy Calculations," *J. Chem. Theory Comput.*, 2012, **8** (9), pp 3314–3321. [110]

MMPBSA.py is a script designed to automate the procedure of performing end-state free energy calculations, as described in Sec. 2.3.3.1.

6.1.1 Motivation

End-state free energy methods—briefly described in Sec. 2.3.3—are popular methods for computing binding free energies for protein-ligand binding, [179–183] protein-protein binding, [101, 102, 183, 184] nucleic acid binding, [183, 185] and relative conformational stabilities. [186, 187] There has been significant effort applied to improving the approximations used in end-state methods, and in some cases it has even approached predictive accuracy. [182, 188]

By 2008, there was a set of perl scripts capable of automating MM-PBSA and MM-GBSA calculations that were written in 2002 for release with Amber 7 and had not been changed since 2003. These scripts will be collectively referred to as *mm_pbsa.pl* from now on. Due to its age, *mm_pbsa.pl* was compatible only with the low-precision, inefficient ASCII trajectory format, and offered only a limited set of the available implicit

solvent models and input parameters that had been developed over the decade that followed its initial release. Furthermore, the input for *mm_pbsa.pl* was very different from the typical input that most other Amber programs expected. Finally, *mm_pbsa.pl* was only capable of running in serial, despite the fact that end-state analyses themselves could be trivially parallelized by computing binding free energies for individual frames simultaneously.

The goal of my project was to revitalize this set of helpful scripts that had fallen out of support and had grown outdated. We wanted to bring the input style in line with the rest of the Amber programs—for example, atom selections should be input via the Amber mask syntax, and groups of related variables should be specified in Fortran-style namelists. Furthermore, we wanted to provide the user with access to new input variables and solvent models. Given the magnitude of the changes required, the recent emergence of Python in the field of computational chemistry, [110, 189–191] and the undocumented, monolithic state of *mm_pbsa.pl*, we decided to build a new script to perform end-state free energy calculations in Python—*MMPBSA.py*. [110]

6.1.2 Capabilities

In this section, I will briefly outline some of the various types of calculations that *MMPBSA.py* is capable of performing.

6.1.2.1 Stability and Binding Free Energy Calculations

End-state calculations are frequently used for two types of analyses—calculating the relative stability of multiple conformations of a system and calculating the binding free energy in a non-covalently bound, receptor-ligand complex, [109] whose thermodynamic cycles were shown in Ch. 2, Fig. 2-11. Stability calculations compare the free energies of multiple conformations to determine their relative stability. If we consider the process of a biomolecule changing conformations from state A to state B, then the free energy associated with that conformational change is simply the difference in the free energies of states A and B. Similarly, the non-covalent binding free energies can

be computed as the difference in free energies of the bound and free states of the two species in solution.

The free energy changes in solution can be decomposed according to

$$\Delta G_{solvated} = E_{gas} + \Delta G_{solvation} - T\Delta S_{solute} \tag{6–1}$$

where $\Delta G_{solvation}$ represents a true free energy, since the solvent degrees of freedom have been averaged by using an implicit solvent model. The free energy of solvation in 6–1 can be further decomposed into a sum of polar and non-polar contributions in most implicit solvent models. Among the solvent models available for end-state calculations in *MMPBSA.py* are the previously mentioned PB and GB implicit solvent models as well as the 3-dimensional reference interaction site model (3D-RISM). [192]

The energies described in Eq. 6–1 are single point energies of the system. However, in practice, end-state calculations estimate these energies according to ensemble averages taken from a simulation. Expressing Eq. 6–1 in terms of an average over a simulated ensemble yields Eq. 6–2.

$$\begin{aligned}\Delta G_{solvated} &\approx \langle E_{gas}\rangle + \langle \Delta G_{solvation}\rangle - T\langle S_{solute}\rangle \\ &= \frac{1}{N}\left\{\sum_{i=1}^{N}[E_{i,gas} + \Delta G_{i,solvation}] - T\sum_{i=1}^{N} S_{i,solute}\right\}\end{aligned} \tag{6–2}$$

where i is the index of a particular frame and N is the total number of analyzed frames.

There are two approaches to generating the necessary ensembles for the bound and unbound state of binding energy calculations—all ensembles can be extracted from a single MD or MC trajectory of the bound complex, or trajectories can be generated for each state using separate simulations. [193] These approaches are called the *single trajectory protocol* (STP) and the *multiple trajectory protocol* (MTP), respectively, and each approach has distinct advantages and disadvantages.

STP is less computationally expensive than MTP, because only a single trajectory is required to generate all three ensembles. Furthermore, the internal potential terms (e.g., bonds, angles, and torsions) cancel exactly in the STP, because the conformations in the bound and unbound ensembles are the same, leading to lower fluctuations and easier convergence in the binding free energy. The STP is appropriate if the receptor and ligand ensembles are comparable in the bound and unbound states. However, the conformations populating the unbound ensembles typically adopt strained configurations when extracted from the bound state ensemble, thereby over-stabilizing the binding, compared to the MTP.

6.1.2.2 Free Energy Decomposition

Amber [141] provides several schemes to decompose calculated free energies into specific residue contributions using either the GB or PB implicit solvent models, [194] following the work of Gohlke et al. [101] Interactions can be decomposed for each residue by including only those interactions in which one of the residue's atoms is involved—a scheme called *per-residue* decomposition. Alternatively, interactions can be decomposed by specific residue pairs by including only those interactions in which one atom from each of the analyzed residues is participating—a scheme called *pairwise* decomposition. These decomposition schemes can provide useful insights into important interactions in free energy calculations. [101]

However, it is important to note that solvation free energies using GB and PB are not strictly pairwise decomposable, since the dielectric boundary defined between the protein and the bulk solvent is inherently nonlocal and depends on the arrangement of all atoms in space. Thus, care must be taken when interpreting free energy decomposition results.

An alternative way of decomposing free energies is to introduce specific mutations in the protein sequence and analyze how binding free energies or stabilities are affected. [112] Alanine scanning, which is a technique in which an amino acid in the system is

mutated to alanine, can highlight the importance of the electrostatic and steric nature of the original side chain. [99] Assuming that the mutation will have a negligible effect on protein conformation, we can incorporate the mutation directly into each member of the original ensemble. This avoids the need to perform an additional MD or MC simulation to generate an ensemble for the mutant.

6.1.2.3 Entropy Calculations

The implicit solvent models used to calculate relative stability and binding free energies in end-state calculations often neglect some contributions to the solute entropy. If we assume that biological systems obey a rigid rotor model, we can calculate the translational and rotational entropies using standard statistical mechanical formulae, [9] and we can approximate the vibrational entropy contribution using one of two methods. First, the vibrational frequencies of normal modes can be calculated at various local minima of the potential energy surface. [9] Alternatively, the eigenvalues of the mass-weighted covariance matrix constructed from every member of the ensemble can be approximated as frequencies of global, orthogonal motions—a technique called the *quasi-harmonic* approximation. [195] Using either the normal mode or quasi-harmonic approximations, we can sum the vibrational entropies of each mode calculated from standard formulae. [9]

Typically, normal mode calculations are computationally demanding for large systems, because they require minimizing every frame, building the Hessian matrix, and diagonalizing it to obtain the vibrational frequencies (eigenvalues). Because of the Hessian diagonalization, normal-mode calculations scale as roughly $(3N)^3$, where N is the number of atoms in the system. While the quasi-harmonic approach is less computationally expensive, a large number of snapshots are typically needed to extrapolate the asymptotic limit of the total entropy for each ensemble, which increases the computational cost of the original simulation. [179]

6.1.3 General Workflow

MMPBSA.py is a program written in Python and *nab* [196] that streamlines the procedure of preparing and calculating free energies for an ensemble generated by MD or MC simulations whose general workflow is shown in Fig. 6-1. The process of calculating binding free energies can be a tedious procedure that *MMPBSA.py* aims to shorten and simplify.

Python is a useful programming language for performing tasks that are not numerically intensive, and because it is available on virtually every platform, Python programs are highly portable. Nucleic Acid Builder (*nab*), [196] which is a molecule-based programming language included with AmberTools, contains functionality pertinent to building, manipulating and performing energy calculations on biological systems, such as proteins and nucleic acids.

End-state calculations often require multiple *topology files* (described later) that contain the parameters corresponding to the force field. Simulations are typically run using explicit solvent with any of the electrostatics methods described in Ch. 2, which would require both solvated and unsolvated topology files to use with *MMPBSA.py*. It is necessary that all topology files have a consistent set of parameters, especially for binding free energy calculations. Therefore, *MMPBSA.py* checks the input topology files prior to binding free energy calculations to prevent erroneous results due to inconsistencies that may not be immediately obvious (*e.g.,* different particle counts, partial charges for the same atoms, etc.). I wrote the Python utility *ante-MMPBSA.py* (also released alongside *MMPBSA.py*), which allows a user to easily create topology files with a consistent set of parameters, including changing the intrinsic implicit solvent radius set to fit the desired solvent model.

The use of *MMPBSA.py* is similar to that of Amber's MD engines *sander* and *pmemd*. The command-line flags common to both *MMPBSA.py* and the MD engines

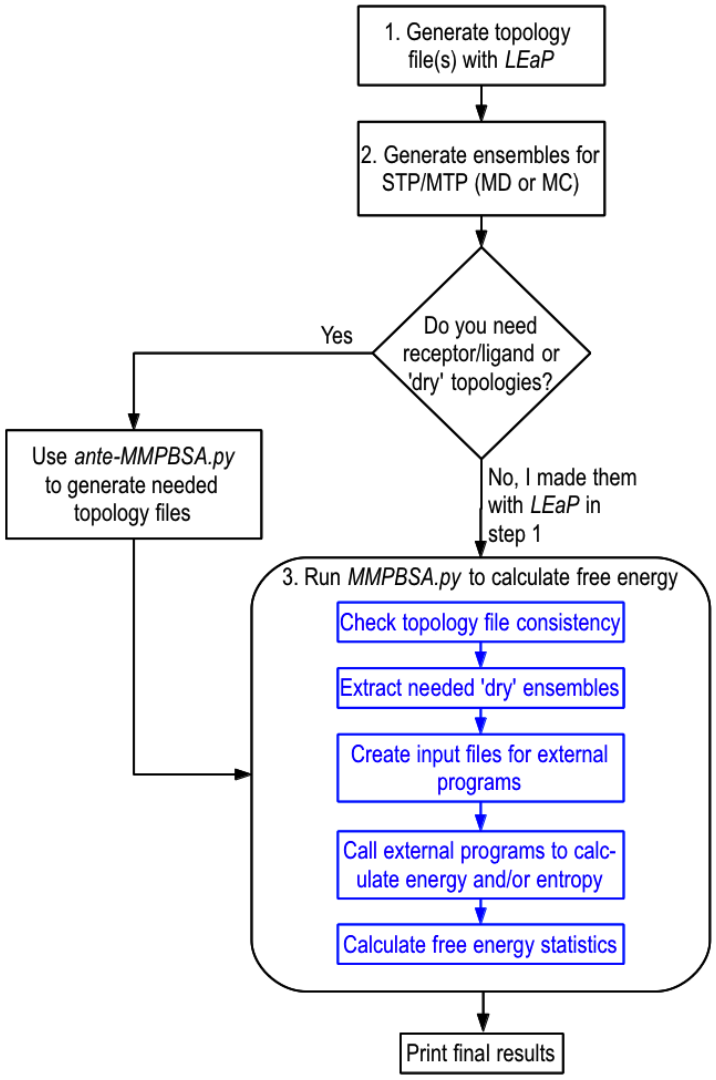

Figure 6-1. General workflow for performing end-state calculations with *MMPBSA.py*. LEaP is a program in Amber used to create topology files for dynamics. The workflow shown in step 3 is the series of steps that *MMPBSA.py* automates. “Dry” topologies and ensembles are systems without explicit solvent that are subsequently treated using an implicit solvent model. “External programs” refers to the executables that perform the energy calculations (*e.g., sander*).

are identical, and input files are separated with similar, Fortran-style namelists, indicated with an ampersand (&) prefix.

The *MMPBSA.py* input file contains a general namelist for variables that control general behavior. For example, variables that control the subset of frames analyzed (startframe, endframe, and interval) and the amount of information printed in the output file (verbose) are specified here. An example of this section is shown below:

```
General MMPBSA.py input file
&general
   startframe=1, endframe=100, interval=2,
   keep_files=0, verbose=1, strip_mask=:WAT:Cl-:Na+,
/
```

6.1.4 Running in Parallel

MMPBSA.py is implemented in parallel, so users with access to multiple processors can speed up their calculations. *MMPBSA.py.MPI* is the parallel implementation of *MMPBSA.py* that uses MPI (described in Appendix C) for Python (*mpi4py*). Since energy calculations for each frame are independent, the calculation can be trivially parallelized, given enough available processors. *MMPBSA.py.MPI* divides frames evenly across all processors, which allows calculations using many frames to scale better than if *MMPBSA.py* invoked parallel executables to calculate free energies. However, perfect scaling is not attained, because certain setups tasks and file input/output can only be done with a single processor. Fig. 6-2 demonstrates scaling for a sample MM-PBSA and MM-GBSA calculation.

6.1.5 Differences to *mm_pbsa.pl*

Both *MMPBSA.py* and *mm_pbsa.pl* allow users to perform free energy calculations using the STP and MTP, although *MMPBSA.py* offers more flexibility when using the MTP. Both programs have the ability to use different PB and GB models contained within Amber and estimate entropic contributions. Finally, *MMPBSA.py* and *mm_pbsa.pl*

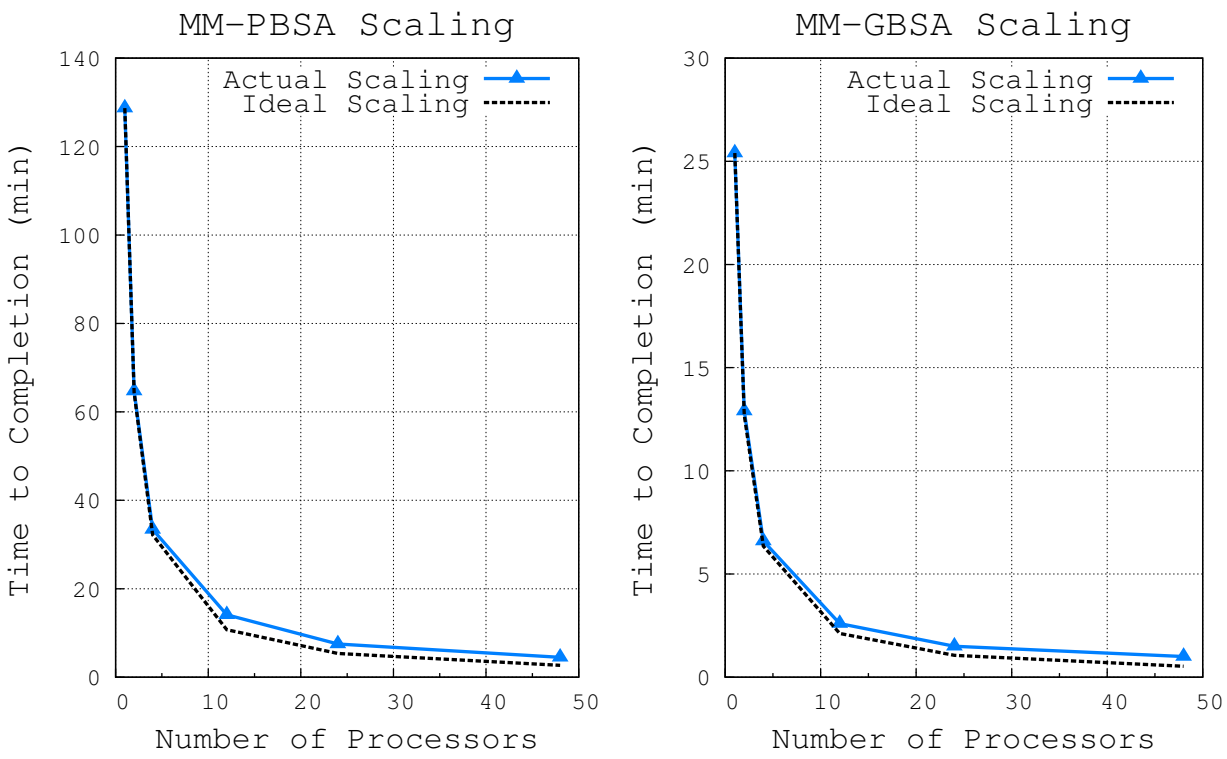

Figure 6-2. *MMPBSA.py* scaling comparison for MM-PBSA and MM-GBSA calculations on 200 frames of a 5910-atom complex. Times shown are the times required for the calculation to finish. Note that MM-GBSA calculations are ∼5 times faster than MM-PBSA calculations. All calculations were performed on NICS Keeneland (2 Intel Westmere 6-core CPUs per node, QDR infiniband interconnect).

can run free energy calculations in parallel, although only *MMPBSA.py* can run on distributed memory systems (*i.e.,* on multiple nodes connected over a network).

Despite their obvious similarities, there are many differences that exist in their accessibility, implementation, and capabilities. *MMPBSA.py* is available free of charge alongside AmberTools, while an Amber license is necessary to obtain *mm_pbsa.pl*. The usage of *MMPBSA.py* is intended to resemble Ambers MD engines for ease of the user, while *mm_pbsa.pl*s input file and usage has its own syntax. Only *MMPBSA.py* has an intuitive mechanism for guessing the ligand and receptor masks of a complex based on the topology files provided and analyzes topology files for parameter consistency. Furthermore, only *MMPBSA.py* can calculate entropic contributions to the free energy

using the quasi-harmonic approximation. An interface to external PB solvers such as Delphi, MEAD, and UHBD is available with *mm_pbsa.pl* only, although both can use the *apbs* program to solve the PB equation. *MMPBSA.py* allows users to provide their own input files for external programs, which gives users the ability to adjust all parameters, not just the variables described in the *MMPBSA.py* manual; in comparison, *mm_pbsa.pl* has no similar functionality without directly altering the source code. Finally, QM/MM-GBSA and MM/3D-RISM calculations are only available through the *MMPBSA.py* implementation.

6.2 ParmEd

ParmEd—short for *Parm*top *Ed*itor—is a program that allows researchers to easily manipulate and extract information from Amber parameter-topology (prmtop) files. The prmtop is a compact ASCII (*i.e.,* pure text) file whose format was optimized for extensibility and Fortran-style parsing. The data structures stored in this file are similar to the data structures used inside the Amber codes that perform MM simulations, making them overly tedious to extract information by simply reading its contents. The full structure and specification of the prmtop is presented in Appendix B.

6.2.1 Motivation

The prmtop files are very complex objects, and there is very little 'locality' in these files. That is, determining which bonds exist and how strong their force constants are is not as simple as looking for the sections labeled with `BOND` in the prmtop. Prior to writing ParmEd, there were no programs released with Amber or AmberTools capable of modifying the topology file in a general way. Changing simple atomic properties—such as the partial charge or the set of intrinsic radii used for implicit solvent models—required the user to modify their original input files to *tleap* and recreate a topology file from their original structure, or in some cases even modify the *tleap* source code directly! Because many input files for *tleap* are shared among all users and original input files help document one's protocol, modifying these files frequently is dangerous.

The tedious and error-prone nature of this process is a deterrent for testing some new hypotheses and methods that require small changes to the topology file. For instance, parameterizing a new GB model by using different intrinsic radii to define the dielectric boundary requires either modifying the topology file by hand—a dangerous and tedious process—or learning and modifying the *tleap* source code and rebuilding the program all in the process of refining a set of parameters. With ParmEd, users and method developers can rapidly prototype a new method in a reliable way. A primary goal of ParmEd is to enable safe, rapid prototyping of new methods that require straightforward changes to the prmtop file.

A second motivator for creating ParmEd was to provide a unified platform for disseminating prmtop modifications that may be required for a particular method. The traditional approach when a method required a prmtop modification was for the developer that released the new code to develop a stand-alone tool in their programming language of choice to be released alongside Amber. These tools often parsed and modified topology files in a minimalistic fashion, and are not used or tested frequently. Such an approach quickly becomes unsustainable as the authors of these tools leave the developer community (*e.g.,* through graduation or retirement). With ParmEd, I sought to create a simple platform to unify prmtop modifying programs within Amber in an attempt to ease the burden of support and simplify the user experience. Therefore, ParmEd should be intuitive to use for experienced Amber users, and written in a way that the code can be easily understood by other developers.

6.2.2 Implementation and Capabilities

I wrote ParmEd as a set of two Python scripts built on top of a common library of functionality. The first, *parmed.py*, is a command-line tool that strongly resembles the popular trajectory analysis programs *ptraj* and *cpptraj* in its use. The second, *xparmed.py*, is a graphical user interface built on the Tcl/Tk toolkit through the *Tkinter*

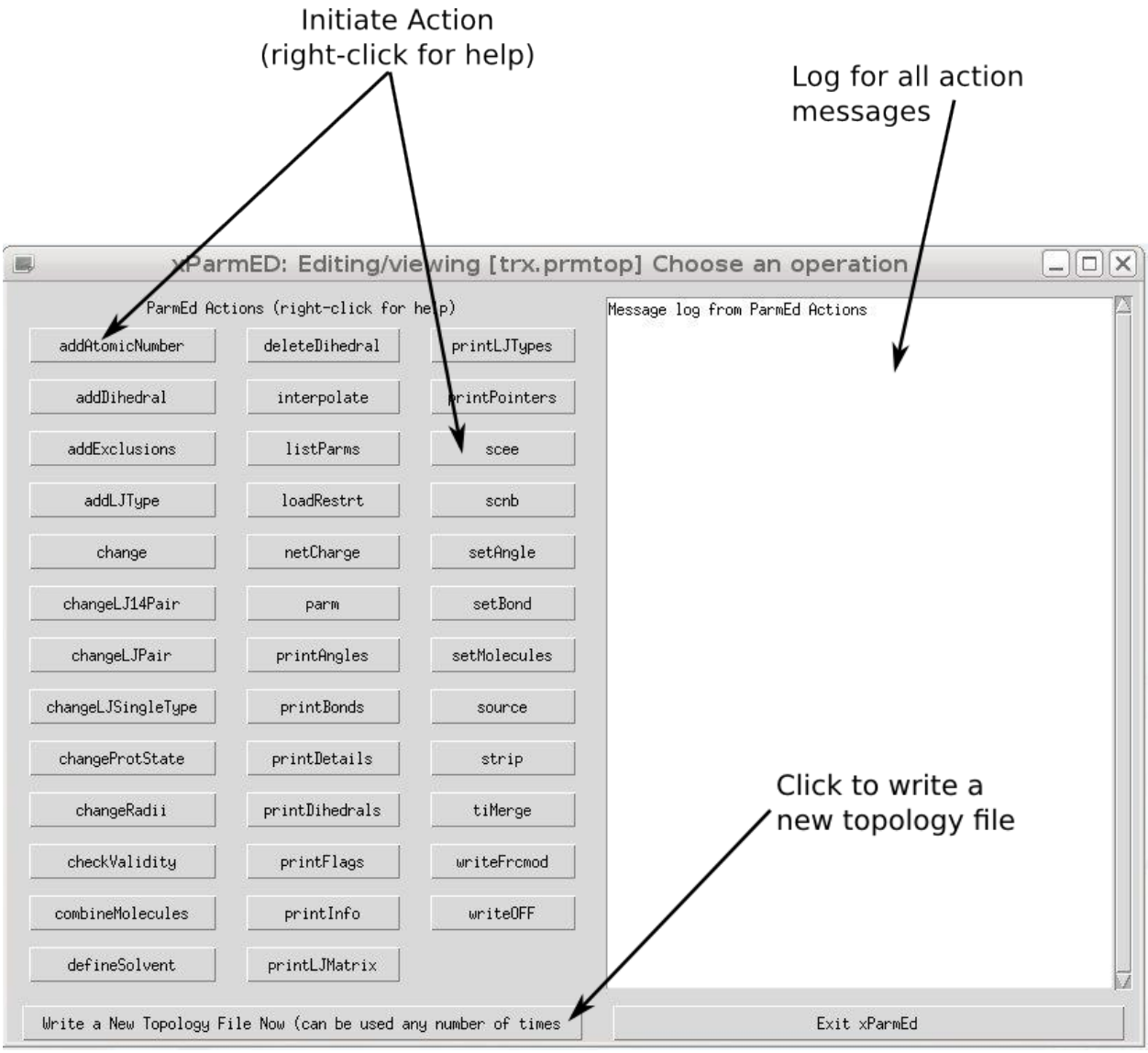

Figure 6-3. Screenshot of the *xparmed.py* GUI window, labeled with the available Actions and a message log.

Python bindings. The GUI, shown in Fig. 6-3, is meant to be a very simple, point-and-click interface for prmtop modification, while *parmed.py* is ideal for scripting purposes. To further simplify the use of ParmEd to those familiar with other Amber programs, the ubiquitous Amber mask syntax is used to specify all necessary atom selections.

The individual capabilities of ParmEd, called Actions, are all subclassed from a common *Action* base class. Each Action interprets its own list of arguments and implements its own, unique functionality. To expand the utility of the ParmEd code,

users can incorporate individual ParmEd Actions into their own Python script through an Application Programmer Interface (API) documented in the AmberTools manual. This allows users to avoid the need to learn the inner-workings of the prmtop file and re-implement existing code in the cases where ParmEd does not handle all of the users' needs.

In the following sections, I will outline some of the Actions and functionality I consider to be particularly helpful or particularly challenging to implement through other programs.

6.2.2.1 Lennard-Jones Parameter Modifications

I will briefly describe here how the radius (r_i) and well depth (ε_i) assigned in the parameter databases for each atom type i is translated into a set of parameters used to compute the LJ potential in the Amber force field. Specifically, between pairs i and j, the well depth $\varepsilon_{i,j}$ is the geometric average and the radius $r_{i,j}$ is the arithmetic average

$$\begin{aligned} \varepsilon_{i,j} &= \sqrt{\varepsilon_i \varepsilon_j} \\ R_{min,i,j} &= R_{min,i} + R_{min,j} \end{aligned} \tag{6–3}$$

These combined radii and depths are then combined into A-coefficients and B-coefficients using the equations

$$\begin{aligned} ACOEF_{i,j} &= \varepsilon_{i,j} r_{i,j}^{-12} \\ BCOEF_{i,j} &= 2\varepsilon_{i,j} r_{i,j}^{-6} \end{aligned} \tag{6–4}$$

Eqs. 6–3 and 6–4 are evaluated in *tleap*, and ε_i and r_i are provided as input in the parameter files. Because there are more ACOEF and BCOEF parameters than there are input parameters, the way *tleap* handles LJ parameters restricts some flexibility in the force field. The A and B coefficients can be thought of as a matrix of pairwise combined terms—defined in Eq. 6–4—in which only the diagonal terms are specified.

The interactions between each pair of atom types cannot be set independently like they can in the CHARMM program via the *NBFIX* keyword, for instance.

I will make a detour here to discuss how *tleap* compresses the number of LJ parameters written to the topology file. Since the LJ potential is composed of pairwise terms, there must be a term for every pair of atoms in the system—a number that becomes astronomically large for large numbers of particles. To avoid printing out on the order of N^2 terms in both coefficient matrices (where N is the total number of atoms), *tleap* assigns each atom to a particular atom type index that it shares with every other atom in the system that has the same set of starting LJ parameters ε_i and r_i. Therefore, each A- and B-coefficient printed in the topology file may be used for numerous other atom pairs in the force and energy evaluations.

I implemented a number of Actions in ParmEd that allow users to query and adjust LJ parameters in a way that is currently impossible with any other program. The `printLJTypes` Action in ParmEd takes an atom selection and prints out every other atom that has been assigned to the same LJ atom type. The *changeLJPair* Action allows users to adjust individual, off-diagonal elements of the A- and B-coefficient matrices for any pair of atoms. The `addLJType` command provides further flexibility by allowing the user to treat a subset of atoms as a different LJ atom type so any off-diagonal changes affect only the desired atoms.

6.2.2.2 Changing Atomic Properties

Another Action implemented in ParmEd—the `change` Action—allows users to change one of the following atomic properties: the partial charge, atomic mass, implicit solvent radius, implicit solvent screening factor, atom name, atom type name, atom type index, or the atomic number. Changing any of these properties without using ParmEd requires the user to modify a number of files, including standard residue libraries, force field databases, and the original starting structure before running those files through

tleap. Even then, care must be taken to ensure that the prmtop was changed the desired way.

This functionality allows rapid prototyping for tasks such as parameterizing new charge or implicit solvent radius sets. Alternatives are currently tedious and error-prone.

6.2.2.3 Setting up for H-REMD Simulations

The H-REMD implementation in Amber—described in Ch. 5—is capable of performing alchemical REFEP calculations provided that the alchemical pathway can be characterized by different topology files with the same atoms. When an atom disappears—like in a pK_a calculation when a proton vanishes—a dummy atom is required in the end state in which that atom is 'missing.' The `interpolate` Action is provided to create a series of prmtops whose charge and LJ parameters are linearly interpolated between two prmtops. Alternative approaches are, again, time consuming and error-prone.

6.2.2.4 Changing Parameters

Perhaps one of the strongest features of ParmEd is its ability to change individual bonded parameters—*i.e.,* bonds, angles, and torsions. The `setBond` and `setAngle` commands can be used to either add or modify a bond or angle parameter, respectively. The `addDihedral` and `deleteDihedral` commands can be used to create, remove, and even change individual torsion parameters. This control over the torsion parameters is particularly useful when attempting to fit new torsion parameters to improve force fields. [23–25]

APPENDIX A
NUMERICAL INTEGRATION IN CLASSICAL MOLECULAR DYNAMICS

A.1 Lagrangian and Hamiltonian Formulations

The *Lagrangian* and *Hamiltonian* formulations of classical mechanics—shown in Eqs. A–1 and A–2, respectively—offer a more convenient formalism than the more popularly known equations derived by Newton. [197] While Newton's equations apply in three-dimensional Cartesian space, they are not generally applicable to other coordinate systems (*e.g.,* polar and spherical-polar coordinates) that may be a more natural way to express certain problems. For instance, polar coordinates more naturally describe the mechanics of orbiting bodies than standard Euclidean space.

Lagrangian Equation. The Lagrangian function, $L = K - V$, where K is the kinetic energy and V is the potential energy, satisfies the Lagrangian equation (Eq. A–1) for m generalized coordinates (q_m). The advantage of Eq. A–1 is that it is derived without any assumption of a specific coordinate system for q_m. Generalized velocities are the first time-derivative of the generalized coordinates, $\dot{q}_m$. These generalized velocities are used to define the kinetic energy in the familiar form $K = 1/2\dot{q}_m^2$.

Another advantage to the Lagrangian formulation of classical mechanics is that the equations are still valid when subject to constraints on the dynamics of the system (as long as there are fewer constraints than particles). [197] This property is crucial for carrying out constrained dynamics, such as those simulations employing the commonly-used SHAKE, [16] RATTLE, [17] or SETTLE [18] algorithms, to name a few.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_m} - \frac{\partial L}{\partial q_m} = 0 \tag{A–1}$$

L in Eq. A–1 is the Lagrangian function mentioned above, q_m are the generalized coordinates of every particle in the system, and $\dot{q}_m$ are the set of corresponding generalized velocities. When applying Eq. A–1 to a system in the standard Cartesian coordinates without constraints, the familiar form of Newton's equations are recovered. [197]

Hamiltonian Equation. The Hamiltonian formulation of classical mechanics builds on the strengths of the Lagrangian formulation and provides a deeper insight into the physical behavior of classical systems. Unlike the Lagrangian, the Hamiltonian is defined as the total energy of the system: $H = K + V$. The Lagrangian of the system, $L = K - V$, plays an important part in Hamilton's formulation. The degrees of freedom in Hamilton's equation (Eq. A–2) are the generalized coordinates q_m as defined in the Lagrangian, and their conjugate momenta, p_m. The generalized coordinates and momenta are said to be canonically conjugate because they obey the relationship given in Eq. A–2. [197]

$$\begin{aligned} q_m &= \frac{\partial H}{\partial p_m} \\ p_m &= -\frac{\partial H}{\partial q_m} \end{aligned} \tag{A–2}$$

Now that a convenient formulation of the laws of classical dynamics are known, I will shift the discussion toward techniques by which these equations are used to integrate these second-order differential equations in typical molecular dynamics simulations.

A.2 Numerical Integration by Finite Difference Methods

The equations of motion are second-order differential equations with respect to the particle coordinates, since the force is proportional to the second time-derivative (*i.e.,* the acceleration) of those particles. Due to the typical size and complexity of the systems and their potentials studied in computational chemistry, MD simulations require numerical integration of the second-order differential equations of motion. In this section, I will describe two common approaches to iteratively integrating Eqs. A–1 and A–2—so-called *predictor-corrector* methods and the Verlet family of integrators.

A.2.1 Predictor-corrector

The predictor-corrector integrators are based on a simple Taylor-series expansion of the coordinates. Knowing that the velocity and acceleration are the first- and second-time derivatives of the particle positions, respectively, the Taylor expansions of each of

these quantities are given below.

$$\begin{aligned}
\vec{r}_p(t_0+\delta t) &= \vec{r}(t_0) + \delta t\,\vec{v}(t_0) + \delta t^2\,\frac{1}{2}\vec{a}(t_0) + \delta t^3\,\frac{1}{6}\frac{d^3\vec{r}(t)}{dt^3} + \ldots \\
\vec{v}_p(t_0+\delta t) &= \vec{v}(t_0) + \delta t\,a(t) + \frac{1}{2}\delta t^2\,\frac{d^3\vec{r}(t)}{dt^3} + \ldots \\
\vec{a}_p(t_0+\delta t) &= \vec{a}(t_0) + \delta t\,\frac{d^3\vec{r}(t)}{dt^3} + \ldots
\end{aligned} \tag{A–3}$$

The subscript p in these equations emphasizes that these are the *predicted* quantities of the positions, velocities, and accelerations at time $t_0 + \delta t$ based on the known values at time t_0.

It is convenient to truncate the Taylor series in Eqs. A–3 after the acceleration term since the acceleration at time t_0 can be easily calculated from the gradient of the potential energy function. Higher order terms are difficult to compute, and contribute a significantly smaller amount as the time step, δt, decreases. However, by truncating the Taylor expansion we used an approximation that will introduce systematic error of our predicted values calculated by Eqs. A–3 compared to their *true* values. There is a way of approximating the magnitude of the deviation of the predicted values from Eqs. A–3, however, that will allow a correction to be applied to the integrated values.

As a reminder, the gradient of the potential was used to calculate the forces—and therefore the acceleration—on each particle when making the initial integration step from t_0. The acceleration may be calculated again using the gradient of the potential at the predicted conformations:

$$\bigtriangledown\, V\left[\vec{r}_p(t_0+\delta t)\right] = m\vec{a}' \tag{A–4}$$

Since systematic error has been introduced by truncating the expansion in Eqs. A–3, $\vec{a}'$ from Eq. A–4 and $\vec{a}_p(t_0+\delta t)$ from Eq. A–3 will differ. The magnitude of this difference can be used to correct the predicted values according to Eqs. A–5.

$$\begin{aligned}
\vec{r}_c(t+\delta t) &= \vec{r}_p(t+\delta t) + c_0 \Delta\vec{a}(t+\delta t) \\
\vec{v}_c(t+\delta t) &= \vec{v}_p(t+\delta t) + c_1 \Delta\vec{a}(t+\delta t) \\
\vec{a}_c(t+\delta t) &= \vec{a}_p(t+\delta t) + c_2 \Delta\vec{a}(t+\delta t)
\end{aligned} \tag{A–5}$$

where the subscripts indicate the relationship between the *c*orrected and *p*redicted quantities, and the coefficients c_0, c_1, and c_2 are parametrized to maximize performance, [198, 199] and have the appropriate units to satisfy each equation. [54] The corrector process can be iterated until the desired level of agreement between the predicted and corrected values is reached.

While the predictor-corrector algorithm allows long time steps to be taken by fixing the resulting systematic error, the corrector step requires a full force evaluation of the system at a set of coordinates, which is the most time-consuming portion of the calculation. As a result, the corrector step is computationally demanding, and predictor-corrector methods have been replaced by other integration schemes in standard practice.

A.2.2 Verlet Integrators

Among the most popular types of integrators in common use today are based on the *Verlet* algorithms. The Verlet algorithm, developed in 1967 by Verlet, utilizes a Taylor series expansion of the particle coordinates about time t_0. The key to the Verlet approach is to use both the forward and reverse time steps, as shown in Eqs. A–6. [54]

$$\begin{aligned}
\vec{r}(t_0+\delta t) &= r(t_0) + \delta t\, \vec{v}(t_0) + \frac{1}{2}\delta t^2\, \vec{a}(t_0) + \ldots \\
\vec{r}(t_0-\delta t) &= r(t_0) - \delta t\, \vec{v}(t_0) + \frac{1}{2}\delta t^2\, \vec{a}(t_0) - \ldots
\end{aligned} \tag{A–6}$$

Combining Eqs. A–6 gives

$$\vec{r}(t_0 + \delta t) = 2\vec{r}(t_0) + \delta t^2\, \vec{a}(t_0) - \vec{r}(t_0 - \delta t) \tag{A–7}$$

where the velocities have been eliminated from the expression and are therefore unnecessary when integrating the equations of motion. Furthermore, like the velocities, the δt^3 term also cancels, so the Verlet algorithm is not only time-reversible given its symmetry around t_0 but also accurate to fourth order in the time step. The velocities are still useful, however, to compute the total kinetic energy and related properties, such as the instantaneous temperature. When necessary, velocities can be approximated as the average velocity over the time period from $t_0 - \delta t$ to $t_0 + \delta t$.

Performing MD using the Verlet algorithm requires storing the current positions, 'old' positions at time $t_0 - \delta t$, and the accelerations at time t_0—a modest cost given the accuracy of the integration scheme. However, the use of Eq. A–7 introduces an issue of numerical precision, since $\vec{r}(t_0)$ and $\vec{r}(t_0 - \delta t)$ are potentially large values, while $\delta t^2\, \vec{a}(t_0)$ is typically quite small since the time step is small. Since real numbers can be stored only to a limited precision, accuracy is potentially lost when a small number is added to a difference of large numbers. [54] To address this issue and improve the way in which velocities are handled, the leap-frog and velocity Verlet methods are discussed below.

Velocity Verlet. In 1982, Swope et al. developed a variant of the Verlet algorithm that sidesteps the potential roundoff errors and naturally stores positions, velocities, and accelerations at the same time. A Taylor series expansion is again used to propagate the positions, but only the $t_0 + \delta t$ step is used, resulting in Eq. A–8.

$$\vec{r}(t_0 + \delta t) = \vec{r}(t_0) + \delta t\, \vec{v}(t_0) + \frac{1}{2}\delta t^2\, \vec{a}(t_0) \tag{A–8}$$

The accelerations of the particles are computed from their positions at time $t_0 + \delta t$, and are used to compute the velocities. To increase the accuracy of the computed velocities, the velocity integration is divided into two half-timesteps, shown in Eqs. A–9. In this

case, the accuracy to δt^4 in the positions obtained by the Verlet algorithm is sacrificed for improved numerical precision for finite-precision computers and a more accurate treatment of system velocities.

$$\begin{aligned}
\vec{v}\left(t_0 + \frac{1}{2}\delta t\right) &= \vec{v}(t_0) + \frac{1}{2}\delta t\,\vec{a}(t_0) \\
\vec{v}(t_0 + \delta t) &= \vec{v}\left(t_0 + \frac{1}{2}\delta t\right) + \frac{1}{2}\delta t\,\vec{a}(t_0 + \delta t) \\
\vec{v}(t_0 + \delta t) &= \vec{v}(t) + \frac{1}{2}\delta t\,[\vec{a}(t) + \vec{a}(t + \delta t)]
\end{aligned} \tag{A–9}$$

The *NAB* and *mdgx* programs of the AmberTools 12 program suite (and earlier versions, where available), utilize the velocity Verlet algorithm for dynamics.

Leap-frog. A common integrator used in MD simulations is the *leap-frog* method, so-called because the computed velocities 'leap' over the computed coordinates in a manner that will be explained shortly. The main dynamics engines in the Amber 12 program suite—pmemd and sander—use the leap-frog integrator.

While similar to the velocity Verlet approach, the leap-frog algorithm computes positions and accelerations of particles at integral time steps, but computes velocities at half-integral time steps according to Eqs. A–10.

$$\begin{aligned}
\vec{r}(t_0 + \delta t) &= \vec{r}(t_0) + \delta t\,\vec{v}\left(t_0 + \frac{1}{2}\delta t\right) \\
\vec{v}\left(t_0 + \frac{1}{2}\delta t\right) &= \vec{v}\left(t_0 - \frac{1}{2}\delta t\right) + \delta t\,\vec{a}(t_0)
\end{aligned} \tag{A–10}$$

If the velocities are required at time t_0, they can be estimated as the average velocities between times $t_0 - 1/2\delta t$ and $t_0 + 1/2\delta t$, which is significantly more accurate than the approximation in Verlet's original algorithm. Like the velocity Verlet algorithm, leap-frog integration sacrifices the 4th-order accuracy in integrated positions to alleviate the aforementioned precision and velocity issues. [54]

APPENDIX B
AMBER PARAMETER-TOPOLOGY FILE FORMAT

This appendix details the Parameter-Topology file format used extensively by the AMBER software suite for biomolecular simulation and analysis, referred to as the *prmtop* file for short. The format specification of the AMBER topology file was written initially over a decade ago and posted on http://ambermd.org/formats.html. I have recently expanded that document to account for the drastic change to the file format that occurred with the 2004 release of Amber 7. The pre-Amber 7 format (*old format*) is described more briefly afterwards, although each section provided in the original format contains exactly the same information as the newer version.

This appendix also details the format changes and additions introduced by *chamber*—the program that translates a CHARMM parameter file (PSF) into a topology file that can be used with the *sander* and *pmemd* programs in AMBER.

This appendix draws from the information on http://ambermd.org/formats.html that was added by both me and others, as well as the experience I gleaned while writing the ParmEd program and working with the various codes in AMBER.

As a warning, the prmtop file is a result of bookkeeping that becomes increasingly complex as the system size increases. Therefore, hand-editing the topology file for all but the smallest systems is discouraged—a program or script should be written to automate the procedure.

B.1 Layout

The first line of the Amber topology file is the version string. An example is shown below in which `XX` is replaced by the actual date and time.

```
%VERSION  VERSION_STAMP = V0001.000  DATE = XX/XX/XX  XX:XX:XX
```

The topology format is divided into several sections in a way that is designed to be parsed easily using simple Fortran code. A consequence of this is that it is difficult for parsers written in other languages (*e.g.,* C, C++, Python, etc.) to strictly adhere to the

standard. These parsers should try, however, to support as much of the standard as possible.

```
%FLAG SECTION
%COMMENT an arbitrary number of optional comments may be put here
%FORMAT(<FORTRAN FORMAT>)
... data formatted according to <FORTRAN FORMAT>
```

All names (*e.g.,* atom names, atom type names, and residue names) are limited to 4 characters and are printed in fields of width *exactly* 4 characters wide, left-justified. This means that names might not be space-delimited if any of the names have 4 characters.

Requirements for prmtop parsers. Parsers, regardless of the language they are written in, should conform to a list of attributes to maximize the likelihood that they are parsed correctly.

- Parsers should expect that some 4-character fields (*e.g.,* atom or residue names) may have some names that have 4 characters and therefore might not be whitespace-delimited.
- Parsers should not expect `SECTION`s in the prmtop to be in any particular order.
- Parsers should not expect or require `%COMMENT` lines to exist, but should properly parse the file if any number of `%COMMENT` lines appear as indicated above
- The topology file may be assumed to have been generated 'correctly' by *tleap* or some other credible source. No graceful error checking is required.

Requirements for modifying `SECTION`s. To minimize the impact of prmtop changes to existing, third-party parsers, the following conventions should be followed.

- Any new `SECTION` should be added to the end of the topology file to avoid conflicts with order-dependent parsers.
- The `<FORTRAN FORMAT>` should be as simple as possible (and avoid adding new formats) to maintain simplicity for non-Fortran parsers.

- Avoid modifying if possible. Consider if this new section or change is truly necessary *and* belongs in the prmtop.

B.2 List of SECTIONs

TITLE

This section contains the title of the topology file on one line (up to 80 characters). While the title serves a primarily cosmetic purpose, this section must be present.

```
%FORMAT(20a4)
```

POINTERS

This section contains the information about how many parameters are present in all of the sections. There are 31 or 32 integer pointers (NCOPY might not be present). The format and names of all of the pointers are listed below, followed by a description of each pointer.

```
%FLAG POINTERS
%FORMAT(10I8)
NATOM  NTYPES NBONH  MBONA  NTHETH MTHETA NPHIH  MPHIA  NHPARM NPARM
NNB    NRES   NBONA  NTHETA NPHIA  NUMBND NUMANG NPTRA  NATYP  NPHB
IFPERT NBPER  NGPER  NDPER  MBPER  MGPER  MDPER  IFBOX  NMXRS  IFCAP
NUMEXTRA NCOPY
```

NATOM Number of atoms

NTYPES Number of distinct Lennard-Jones atom types

NBONH Number of bonds containing Hydrogen

MBONA Number of bonds not containing Hydrogen

NTHETH Number of angles containing Hydrogen

MTHETA Number of angles not containing Hydrogen

NPHIH Number of torsions containing Hydrogen

MPHIA Number of torsions not containing Hydrogen

NHPARM Not currently used for anything

NPARM Used to determine if this is a LES-compatible prmtop

NNB Number of excluded atoms (length of total exclusion list)

NRES Number of residues

NBONA MBONA + number of constraint bonds [1]

NTHETA MTHETA + number of constraint angles [1]

NPHIA MPHIA + number of constraint torsions [1]

NUMBND Number of unique bond types

NUMANG Number of unique angle types

NPTRA Number of unique torsion types

NATYP Number of SOLTY terms. Currently unused.

NPHB Number of distinct 10-12 hydrogen bond pair types [2]

IFPERT Set to 1 if topology contains residue perturbation information. [3]

NBPER Number of perturbed bonds [3]

NGPER Number of perturbed angles [3]

NDPER Number of perturbed torsions [3]

MBPER Number of bonds in which both atoms are being perturbed

MGPER Number of angles in which all 3 atoms are being perturbed

MDPER Number of torsions in which all 4 atoms are being perturbed [1]

IFBOX Flag indicating whether a periodic box is present. Values can be 0 (no box), 1 (orthorhombic box) or 2 (truncated octahedron)

NMXRS Number of atoms in the largest residue

IFCAP Set to 1 if a solvent CAP is being used

NUMEXTRA Number of extra points in the topology file

---

[1] AMBER codes no longer support constraints in the topology file.

[2] Modern AMBER force fields do not use a 10-12 potential

[3] No AMBER codes support perturbed topologies anymore

NCOPY Number of PIMD slices or number of beads

ATOM_NAME

This section contains the atom name for every atom in the prmtop.

`%FORMAT(20a4)` There are `NATOM` 4-character strings in this section.

CHARGE

This section contains the charge for every atom in the prmtop. Charges are multiplied by 18.2223 ($\sqrt{k_{ele}}$ where k_{ele} is the electrostatic constant in $kcal\,\mathring{A}\,mol^{-1}\,q^{-2}$, where q is the charge of an electron).

`%FORMAT(5E16.8)`

There are `NATOM` floating point numbers in this section.

ATOMIC_NUMBER

This section contains the atomic number of every atom in the prmtop. This section was first introduced in AmberTools 12. [141]

`%FORMAT(10I8)`

There are `NATOM` integers in this section.

MASS

This section contains the atomic mass of every atom in $g\,mol^{-1}$.

`%FORMAT(5E16.8)`

There are `NATOM` floating point numbers in this section.

ATOM_TYPE_INDEX

This section contains the Lennard-Jones atom type index. The Lennard-Jones potential contains parameters for every pair of atoms in the system. To minimize the memory requirements of storing `NATOM` $\times$ `NATOM` [2] Lennard-Jones A-coefficients and B-coefficients, all atoms with the same σ and ε parameters are assigned to the same type

[2] Only half this number would be required, since $a_{i,j} \equiv a_{j,i}$

(regardless of whether they have the same AMBER_ATOM_TYPE). This significantly reduces the number of LJ coefficients which must be stored, but introduced the requirement for bookkeeping sections of the topology file to keep track of what the LJ type index was for each atom.

This section is used to compute a pointer into the NONBONDED_PARM_INDEX section, which itself is a pointer into the LENNARD_JONES_ACOEF and LENNARD_JONES_BCOEF sections (see below).

%FORMAT(10I8)

There are NATOM integers in this section.

NUMBER_EXCLUDED_ATOMS

This section contains the number of atoms that need to be excluded from the non-bonded calculation loop for atom *i* because *i* is involved in a bond, angle, or torsion with those atoms. Each atom in the prmtop has a list of excluded atoms that is a subset of the list in EXCLUDED_ATOMS_LIST (see below). The *i*th value in this section indicates how many elements of EXCLUDED_ATOMS_LIST belong to atom *i*.

For instance, if the first two elements of this array is 5 and 3, then elements 1 to 5 in EXCLUDED_ATOMS_LIST are the exclusions for atom 1 and elements 6 to 8 in EXCLUDED_ATOMS_LIST are the exclusions for atom 2. Each exclusion is listed only once in the topology file, and is given to the atom with the smaller index. That is, if atoms 1 and 2 are bonded, then atom 2 is in the exclusion list for atom 1, but atom 1 is *not* in the exclusion list for atom 2. If an atom has no excluded atoms (either because it is a monoatomic ion or all atoms it forms a bonded interaction with have a smaller index), then it is given a value of 1 in this list which corresponds to an exclusion with (a non-existent) atom 0 in EXCLUDED_ATOMS_LIST.

The exclusion rules for extra points are more complicated. When determining exclusions, it is considered an ‘extension’ of the atom it is connected (bonded) to.

Therefore, extra points are excluded not only from the atom they are connected to, but also from every atom that its parent atom is excluded from.

NOTE: The non-bonded interaction code in *sander* and *pmemd* currently (as of Amber 12) recalculates the exclusion lists for simulations of systems with periodic boundary conditions, so this section is effectively ignored. The GB code uses the exclusion list in the topology file.

%FORMAT(10I8)

There are NATOM integers in this section.

NONBONDED_PARM_INDEX

This section contains the pointers for each pair of LJ atom types into the LENNARD_JONES_ACOEF and LENNARD_JONES_BCOEF arrays (see below). The pointer for an atom pair in this array is calculated from the LJ atom type index of the two atoms (see ATOM_TYPE_INDEX above).

The index for two atoms i and j into the LENNARD_JONES_ACOEF and LENNARD_JONES_BCOEF arrays is calculated as

$$index = \texttt{NONBONDED\_PARM\_INDEX}\,[\texttt{NTYPES} \times (\texttt{ATOM\_TYPE\_INDEX}(i) - 1) + \texttt{ATOM\_TYPE\_INDEX}(j)] \quad \text{(B–1)}$$

Note, each atom pair can interact with either the standard 12-6 LJ potential *or* via a 12-10 hydrogen bond potential. If *index* in Eq. B–1 is negative, then it is an index into HBOND_ACOEF and HBOND_BCOEF instead (see below).

%FORMAT(10I8)

There are NTYPES $\times$ NTYPES integers in this section.

RESIDUE_LABEL

This section contains the residue name for every residue in the prmtop. Residue names are limited to 4 letters, and might not be whitespace-delimited if any residues have 4-letter names.

%FORMAT(20a4)

There are NRES 4-character strings in this section.

RESIDUE_POINTER

This section lists the first atom in each residue.

%FORMAT(10i8)

There are NRES integers in this section.

BOND_FORCE_CONSTANT

Bond energies are calculated according to the equation

$$E_{bond} = \frac{1}{2} k \left(\vec{r} - \vec{r}_{eq}\right)^2 \tag{B–2}$$

This section lists all of the bond force constants (k in Eq. B–2) in units $kcal\ mol^{-1}\ \AA^{-2}$ for each unique bond type. Each bond in BONDS_INC_HYDROGEN and BONDS_WITHOUT_HYDROGEN (see below) contains an index into this array.

%FORMAT(5E16.8)

There are NUMBND floating point numbers in this section.

BOND_EQUIL_VALUE

This section lists all of the bond equilibrium distances ($\vec{r}_{eq}$ in Eq. B–2) in units of Å for each unique bond type. This list is indexed the same way as BOND_FORCE_CONSTANT.

%FORMAT(5E16.8)

There are NUMBND floating point numbers in this section.

ANGLE_FORCE_CONSTANT

Angle energies are calculated according to the equation

$$E_{angle} = \frac{1}{2} k_\theta \left(\theta - \theta_{eq}\right)^2 \tag{B–3}$$

This section lists all of the angle force constants (k_θ in Eq. B–3) in units of $kcal\ mol^{-1}\ rad^2$ for each unique angle type. Each angle in ANGLES_INC_HYDROGEN and ANGLES_WITHOUT_HYDROGEN contains an index into this (and the next) array.

%FORMAT(5E16.8)

There are NUMANG floating point numbers in this section.

ANGLE_EQUIL_VALUE

This section contains all of the angle equilibrium angles (θ_{eq} in Eq. B–3) in radians. *NOTE:* the AMBER parameter files list equilibrium angles in degrees and are converted to radians in *tleap*. This list is indexed the same way as ANGLE_FORCE_CONSTANT.

%FORMAT(5E16.8)

There are NUMBND floating point numbers in this section.

DIHEDRAL_FORCE_CONSTANT

Torsion energies are calculated for each term according to the equation

$$E_{torsion} = k_{tor} \cos(n\phi + \psi) \tag{B–4}$$

This section lists the torsion force constants (k_{tor} in Eq. B–4) in units of $kcal\ mol^{-1}$ for each unique torsion type. Each torsion in DIHEDRALS_INC_HYDROGEN and DIHEDRALS_WITHOUT_HYDROGEN has an index into this array.

Amber parameter files contain a dividing factor and barrier height for each dihedral. The barrier height in the parameter files are divided by the provided factor inside *tleap* and then discarded. As a result, the torsion barriers in this section might not match those in the original parameter files.

%FORMAT(5E16.8)

There are NPTRA floating point numbers in this section.

DIHEDRAL_PERIODICITY

This section lists the periodicity (n in Eq. B–4) for each unique torsion type. It is indexed the same way as DIHEDRAL_FORCE_CONSTANT. *NOTE:* only integers are read by *tleap*, although the AMBER codes support non-integer periodicities.

%FORMAT(5E16.8)

There are NPTRA floating point numbers in this section.

DIHEDRAL_PHASE

This section lists the phase shift (ψ in Eq. B–4) for each unique torsion type in radians. It is indexed the same way as DIHEDRAL_FORCE_CONSTANT.

%FORMAT(5E16.8)

There are NPTRA floating point numbers in this section.

SCEE_SCALE_FACTOR

This section was introduced in Amber 11. In previous versions, this variable was part of the input file and set a single scaling factor for every torsion.

This section lists the factor by which 1-4 electrostatic interactions are divided (*i.e.,* the two atoms on either end of a torsion). For torsion types in which 1-4 non-bonded interactions are not calculated (*e.g.,* improper torsions, multi-term torsions, and those involved in ring systems of 6 or fewer atoms), a value of 0 is assigned by *tleap*. This section is indexed the same way as DIHEDRAL_FORCE_CONSTANT.

%FORMAT(5E16.8)

There are NPTRA floating point numbers in this section.

SCNB_SCALE_FACTOR

This section was introduced in Amber 11. In previous versions, this variable was part of the input file and set a single scaling factor for every torsion.

This section lists the factor by which 1-4 van der Waals interactions are divided (*i.e.,* the two atoms on either end of a torsion). This section is analogous to SCEE_SCALE_FACTOR described above.

%FORMAT(5E16.8)

There are NPTRA floating point numbers in this section.

SOLTY

This section is currently unused, and while 'future use' is planned, this assertion has lain dormant for some time.

%FORMAT(5E16.8)

There are NATYP floating point numbers in this section.

LENNARD_JONES_ACOEF

LJ non-bonded interactions are calculated according to the equation

$$E_{LJ} = \frac{a_{i,j}}{r^{12}} - \frac{b_{i,j}}{r^6} \tag{B–5}$$

This section contains the LJ A-coefficients ($a_{i,j}$ in Eq. B–5) for all pairs of distinct LJ types (see sections ATOM_TYPE_INDEX and NONBONDED_PARM_INDEX above).

%FORMAT(5E16.8)

There are $[\text{NTYPES} \times (\text{NTYPES} + 1)]\,/2$ floating point numbers in this section.

LENNARD_JONES_BCOEF

This section contains the LJ B-coefficients ($b_{i,j}$ in Eq. B–5) for all pairs of distinct LJ types (see sections ATOM_TYPE_INDEX and NONBONDED_PARM_INDEX above).

%FORMAT(5E16.8)

There are $[\text{NTYPES} \times (\text{NTYPES} + 1)]\,/2$ floating point numbers in this section.

BONDS_INC_HYDROGEN

This section contains a list of every bond in the system in which at least one atom is Hydrogen. Each bond is identified by 3 integers—the two atoms involved in the bond and the index into the BOND_FORCE_CONSTANT and BOND_EQUIL_VALUE. For run-time efficiency, the atom indexes are actually indexes into a coordinate array, so the actual atom index A is calculated from the coordinate array index N by $A = N/3 + 1$. (N is the value in the topology file)

%FORMAT(10I8)

There are $3 \times$ NBONH integers in this section.

BONDS_WITHOUT_HYDROGEN

This section contains a list of every bond in the system in which neither atom is Hydrogen. It has the same structure as BONDS_INC_HYDROGEN described above.

%FORMAT(10I8)

There are $3 \times$ NBONA integers in this section.

ANGLES_INC_HYDROGEN

This section contains a list of every angle in the system in which at least one atom is Hydrogen. Each angle is identified by 4 integers—the three atoms involved in the angle and the index into the ANGLE_FORCE_CONSTANT and ANGLE_EQUIL_VALUE. For run-time efficiency, the atom indexes are actually indexes into a coordinate array, so the actual atom index A is calculated from the coordinate array index N by $A = N/3 + 1$. (N is the value in the topology file)

%FORMAT(10I8)

There are $4 \times$ NTHETH integers in this section.

ANGLES_WITHOUT_HYDROGEN

This section contains a list of every angle in the system in which no atom is Hydrogen. It has the same structure as ANGLES_INC_HYDROGEN described above.

%FORMAT(10I8)

There are $4 \times$ NTHETA integers in this section.

DIHEDRALS_INC_HYDROGEN

This section contains a list of every torsion in the system in which at least one atom is Hydrogen. Each torsion is identified by 5 integers—the four atoms involved in the torsion and the index into the DIHEDRAL_FORCE_CONSTANT, DIHEDRAL_PERIODICITY, DIHEDRAL_PHASE, SCEE_SCALE_FACTOR and SCNB_SCALE_FACTOR arrays. For run-time efficiency, the atom indexes are actually indexes into a coordinate array, so the actual atom index A is calculated from the coordinate array index N by $A = N/3 + 1$. (N is the value in the topology file)

If the third atom is negative, then the 1-4 non-bonded interactions for this torsion is not calculated. This is required to avoid double-counting these non-bonded interactions in some ring systems and in multi-term torsions.

If the fourth atom is negative, then the torsion is improper.

NOTE: The first atom has an index of zero. Since 0 cannot be negative and the 3rd and 4th atom indexes are tested for their sign to determine if 1-4 terms are calculated, the first atom in the topology file must be listed as either the first or second atom in whatever torsions it is defined in. The atom ordering in a torsion can be reversed to accommodate this requirement if necessary.

%FORMAT(10I8)

There are $5 \times$ NPHIH integers in this section.

DIHEDRALS_WITHOUT_HYDROGEN

This section contains a list of every torsion in the system in which no atom is Hydrogen. It has the same structure as DIHEDRALS_INC_HYDROGEN described above.

%FORMAT(10I8)

There are $5 \times$ NPHIA integers in this section.

EXCLUDED_ATOMS_LIST

This section contains a list for each atom of excluded partners in the non-bonded calculation routines. The subset of this list that belongs to each atom is determined from the pointers in NUMBER_EXCLUDED_ATOMS—see that section for more information.

NOTE: The periodic boundary code in *sander* and *pmemd* currently recalculates this section of the topology file. The GB code, however, uses the exclusion list defined in the topology file.

%FORMAT(10I8)

There are NNB integers in this section.

HBOND_ACOEF

This section is analogous to the LENNARD_JONES_ACOEF array described above, but refers to the A-coefficient in a 12-10 potential instead of the familiar 12-6 potential. This term has been dropped from most modern force fields.

%FORMAT(5E16.8)

There are NPHB floating point numbers in this section.

HBOND_BCOEF

This section is analogous to the LENNARD_JONES_BCOEF array described above, but refers to the B-coefficient in a 12-10 potential instead of the familiar 12-6 potential. This term has been dropped from most modern force fields.

%FORMAT(5E16.8)

There are NPHB floating point numbers in this section.

HBCUT

This section used to be used for a cutoff parameter in the 12-10 potential, but is no longer used for anything.

%FORMAT(5E16.8)

There are NPHB floating point numbers in this section.

AMBER_ATOM_TYPE

This section contains the atom type name for every atom in the prmtop.

%FORMAT(20a4)

There are NATOM 4-character strings in this section.

TREE_CHAIN_CLASSIFICATION

This section contains information about the tree structure (borrowing concepts from *graph theory*) of each atom. Each atom can have one of the following character indicators:

M This atom is part of the “main chain”

S This atom is part of the “sidechain”

E This atom is a chain-terminating atom (*i.e.,* an “end” atom)

3 The structure branches into 3 chains at this point

BLA If none of the above are true

%FORMAT(20a4)

There are NATOM 4-character strings in this section.

JOIN_ARRAY

This section is no longer used and is currently just filled with zeros.

%FORMAT(10I8)

There are NATOM integers in this section.

IROTAT

This section is not used and is currently just filled with zeros.

%FORMAT(10I8)

There are NATOM integers in this section.

SOLVENT_POINTERS

This section is only present if IFBOX is greater than 0 (*i.e.,* if the system was set up for use with periodic boundary conditions). There are 3 integers present in this section—the final residue that is part of the solute (IPTRES), the total number of 'molecules' (NSPM), and the first solvent 'molecule' (NSPSOL).

A 'molecule' is defined as a closed graph—that is, there is a pathway from every atom in a molecule to every other atom in the molecule by traversing bonds, and there are no pathways to 'other' molecules.

%FLAG SOLVENT_POINTERS

%FORMAT(3I8)

IPTRES NSPM NSPSOL

ATOMS_PER_MOLECULE

This section is only present if IFBOX is greater than 0 (*i.e.,* if the system was set up for use with periodic boundary conditions). This section lists how many atoms are present in each 'molecule' as defined in the SOLVENT_POINTERS section above.

%FORMAT(10I8)

There are NSPM integers in this section (see the SOLVENT_POINTERS section above).

BOX_DIMENSIONS

This section is only present if IFBOX is greater than 0 (*i.e.,* if the system was set up for use with periodic boundary conditions). This section lists the box angle (OLDBETA) and dimensions (BOX(1)×BOX(2)×BOX(3)). The values in this section are deprecated now since newer and more accurate information about the box size and shape is stored in the coordinate file. Since constant pressure simulations can change the box dimensions, the values in the coordinate file should be trusted over those in the topology file.

```
%FLAG BOX_DIMENSIONS
%FORMAT(5E16.8)
OLDBETA          BOX(1)          BOX(2)          BOX(3)
```

CAP_INFO

This section is present only if IFCAP is not 0. If present, it contains a single integer which is the last atom before the water cap begins (NATCAP)

```
%FORMAT(10I8)
```

CAP_INFO2

This section is present only if IFCAP is not 0. If present, it contains four numbers—the distance from the center of the cap to outside the cap (CUTCAP), and the Cartesian coordinates of the cap center.

```
%FLAG CAP_INFO2
%FORMAT(5E16.8)
CUTCAP           XCAP            YCAP            ZCAP
```

RADIUS_SET

This section contains a one-line string (up to 80 characters) describing the intrinsic implicit solvent radii set that are defined in the topology file. The available radii sets with their 1-line descriptions are:

bondi `Bondi radii (bondi)`

amber6 `amber6 modified Bondi radii (amber6)`

mbondi `modified Bondi radii (mbondi)`

mbondi2 `H(N)-modified Bondi radii (mbondi2)`

mbondi3 `ArgH and AspGlu0 modified Bondi2 radii (mbondi3)`

`%FORMAT(1a80)`

There is a single line description in this section.

RADII

This section contains the intrinsic radii of every atom used for implicit solvent calculations (typically Generalized Born).

`%FORMAT(5E16.8)`

There are `NATOM` floating point numbers in this section.

IPOL

This section was introduced in Amber 12. In previous versions of Amber, this was a variable in the input file.

This section contains a single integer that is 0 for fixed-charge force fields and 1 for force fields that contain polarization.

POLARIZABILITY

This section is only present if `IPOL` is not 0. It contains the atomic polarizabilities for every atom in the prmtop.

`%FORMAT(5E16.8)`

There are `NATOM` floating point numbers in this section.

`%FORMAT(1I8)`

B.3 Deprecated Sections

All of the sections of the topology file listed here are only present if `IFPERT` is 1. However, no modern programs support such prmtops so these sections are rarely (if ever) used. They are included in Table B-1 for completeness, only.

More info can be found online at http://ambermd.org/formats.html

Table B-1. List of all of the perturbed topology file sections.

FLAG name	%FORMAT	of values	Description
PERT_BOND_ATOMS	10I8	$2 \times$ NBPER	perturbed bond list
PERT_BOND_PARAMS	10I8	$2 \times$ NBPER	perturbed bond pointers
PERT_ANGLE_ATOMS	10I8	$3 \times$ NGPER	perturbed angle list
PERT_ANGLE_PARAMS	10I8	$2 \times$ NGPER	perturbed angle pointers
PERT_DIHEDRAL_ATOMS	10I8	$4 \times$ NDPER	perturbed torsion list
PERT_DIHEDRAL_PARAMS	10I8	$2 \times$ NDPER	perturbed torsion pointers
PERT_RESIDUE_NAME	20a4	NRES	end state residue names
PERT_ATOM_NAME	20a4	NATOM	end state atom names
PERT_ATOM_SYMBOL	20a4	NATOM	end state atom types
ALMPER	5E16.8	NATOM	Unused
IAPER	10I8	NATOM	Is Atom PERturbed?
PERT_ATOM_TYPE_INDEX	10I8	NATOM	Perturbed LJ Type
PERT_CHARGE	5E16.8	NATOM	Perturbed charge

B.4 CHAMBER Topologies

Here we will describe the general format of topology files generated by the *chamber* program. The *chamber* program was developed to translate CHARMM topology (PSF) files into Amber topology files for use with the AMBER program suite.

Due to differences in the CHARMM force field (*e.g.,* the extra CMAP and Urey-Bradley terms and the different way that improper dihedrals are treated), chamber topologies contain more sections than Amber topologies. Furthermore, to ensure rigorous reproduction of CHARMM energies inside the AMBER program suites, some of the sections that are common between AMBER and CHARMM topology files have a different format for their data to support a different level of input data precision.

Due to the differences in the *chamber* topology files, a mechanism to differentiate between *chamber* topologies and AMBER topologies was introduced. If the topology file has a %FLAG TITLE then it is an AMBER topology. If it has a %FLAG CTITLE instead, then it is a *chamber* topology.

The following sections of the *chamber* topology are exacly the same as those from the AMBER topology files:

- POINTERS

- ATOM_NAME
- MASS
- ATOM_TYPE_INDEX
- NUMBER_EXCLUDED_ATOMS
- EXCLUDED_ATOMS_LIST
- NONBONDED_PARM_INDEX
- RESIDUE_LABEL
- BOND_FORCE_CONSTANT
- BOND_EQUIL_VALUE
- ANGLE_FORCE_CONSTANT
- DIHEDRAL_FORCE_CONSTANT
- DIHEDRAL_PERIODICITY
- DIHEDRAL_PHASE
- SCEE_SCALE_FACTOR
- SCNB_SCALE_FACTOR
- SOLTY
- BONDS_INC_HYDROGEN
- BONDS_WITHOUT_HYDROGEN
- ANGLES_INC_HYDROGEN
- ANGLES_WITHOUT_HYDROGEN
- DIHEDRALS_INC_HYDROGEN
- DIHEDRALS_WITHOUT_HYDROGEN
- HBOND_ACOEF
- HBOND_BCOEF
- HBCUT
- AMBER_ATOM_TYPE

Table B-2. List of flags that are common between Amber and chamber topology files, but have different FORMAT identifiers.

FLAG name	AMBER Format	chamber Format
CHARGE	5E16.8	3E24.16
ANGLE_EQUIL_VALUE	5E16.8	3E25.17
LENNARD_JONES_ACOEF	5E16.8	3E24.16
LENNARD_JONES_BCOEF	5E16.8	3E24.16

- TREE_CHAIN_CLASSIFICATION [3]
- JOIN_ARRAY
- IROTAT
- RADIUS_SET
- RADII
- SCREEN
- SOLVENT_POINTERS
- ATOMS_PER_MOLECULE

In Table B-2 is a list of sections that have the same name and the same data, but with a different Fortran format identifier.

FORCE_FIELD_TYPE

This section is a description of the CHARMM force field that is parametrized in the topology file. It is a single line (it can be read as a single string of length 80 characters). It does not affect any numerical results.

%FORMAT(i2,a78)

CHARMM_UREY_BRADLEY_COUNT

This section contains the number of Urey-Bradley parameters printed in the topology file. It contains two integers, the total number of Urey-Bradley terms (NUB) and the number of unique Urey-Bradley types (NUBTYPES).

[3] Not really supported. Every entry is BLA

%FLAG CHARMM_UREY_BRADLEY_COUNT

%FORMAT(2i8)

NUB NUBTYPES

CHARMM_UREY_BRADLEY

This section contains all of the Urey-Bradley terms. It is formatted exactly like BONDS_INC_HYDROGEN and BONDS_WITHOUT_HYDROGEN.

%FORMAT(10i8)

There are $3 \times$ NUB integers in this section.

CHARMM_UREY_BRADLEY_FORCE_CONSTANT

This section contains all of the force constants for each unique Urey-Bradley term in *kcal mol*$^{-1}$ *Å*^{2}. It is formatted exactly the same as BOND_FORCE_CONSTANT.

%FORMAT(5E16.8)

There are NUBTYPES floating point numbers in this section.

CHARMM_UREY_BRADLEY_EQUIL_VALUE

This section contains all of the equilibrium distances for each unique Urey-Bradley term in Å. It is formatted exactly the same as BOND_EQUIL_VALUE.

%FORMAT(5E16.8)

There are NUBTYPES floating point numbers in this section.

CHARMM_NUM_IMPROPERS

This section contains the number of improper torsions in the topology file. It contains one integer, the total number of improper torsions.

%FLAG CHARMM_NUM_IMPROPERS

%FORMAT(i8)

NIMPHI

CHARMM_IMPROPERS

This section contains all of the improper torsion terms. It is formatted exactly like DIHEDRALS_INC_HYDROGEN and DIHEDRALS_WITHOUT_HYDROGEN.

%FORMAT(10i8)

There are $5 \times$ NIMPHI integers in this section.

CHARMM_NUM_IMPROPER_TYPES

This section contains the number of unique improper torsion types in the topology file. It contains one integer, the total number of improper torsions types.

%FLAG CHARMM_NUM_IMPROPERS

%FORMAT(i8)

NIMPRTYPES

CHARMM_IMPROPER_FORCE_CONSTANT

This section contains the force constant for each unique improper torsion type. It is formatted exactly like DIHEDRAL_FORCE_CONSTANT.

%FORMAT(5E16.8)

There are NIMPRTYPES integers in this section.

CHARMM_IMPROPER_PHASE

This section contains the phase shift for each unique improper torsion type. It is formatted exactly like DIHEDRAL_PHASE

%FORMAT(5E16.8)

There are NIMPRTYPES integers in this section.

LENNARD_JONES_14_ACOEF

Instead of scaling the 1-4 van der Waals interactions, the CHARMM force field actually assigns entirely different LJ parameters to each atom type. Therefore, *chamber* topologies have two extra sections that correspond to the set of LJ parameters for 1-4 interactions. The way these tables are set up is identical to the way LENNARD_JONES_ACOEF and LENNARD_JONES_BCOEF are set up in chamber topologies.

%FORMAT(5E16.8)

There are $[\text{NTYPES} \times (\text{NTYPES} + 1)]\,/2$ floating point numbers in this section.

LENNARD_JONES_14_BCOEF

This section contains the LJ B-coefficients for 1-4 interactions. See LENNARD_JONES_14_ACOEF above.

%FORMAT(5E16.8)

There are $[\texttt{NTYPES} \times (\texttt{NTYPES} + 1)]\,/2$ floating point numbers in this section.

CHARMM_CMAP_COUNT

This section contains two integers—the number of total correction map (CMAP terms and the number of unique CMAP 'types.'

%FLAG CHARMM_CMAP_COUNT

%FORMAT(2i8)

CMAP_TERM_COUNT CMAP_TYPE_COUNT

CHARM_CMAP_RESOLUTION

This section stores the resolution (*i.e.,* number of steps along each phi/psi CMAP axis) for each CMAP grid.

%FORMAT(20I4)

There are CMAP_TERM_COUNT integers in this section.

CHARMM_CMAP_PARAMETER_

There are CMAP_TYPE_COUNT of these sections, where is replaced by a 2-digit integer beginning from 01. It is a 2-dimensional Fortran array whose 1-D sequence is stored in column-major order.

%FORMAT(8(F9.5))

There are $\texttt{CHARMM\_CMAP\_RESOLUTION(i)}^2$ floating point numbers in this section, where i is the in the FLAG title.

APPENDIX C
MESSAGE PASSING INTERFACE

In this appendix, I will briefly describe the Message Passing Interface (MPI) model that is frequently used to parallelize programs in the field of computational chemistry. The MPI is used extensively in the field of computational chemistry to enable large-scale parallelism on modern supercomputer architecture. Pachecho authored a particularly useful text for learning MPI programming. [202]

C.1 Parallel Computing

C.1.1 Data Models

Generally speaking, programs fall into one of two categories with regards to how handling and processing data is parallelized. The first approach refers to using multiple threads to run the same program or executable, each of which work on a different set of data—an approach called Single Program Multiple Data (SPMD). The second approach refers to multiple threads each running *different* programs on different sets of data—an approach called Multiple Program Multiple Data (MPMD).

MPI supports both SPMD and MPMD data models, with support for MPMD being introduced with the adoption of the MPI-2 standard. With the exception of some specialized QM/MM functionality in *sander*, MPI-enabled programs in Amber use the SPMD approach to parallelization, including all codes I contributed.

C.1.2 Memory Layout

In addition to the various approaches for parallelizing data processing, parallel programs fall into one of two broad families with respect to memory layout and access. An approach in which all processors share a common memory bank is called *shared memory parallelization* (SMP). This is the approach used by the OpenMP API that is implemented by most standard C and Fortran compilers. The other approach, called *distributed memory parallelization*, defines separate memory buffers for each process,

and each process can only modify its own memory buffer. The MPI implements the latter form of parallelism.

Shared memory and distributed memory parallelism each offer different advantages and disadvantages with respect to each other. In SMP, one thread can access data that has previously been manipulated by a different process without requiring that the result be copied and passed between processes. In distributed parallelism, however, the lack of required shared memory means that not all processes need access to the same memory bank, allowing tasks to be distributed across different physical computers.

The difference between distributed and shared memory parallelization can be visualized by considering a number of talented craftsmen constructing a complex machine in a workshop. SMP is analogous to crowding multiple workers around a single workbench with a single set of tools or instruments. Each worker can perform a separate task toward completing the project at the same time other workers are performing their tasks. Furthermore, as soon as one worker finishes their task and returns the result to the workbench, the result is immediately accessible to every other worker at the table. Of course, the number of workers that can work at the table and the physical size of the total project is limited by the number of tools present at the workbench and the size of that table, respectively. By analogy, the number of tools can be thought of as the number of processing cores available, while the size of the table is analogous to the amount of available shared memory.

Distributed memory parallelization schemes like MPI, on the other hand, are akin to providing each worker with their own workbench where they perform whatever tasks are assigned to them. When one worker's task requires the result of another's work, the required materials must be transported, or 'communicated,' between the two workers. This inter-workbench communication introduces a latency that is not present in SMP. However, the size of the project is no longer limited by the size of the workbench, but rather by whether or not the individual *pieces* can fit on any of the available workbenches. In

this case, the room is a cluster of computers, and each table is a separate processing core available in that cluster. Since most modern supercomputers are composed of large numbers of smaller, interconnected computers, distributed memory programs must be used for large, scalable applications.

Unsurprisingly, peak parallel performance leverages the capabilities of both distributed and shared memory parallelization to optimize load balancing across the available resources and to minimize communication requirements. On a typical computer cluster or supercomputer, there are a small number of cores on each individual machine—between 8 and 48 are currently commonplace—that are placed in a network connecting hundreds, thousands, or even tens of thousands of these machines. Using SMP within a single node as part of a larger, distributed application allows programs to take advantage of the strengths of both programming models. [203] Using the analogy above, this approach is equivalent to using multiple workers each around multiple workbenches, such that SMP takes place within a single workbench, and data and materials have to be 'communicated' between different ones.

C.1.3 Thread Count

A *process*, or *thread*, is an instance of an instruction set by which a processing unit operates on data. Drawing again on our analogy, a thread is equivalent to a single worker at a single workbench. Some parallel programming APIs use a dynamic thread count, so that new threads are launched when they are needed and ended when they are not. The OpenMP API operates this way. This is akin to more workers being called to work on the complex, labor-intensive parts of the manufacturing process and having them leave after that part of the task is finished. This way, a parallelization strategy is only necessary for particularly time-consuming parts of the computational process.

The MPI approach, on the other hand, employs a static thread count set before the program is initially launched, and this number never changes. In this case, the workers are brought into the workroom and the room is then locked. Each worker is assigned a

workbench and a set of instructions to follow based on the ID card they received when they entered the room.

C.2 The Mechanics of MPI

At its most basic level, MPI consists of a series of API calls that allow threads to communicate contents of their memory between each other so that they may coordinate efforts on a single task. While any parallel program may be constructed by simply allowing any two threads to send and receive data, MPI provides an extensive set of communication options to simplify creating efficient parallel programs.

C.2.1 Messages

In MPI, data that is sent and received between threads is referred to as a *message*, and the act of passing data between threads is called *communication*. The following sections will describe how messages are passed via communication within MPI.

C.2.2 Communicators

A communicator is a grouping of threads within an MPI universe between which messages may be passed. All messages sent and received in an MPI program do so through a particular communicator. Each thread within a communicator is given a unique identity within that communicator, called its *rank*, that is an integer value between 0 and N - 1, where N is the *size* of the communicator (*i.e.,* the number of threads that define it). The ranks of the communicators can be used to assign different processors to different portions of total work.

Communicators can be assigned and destroyed as desired within an MPI program, and are very useful tools for assigning a subset of the available threads to a particular task. There is one communicator, *MPI_COMM_WORLD*, that is created when an MPI program is launched that links every thread.

C.2.3 Communications

Communicating data between threads is the heart of parallelizing a program using MPI. As mentioned above, a program may be fully parallelized using MPI by only

defining simple send and receive calls between two threads. However, the optimal set of sends and receives depends strongly on where the threads are placed, the bandwidth and latency of the connection between them, and the pure number of such calls that are required for a particular task.

To facilitate the creation of portable, efficient parallel programs, MPI provides an expansive set of functions to communicate data to abstract the complexity of optimizing communications. The following sections will briefly describe the three main families of communications as well as some representative examples within those families.

C.2.3.1 Point-to-point

The simplest set of communication involves exchanging data between two threads. These communications are the cheapest individual MPI communications to use, since they require communication between the minimum number of threads—two. Example functions in this family include *MPI_Send*, *MPI_Recv*, and *MPI_Sendrecv*. The first two allow data to be sent from one process to another, and the second explicitly receives sent data. Every send must have a corresponding receive call on the destination thread to complete the communication. The last function, *MPI_Sendrecv* combines a send and receive in the same function. The effect of these functions are shown in Fig. C-1.

C.2.3.2 All-to-one and One-to-all

The next family of communication occurs between a specified *root* thread and every other thread within a communicator. These functions involve more costly communication than the point-to-point communications described above since it requires at least as many messages be sent as there are threads in the communicator. However, specific MPI implementations can optimize these functions with respect to the naïve implementation, typically making them more efficient than alternatives implemented via a series of point-to-point communications.

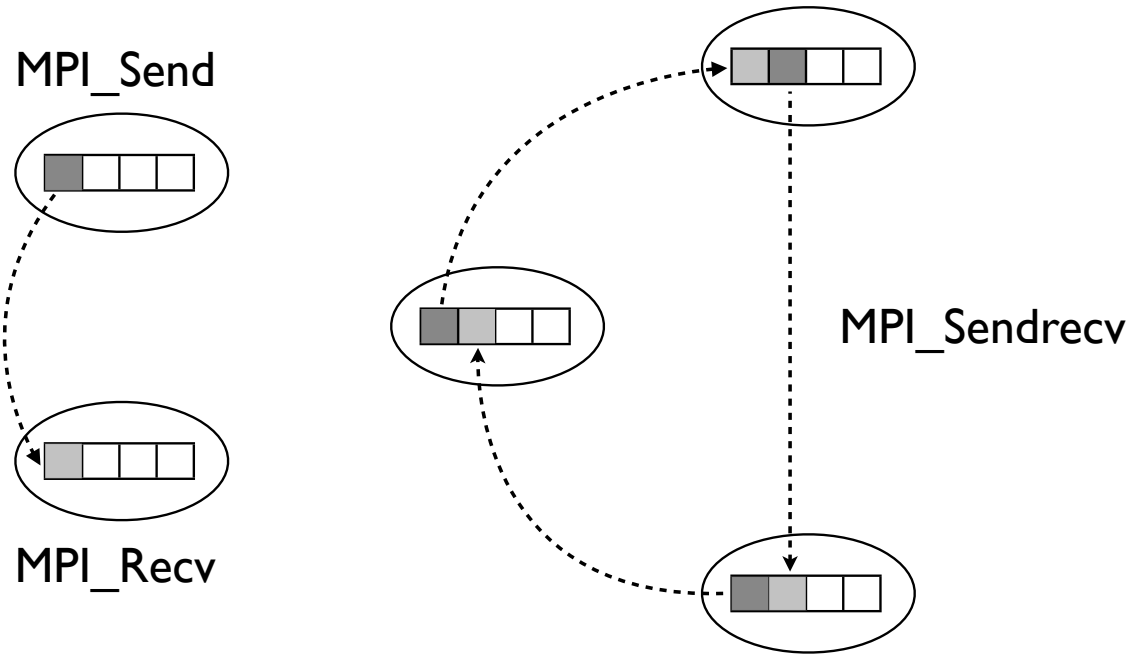

Figure C-1. Schematic of different point-to-point communications. Threads and data are shown as ovals and boxes, respectively, with arrows indicating the lines of communication

Examples in this family include *MPI_Bcast*, *MPI_Gather*, *MPI_Scatter*, and *MPI_Reduce*. *MPI_Bcast* is a broadcast that sends data from the root thread to every other thread in a communicator. *MPI_Gather* collects data from all threads into an array on the root thread. *MPI_Scatter* operates similarly to *MPI_Bcast*, except that it divides the data sent by the root into equal-sized chunks that are sent out to every thread in the communicator (this is effectively the inverse of an *MPI_Gather* call). Finally, *MPI_Reduce* takes an array of data on each thread and combines them via some mathematical operation (*i.e.,* addition, subtraction, etc.) into the final result on the root thread. These functions are demonstrated diagrammatically in Fig. C-2.

C.2.3.3 All-to-all

The last family of communication involves transferring data from every thread in a communicator to every other thread. Examples include *MPI_Allgather*, *MPI_Allreduce*, and *MPI_Alltoall*. These are the most expensive of all MPI communications since

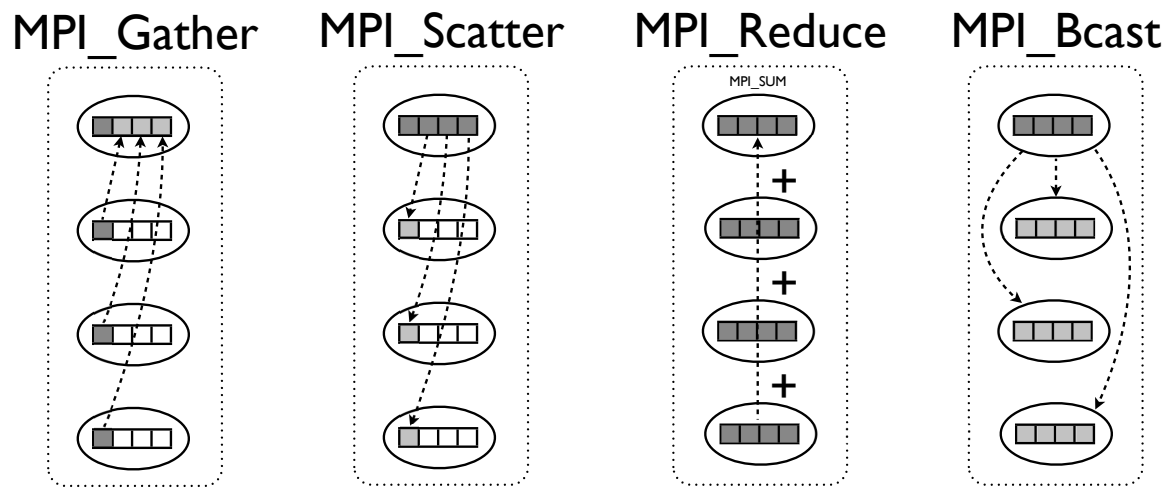

Figure C-2. Schematic of different all-to-one and one-to-all communications. Threads and data are shown as ovals and boxes, respectively, with arrows indicating the lines of communication. Communicators are shown as dotted lines enclosing all the threads in the communicator. The 'root' thread in all communications is the top oval.

they involve the most amount of communication. As a result, they should be avoided whenever possible. However, due to the complexity of the required communication, these functions are the best candidates for performance optimization and tuning within an MPI implementation. As a result, when such communication is required, programs should *not* attempt to implement their own, equivalent alternatives.

MPI_Allgather and *MPI_Allreduce* are logically equivalent to invoking an *MPI_Bcast* call from the root thread following either an *MPI_Gather* or *MPI_Reduce* call to that root. The *MPI_Alltoall* function behaves like an *MPI_Gather* to a root process followed by a *MPI_Scatter* from that root. The *MPI_Allgather* is the most expensive of the all-to-all communications given the increased amount of data that must be transmitted between threads. Fig. C-3 illustrates how these all-to-all communications work.

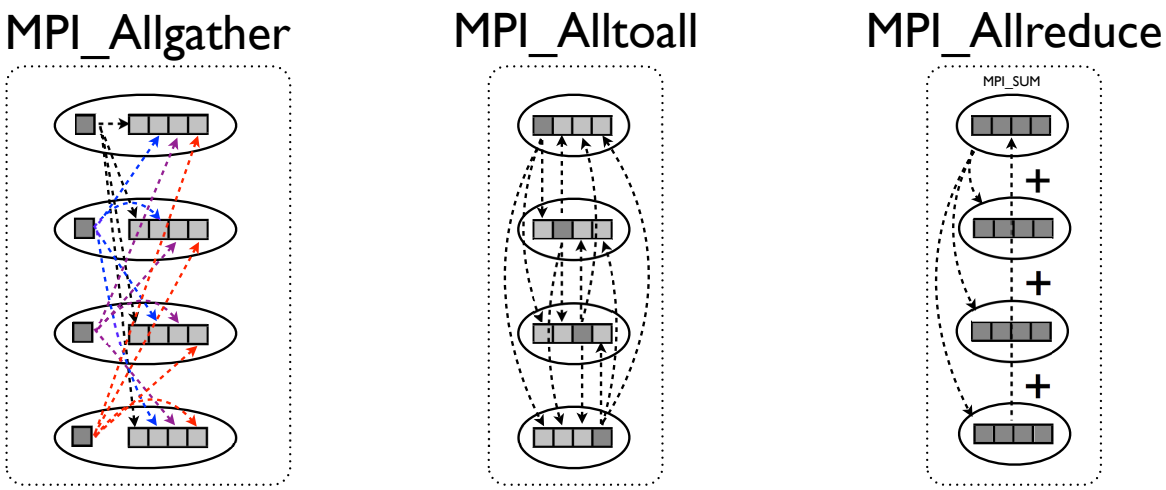

Figure C-3. Schematic of different all-to-all communications. Threads and data are shown as ovals and boxes, respectively, with arrows indicating where data is transferred to and from.

C.2.4 Blocking vs. Non-blocking Communications

In general, communications within MPI fall into one of two categories: so-called *blocking* and *non-blocking* communications. Blocking communications require the communication complete before the program can continue. Non-blocking communications, on the other hand, return instantaneously and allow the program to continue executing code while waiting for the communication to complete. All communications involving more than two threads—*i.e.,* one-to-all and all-to-all—are blocking.

There is a special MPI function, *MPI_Barrier* whose sole purpose is to block all threads within a communicator from advancing past the barrier until each thread has reached it. Similarly, the *MPI_Wait* functions prevent a thread from continuing its computations until after the specified non-blocking communications complete.

REFERENCES

[1] Lide, D. R., Frederikse, H. P. R., Brewer, L., Koetzle, T. F., Craig, N. C., Lineberger, W. C., Donnelly, R. J., Smith, A. L., Goldberg, R. N., Westbrook, J. H., Eds. *CRC Handbook of Chemistry and Physics*; CRC Press, Inc.: New York, 1997.

[2] Schrödinger, E. *Phys. Rev.* **1926**, *28*, 1049–1070.

[3] McQuarrie, D. A.; Simon, J. D. *Physical Chemistry: A Molecular Approach*; University Science Books: Sausalito, CA, 1997.

[4] Jeletic, M. S.; Lowry, R. J.; Swails, J. M.; Ghiviriga, I.; Veige, A. S. *J. Organomet. Chem.* **2011**, *696*, 3127–3134.

[5] Chandrasekhar, J.; Smith, S. F.; Jorgensen, W. L. *J. Am. Chem. Soc.* **1985**, *107*, 154–163.

[6] Watson, T. J.; Bartlett, R. J. *Chem. Phys. Lett.* **2013**, *555*, 235–238.

[7] Range, K.; Riccardi, D.; Cui, Q.; Elstner, M.; York, D. M. *Phys. Chem. Chem. Phys.* **2005**, *7*, 3070–3079.

[8] Hehre, W.; Radom, L.; von Schleyer, P.; Pople, J. *Ab inito Molecular Orbital Theory*; John Wiley and Sons: New York, 1986.

[9] McQuarrie, D. A. *Statistical Mechanics*; University Science Books: Mill Valley, CA, 1973.

[10] Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H. *J. Chem. Phys.* **1953**, *21*, 1087–1092.

[11] Tuckerman, M. E. *Statistical Mechanics: Theory and Molecular Simulation*; Oxford University Press, 2010.

[12] Leach, A. R. *Molecular Modelling: Principles and Applications*, 2nd ed.; Prentice Hall, 2001.

[13] McCammon, J. A.; Gelin, B. R.; Karplus, M. *Nature* **1977**, *267*, 585–590.

[14] Woodcock, L. V. *Chem. Phys. Lett.* **1971**, *10*, 257–261.

[15] Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; Dinola, A.; Haak, J. R. *J. Chem. Phys.* **1984**, *81*, 3684–3690.

[16] Ryckaert, J. P.; Ciccotti, G.; Berendsen, H. J. C. *J. Comput. Phys.* **1977**, *23*, 327–341.

[17] Andersen, H. C. *J. Comp. Phys.* **1983**, *52*, 24–34.

[18] Miyamoto, S.; Kollman, P. A. *J. Comput. Chem.* **1992**, *13*, 952–962.

[19] Forester, T. R.; Smith, W. *J. Comput. Chem.* **1998**, *19*, 102–111.

[20] Lee, S.-H.; Palmo, K.; Krimm, S. *J. Comput. Phys.* **2005**, *210*, 171–182.

[21] Kolos, W.; Wolniewicz, L. *J. Chem. Phys.* **1964**, *41*, 3663–3673.

[22] Cramer, C. J. *Essentials of Computational Chemisrty: Theories and Models*, 2nd ed.; John Wiley & Sons, Ltd.: 111 River St., Hoboken, NJ 07030, USA, 2004.

[23] Hornak, V.; Abel, R.; Okur, A.; Strockbine, B.; Roitberg, A.; Simmerling, C. *Proteins* **2006**, *65*, 712–725.

[24] Pérez, A.; Marchán, I.; Svozil, D.; Šponer, J.; Cheatham III, T. E.; Laughton, C. A.; Orozco, M. *Biophys. J.* **2007**, *92*, 3817–3829.

[25] Lindorff-Larsen, K.; Stefano, P.; Palmo, K.; Maragakis, P.; Klepeis, J. L.; Dror, R. O.; Shaw, D. E. *Proteins* **2010**, *78*, 1950–1958.

[26] Bayly, C. I.; Cieplak, P.; Cornell, W. D.; Kollman, P. A. *J. Phys. Chem.* **1993**, *97*, 10269–10280.

[27] Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Kollmann, P. A. *J. Am. Chem. Soc.* **1993**, *115*, 9620–31.

[28] Cieplak, P.; Cornell, W. D.; Bayly, C.; Kollman, P. A. *J. Comput. Chem.* **1995**, *16*, 1357–1377.

[29] Mackerell, Jr., A. D.; Feig, M.; Brooks, III, C. L. *J. Comput. Chem.* **2004**, *25*, 1400–1415.

[30] MacKerell, Jr., A. D. et al. *J. Phys. Chem. B* **1998**, *102*, 3586–3616.

[31] Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. *J. Am. Chem. Soc.* **1995**, *117*, 5179–5197.

[32] Duan, Y.; Wu, C.; Chowdhury, S.; Lee, M. C.; Xiong, G.; Zhang, W.; Yang, R.; Cieplak, P.; R., L.; Lee, T. *J. Comput. Chem.* **2003**, *24*, 1999–2012.

[33] Case, D. A.; Cheatham III, T. E.; Darden, T.; Gohlke, H.; Luo, R.; Merz, K. M.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. *J. Comput. Chem.* **2005**, *26*, 1668–1688.

[34] Wang, J.; Cieplak, P.; Kollman, P. A. *J. Comput. Chem.* **2000**, *21*, 1049–1074.

[35] Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. *J. Comput. Chem.* **2004**, *25*, 1157–1174.

[36] Zgarbová, M.; Otyepka, M.; Šponer, J.; Mládek, A.; Banáš, P.; Cheatham III, T. E.; Jurečka, P. *J. Chem. Theory Comput.* **2011**, *7*, 2886–2902.

[37] Sitkoff, D.; Sharp, K. A.; Honig, B. *J. Phys. Chem.* **1994**, *98*, 1978–1988.

[38] Klapper, I.; Hagstrom, R.; Fine, R.; Sharp, K.; Honig, B. *Proteins* **1986**, *1*, 47–59.

[39] Gilson, M. K.; Sharp, K. A.; Honig, B. H. *J. Comput. Chem.* **1988**, *9*, 327–335.

[40] Baker, N. A.; Sept, D.; Joseph, S.; Holst, M. J.; McCammon, J. A. *Proc. Natl. Acad. Sci. USA* **2001**, *98*, 10037–10041.

[41] Nielsen, J. E.; Vriend, G. *Proteins* **2001**, *43*, 403–412.

[42] Wang, J.; Qin, C.; Li, Z.-L.; Zhao, H.-K.; Luo, R. *Chem. Phys. Lett.* **2009**, *468*, 112–119.

[43] Still, W. C.; Tempczyk, A.; Hawley, R. C.; Hendrickson, T. *J. Am. Chem. Soc.* **1990**, *112*, 6127–6129.

[44] Qiu, D.; Shenkin, P. S.; Hollinger, F. P.; Still, W. C. *J. Phys. Chem. A* **1997**, *101*, 3005–3014.

[45] Onufriev, A.; Bashford, D.; Case, D. A. *J. Phys. Chem. B* **2000**, *104*, 3712–3720.

[46] Bashford, D.; Case, D. A. *Annu. Rev. Phys. Chem.* **2000**, *51*, 129–152.

[47] Onufriev, A.; Case, D. A.; Bashford, D. *J. Comput. Chem.* **2002**, *23*, 1297–1304.

[48] Onufriev, A. V.; Sigalov, G. *J. Chem. Phys* **2011**, *134*, 164104.

[49] Onufriev, A.; Bashford, D.; Case, D. A. *Proteins* **2004**, *55*, 383–394.

[50] Mongan, J.; Simmerling, C.; McCammon, J. A.; Case, D. A.; Onufriev, A. *J. Chem. Theory Comput.* **2007**, *3*, 156–169.

[51] Nguyen, H.; Roe, D. R.; Simmerling, C. *J. Chem. Theory Comput.* **2013**,

[52] Weiser, J.; Shenkin, P. S.; Still, W. C. *J. Comput. Chem.* **1999**, *20*, 217–230.

[53] Mei, C.; Sun, Y.; Zheng, G.; Bohm, E. J.; Kale, L. V.; Phillips, J. C.; Harrison, C. Enabling and scaling biomolecular simulations of 100 million atoms on petascale machines with a multicore-optimized message-driven runtime. 2011; http://doi.acm.org/10.1145/2063384.2063466.

[54] Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Oxford science publications; Oxford University Press, USA, 1989.

[55] Schreiber, H.; Steinhauser, O. *J. Mol. Biol.* **1992**, *228*, 909–923.

[56] Schreiber, H.; Steinhauser, O. *Biochemistry* **1992**, *31*, 5856–5860.

[57] Saito, M. *J. Chem. Phys.* **1994**, *101*, 4055–4061.

[58] Auffinger, P.; Beveridge, D. L. *Chem. Phys. Lett.* **1995**, *234*, 413–415.

[59] Cheatham III, T. E.; Miller, J. L.; Fox, T.; Darden, T. A.; Kollman, P. A. *J. Am. Chem. Soc.* **1995**, *117*, 4193–4194.

[60] Feller, S. E.; Pastor, R. W.; Rojnuckarin, A.; Bogusz, S.; Brooks, B. R. *J. Phys. Chem.* **1996**, *100*, 17011–17020.

[61] Patra, M.; Karttunen, M.; Hyvönen, M. T.; Falck, E.; Lindqvist, P.; Vattulainen, I. *Biophys. J.* **2003**, *84*, 3636–3645.

[62] Steinbach, P. J.; Brooks, B. R. *J. Comput. Chem.* **1994**, *15*, 667–683.

[63] Ewald, P. P. *Ann. Phys.* **1921**, *64*, 253–268.

[64] Darden, T.; Perera, L.; Li, L.; Pedersen, L. *Structure* **1999**, *7*, R55–R60.

[65] Miaskiewicz, K.; Osman, R.; Weinstein, H. *J. Am. Chem. Soc.* **1993**, *115*, 1526–1537.

[66] McConnell, K. J.; Nirmala, R.; Young, M. A.; Ravishanker, G.; Beveridge, D. L. *J. Am. Chem. Soc.* **1994**, *116*, 4461–4462.

[67] unenberger, P. H. H.; McCammon, J. A. *J. Chem. Phys.* **1999**, *110*, 1856–1872.

[68] Cerutti, D. S.; Case, D. A. *J. Chem. Theory Comput.* **2010**, *6*, 443–458.

[69] Wu, X.; Brooks, B. R. *J. Chem. Phys.* **2005**, *122*, 044107.

[70] Tironi, I. G.; Sperb, R.; Smith, P. E.; van Gunsteren, W. F. *J. Chem. Phys.* **1995**, *102*, 5451–5459.

[71] Shaw, D. E. et al. *SIGARCH Comput. Archit. News* **2007**, *35*.

[72] Shaw, D. E.; Maragakis, P.; Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Eastwood, M. P.; Bank, J. A.; Jumper, J. M.; Salmon, J. K.; Shan, Y.; Wriggers, W. *Science* **2010**, *330*, 341–346.

[73] Grossfield, A. WHAM: the weighted histogram analysis method, version 2.0.4. 2005; http://membrane.urmc.rochester.edu/content/wham.

[74] Shirts, M. R.; Chodera, J. D. *J. Chem. Phys.* **2008**, *129*, 124105.

[75] Lee, T.; Radak, B.; Pabis, A.; York, D. M. *J. Chem. Theory Comput.* **2013**, *9*, 153–164.

[76] Jarzynski, C. *Phys. Rev. Lett.* **1997**, *78*, 2690–2693.

[77] Lyubartsev, A. P.; Martsinovksi, A. A.; Shevkunov, S. V.; Vorontsov-Velyaminov, P. N. *J. Chem. Phys.* **1992**, *96*, 1776.

[78] Sugita, Y.; Okamoto, Y. *Chem. Phys. Lett.* **1999**, *314*, 141–151.

[79] Babin, V.; Roland, C.; Sagui, C. *J. Chem. Phys.* **2008**, *128*, 134101.

[80] Sugita, Y.; Kitao, A.; Okamoto, Y. *J. Chem. Phys.* **2000**, *113*, 6042–6051.

[81] Fukunishi, H.; Watanabe, O.; Takada, S. *J. Chem. Phys.* **2002**, *116*, 9058–9067.

[82] Fajer, M.; Swift, R. V.; McCammon, J. A. *J Comput Chem* **2009**, *30*, 1719–1725.

[83] Arrar, M.; de Oliveira, C. A. F.; Fajer, M.; Sinko, W.; McCammon, J. A. *J. Chem. Theory Comput.* **2013**, *9*, 18–23.

[84] Jiang, W.; Roux, B. *J. Chem. Theory Comput.* **2010**, *6*, 2559–2565.

[85] Meng, Y.; Dashti, D.; Roitberg, A. E. *J. Chem. Theory Comput.* **2011**, *7*, 2721–2727.

[86] Wallace, J. A.; Shen, J. K. *J. Chem. Theory Comput.* **2011**, *7*, 2617–2629.

[87] Itoh, S. G.; Damjanović, A.; Brooks, B. R. *Proteins* **2011**, *79*, 3420–3436.

[88] Swails, J. M.; Roitberg, A. E. *J. Chem. Theory Comput.* **2012**, *8*, 4393–4404.

[89] Dashti, D.; Roitberg, A. *J. Phys. Chem. B* **2012**, *116*, 8805–8811.

[90] Wu, X.; Hodoscek, M.; Brooks, B. R. *J. Chem. Phys.* **2012**, *137*, 044106.

[91] Bolhuis, P. G. *J. Chem. Phys.* **2008**, *129*, 114108.

[92] Vorobjev, Y. N.; Almagro, J. C. *Proteins* **1998**, *32*, 399–413.

[93] Head, M. S.; Given, J. A.; Gilson, M. K. *J. Phys. Chem. A* **1997**, *101*, 1609–1618.

[94] Yang, A. S.; Honig, B. *J. Mol. Biol.* **1995**, *252*, 351–365.

[95] Yang, A. S.; Honig, B. *J. Mol. Biol.* **1995**, *252*, 366–376.

[96] Portman, J. J.; Takada, S.; Wolynes, P. G. *Phys. Rev. Lett.* **1998**, *81*, 5237–5240.

[97] Eisenberg, D.; McLachlan, A. D. *Nature* **1986**, *319*, 199–203.

[98] Jean-Charles, A.; Anthony, N.; Sharp, K.; Honig, B.; Tempczyk, A.; Hendrickson, T. F.; Still, W. C. *J. Am. Chem. Soc.* **1991**, *113*, 1454–1455.

[99] Massova, I.; Kollman, P. A. *J. Am. Chem. Soc.* **1999**, *121*, 8133–8143.

[100] Woo, H.-J.; Roux, B. *Proc. Natl. Acad. Sci.* **2005**, *102*, 6825–6830.

[101] Gohlke, H.; Kiel, C.; Case, D. A. *J. Mol. Biol.* **2003**, *330*, 891–913.

[102] Gohlke, H.; Case, D. A. *J. Comput. Chem.* **2004**, *25*, 238–250.

[103] Meirovitch, H. *Curr. Opin. Struct. Biol.* **2007**, *17*, 181–186.

[104] Davies, J.; Doltsinis, N.; Kirby, A.; Roussev, C.; Sprik, M. *J. Am. Chem. Soc.* **2002**, *124*, 6594–6599.

[105] Steinbrecher, T.; Joung, I.; Case, D. A. *J. Comp. Chem.* **2011**, *32*, 3253–3263.

[106] Beutler, T. C.; Mark, A. E.; René C. van Schaik and Paul R. Gerber and Wilfred F. van Gunsteren, *Chem. Phys. Lett.* **1994**, *222*, 529–539.

[107] Steinbrecher, T.; Mobley, D. L.; Case, D. A. *J. Chem. Phys.* **2007**, *127*, 214108.

[108] Zwanzig, R. W. *J. Chem. Phys.* **1954**, *22*, 1420–1426.

[109] Homeyer, N.; Gohlke, H. *Mol. Inf.* **2012**, *31*, 114–122.

[110] Miller III, B. R.; McGee Jr., T. D.; Swails, J. M.; Homeyer, N.; Gohlke, H.; Roitberg, A. E. *J. Chem. Theory Comput.* **2012**, *8*, 3314–3321.

[111] Srinivasan, J.; Cheatham, III, T. E.; Cieplak, P.; Kollman, P. A.; Case, D. A. *J. Am. Chem. Soc.* **1998**, *129*, 9401–9409.

[112] Massova, I.; Kollman, P. A. *Perspect. Drug Discov.* **2000**, *18*, 113–135.

[113] Åqvist, J.; Medina, C.; Samuelsson, J.-E. *Protein Eng.* **1994**, *7*, 385–391.

[114] Hansson, T.; Marelius, J.; Åqvist, J. *J. Comput. Aid. Mol. Des.* **1998**, *12*, 27–35.

[115] Marcus, R. A. *J. Chem. Phys.* **1955**, *24*, 966–979.

[116] Cornish-Bowden, A. J.; Knowles, J. R. *Biochem. J* **1969**, *113*, 353–362.

[117] White Jr., F. H.; Anfinsen, C. B. *Ann. NY Acad. Sci.* **1959**, *81*, 515–523.

[118] Tanford, C.; Kirkwood, J. G. *J. Am. Chem. Soc.* **1957**, *79*, 5333–5339.

[119] Olsson, M. H. M.; Söndergaard, C. R.; Rostkowski, M.; Jensen, J. H. *J. Chem. Theory Comput.* **2011**, *7*, 525–537.

[120] Myers, J.; Grothaus, G.; Narayanan, S.; Onufriev, A. *Proteins* **2006**, *63*, 928–938.

[121] Bashford, D.; Karplus, M. *Biochemistry* **1990**, *29*, 10219–10225.

[122] Bashford, D.; Gerwert, K. *J. Mol. Biol.* **1992**, *224*, 473–486.

[123] Antosiewicz, J.; McCammon, J. A.; Gilson, M. K. *J. Mol. Biol.* **1994**, *238*, 415–436.

[124] Song, Y.; Mao, J.; Gunner, M. R. *J. Comput. Chem.* **2009**, *30*, 2231–2247.

[125] Baptista, A. M.; Martel, P. J.; Petersen, S. B. *Proteins* **1997**, *27*, 523–544.

[126] Baptista, A. M.; Teixeira, V. H.; Soares, C. M. *J. Chem. Phys.* **2002**, *117*, 4184–4200.

[127] Bürgi, R.; Kollman, P. A.; van Gunsteren, W. F. *Proteins* **2002**, *47*, 469–480.

[128] Lee, M. S.; Salsbury, Jr., F. R.; Brooks III, C. L. *Proteins* **2004**, *56*, 738–752.

[129] Börjesson, U.; Hünenberger, P. H. *J. Phys. Chem. B* **2004**, *108*, 13551–13559.

[130] Mongan, J.; Case, D. A.; McCammon, J. A. *J. Comput. Chem.* **2004**, *25*, 2038–2048.

[131] Khandogin, J.; Brooks III, C. L. *Biophys. J.* **2005**, *89*, 141–157.

[132] Alexov, E.; Mehler, E. L.; Baker, N.; Huang, Y.; Milletti, F.; Nielsen, J. E.; Farrell, D.; Carstensen, T.; Olsson, M. H. M.; Shen, J. K.; Warwicker, J.; Williams, S.; Word, J. M. *Proteins* **2011**, *79*, 3260–3275.

[133] Machuqueiro, M.; Baptista, A. M. *Proteins* **2011**, *79*, 3437–3447.

[134] Hamelberg, D.; Mongan, J.; McCammon, J. A. *J. Chem. Phys.* **2004**, *120*, 11919–11929.

[135] Williams, S. L.; de Oliveira, C. A. F.; McCammon, J. A. *J. Chem. Theory Comput.* **2010**, *6*, 560–568.

[136] Webb, H.; Tynan-Connolly, B. M.; Lee, G. M.; Farrell, D.; O'Meara, F.; Sondergaard, C. R.; Teilum, K.; Hewage, C.; McIntosh, L. P.; Nielsen, J. E. *Proteins* **2011**, *79*, 685–702.

[137] Pitera, J. W.; Swope, W. *Proc. Natl. Acad. Sci. USA* **2003**, *100*, 7587–7592.

[138] Chodera, J. D.; Shirts, M. R. *J. Chem. Phys.* **2011**, *135*, 194110.

[139] Nadler, W.; Meinke, J. H.; Hansmann, U. H. E. *Phys. Rev. E* **2008**, *78*, 061905.

[140] Meng, Y.; Roitberg, A. E. *J. Chem. Theory Comput.* **2010**, *6*, 1401–1412.

[141] Case, D. A.; Darden, T. A.; Cheatham III, T. E.; Simmerling, C. L.; Wang, J.; Duke, R. E.; Luo, R.; Walker, R. C.; Zhang, W.; Merz, K. M.; Roberts, B.; Hayik, S.; Roitberg, A.; Seabra, G.; Swails, J.; Götz, A. W.; Kolossváry, I.; Wong, K. F.; Paesani, F.; Vanicek, J.; Wolf, R. M.; Liu, J.; Wu, X.; Brozell, S. R.; Steinbrecher, T.; Gohlke, H.; Cai, Q.; Ye, X.; Wang, J.; Hsieh, M.-J.; Cui, G.; Roe, D. R.; Mathews, D. H.; Seetin, M. G.; Salomon-Ferrer, R. Sagui, C.; Babin, V.; Luchko, T.; Gusarov, S.; Kovalenko, A.; Kollman, P. A. AMBER 12. University of California, San Francisco: San Francisco, CA, 2012.

[142] Sindhikara, D.; Meng, Y.; Roitberg, A. E. *J. Chem. Phys.* **2008**, *128*, 024103–024103–10.

[143] Sindhikara, D. J.; Emerson, D. J.; Roitberg, A. E. *J. Chem. Theory Comput.* **2010**, *6*, 2804–2808.

[144] Takahashi, T.; Nakamura, H.; Wada, A. *Biopolymers* **1992**, *32*, 897–909.

[145] Bartik, K.; Redfield, C.; Dobson, C. M. *Biophys. J.* **1994**, *66*, 1180–1184.

[146] Demchuk, E.; Wade, R. C. *J. Phys. Chem.* **1996**, *100*, 17373–17387.

[147] Artymiuk, P. J.; Blake, C. C. F.; W., R. D.; S., W. K. *Acta Cryst. B* **1982**, *38*, 778–783.

[148] Vocadlo, D. J.; Davies, G. J.; Laine, R.; Withers, S. G. *Nature* **2001**, *412*, 835–838.

[149] Sindhikara, D. J.; Kim, S.; Voter, A. F.; Roitberg, A. E. *J. Chem. Theory Comput.* **2009**, *5*, 1624–1631.

[150] Hamacher, K. *J. Comp. Chem.* **2007**, *28*, 2576–2580.

[151] McClendon, C. L.; Hua, L.; Barreiro, G.; Jacobson, M. P. *J. Chem. Theory Comput.* **2012**, *8*, 2115–2126.

[152] Vetter, J. S.; Glassbrook, R.; Dongarra, J.; Schwan, K.; Loftis, B.; McNally, S.; Meredith, J.; Rogers, J.; Roth, P.; Spafford, K.; Yalamanchili, S. *Computing in Science and Engg.* **2011**, *13*, 90–95.

[153] Cheatham, III, T. E.; A.Young, M. *Biopolymers* **2001**, *56*, 232–256.

[154] Várnai, P.; Djuranovic, D.; Lavery, R.; Hartmann, B. *Nucleic Acids Res.* **2002**, *30*, 5398–5406.

[155] Klepeis, J. L.; Lindorff-Larsen, K.; Dror, R. O.; Shaw, D. E. *Curr. Opin. Struct. Biol.* **2009**, *19*, 120–127.

[156] Zhou, R. *Proteins* **2003**, *53*, 148–161.

[157] Geney, R.; Layten, M.; Gomperts, R.; Hornak, V.; Simmerling, C. *J. Chem. Theory Comput.* **2006**, *2*, 115–127.

[158] Donnini, S.; Tegeler, F.; Groenhof, G.; Grubmüller, H. *J. Chem. Theory Comput.* **2011**, *7*, 1962–1978.

[159] Goh, G. B.; Knight, J. L.; Brooks, C. L. *J. Chem. Theory Comput.* **2012**, *8*, 36–46.

[160] Wallace, J. A.; Shen, J. K. *J. Chem. Phys.* **2012**, *137*, 184105.

[161] Machuqueiro, M.; Baptista, A. M. *Proteins* **2008**, *72*, 289–298.

[162] Baptista, A. M.; Soares, C. M. *J. Phys. Chem. B* **2001**, *105*, 293–309.

[163] Gregory D. Hawkins, C. C.; Truhlar, D. *Chem. Phys. Lett.* **1995**, *246*, 122–129.

[164] Hawkins, G. D.; Cramer, C. J.; Truhlar, D. G. *J. Phys. Chem.* **1996**, *100*, 19824–19839.

[165] Shang, Y.; Nguyen, H.; Wickstrom, L.; Okur, A.; Simmerling, C. *J. Mol. Graphics* **2011**, *29*, 676–684.

[166] Frantz, C.; Barreiro, G.; Dominguez, L.; Xiaoming, C.; Eddy, R.; Condeelis, J.; Kelly, M. J. S.; Jacobson, M. P.; Barber, D. L. *J. Cell Biol.* **2008**, *183*, 865–871.

[167] Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926–935.

[168] Young, A. C. M.; Dewan, J. C. *J. Appl. Cryst.* **1993**, *26*, 309–319.

[169] Berisio, R.; Sica, F.; Lamzin, V. S.; Wilson, K. S.; Zagari, A.; Mazzarella, L. *Acta Crystallogr. D.* **2002**, *58*, 441–450.

[170] Wlodawer, A.; Svensson, L. A.; Sjölin, L.; Gilliland, G. L. *Biochemistry* **1988**, *27*, 2705–2717.

[171] Uberuaga, B. P.; Anghel, M.; Voter, A. F. *J. Chem. Phys.* **2004**, *120*, 6363–6374.

[172] Darden, T.; York, D.; Pedersen, L. *J. Chem. Phys.* **1993**, *98*, 10089–10092.

[173] Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Hsing, L.; Pedersen, L. G. *J. Chem. Phys.* **1995**, *103*, 8577–8593.

[174] Baker, W. R.; Kintanar, A. *Arch. Biochem. Biophys.* **1996**, *327*, 189–199.

[175] Patriksson, A.; van der Spoel, D. *Phys. Chem. Chem. Phys.* **2008**, *10*, 2073–2077.

[176] Okur, A.; Wickstrom, L.; Layten, M.; Geney, R.; Song, K.; Hornak, V.; Simmerling, C. *J. Chem. Theory Comput.* **2006**, *2*, 420–433.

[177] Chodera, J. D.; Swope, W. C.; Pitera, J. W.; Seok, C.; Dill, K. A. *J. Chem. Theory Comput.* **2007**, *3*, 26–41.

[178] Cheng, X.; Cui, G.; Hornak, V.; Simmerling, C. *J. Phys. Chem. B* **2005**, *109*, 8220–8230.

[179] Wang, J.; Morin, P.; Wang, W.; Kollman, P. A. *J. Am. Chem. Soc.* **2001**, *123*, 5221–5230.

[180] Kuhn, B.; Gerber, P.; Schulz-Gasch, T.; Stahl, M. *J. Med. Chem.* **2005**, *48*, 4040–4048.

[181] Weis, A.; Katebzadeh, K.; Söderhjelm, P.; Nilsson, I.; Ryde, U. *J. Med. Chem.* **2006**, *49*, 6596–6606.

[182] Genheden, S.; Ryde, U. *J. Comp. Chem.* **2009**, *31*, 837–846.

[183] Wang, W.; Donini, O.; Reyes, C. M.; Kollman, P. A. *Annu. Rev. Biophys. Biomol. Struct.* **2001**, *30*, 211–243.

[184] Bradshaw, R. T.; Patel, B. H.; Tate, E. W.; Leatherbarrow, R. J.; Gould, I. R. *Bioinformatics* **2010**, *24*, 197–207.

[185] Gouda, H.; Kuntz, I. D.; Case, D. A.; Kollman, P. A. *Biopolymers* **2003**, *68*, 16–34.

[186] Combelles, C.; Gracy, J.; Heitz, A.; Craik, D. J.; Chiche, L. *Proteins* **2008**, *73*, 87–103.

[187] Brice, A. R.; Dominy, B. N. *J. Comp. Chem.* **2011**, *32*, 1431–1440.

[188] Mikulskis, P.; Genheden, S.; Rydberg, P.; Sandberg, L.; Olsen, L.; Ryde, U. *J. Comput. Aided Mol. Des.* **2012**, *26*, 527–541.

[189] Sanner, M. F. *J. Mol. Graph Model.* **1999**, *17*, 57–61.

[190] Cock, P. J. A.; Antao, T.; Chang, J. T.; Chapman, B. A.; Cox, C. J.; Dalke, A.; Friedberg, I.; Hamelryck, T.; Kauff, F.; Wilczynski, B.; de Hoon, M. J. L. *Bioinformatics* **2009**, *25*, 1422–1423.

[191] Michaud-Agrawal, N.; Denning, E. J.; Woolf, T. B.; Beckstein, O. *J. Comp. Chem.* **2011**, *32*, 2319–2327.

[192] Genheden, S.; Luchko, T.; Gusarov, S.; Kovalenko, A.; Ryde, U. *J. Phys. Chem. B* **2010**, *114*, 8505–8516.

[193] Wang, J.; Hou, T.; Xu, X. *Curr. Comput.-Aid. Drug* **2006**, 3.

[194] Metz, A.; Pfleger, C.; Kopitz, H.; Pfeiffer-Marek, S.; Baringhaus, K.-H.; Gohlke, H. *J. Chem. Inf. Model.* **2012**, *52*, 120–133.

[195] Brooks, B. R.; Janežič, D.; Karplus, M. *J. Comput. Chem.* **1995**, *16*, 1522–1542.

[196] Macke, T. J.; Case, D. A. In *Molecular Modeling of Nucleic Acids*; Leontis, N. B., SantaLucia, J., Eds.; American Chemical Society: Washington, DC, 1997; Chapter 25, pp 379–393.

[197] Corben, H. C.; Stehle, P. *Classical Mechanics*, 2nd ed.; Dover Publications, Inc.: New York, 1950.

[198] Gear, C. W. *The numerical integration of ordinary differential equations of various orders*; 1966.

[199] Gear, C. W. *Numerical Initial Value Problems in Ordinary Differential Equations*, 1st ed.; Prentice Hall, 1971.

[200] Verlet, L. *Phys. Rev.* **1967**, *159*, 98–103.

[201] Swope, W. C.; Andersen, H. C.; Berens, P. H.; Wilson, K. R. *J. Chem. Phys.* **1982**, *76*, 637–649.

[202] Pachecho, P. *Parallel Programming with MPI*; Morgan Kaufmann Publishers, Inc.: San Francisco, CA, 1997.

[203] Lusk, E.; Chan, A. *Lec. Notes in Comp. Sci.* **2008**, *5004*, 36–47.

BIOGRAPHICAL SKETCH

Jason M. Swails was born in Binghamton, NY and grew up in Vestal, NY. He attended Binghamton University for his undergraduate studies where he majored in chemistry. In the summer of 2007 after his junior year at Binghamton, he went to the University of Florida and worked in Professor Adrian Roitberg's research lab under the NSF REU program.

The next summer following his senior year at Binghamton University, Jason studied at the University of Buenos Aires in Argentina under an international NSF REU program funded through the University of Florida. That fall he began graduate studies at the University of Florida, and was awarded the NSF GRFP fellowship. He received his Ph.D. from the University of Florida in the summer of 2013.

On July 9, 2011, Jason married Roxy J. Lowry, a graduate from the University of Florida, near Boise, Idaho.