Spring 2017

MOLECULAR DYNAMICS STUDIES OF BIOMIMETIC MEMBRANES

Daniel Ryan Barden
University of New Hampshire, Durham

MOLECULAR DYNAMICS STUDIES OF BIOMIMETIC MEMBRANES

By

D. Ryan Barden
B.S. Chemical Engineering, University of New Hampshire, 2016

Thesis

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science

in

Chemical Engineering

May, 2017

This thesis has been examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Chemical Engineering by:

Thesis Director, Dr. Harish Vashisth, Assistant Professor of Chemical Engineering

Dr. Russell T. Carr, Professor and Chair of Chemical Engineering

Dr. Kang Wu, Assistant Professor of Chemical Engineering

On May 12, 2017

Original approval signatures are on file with the University of New Hampshire Graduate School.

Acknowledgements

Thank you to Danny, Annie, Becki, Rissi, and Petey, love and gratitude for all we share.

I would like to acknowledge and thank my thesis advisor, Dr. Harish Vashisth, for his guidance and insight throughout this project. Thank you to Dr. Russell T. Carr and Dr. Kang Wu for serving on my thesis committee, and thanks also to the entire faculty and staff of the Chemical Engineering Department at the University of New Hampshire. I am grateful for all this department has done for me during my time here. I wish the department and all of its members nothing but success. I would also like to acknowledge and thank my fellow research group member Hossein Mohammadiarani for providing me with valuable references used in this thesis. To the rest of my fellow research group members, Michael Beck, Rhiannon Jacobs, Lev Levintov, and Mohammadjavad Mohammadi, thank you and good luck.

I would like to acknowledge and thank Dr. Manish Kumar and Dr. Yuexiao Shen of *The Kumar Research Group* at Penn State University for their work on block copolymer membranes and collaboration on this project. I would also like to thank and acknowledge Karl Decker of *The Aksimentiev Group* at the University of Illinois at Urbana-Champaign for providing the PAP structure and parameters. Finally, I would like to thank Dr. Fangqiang Zhu for taking the time to answer my questions about his collective diffusion model and providing me with his original structures to test my code.

To everyone mentioned here, thank you all for your various contributions and I wish you well in the future. Good luck separating the signal from the noise.

Contents

List of Figures

List of Tables

Abstract

MOLECULAR DYNAMICS STUDIES OF BIOMIMETIC MEMBRANES

By

D. Ryan Barden

University of New Hampshire, May, 2017

We have explored the conformational dynamics of the peptide-appended pillar[5]arene (PAP) channel in lipid and block copolymer (BCP) membranes through the use of molecular dynamics (MD) simulations. The novel polymeric structures trans-1,4-polybutadiene (PB), trans-1,4-polyisoprene (PI), and poly-2-methyl-2-oxazoline (PMOXA) were created and parameterized. These structures were then used to build and simulate pure $PB_{12}PEO_9$, $PB_{23}PEO_{16}$, $PI_{12}PEO_9$, and $PI_{23}PEO_{16}$ synthetic BCP membranes. In addition, simulations of the PAP channel inserted into lipid (POPC), $PB_{12}PEO_9$, and $PB_{23}PEO_{16}$ membranes were conducted. Results of simulations containing PAP suggest that the membrane environment can affect the channel dynamics and potentially its diffusive as well as transport characteristics. Next, we began to explore the microscopic structure of block copolymer membranes using coarse-grained methods. We tested original MARTINI force-field parameters by simulating the self-assembly of a POPC lipid bilayer. We then used the MARTINI force-field to build and simulate coarse-grained models of $PB_{12}PEO_9$. The original MARTINI force-field was unable to show the self-assembly of $PB_{12}PEO_9$ and must therefore be further optimized to observe the desired behavior.

Chapter 1

Introduction

The demand for clean water continues to rise beyond available resources. In 2008, Shannon et al. reported that an estimated 1.2 billion people didn't have access to safe drinking water [1]. Current estimates project that two-thirds of the worlds population will live in water stressed countries by the year 2025 [2]. Improving desalination and water purification technologies has long been a focus of scientists and engineers. The necessity for efficient and economical methods goes beyond personal consumption, having large environmental and industrial impacts. Water is ubiquitous throughout process engineering, both for its properties as a powerful solvent and efficient energy storage medium. In an effort to meet the ever growing demand for water, we seek new and improved water purification methods to expand our resources to those available in seawater and saline aquifiers which represent 97.5% of all water on the Earth [1].

As of 2011, global water production by desalination was projected to exceed 38 billion m^3 per year, twice the production rate in 2008 [2]. Membrane based separation technologies have long been utilized in process engineering and the majority of desalination is performed using reverse osmosis (RO) membrane technology. In the case of RO, water is forced through a semi-permeable membrane by a pressure gradient. The membrane is designed to allow selective transport of water while rejecting any unwanted solutes. Major problems with this method include high energy consumption, both in pretreatment of the water and filtration, and the negative environmental impacts of the wastewater. Moreover, many harsh chemicals are introduced during pretreatment and large pressure gradients are needed to induce an appreciable flux of clean water across the membrane. These stages have the most opportunity to improve efficiency and reduce the environmental impact of this

process. It is desired to increase the permeability of RO membranes while retaining high selectivity, as well as design them to be more chemically robust to decrease the amount of pretreatment necessary. This technology will present an alternative for water decontamination without the use of harsh chemicals.

The efficiency of RO membranes has undergone continuous improvement since their inception in the 1960s. The current design is thought to have reached a maximum efficiency [2]. Therefore, to achieve the improvements we require, novel membrane materials must be considered. The transport characteristics of biological membranes are of particular interest when engineering next generation materials for use in purifications. It is well known that lipids spontaneously self-assemble in aqueous environments to form lipid bilayers. In cells, the bilayer not only serves as an outer cell envelope but also allows insertion of several membrane proteins including those that transport water molecules, such as Aquaporin (AQP). AQP allows for the selective and passive transport of water molecules on the order of $\sim 10^9$ molecules/s [3]. AQPs have been used to generate membranes with an order of magnitude increase in permeability over current reverse osmosis membranes [4]. While these systems offer proof that more advanced membranes are possible, they cannot be directly used because these systems lack the chemical and mechanical stability needed for process engineering applications.

1.1 Biomimetics

While a significant amount of research has been conducted on lipid based membranes and AQPs [3–7], a larger portion of materials based research is dedicated to the design of biomimetic materials that are inspired from biological systems. Several areas in which biomimetic materials are being used include molecular-scale devices, energy conversion and conservation, and biological self-assembly [8] because nature has the ability to reduce energy use through complex structures [9]. Structures typically become less robust as they increase in complexity. For example, AQP is a complex protein structure, and has been shown to have problems with functionality in different

membrane environments [10]. Biomimetics capture the specialized abilities of complex structures in simple robust synthetic structures. Research on synthetic materials relevant to membrane based separations includes work done on block copolymers [3, 11–15], carbon nanotubes (CNTs) [16, 17], and derivatives of imidazoles [18–20] and pillararenes [3, 20–22].

1.2 Synthetic Transport Channels

Due to their tubular architecture, CNTs have been used in transport applications, and have a reported single channel permeability of $\sim 9.0 \times 10^8$ molecules/s [3]. Entire membranes can be constructed using CNTs and they have been explored as a replacement for biological transport channels with mixed results. Simulation studies have shown promise, with CNTs transporting water at high rates while rejecting solutes [16]. In practice however, CNT membranes have had problems with saline rejection and are difficult to manufacture on large scales [11].

Imidazole will also self-assemble into superstructures capable of selective water transport in membranes. In lipid membranes, these imidazole quartet (I-quartet) channels can passively transport water at a rate of $\sim 10^6$ molecules/s while rejecting all solutes except protons [18]. While the permeability is 2 orders of magnitude lower than that of AQPs, these structures offer insight into designing simple synthetic structures capable of high water selectivity [18]. Pillarene derivatives, in particular peptide appended pillar[5]arene (PAP), have been shown to insert into lipid membranes and passively transport water molecules at a rate of $3.5(\pm 1.0) \times 10^8$ molecules/s [3]. The resulting membranes have a pore density 2 orders of magnitude higher than CNT membranes [3]. PAP channels currently lack the necessary selectivity but can be modified further to change functionality [3]. Simulation studies of PAP channels are a major focus of this thesis.

1.3 Block Copolymers

Polymers have long held an important place in process engineering. A great deal of work has been done in developing theories of polymer chemistry, thermodynamics [23], and structural behavior.

New developments in understanding polymeric structures would be highly impactful in the fields of chemistry, material science, and chemical engineering. Block copolymers (BCPs) are a combination of two or more different polymer species in a single chain. BCPs have the potential to create versatile and highly customized materials engineered for specific needs [24]. For this reason, they represent a large industry with applications including pressure sensitive-adhesives, high-impact plastics, foams, oil additives, and a variety of automobile parts [13]. BCPs also show promise in the field of biosensors due to the diversity of properties which can be selected [25].

Importantly, BCPs also have the ability to self-assemble into different types of structures, depending on the relative lengths and chemical properties of the constituent chains [15]. These structures include synthetic membranes. These synthetic membranes can exhibit bending and area expansion moduli comparable to lipid membranes while having an order of magnitude increase in area strain before rupturing [12, 26]. They have also been shown to have a decrease in water permeability by an order of magnitude [12, 26] when compared with lipid membranes. With such diverse applications resulting from the structural behavior of different BCPs, there is a need for further investigation into the dynamics that invoke these material properties [13].

1.4 Project Goals

In this work, we seek to investigate the dynamic behavior of synthetic BCP membranes and PAP transport channels. These investigations were conducted by first curating a library of different BCP structures along with the corresponding force-field information necessary to conduct molecular dynamics simulations. We offer measurements from these simulations so that the work presented is useful for future experimental investigations and comparison. We also provide information from simulations to quantify the dynamic behavior of these structures with the intention of providing insight when considering BCPs of interest and design refinements to the structures.

While working on the goals outlined above, we faced the following challenges. Certain BCP structures needed to be generated specifically for this work as prior computer models do not exist.

We also had to develop the proper force-field needed to study their dynamic behavior through simulations. We also needed to devise a strategy that would allow us to quantify the dynamic and structural information contained in the simulation trajectories. The structural information about the self-assembly and membrane organization is largely unknown. Self assembly simulations would allow us to analyze the membrane structure predicted by our force-field. Simulations that show self assembly were not feasible with the available computational resources. For this reason we needed to explore the development of a "coarse grained" force-field for our BCPs. In the following chapters, we outline the strategies used to meet these challenges and accomplish the proposed goals.

1.5 Thesis Outline

In Chapter 2, we introduce underlying details of all computational methods and software tools used. These include the basics of statistical mechanics, simulation software, and computational techniques. Chapter 3 describes the building of *de novo* BCP structures suitable for use in molecular dynamics (MD) simulations, and Chapter 4 contains the results of simulations performed with structures built in Chapter 3. Chapter 5 contains a brief exploration into building and simulating "coarse grained" structures. Each chapter begins with a brief introduction describing background information. The introduction of each chapter is followed by sections detailing computational methods/procedures and results with the exception of Chapter 2, which contains no results. Appendices contain simulation details, example scripts, and mini software tutorials.

Chapter 2

Methods

In this Chapter, we provide the basic information on simulation methods necessary to understand work presented in Chapters 3-5. Sections 2.1 through 2.2 contain information fundamental to Molecular Dynamics (MD) simulations. This information is broad and in general will apply to all types of MD simulation work. The remaining sections contain information specific to the work presented in this thesis. These sections will include discussions about certain types of simulations that were employed, as well as descriptions of the various analysis techniques that we used. Briefly, the information contained in this chapter has been limited to what we feel is necessary to understand these topics at their most basic level. If further clarification is needed, or the reader desires a more complete description of any of these topics, we suggest consulting cited sources.

2.1 Statistical Mechanics and Molecular Dynamics Simulations

Developments in computer science have played a key role in the efficiency of molecular dynamics simulations. Equally as important is the subject of statistical mechanics. Statistical mechanics provides the mathematical bridge connecting microscopic behavior to macroscopic thermodynamics. In this section, we briefly introduce some key concepts of the subject to familiarize the reader. A more complete treatment can be found in the following texts [27, 28].

2.1.1 Systems and Phase Space

To understand the fundamentals of statistical mechanics, we first introduce the concept of defining a collection of atoms as a **system**. We can describe the state of this system by specifying the number of atoms (**N**), volume (**V**), and energy (**E**) of the system. We then make the assumption that the total energy of the system depends on each atom's position ($\vec{r_i}$) and momentum ($\vec{p_i}$), i.e. $E(\vec{r_i}, \vec{p_i})$. The **phase space** of our system is then defined as all possible values of the variables $(\vec{r_i}, \vec{p_i})$. A set of phase space positions of all atoms is called a microstate of the system. We now present the argument that there are a great many microstates which can result in the same values of N, V, and E. This collection of microstates is called an ensemble. In this thesis, we performed MD simulations by sampling phase space in two specific ensembles, NVT (constant number of particles, volume, and temperature) and NPT (constant number of particles, pressure, and temperature).

2.1.2 Entropy, Probability, and The Partition Function

The connection between entropy and macroscopic thermodynamic variables has been well established through the relations first derived by James Clerk Maxwell [29]. The third law of thermodynamics (Equation 2.1) provides us with a definition of entropy (S) in terms of the microscopic variable $\Omega(E)$, called the density of states, and Boltzmann's constant (k_b).

$$S = k_b \ln \Omega(E) \tag{2.1}$$

In the above equation, we see by definition the density of states is some function of the systems energy. It represents the number of all available states to a system with energy E. We note the postulate that a state with energy E_s is available to a system with energy E if $E_s < E$. If we let $\Omega(E_s)$ represent the total number of states with energy E_s, the probability of the system being in one of those states is given as Equation 2.2.

$$P(E_s) = \frac{\Omega(E_s)}{\Omega(E)} \tag{2.2}$$

It should be obvious that even for microscopic systems of modest size, counting the total number of available states is impossible. Through some mathematical reasoning, Equation 2.2 for the NVT ensemble can be estimated by Equation 2.3 where T is the temperature of the system, the variable $\beta \equiv \frac{1}{k_b T}$, and Z is a new mathematical object called the partition function.

$$P(E_s) = \frac{e^{-\beta E_s(\vec{r},\vec{p})}}{Z} \tag{2.3}$$

With the requirement that Equation 2.3 is normalized, the partition function must have the following form.

$$Z = \sum_s e^{-\beta E_s(\vec{r},\vec{p})} \tag{2.4}$$

The partition function is a powerful tool and allows us to connect our sampling of phase space $(\vec{r_i}, \vec{p_i})$ to macroscopic thermodynamic properties.

2.1.3 Ensemble Averaging

When measuring certain properties in our simulations, it is likely that the value of the property will be fluctuating rapidly. This is due to the chaotic nature of many body physics. The macroscopic value of this property is taken as a time average of all fluctuations. The time average of some property $M(t)$, is given by the following equation.

$$\overline{M} = \lim_{\tau\to\infty} \frac{1}{\tau} \int_0^\tau M(t)\ dt \tag{2.5}$$

In Equation 2.5 we explicitly defined M as a function of time, $M(t)$, but M can also be considered as a function of the overall phase space, $M(\vec{r},\vec{p})$. We define a new average over all of the phase space variables called the ensemble average given in Equation 2.6, where $P(\vec{r},\vec{p})$ is the probability distribution introduced in Equation 2.3.

$$\langle M \rangle = \iint M(\vec{r},\vec{p}) P(\vec{r},\vec{p})\ d\vec{r}d\vec{p} \tag{2.6}$$

For certain properties of any system, it holds true that $\overline{M} = \langle M \rangle$. Properties that satisfy this condition are said to be *ergodic*. We can take advantage of this by keeping our system sizes relatively small, and combining results from multiple independent trajectories. It should be evident from Equation 2.6 that we would like to perform simulations that maximize the sampling of phase space. In Section 2.2 we discuss different tools and methods we used to perform our simulations.

2.2 Simulation Software and Force-Field

Work presented in this thesis was completed using various computational resources. In this section, we provide a brief description of each of the resources used.

2.2.1 Visualization Software: VMD

Visual Molecular Dynamics (VMD) [30] software was an integral part of all research we conducted. Developed and maintained by *The Theoretical and Computational Biophysics Group* (TCBG) at the University of Illinois at Urbana-Champaign, VMD contains a wealth of useful tools and resources for anyone performing MD simulations. VMD was used to visualize and analyze MD trajectories as well as build and parameterize novel structures.

2.2.2 The CHARMM Force-Field

All classical MD simulations were performed using CHARMM (**C**hemistry at **HAR**vard **M**acromolecular **M**echanics) [31] family of force-fields. All CHARMM [31] force-fields use the same set of pairwise potential functions to define the potential energy of any simulation system. The general form of this force-field is given by Equations 2.7 through 2.13.

$$U = U_{bonds} + U_{angles} + U_{dihedrals} + U_{impropers} + U_{Urey-Bradley} + U_{non-bonded} \tag{2.7}$$

$$U_{bonds} = \sum_{bonds} k_b(b - b_o)^2 \tag{2.8}$$

$$U_{angles} = \sum_{angles} k_\theta(\theta - \theta_o)^2 \tag{2.9}$$

$$U_{dihedrals} = \sum_{dihedrals} k_\phi[1 + \cos(n\phi - \delta)] \tag{2.10}$$

$$U_{impropers} = \sum_{impropers} k_\psi(\psi - \psi_o)^2 \tag{2.11}$$

$$U_{Urey-Bradley} = \sum_{Urey-Bradley} k_u(u - u_o)^2 \tag{2.12}$$

$$U_{non-bonded} = \sum_i \sum_{j>i} \epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \sum_i \sum_{j>i} \frac{q_i q_j}{4\pi\epsilon_o r_{ij}} \tag{2.13}$$

The bond, angle, improper, and Urey-Bradley type terms are modeled after harmonic oscillators, with the potential energy equal to a force constant, k_i, multiplied by the square of the displacement from an equilibrium position, i_o (i represents b, θ, ψ, and u). Impropers account for out of plane bending and Urey-Bradley terms account for 1,3 non-bonded interactions causing angle bending. The dihedral term accounts for any bond rotations. As bonds rotate, they sample an energy landscape which is periodic in increments of 2π. In this term, k_ϕ is a force constant, n is the periodicity, ϕ is the rotation angle, and δ is the phase shift. Multiple terms with different periodicities and phase shifts can be used to parametrize a single bond rotation. The non-bonded terms recreate electrostatic and van der Waals type forces between non-bonded atoms. The electrostatic interactions are represented by a Coulombic potential where q represents the charge of an atom, ϵ_o is the permittivity of free space, and r_{ij} is the distance between the centers of mass of two atoms. The van der Waals forces are represented by a Lennard-Jones potential function. In this equation, ϵ_{ij} represents the energy well depth, σ_{ij} is the hard sphere radius, and r_{ij} is the distance between the centers of mass of two atoms. Different versions of this force-field contain parameters that have been optimized for simulating specific types of molecules. In addition to parameters generated and optimized manually, all simulations were performed using the CHARMM36_prot, CHARMM36_cgenff, and CHARMM35_ethers force-fields. Parameters to simulate the PAP channel were previously used in MD simulations and were adopted as is [3].

2.2.3 The MARTINI Force-Field

The MARTINI [32] force-field is a coarse-grained (CG) force-field first developed to simulate lipids and proteins. This force-field uses some of the same pairwise potential terms as the CHARMM [31] force-field, specifically those defined for bonds, angles, dihedrals, and non-bonded interactions. The MARTINI [32] force field saves computational resources by defining groups of individual atoms as "super atoms". This reduces the number of degrees of freedom (DOF) in a given system and accelerates simulation time. This strategy of reducing the DOF is what classifies it as a coarse grained force-field. This approach is not suitable for capturing all types of dynamic behavior but can be effective for certain applications. In Chapter 5, we briefly discuss CG simulations using the MARTINI [32] force-field.

2.2.4 Simulation Software: NAMD

All simulations were performed with NAMD (**NA**noscale **M**olecular **D**ynamics) software [33]. NAMD generates simulation trajectories by numerically integrating Newtons equations of motion given in Equation 2.14.

$$m_i \frac{d^2 \vec{r_i}}{dt^2} = -\nabla U(\vec{r_i}) \tag{2.14}$$

Here the force on each atom is derived from the potential function defined in Equations 2.7 through 2.13. In Equation 2.14, m_i represents the mass of each atom and $\vec{r_i}$ is each atom's radial position vector. The integration algorithm in NAMD is designed for parallel computations and results in negligible violations in energy conservation across large numbers of integration steps, preserving the time reversibility of the equations of motion. The integrator includes methods to control system volume, pressure, and temperature and gives users the ability to define forces through `tcl` scripting. The source code is freely available for academic use and remains open source for users to modify. Details about the integration algorithm, computational performance, and the various features of NAMD can be found in this 2005 review paper [33].

2.3 Simulations of the Membrane Transport Channel

MD Simulations involving membranes with embedded structures such as proteins or synthetic transport channels require careful considerations for stability. Simulations of this type must be carried out in three separate steps. Simulations performed in Chapter 4 that included a synthetic transport channel were carried out with the same procedure described in this section. A more complete description of this procedure can be found in a tutorial on the subject [34].

All simulations containing a synthetic transport channel inserted into a membrane were conducted by first creating a PDB and PSF file of an equilibrated membrane structure. Detailed descriptions of PDB and PSF files can be found in Sections 3.1.1 and 3.1.3, respectively. For POPC membranes, the membrane builder plugin of VMD [30] was used to generate these files. Simulations using block copolymer (BCP) membranes required additional simulations of membrane structures which is described in Section 4.1.1. The equilibrated structures of membranes were then combined with the transport channel structure by aligning the center of mass of each to the origin and orienting the channel axis along the z direction. All steric conflicts must be removed from the newly combined structure. This was done by removing any lipids or BCPs within a certain distance of the transport channel ($\sim$2-5 Å). Care is taken to choose a cutoff distance which is not so large as to remove too many lipids/BCPs while still reconciling any structural issues. The combined structure was then solvated and any water molecules within the hydrophobic layer of the membrane were manually removed from the system.

During the first step of each simulation, all atoms in the system are fixed with the exception of the hydrophobic section of the membrane. This allows the hydrophobic chains to acclimate to the newly inserted channel and fill any voids. We experienced problems with systems crashing during the first step. The cause of the crashing was hydrophobic chains starting too close to the transport channel. As a result, atoms would have velocities too high for the system to handle at a 2 fs time step. To fix this, a 1 fs time step was used in the first phase of all simulations of this type. These simulations are short, lasting only 0.25 ns so the smaller time step resulted in a negligible change

in simulation time. These simulations were conducted in the NVT ensemble.

The second step of the simulation was carried out in the NPT ensemble with the area of the periodic cell constrained to a constant value. During this step of the simulation, new constraints were placed on the atoms in the system. Atoms in the transport channel were held fixed and external forces were applied to keep water from entering the hydrophobic section of the membrane. All other atoms were allowed to move freely. A 2-fs time step was used and simulations ran for 0.25 ns. The third and final steps of simulations were carried out in the NPT ensemble with a fixed area. A 2-fs time step was employed and all atoms were allowed to move freely. These were long scale simulations which ranged from ∼200-1000 ns in length.

2.4 Trajectory Analysis Techniques

Various techniques were used to analyze all simulations performed. In this section, we provide a brief description of each of these techniques and the significance of the information.

2.4.1 Density of Species

Understanding the overall structure of any given membrane was an important goal of the simulation work we performed. This can be done by visually inspecting individual snapshots of our simulations. This is decidedly insufficient as it will only give us information about the membrane structure at one specific instant in time. To understand the average cross-sectional structure, we developed a way of combining information of this type from all runs. Our code employed a coordinate system whose origin is always aligned with the center of mass (COM) of the membrane. Moving in a direction normal to the membrane, the membrane was separated into 1 Å thick slices. The number of atoms in each of the slices is counted and sorted by species. This is done for each slice of the simulation space for each frame in a simulation trajectory. Then the values for each slice across all trajectory frames are averaged. An example of this code can be found in Appendix B.7.

2.4.2 Root Mean Squared Displacement

Measuring the root mean squared displacement (RMSD) of a structure is a basic technique used to quantify conformational changes during MD simulations. At each frame in the trajectory, the structure is aligned with its initial structure or some reference structure. The displacement of each atom is squared and averaged together. The square root of this value is taken and we plot the RMSD of the structure as a function of simulation time. The RMSD values will initially change rapidly but then start to fluctuate around an average value as the system reaches equilibrium. As a result, this measurement also provides information about when a system reaches equilibrium. An example of this code can be found in Appendix B.8.

2.4.3 PAP Orientation

In an effort to understand the effect of the membrane environment on the PAP channel, we tracked the orientation of the channel within each membrane throughout all simulations. Simulations were set up so that the z-coordinate was always normal to the surface of the membrane. A vector was defined pointing through the center and axis of the cylindrical PAP channel, and its angle with respect to the z-coordinate was measured as a function of time. An example of this script can be found in Appendix B.9.

2.4.4 PAP Diffusivity

To understand the mobility of PAP channels in different membrane environments, we used measurements of the diffusion coefficient. Calculations of the diffusion coefficient were obtained by taking advantage of the Einstein relation [35] shown as Equation 2.15, where where D is the diffusion coefficient, d is the dimensionality of the system, and $\langle r^2 \rangle$ is the mean squared displacement (MSD).

$$\frac{\partial \langle r^2 \rangle}{\partial t} = 2dD \tag{2.15}$$

To obtain the MSD, coordinates of all trajectories were first unwrapped to remove the effect of periodic cell boundary conditions. System drift was removed by tracking the center of mass of the system and adjusting the displacement of the PAP channel accordingly. The squared displacement of the channel, r^2, was then plotted as function of time. The MSD as a function of time, $\langle r^2(t) \rangle$, was obtained by ensemble averaging the plots of $r^2(t)$. Each plot was split into 20 ns segments which were then realigned and averaged. Note that r is a vector containing (x, y, z) coordinates. Due to the Einstein relation's dependence on dimensionality, we could measure the PAP channel's diffusivity in any combination of the three spatial dimensions. We chose three different combinations, overall (x, y, z), lateral in the membrane plane (x, y), and normal to the membrane surface (z). An example of the script used to obtain r and its components can be found in Appendix B.10.

2.4.5 Permeability

The permeability of the PAP channel was measured by a widely used collective diffusion model developed by Zhu et al [36]. This method has been used previously to measure the permeability of the PAP channel in a lipid membrane [3]. A new variable, n, is defined by Equation 2.16 to characterize the movement of water in the channel, where $S(t)$ represents the set of water molecules located inside a volume of interest within the channel at a time t, dz_i represents each water molecules displacement in the z direction, and L is the length of the volume of interest.

$$dn = \sum_{i \in S(t)} \frac{dz_i}{L} \tag{2.16}$$

Equation 2.16 can be integrated to obtain $n(t)$ which obeys the Einstein relation [35] given as Equation 2.15. Using this relationship, we can obtain the diffusion coefficient of n. The following relationship between the diffusion coefficient, D_n, volume of a single water molecule, υ_w, and the channel permeability, p_f, has been shown by Zhu et al [36].

$$p_f = \upsilon_w D_n \tag{2.17}$$

To obtain the MSD as a function of time, $\langle n^2(t) \rangle$, the volume of interest was defined as a cylinder 6 Å in diameter, 8 Å in length, and aligned with the center of mass of the carbon atoms of the dimethoxy benzene ring of the channel. This is the same volume used in previous PAP channel simulations [3]. The square of the collective displacement coordinate, $n^2(t)$, was calculated for each trajectory. The $n^2(t)$ trajectories were split into 1 ns segments and aligned so $n^2(0) = 0$ for all segments. The aligned segments were averaged together to give $\langle n^2(t) \rangle$. This value was plotted to determine the permeability of the channel. An example of the script used to obtain $n(t)$ can be found in Appendix B.11.

Chapter 3

Construction and Parameterization of Novel Polymeric Structures

Polymer science has a broad range of applications in process engineering. Polymers are often the desired product of chemical processes, or used in the actual processes themselves. A significant amount of research has been devoted to this field and there is vast information about the chemical behavior of many polymers. For certain applications, it is of interest to understand these structures on a more mechanical level. To gain molecular-scale insights into dynamic behavior of polymers, Molecular Dynamics (MD) simulations are emerging as an important tool [3, 37–40]

MD simulations of biological molecules are typically carried out using structures obtained by experimental methods (For example, X-ray crystallography and NMR spectroscopy). These structures can be obtained from the Protein Data Bank (PDB), which is a repository for experimentally determined structures of biomolecules [41]. The PDB does not contain structures of the polymers we wish to simulate. Visual Molecular Dynamics (VMD) [30] is a freely-available software that contains built-in tools which generate accurate theory based molecular structures of small organic molecules. In this chapter, we take advantage of these tools to construct the small monomers of trans-1,4-polybutadiene (PB), trans-1,4-polyisoprene (PI), and poly-2-methyl-2-oxazoline (PMOXA) that serve as building blocks for the desired polymers.

Building chemical models suitable for both visualization and simulation presented in this thesis requires generating four different types of files. The content of these four file types (PDB coordinate file, Protein Structure file, Topology file, and Parameter file) is outlined in Sections 3.1.1 through

3.1.4. The molefacture plug-in of VMD [30] allows a user to generate a new structure from small predefined organic compounds and groups. Users can *de novo* build the chemical structure of any molecule from these fragments. This results in preliminary PDB and topology files which can be further refined by the user by employing advanced simulation protocols. Generating force-field parameters requires the use of the force field tool kit (ffTK) [42] plug-in in VMD [30]. The plug-in allows users to easily identify missing parameters for the new structure, and utilizes the quantum chemistry program Gaussian [43] to generate and optimize these missing parameters. It is important to note that successful completion of this stage will require access to Gaussian [43].

3.1 Description of Necessary Files (.pdb, .psf, .top, .par)

Building a new molecule requires detailed knowledge of various types of files. Therefore, automated tools contained within VMD [30] are useful when beginning this process. The files generated with these tools will require careful inspection and some manual editing/troubleshooting throughout the process. A thorough understanding of the format and information contained within each of these files will make the whole process efficient. The information presented in Sections 3.1.1 through 3.1.4 is taken from the NAMD tutorial [44]. A summary of the most relevant information is presented in this chapter as a concise reference. If a more detailed description of the files is desired, the reader is referred to the link: `http://www.ks.uiuc.edu/Training/TutorialsOverview/namd/namd-tutorial-win-html/namd-tutorial-win.html`.

3.1.1 The PDB file

PDB files, sometimes referred to as coordinate files, contain the spatial information of all atoms within a molecule. In addition to the position of each atom, the PDB file identifies element types (i.e. C, N, O...). Each line in this file represents information for a single atom. The information can be split in two different categories, unique identifiers and non-unique identifiers. The first number at the beginning of each line represents the unique index number of each atom. Every

atom within a PDB file will have a unique index number as well as a set of unique Cartesian (x, y, z) coordinates. Non-unique identifiers include residue name (resname), residue id (resid), chain, segment name/id (segname/segid), and element. Non-unique identifiers will not identify a specific atom on their own, however no two atoms will have the same values in all of these categories. Below is an example of excerpts from a PDB file. From left to right, the information in each line is record type, atom ID, atom name, residue name (resname), chain name (chain), residue ID (resid), x, y, and z coordinates, occupancy, temperature factor (called β-factor), segment name (segname), and element type.

```
ATOM     1  C1   BDE X   1       0.000   0.000   0.000  1.00  0.00      1     C
ATOM     2  C3   BDE X   1       2.130   1.308  -0.066  1.00  0.00      1     C
ATOM     3  H11  BDE X   1      -0.326  -0.473   0.819  1.00  0.00      1     H
ATOM     4  H12  BDE X   1      -0.326  -0.473  -0.819  1.00  0.00      1     H
```

It is important to note that a PDB file does not contain information on inter-atomic bonds. These files also treat all elements on equal ground, that is to say given only a PDB file, there is no information to distinguish an sp^3 hybridized carbon from an sp^2 hybridized carbon. This information is essential to modeling the dynamic behavior of molecules. It is obvious that a model comprised of only a PDB file is incomplete for the purpose of studying dynamics.

3.1.2 The Topology File

A topology file is perhaps the most important file type for new users to become familiar with. Topology files contain information about each atom's orbital structure, charge, mass, and bonding. A table appears at the beginning of all topology files that gives the mass in amu of each of the different atom types appearing in the file.

```
MASS     1 CG321  12.01100  C
MASS     1 CG331  12.01100  C
MASS     1 CG2D1  12.01100  C
MASS     1 HGA2   1.00794  H
MASS     1 HGA4   1.00794  H
MASS     1 HGA3   1.00794  H
```

Entries in the topology file are organized by the residue name. The letters RESI signal the begining of a new residue. This line contains the name and overall charge of the residue. Each atom in a residue must be given a unique name. In the example topology file below, there is only one C2 atom. Next to the atom name is the atom type followed by the atoms charge. The atom type is determined by electron orbital hybridization. The lines beginning with BOND contain information about which atoms are bonded together in the residue. There is no need to specify the bond type as this information is contained within the atom type. The next lines beginning with IC are the internal coordinates of the residue. These will become important when we generate polymers using scripts.

```
AUTO ANGLES DIHE
DEFAULT FIRST HBD1 LAST HBD4
RESI  BDE     0.00
GROUP
ATOM  C1  CG321  -0.18000 !   H11 H12
ATOM  C3  CG2D1  -0.15000 !     \ |
ATOM  H11 HGA2    0.09000 ! (C4)-C1      H31
ATOM  H12 HGA2    0.09000 !        \     /
ATOM  C2  CG2D1  -0.15000 !         C2=C3
ATOM  H21 HGA4    0.15000 !        /     \
ATOM  H31 HGA4    0.15000 !      H21      C4-(C1)
ATOM  C4  CG321  -0.18000 !               | \
ATOM  H41 HGA2    0.09000 !             H41 H42
ATOM  H42 HGA2    0.09000 !
BOND C1 H11  C1 H12  C1 C2
BOND C3 C2  C3 H31  C3 C4  C2 H21
BOND C4 H41  C4 H42
BOND C1 -C4

IC -C3 -C4 C1  C2     0.000  0.000  180.0  0.000  0.000
IC -C4  C1 C2  C3     0.000  0.000  180.0  0.000  0.000
IC  C1  C2 C3  C4     0.000  0.000  180.0  0.000  0.000
IC  C2  C3 C4 +C1     0.000  0.000  180.0  0.000  0.000
```

Entries designated as PRES are called patches. These patches are extremely important when building polymers by "patching" monomers together. These entries are typically created manually.

A detailed description of patches can be found in Section 3.2.2. It is important to note that topology files are not unique to a single PDB file.

```
PRES HBD4            0.00  ! Complete terminal methyl at C4
                           !
ATOM H43   HGA3     0.09   !
ATOM C4    CG331   -0.27   !
ATOM H41   HGA3     0.09   !
ATOM H42   HGA3     0.09   !
BOND H43 C4
```

3.1.3 The PSF File

Protein structure files (PSF) are companions to PDB files. They contain information needed to properly model atomic interactions described by the inter-atomic potential, i.e., a force-field. All information in a PSF file can be split into 7 different sections. The first line in each section begins with a number representing the total number of entries that follow. The "!" is included to comment out the section title. It is important to notice that none of the sections then contain identifying information when read by the necessary software. For this reason, PSF files must always contain the same sections in the same order. We will go through each of these sections in the order they appear in these files. The first section designated !NTITLE, contains remarks about different topology files and patches that were used when the PSF file was generated.

```
     3 !NTITLE
 REMARKS original generated structure x-plor psf file
 REMARKS topology /home/bardenr/thesis/pmoxa_build/molfac/mox_1.top
 REMARKS segment 1 { first NONE; last NONE; auto angles dihedrals }
```

The next section designated !NATOM contains information about all the atoms in the structure. This particular structure contains 15 total atoms; only partial entries are shown below. Each line represents a different atom and the entries from left to right are atom ID, segment name (segname), residue ID (resid), residue name (resname), atom name, atom type, charge, mass, and an unused 0. Atoms in the PSF files are paired with their PDB counterpart by atom ID.

```
15 !NATOM
   1   1     1      PMX      O      OG2D1   -0.510000     15.9994   0
   2   1     1      PMX      C1     CG201    0.510000     12.0110   0
   3   1     1      PMX      C2     CG331   -0.270000     12.0110   0
```

It is followed by the section with information on bonds. This particular example has 14 total bonds. Each line contains a list of no more than 4 bonded atoms designated by their atom ID's. Every two atom's listed represent a bonded pair. For example, pairs in the first line are atoms 1-2, 2-3, 2-4, and 3-5.

```
14 !NBOND: bonds
   1        2        2        3        2        4        3        5
   3        6        3        7        4        8        4        9
   8       13        8       14        8       15        9       10
   9       11        9       12
```

The next section contains information on all angles for the structure. This example has 24 total angles but has been truncated. Atoms are again represented by their atom ID number. In this section every three atoms represents a group defining an angle with no more than three angle groups per line. For example, the groupings on the first line are 1-2-4, 1-2-3, and 2-4-9.

```
24 !NTHETA: angles
   1        2        4        1        2        3        2        4        9
   2        4        8        2        3        7        2        3        6
   2        3        5        3        2        4        4        9       12
```

The next section contains information on all dihedral angles for the structure. Dihedral angles are defined by groups of four atoms. This example has a total of 22 dihedral angles but has been truncated. Atoms are again identified by an atom ID number. Each line contains two dihedral groups. For example, the first line defines the following two dihedral angles 1-2-3-5 and 1-2-3-6.

```
22 !NPHI: dihedrals
   1        2        3        5        1        2        3        6
   1        2        3        7        1        2        4        8
   1        2        4        9        2        4        8       13
```

The remaining sections are not required for the work presented in Chapter 3. When generating new polymers, these entries will likely remain blank. An example of the remaining blank entries can be seen below. If the reader desires more information about each of these, it can be found in the NAMD tutorial [44].

```
0 !NIMPHI: impropers

0 !NDON: donors

0 !NACC: acceptors

0 !NNB

1       0 !NGRP
```

3.1.4 The Parameter File

The parameter file contains information relevant to the force-field. Unlike the files we have previously discussed, it contains no information about the molecular structure. Similar to a topology file, parameter files do not pair with specific PDB or PSF files. A single parameter file can contain sufficient information to define a force-field for many different structures. Atoms are identified by each atom-type in these files. Typical parameter files used in this work contain entries for each of the terms in the CHARMM [31] force-field (bond, angle, dihedral, improper, and non-bonded). A detailed description of the CHARMM [31] force-field can be found in Sections 2.2.2 and 3.3.1. Each section of the parameter file contains a description of the force-field term as well as headings for the information. For this reason, parameter files are easier to navigate than other previously discussed file types. These files can become very large as there are many different ways to make combinations of existing atom types. An example of a bonds section is shown in the following. Sections for all other terms follow a similar format. The first line identifies the term. The next line is a mathematical representation of the potential function followed by entries identifying the units of the parameters. The next line contains headings for entries which is then followed by a list

of all atom combinations defined in the parameter file for that term. In the example below, the last line defines a bond between atom types CG331 and HGA2. This bond has a force constant $k_b = 999.508 \frac{kcal}{mol \cdot \mathring{A}^2}$ and a bond length $b_0 = 1.097$ Å.

```
BONDS
!V(bond) = Kb(b - b0)**2
!
!Kb: kcal/mole/A**2
!b0: A
!
!atom type Kb b0
!
CG331  HGA2  999.508    1.097    !
```

3.2 Constructing A New Polymer

With the information provided in Section 3.1, it is possible to generate the chemical structure of a new molecule using any text editor. This becomes tedious as molecules increase in size and complexity. VMD [30] contains a plug-in called molefacture which can automate the process of generating new PDB, PSF, and topology files. Section 3.2.1 provides information needed to successfully complete the work presented in Chapter 3. For a more detailed description, the reader is referred to the following link: `www.ks.uiuc.edu/Training/Tutorials/`.

3.2.1 Molefacture

The most efficient way to build polymers of any size is to first build the corresponding monomer in molefacture. Molefacture is better suited to building relatively small and simple molecules. It is natural then to first build the monomer, which will act as a building block for generating polymer chains. This monomer can then be used with `tcl` scripts to automatically generate polymer chains of any size. This is much faster than building a complete polymer in molefacture every time a new chain is desired. This strategy was used for all polymers built in Chapter 3. In this section,

we include important details of the trans-1,4-polybutadiene (PB) build. A more detailed tutorial version of the PB build can be found in Appendix C.1.

To build a monomer of PB, we used butane as a base molecule. The monomer for PB is essentially 2-butene. This makes the build relatively straight forward. We deleted one hydrogen atom from the carbon atoms at positions 2 and 3. The single bond between the second and third carbon was converted to a double bond. After raising the bond order, the purple markers signifying empty orbitals are gone, while the bond has a cylinder surrounding a portion of it representing the new double bond (see Figure 3.1). Finally, we deleted one hydrogen atom from each terminal carbon. We did this because we are modeling a single unit of a polymer. The empty orbitals are addressed in Section 3.2.2 when we discuss building patches.

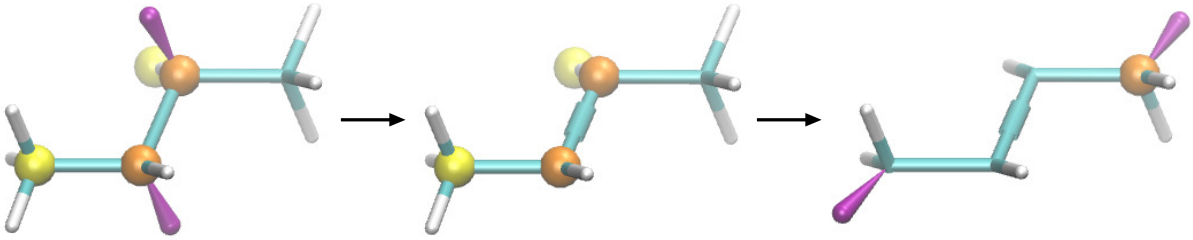


FIGURE 3.1: Snapshots of modifications to the 2-butene fragment in Molefacture

The most important part of building a molecule in molefacture is manually editing each atom in the atom list. We changed the name, type, and charge for each atom in the new molecule. For our monomer, we named our carbons C1, C2, C3, and C4. Hydrogens were named using their periodic table symbol and 2 additional numbers. The first corresponds to the number of the carbon to which the hydrogen is bonded, while the second is a unique identifying number. For example, the three hydrogens bonded to carbon C1 were named H11, H12, and H13. Every atom in our structure needs a unique name for the purposes of applying patches and generating PSF files.

Atom types are determined by the orbital bond structure. When editing the atom type and charge, we browsed structures previously defined in existing topology files. It is important to stay

consistent when assigning atom types as they are not arbitrary. Atom types are what specify force-field parameters. In the `top_all_36cgenff.rtf` topology file, there is an entry for 2-hexene which is included below. This molecule has all the information we needed to define our monomer as well as the patches we built in Section 3.2.2.

```
RESI HXE2         0.00 ! C6H12 2-hexene, yin/adm jr.
GROUP
ATOM  C1  CG331  -0.27 !  H12 H13
ATOM  H11 HGA3    0.09 !    \ |
ATOM  H12 HGA3    0.09 ! H13-C1    H31 H51 H52
ATOM  H13 HGA3    0.09 !       \    /    \ /
GROUP                  !        C2=C3    C5    H61
ATOM  C2  CG2D1  -0.15 !       /     \  /  \  /
ATOM  H21 HGA4    0.15 !     H21      C4    C6-H62
GROUP                  !             / \      \
ATOM  C3  CG2D1  -0.15 !           H41 H42     H63
ATOM  H31 HGA4    0.15 !
GROUP
ATOM  C4  CG321  -0.18
ATOM  H41 HGA2    0.09
ATOM  H42 HGA2    0.09
GROUP
ATOM  C5  CG321  -0.18
ATOM  H51 HGA2    0.09
ATOM  H52 HGA2    0.09
GROUP
ATOM  C6  CG331  -0.27
ATOM  H61 HGA3    0.09
ATOM  H62 HGA3    0.09
ATOM  H63 HGA3    0.09
```

Atoms C2 and C3 were assigned atom types CG2D1 with a charge of -0.15 as they are defined above. The hydrogens bonded to those atoms were assigned types consistent with those described as well. When assigning atom types to the C1 and C4 atoms in our structure, we used the information for the C4 atom in 2-hexene. We did so as this structure is modeled after a single unit appearing within a larger polymer chain. We changed the entry for resname to “BDE” and chain to “B”, then

had molefacture generate PDB, PSF, and topology files. An example of the resulting topology file can be found in Section 3.2.2.

3.2.2 Patches and Topology File Modifications

Patches are entries that appear at the end of topology files. They contain information that allows larger structures to be built from multiple residues or the creation of modified residues. This is done with the psfgen tool included in VMD [30]. A brief description of how we utilized the psfgen feature is outlined in Section 3.2.3, while a more comprehensive tutorial can be found online. In this section, we describe how we manually built a patch and edited the topology file we created using molefacture. A copy of the original topology file generated by Molefacture is presented below. This is followed by the full modified file and a numbered list containing detailed information of all the changes made to the file. The modified file contains numbered entries next to the new/modified lines. The numbers correspond to the detailed description in the list of changes.

Original Topology File

```
*>>>>>> CHARMM topology file generated by Molefacture <<<<<<
27 1

MASS     1 CG321  12.01100  C
MASS     1 HGA2   1.00794  H
MASS     1 CG2D1  12.01100  C
MASS     1 HGA4   1.00794  H

AUTO ANGLES DIHE

RESI  BDE     0.00
GROUP
ATOM   C1 CG321 -0.18000
ATOM  H11 HGA2  0.09000
ATOM  H12 HGA2  0.09000
ATOM   C2 CG2D1 -0.15000
ATOM  H21 HGA4  0.15000
ATOM   C3 CG2D1 -0.15000
ATOM  H31 HGA4  0.15000
ATOM   C4 CG321 -0.18000
ATOM  H41 HGA2  0.09000
ATOM  H42 HGA2  0.09000
BOND C1 H11  C1 H12  C1 C2
BOND C2 H21  C2 C3  C3 H31  C3 C4
BOND C4 H41  C4 H42

END
```

Modified Topology File

```
*>>>>>> Modified CHARMM topology file generated by Molefacture <<<<<<
27 1

MASS     1 CG321  12.01100  C
MASS     1 CG331  12.01100  C  ! 1. New mass entry for atom type CG331.
MASS     1 CG2D1  12.01100  C
MASS     1 HGA2   1.00794  H
MASS     1 HGA4   1.00794  H
MASS     1 HGA3   1.00794  H   ! 2. New mass entry for atom type HGA3.

AUTO ANGLES DIHE
DEFAULT FIRST HBD1 LAST HBD4   ! 3. Line specifying the default patches
                               ! applied to the residue.
RESI  BDE     0.00
GROUP                          ! 4. Map of the atoms.
ATOM  C1  CG321  -0.18000 !   H11 H12
ATOM  C3  CG2D1  -0.15000 !     \ |
ATOM  H11 HGA2    0.09000 ! (C4)-C1      H31
ATOM  H12 HGA2    0.09000 !        \     /
ATOM  C2  CG2D1  -0.15000 !         C2=C3
ATOM  H21 HGA4    0.15000 !        /     \
ATOM  H31 HGA4    0.15000 !      H21      C4-(C1)
ATOM  C4  CG321  -0.18000 !               | \
ATOM  H41 HGA2    0.09000 !             H41 H42
ATOM  H42 HGA2    0.09000 !
BOND C1 H11  C1 H12  C1 C2
BOND C3 C2  C3 H31  C3 C4  C2 H21
BOND C4 H41  C4 H42
BOND C1 -C4                    ! 5. Bond connecting two monomers
                               ! 6. Internal coordinates table
IC -C3 -C4 C1  C2     0.000  0.000  180.0  0.000  0.000
IC -C4  C1 C2  C3     0.000  0.000  180.0  0.000  0.000
IC  C1  C2 C3  C4     0.000  0.000  180.0  0.000  0.000
IC  C2  C3 C4 +C1     0.000  0.000  180.0  0.000  0.000
```

```
                                ! 7. Two new patches for terminal residues
PRES HBD4            0.00  ! Complete terminal methyl at C4
                           !
ATOM H43   HGA3     0.09   !
ATOM C4    CG331   -0.27   !
ATOM H41   HGA3     0.09   !
ATOM H42   HGA3     0.09   !
BOND H43 C4

PRES HBD1            0.00  ! Complete terminal methyl at C1
                           !
ATOM H13   HGA3     0.09   !
ATOM C1    CG331   -0.27   !
ATOM H11   HGA3     0.09   !
ATOM H12   HGA3     0.09   !
BOND H13 C1

END
```

List of Modifications

1. New mass entry for atom type CG331: Patches are essentially used to change existing atom types, add new atoms/bonds, and delete existing atoms/bonds. Our terminal carbon atoms are a different atom-type than those we defined for our interior. We need to define mass for this terminal carbon atom-type (CG331). This atom type can be seen in the topology for 2-hexene included in Section 3.2.1.

2. New mass entry for atom type HGA3: Similar to the carbon atom, we need to define mass for the hydrogens with atom-type HGA3. This atom type can be seen in the topology for 2-hexene included in Section 3.2.1.

3. Line specifying the default patches applied to the residue: The psfgen tool in VMD [30] builds structures in multiple segments, where several residues make up each segment. This line will instruct psfgen to automatically apply patch HBD1 to a BDE residue if it is the first residue in the segment. The line also instructs psfgen to apply patch HBD4 to a BDE residue if it is the last residue on a segment. Both of these patches are defined in item 7.

4. Map of the atoms: This addition is optional and will not be read by any software. It is advisable as it will give anyone who is viewing the file a clear picture of which atoms are connected.

5. Bond connecting two monomers: We created our monomer with the intent of making a larger polymer. Adding this bond will instruct psfgen to connect the C4 atom of a BDE residue with the C1 atom of the next BDE residue. The "-" appearing prior to C4 informs psfgen that these atoms are not on the same residue. If the "-" was not present, psfgen would erroneously create a bond between the C1 and C4 atom of the same residue.

6. Internal coordinates table: As residues are added to a segment, psfgen will guess their positions. You will likely end up with an initial residue that looks good, while the rest will have many overlapping atoms. The internal coordinates table will give psfgen instructions on how to place atoms relative to one another. This requires some trial and error to get right, but the result does not have to be perfect. Atoms just need to be separated from one another. Energy minimization prior to simulation will adjust any unrealistic bond lengths.

7. Two new patches for terminal residues: Patches are identified by the heading PRES, short for patch residue. On the same line as the heading is the patch name and the overall charge of the patch. Our patches are named HBD4 and HBD1. The HBD4 patch will be applied to the C4 carbon while the HBD1 will be applied to the C1 carbon. The patches will override the information contained in the main RESI entry. In the HBD4 patch, we can see that the C4, H41, and H42 atoms will be assigned new atom types. There is also a new atom, H43, defined to cap the terminal carbon. The last line of the patch defines the bond between the C4 atom and the new H43 atom. These patches were modeled after the C1 atom in the topology for 2-hexene included in section 3.2.1.

Patches Connecting Two Different Polymer Segments

To combine two different types of polymer segments, a unique patch must be created for that combination. In this section, we use the example of a patch created to combine a segment of PB with a segment of polyethyleneoxide (PEO). It is convenient to store patches of these type together in a separate topology file. The topology file can be appended as new polymers are added to your structure library. These patches start with the heading PRES followed by the unique patch name and charge. In this example, our patch is given the name PBEO and carries a charge of 0.17. An important difference to note in these patches is the number in front of the atom name. It is likely that two different residues will contain atoms with the same name. To ensure the patch works as intended, each residue is identified by a number. In this example, our PB monomer is denoted with a 1 while the PEO monomer is denoted with a 2. After the new atom types are specified, the next two lines instruct psfgen to delete a hydrogen atom from each of the residues to make space for the new bond. The new bond is defined in the next line followed by an internal coordinates table.

```
PRES PBEO    0.17

GROUP                          !    H41  H1A
ATOM 1C4  CG321 -0.18          !     |    |
ATOM 1H41 HGA2   0.09          !-C3--C4---C1-O1
ATOM 1H42 HGA2   0.09          !     |    |
GROUP                          !    H42  H1B
ATOM 2C1  CC32A -0.01          !
ATOM 2H1A HCA2A  0.09          !
ATOM 2H1B HCA2A  0.09          !

DELE ATOM 1H43
DELE ATOM 2H1C

BOND 1C4 2C1

IC 1C2  1C3  1C4  2C1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C3  1C4  2C1  2O1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C4  2C1  2O1  2C2   0.0000  0.0000  180.0000  0.0000  0.0000
```

3.2.3 Automated Polymer Construction

After we made the necessary changes to our topology files and created the proper patches, we used psfgen to create polymer chains of any desired length. In this section, we outline the procedure to use psfgen to create our polymers. A `tcl` script that will generate a trans-1,4-polybutadiene and polyethyleneoxide block copolymer can be found in appendix B.5. Examples contained in this section are excerpts from that psfgen script.

When creating our block copolymers, each polymer is first created separately. The following `tcl` script was used to generate PB polymer chains. A description of the contents of this script can be found in Appendix C.2.

```
#######################################
#generating PB section                #
#######################################

package require psfgen
topology  ./top_patch/top_all36_cgenff.rtf
topology  ./top_patch/top_all35_ethers.rtf
topology  ./top_patch/BDE.top

segment B {

for {set i 1} {$i <= $nPB} {incr i} {
    residue $i BDE
}
}
coordpdb ./monomers/newBUD_carbon.pdb B

regenerate angles dihedrals
guesscoord
writepdb polyPB.pdb
writepsf polyPB.psf
mol delete all
resetpsf
```

Once we created two different polymers of the desired length, we combined them into a single block copolymer. In the example `tcl` script below, psfgen is instructed to apply the patch PBEO to the

last residue of segment B with the first residue of segment E. Note that this example was taken from an automated script, `${resLast}` calls the stored variable `reslast`. For instance, if we connected a PB segment with 12 units to a PEO segment, `reslast` would be 12. Other scripts similar to this were used to connect all polymer chains. A description of the contents of this script can be found in Appendix C.2.

```
#######################################
#Combining the two chains             #
#######################################

package require psfgen
topology  ./top_patch/top_all36_cgenff.rtf
topology  ./top_patch/top_all35_ethers.rtf
topology  ./top_patch/BDE.top
topology  ./top_patch/patch.top

readpsf    polyPB.psf
coordpdb   polyPB.pdb

readpsf    polyPEO.psf
coordpdb   moved_polyPEO.pdb

patch PBEO B:${resLast} E:1
regenerate angles dihedrals
guesscoord

writepdb pb${resLast}peo${resFirst}.pdb
writepsf pb${resLast}peo${resFirst}.psf
```

3.3 Generating Force-Field Parameters

After we finished building our polymer, we generated any missing parameters for conducting simulations. This was done using the force-field tool kit (ffTK) [42] plugin in VMD [30] as well as the quantum chemistry software Gaussian [43]. In Sections 3.3.2 through 3.3.6, we outline details on the usage of these resources. A more detailed tutorial version of the process can be found in Appendix C.3.

3.3.1 The CHARMM Force-Field

The CHARMM [31] force field uses additive, pairwise potential functions to capture the overall potential energy of all atoms in a given system. This section provides a brief overview of the CHARMM force-field terms relevant to this chapter. A complete description of the CHARMM force-field is given in Section 2.2.2. All potential terms in the force-field can be separated into two main categories: bonded and non-bonded.

$$U = U_{bonded} + U_{non-bonded} \tag{3.1}$$

The bonded terms we are concerned with consist of bond, angle, and dihedral terms. Bonds and angles are represented as harmonic oscillators, with the overall force being proportional to a displacement from some equilibrium position or angle. The dihedral term is a bit different. It is meant to represent a potential energy landscape that is periodic in increments of 2π. More information on the dihedral term can be found in Section 3.3.6.

$$U_{bonded} = \sum_{bonds} k_b(b - b_o)^2 + \sum_{angles} k_\theta(\theta - \theta_o)^2 + \sum_{dihedrals} k_\phi[1 + \cos(n\phi - \delta)] \tag{3.2}$$

The non-bonded terms recreate electrostatic and van der Waal's type forces between non-bonded atoms. The electrostatic interactions are represented by a Coulombic potential where q represents the charge of each atom, ϵ_o is the permeativity of free space, and r_{ij} is the distance between centers of mass of two atoms. The van der Waal's forces are represented by a Lennard-Jones potential function. In this equation, ϵ_{ij} represents the energy well depth, σ_{ij} is the hard sphere radius, and r_{ij} is the distance between the two centers of masses of two atoms.

$$U_{non-bonded} = \sum_i \sum_{j>i} \epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \sum_i \sum_{j>i} \frac{q_i q_j}{4\pi\epsilon_o r_{ij}} \tag{3.3}$$

3.3.2 Force Field Toolkit (ffTK)

When generating parameters, the size of each structure was kept as small as possible while still retaining necessary atoms to find missing parameters. We found that dimers were sufficient for most polymers, however, there were a few exceptions. In Sections 3.3.2 through 3.3.6, we will walk through an example of generating the missing parameters for a PB and PMOXA dimer. Missing parameters were identified by using ffTK [42] to scan existing parameter files. For the PB dimer, we were only missing one dihedral angle (Figure D.2). All other bonds, angles and non-bonded parameters were defined in the CHARMM36 [31] force field. This was not typical of all dimers as most of them were missing bond and angle parameters. Non-bonded parameters will always be defined if atom types are taken from previously defined topology files.

Instructions on how to generate and optimize bond/angle parameters are included in Appendix C.3 and the ffTK tutorial [45] located at `www.ks.uiuc.edu/Training/Tutorials/`. Once we have identified missing parameters, we generated an output parameter file for our newly developed parameters.

3.3.3 Geometry Optimization

After identifying the missing parameters, we used Gaussian [43] to optimize the geometry of our structures. This is an important step as it corrects any unrealistic structures created by psfgen. If this step is skipped, any Gaussian [43] runs to calculate bonds/angles and dihedrals will likely fail. An example of the Gaussian [43] input file generated by ffTK [42] is included Appendix B.1. All necessary output information, including the optimized geometry, was written to a log file. The information in the log file was used by ffTK [42] to create new PDB files with optimized geometry. A geometry optimized PDB was used in all remaining steps.

3.3.4 Charge Optimization

Gaussian input files were generated to optimize atomic charges in each structure. An example of some of these files can be seen in Appendix B.2. These files were run in Gaussian [43], again directing the output to a log file. The log files were used by ffTK [42] to optimize the charge of each atom. Successive iterations of the optimization process were performed until there was a negligible difference between the "Prev. Charge" and "Final Charge" values. Once the charges had converged, a new PSF file containing the optimized charges was created. If the charges were significantly different than those in the initial PSF, the original topology file was modified with the new charges. There was a negligible change in the charges for all structures except for PMOXA.

3.3.5 Bond Angle Optimization

The required Gaussian [43] input file for each structure was generated by ffTK [42]. An example of one of the files is included in Appendix B.3. These files were run in Gaussian [43] and outputs were saved to log files. The output files were then used to optimize any missing bond/angle parameters through an iterative procedure described in Appendix C.3.3. The optimized parameters were then used to update the initial parameter files we created.

3.3.6 Dihedral Optimization

Gaussian [43] dihedral scan inputs for all missing dihedral parameters were generated using ffTK [42]. Dihedrals terminating with hydrogens were not scanned explicitly unless the hydrogens carried a charge other than +0.09. The following settings were used for all dihedral scans performed in Chapter 3: "Scan +/- (°)" was set to 180 and "Step Size (°)" was set to 10. These modifications were made at the suggestion of the ffTK tutorial [45]. Each dihedral angle resulted in two scan input files, one positive scan and one negative scan. Each of these files were run in Gaussian [43] and outputs were saved to log files. An example of a positive scan input is included in Appendix B.4.

Once Gaussian [43] completed all the necessary scans, the output files were used to optimize missing parameters through an iterative procedure described in Appendix C.3.4. When parameterizing a PMOXA dimer, there were 8 missing dihedral angle terms. The resulting energy landscape was fairly complex and therefore we developed a new strategy to optimize molecules that have a large number of missing dihedral parameters. The steps are outlined here as well as in Appendix B.4.

1. Before running the initial optimization, we duplicated every dihedral angle 4 times and changed the "Periodicity (n)" value in each of the three duplicates to 2, 3, 4, and 6. Now, every angle had 5 terms with different periodicity values. The first optimization gave a poor fit and had a root mean squared error (RMSE) of 5.241.

2. We used "Set As Refit Input" and "Run Refitting/Refinement" to perform successive iterations. The RMSE initially decreased by large amounts. The decreases in RMSE continued to become smaller and smaller. We continued performing iterations until there was little to no change in RMSE or the RMSE began to increase.

3. We then carefully went through the list of parameters and identified the smallest "Force Constant (k)" value. The term with the smallest "k" value was removed as this essentially weights the terms in overall contribution to the energy profile. When removing terms, we made sure to always keep at least one term for every dihedral. We continued to repeat steps 2 and 3 until the smallest "k" value was larger than 0.01.

4. We then changed the "Mode" to downhill and "Tol" to 0.0001 and ran the refitting optimization.

5. We deleted the smallest "k" value and ran the refitting/refinement in "Simulated Annealing" mode. The RMSE increased and we set this as the refit input and ran the refitting/refinement in "Downhill" mode. We then compared the final RMSE with the RMSE prior to deleting

the last term and refitting in "Simulated Annealing". This step was repeated as long as the RMSE continued to decrease.

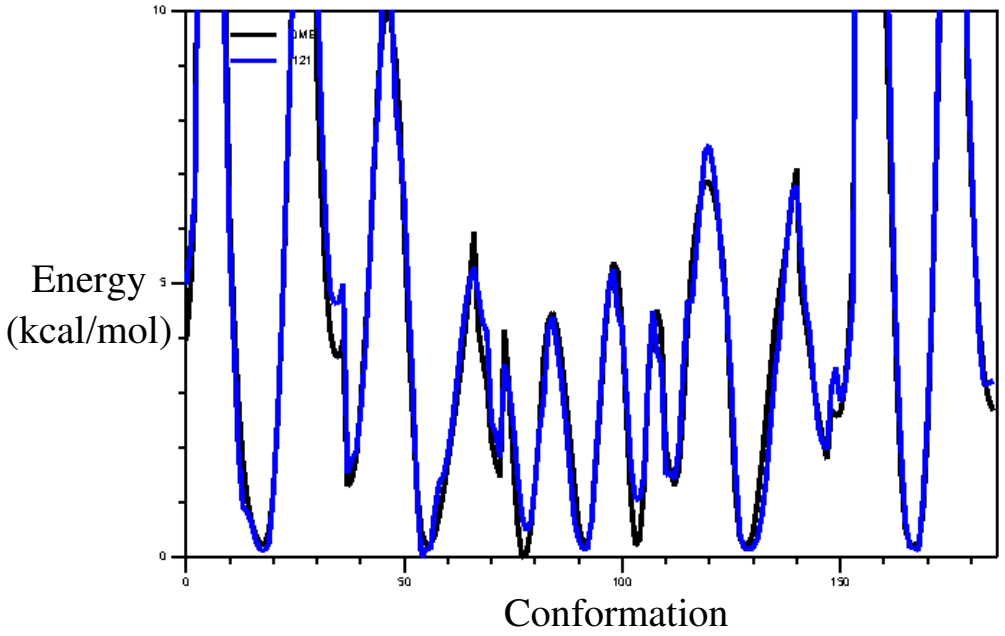


FIGURE 3.2: Energy profile of the missing poly-2-methyl-2-oxazoline dihedral angles (black) as well as the final (blue) fittings.

Using the steps outlined above, we were able to produce the energy profile (blue), shown in Figure 3.2, after approximately 60 iterations. This was time consuming but it took away a lot of guess work. The final fit has an RMSE of 0.524 which is higher than usual but acceptable given the complexity of the profile. This fit also closely reproduces all energetic minimas which is important. We were able to optimize all dihedral angles except three needed to simulate PB-PMOXA and PI-PMOXA (Table 3.4). We were unable to to reproduce a suitable fit for the energy profile using the above strategy. New strategies will need to be explored to obtain these parameters.

3.4 Individual Builds

Each of the structures in this section were built using the same method outlined in section 3.2. Sections 3.4.1 through 3.4.3 contain the resulting topology files and any supplemental information specific to each polymer build.

3.4.1 Trans-1,4-polybutadiene

A detailed description of the trans-1,4-polybutadiene build is outlined as an example in section 3.2.

```
*>>>>>> CHARMM topology file generated by Molefacture <<<<<<
27 1

MASS     1 CG321  12.01100  C
MASS     1 CG331  12.01100  C
MASS     1 CG2D1  12.01100  C
MASS     1 HGA2   1.00794  H
MASS     1 HGA4   1.00794  H
MASS     1 HGA3   1.00794  H

AUTO ANGLES DIHE
DEFAULT FIRST HBD1 LAST HBD4

RESI  BDE     0.00
GROUP
ATOM  C1  CG321  -0.18000 !   H11 H12
ATOM  C3  CG2D1  -0.15000 !     \ |
ATOM  H11 HGA2    0.09000 ! (C4)-C1      H31
ATOM  H12 HGA2    0.09000 !        \     /
ATOM  C2  CG2D1  -0.15000 !         C2=C3
ATOM  H21 HGA4    0.15000 !        /     \
ATOM  H31 HGA4    0.15000 !      H21      C4-(C1)
ATOM  C4  CG321  -0.18000 !               | \
ATOM  H41 HGA2    0.09000 !             H41 H42
ATOM  H42 HGA2    0.09000 !
```

```
BOND C1 H11  C1 H12  C1 C2
BOND C3 C2  C3 H31  C3 C4  C2 H21
BOND C4 H41  C4 H42
BOND C1 -C4

IC -C3 -C4 C1  C2     0.000  0.000  180.0  0.000  0.000
IC -C4  C1 C2  C3     0.000  0.000  180.0  0.000  0.000
IC  C1  C2 C3  C4     0.000  0.000  180.0  0.000  0.000
IC  C2  C3 C4 +C1     0.000  0.000  180.0  0.000  0.000

PRES HBD4            0.00  ! Complete terminal methyl at C4
                           !
ATOM H43   HGA3     0.09   !
ATOM C4    CG331   -0.27   !
ATOM H41   HGA3     0.09   !
ATOM H42   HGA3     0.09   !
BOND H43 C4

PRES HBD1            0.00  ! Complete terminal methyl at C1
                           !
ATOM H13   HGA3     0.09   !
ATOM C1    CG331   -0.27   !
ATOM H11   HGA3     0.09   !
ATOM H12   HGA3     0.09   !
BOND H13 C1

END
```

3.4.2 Trans-1,4-polyisoprene

A single monomer of trans-1,4-polyisoprene was constructed in molefacture, atom types were modeled after 2-hexene.

```
*>>>>>> CHARMM topology file generated by Molefacture <<<<<<
27 1

MASS     1 HGA3   1.00794  H
MASS     1 CG331  12.01100  C
MASS     1 CG2D1  12.01100  C
MASS     1 HGA4   1.00794  H
MASS     1 HGA2   1.00794  H
MASS     1 CG321  12.01100  C

AUTO ANGLES DIHE
DEFAULT FIRST HSP1 LAST HSP3

RESI  ISP     0.008
GROUP

ATOM  H12 HGA2    0.09000        !           H31 H32
ATOM  H13 HGA2    0.09000        !             \ /
ATOM   C3 CG321  -0.18000        !     H21      C3-(C1)
ATOM  H31 HGA2    0.09000        !       \     /
ATOM   C5 CG2D1   0.00800        !        C2=C5
ATOM   C2 CG2D1  -0.03900        !       /     \
ATOM  H21 HGA4    0.03900        ! (C3)-C1      C4-H43
ATOM   C1 CG321  -0.18000        !     / \     / \
ATOM   C4 CG331  -0.27000        !   H13 H12 H41 H42
ATOM  H41 HGA3    0.09000        !
ATOM  H42 HGA3    0.09000        !
ATOM  H43 HGA3    0.09000        !
ATOM  H32 HGA2    0.09000        !

BOND C1 -C3  H12 C1
BOND C3 H31  C3 C5  C3 H32  C5 C2
BOND C5 C4  C2 H21  C2 C1  C1 H13
BOND C4 H41  C4 H42  C4 H43
```

```
IC -C3 C1  C2  C5     0.000  0.000  180.0  0.000  0.000
IC  C1 C2  C5  C3     0.000  0.000  180.0  0.000  0.000
IC  C1 C2  C5  C4     0.000  0.000  000.0  0.000  0.000
IC  C2 C5  C3 +C1     0.000  0.000  180.0  0.000  0.000
IC  C5 C3 +C1 +C2     0.000  0.000  180.0  0.000  0.000

PRES HSP3            0.00  ! Complete terminal methyl at C3
                           !
ATOM H33   HGA3     0.09   !
ATOM C3    CG331   -0.27   !
ATOM H31   HGA3     0.09   !
ATOM H32   HGA3     0.09   !
BOND H33 C3

PRES HSP1            0.00  ! Complete terminal methyl at C1
                           !
ATOM H11   HGA3     0.09   !
ATOM C1    CG331   -0.27   !
ATOM H12   HGA3     0.09   !
ATOM H13   HGA3     0.09   !
BOND H11 C1

END
```

3.4.3 Poly-2-methyl-2-oxazoline

A single monomer of the PMOXA chain was constructed in molefacture. The ketone group attached to the nitrogen was named using acetamide. The necessary parameters to simulate PMOXA linked with PB and PI blocks could not be optimized.

```
*>>>>>> CHARMM topology file generated by Molefacture <<<<<<
27 1

MASS     1 OG2D1  15.99940  O
MASS     1 CG201  12.01100  C
MASS     1 CG331  12.01100  C
MASS     1 NG2S1  14.00674  N
MASS     1 HGA3   1.00794  H
MASS     1 HGA2   1.00794  H
MASS     1 CG321  12.01100  C

AUTO ANGLES DIHE
DEFAULT FIRST HSP3 LAST HSP4

RESI  PMX    -0.15
GROUP
ATOM    O OG2D1  -0.61300        !  H32  H31 H41 H43
ATOM   C1 CG201   0.66700        !     \ /    \ /
ATOM   C2 CG331  -0.25800        ! (C4)-C3     C4-(C3)
ATOM    N NG2S1  -0.24400        !        \   /
ATOM  H21 HGA3    0.09000        !          N    H21
ATOM  H22 HGA3    0.09000        !           \   /
ATOM  H23 HGA3    0.09000        !           C1-C2-H22
ATOM   C3 CG321  -0.18000        !          //   \
ATOM   C4 CG321  -0.18000        !          O    H23
ATOM  H41 HGA2    0.09000        !
ATOM  H43 HGA2    0.09000        !
ATOM  H32 HGA2    0.09000        !
ATOM  H31 HGA2    0.09000        !

BOND O C1  C1 C2  C1 N
BOND C2 H21  C2 H22  C2 H23  N C3
BOND N C4  C3 H32  C3 H31
BOND C4 H41  C4 H43  C3 -C4
```

```
IC -C4 C3   N  C1     0.000  0.000  000.0  0.000  0.000
IC -C4 C3   N  C4     0.000  0.000  180.0  0.000  0.000
IC  C3  N  C1   O     0.000  0.000  000.0  0.000  0.000
IC  C3  N  C1  C2     0.000  0.000  180.0  0.000  0.000
IC  C3  N  C4 +C3     0.000  0.000  180.0  0.000  0.000
IC  C4  N  C1   O     0.000  0.000  180.0  0.000  0.000
IC  C4  N  C1  C2     0.000  0.000  180.0  0.000  0.000
IC  C1  N  C3 -C4     0.000  0.000  180.0  0.000  0.000
IC  C1  N  C4 +C3     0.000  0.000  180.0  0.000  0.000
IC   N C3 -C4 -C3     0.000  0.000  180.0  0.000  0.000

PRES HSP3            0.00  ! Complete terminal methyl at C3
                           !
ATOM H33   HGA3     0.09   !
ATOM C3    CG331   -0.181  !
ATOM H31   HGA3     0.09   !
ATOM H32   HGA3     0.09   !
BOND H33 C3

PRES HSP4            0.00  ! Complete terminal methyl at C4
                           !
ATOM H43   HGA3     0.09   !
ATOM C4    CG331   -0.181  !
ATOM H41   HGA3     0.09   !
ATOM H42   HGA3     0.09   !
BOND H42 C4

END
```

3.4.4 Patches For Block Copolymers

```
PRES PBEO    0.17

GROUP                        !    H41  H1A
ATOM 1C4  CG321 -0.18        !     |    |
ATOM 1H41 HGA2   0.09        !-C3--C4---C1-O1
ATOM 1H42 HGA2   0.09        !     |    |
GROUP                        !    H42  H1B
ATOM 2C1  CC32A -0.01        !
ATOM 2H1A HCA2A  0.09        !
ATOM 2H1B HCA2A  0.09        !

DELE ATOM 1H43
DELE ATOM 2H1C

BOND 1C4 2C1

IC 1C2  1C3  1C4  2C1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C3  1C4  2C1  2O1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C4  2C1  2O1  2C2   0.0000  0.0000  180.0000  0.0000  0.0000

PRES PBOX    0.00

GROUP                        !    H41  H32
ATOM 1C4  CG321 -0.18        !     |    |
ATOM 1H41 HGA2   0.09        !-C3--C4---C3-N
ATOM 1H42 HGA2   0.09        !     |    |
GROUP                        !    H42  H31
ATOM 2C3  CG321 -0.18        !
ATOM 2H32 HGA2   0.09        !
ATOM 2H31 HGA2   0.09        !

DELE ATOM 1H43
DELE ATOM 2H33

BOND 1C4 2C3

IC 1C2  1C3  1C4  2C3   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C3  1C4  2C3  2N    0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C4  2C3  2N   2C4   0.0000  0.0000  180.0000  0.0000  0.0000
```

```
PRES PIEO    0.17

GROUP                        !    H31  H1A
ATOM 1C3  CG321 -0.18        !     |    |
ATOM 1H31 HGA2   0.09        !-C5--C3---C1-O1
ATOM 1H32 HGA2   0.09        !     |    |
GROUP                        !    H32  H1B
ATOM 2C1  CC32A -0.01        !
ATOM 2H1A HCA2A  0.09        !
ATOM 2H1B HCA2A  0.09        !

DELE ATOM 1H33
DELE ATOM 2H1C

BOND 1C3 2C1

IC 1C2  1C5  1C3  2C1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C5  1C3  2C1  2O1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C3  2C1  2O1  2C2   0.0000  0.0000  180.0000  0.0000  0.0000

PRES PIOX    0.00

GROUP                        !    H41  H32
ATOM 1C3  CG321 -0.18        !     |    |
ATOM 1H31 HGA2   0.09        !-C5--C3---C3-N
ATOM 1H32 HGA2   0.09        !     |    |
GROUP                        !    H42  H31
ATOM 2C3  CG321 -0.18        !
ATOM 2H32 HGA2   0.09        !
ATOM 2H31 HGA2   0.09        !

DELE ATOM 1H33
DELE ATOM 2H33

BOND 1C3 2C3

IC 1C2  1C5  1C3  2C3   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C5  1C3  2C3   2N   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C3  2C3   2N  2C4   0.0000  0.0000  180.0000  0.0000  0.0000

END
```

3.5 Parameters and Conclusions

All parameters in Tables 3.1-3.4 were generated using the methods described in Section 3.3. All the parameters were obtained by parameterizing the following molecules: PB dimer, PB_1PEO_1, PB_1PEO_2, PI monomer, PI_2PEO_1, PMOXA monomer, PMOXA dimer, and PB_1PMOXA_2. Tables 3.1-3.4 contain all parameters needed to perform simulations of these molecules with the exception of three dihedral angles involved in cross-linking PMOXA to PB and PI. We were unable to optimize these parameters using ffTK [42] and they are denoted with an (*) in Table 3.4.

3.5.1 Summary and Conclusions

- Complete knowledge of the contents and purpose of PDB, PSF, topology, and parameter files is necessary for successful construction and parameterization of novel molecular structures. These files will require manual editing and inspection.
- All current CHARMM topology files should be scanned to find similar structures for assigning atom types (Section 3.2.1). This will reduce the number of new parameters needed.
- When obtaining new parameters, use multiple small and simple molecular structures rather than one large and complex structure.

3.5.2 Parameters

TABLE 3.1: Bonded potential parameters generated with ffTK [42] and Gaussian [43] software. Parameters are organized by atom type which can be cross-referenced in Figure 3.3

Bonds	Atom Types	k_b $\left[\frac{kcal}{mol \cdot \mathring{A}^2}\right]$	b_o [$\mathring{A}$]
$U_{bond} = k_b(b - b_o)^2$	CG321-CC32A	282.912	1.511
	OG2D1-CG201	774.515	1.240
	CG201-CG331	309.030	1.528
	CG201-NG2S1	436.936	1.363

TABLE 3.2: Bond angle potential parameters generated with ffTK [42] and Gaussian [43] software. Parameters are organized by atom type which can be cross-referenced in Figure 3.3

Angles	Atom Types	k_θ $\left[\frac{kcal}{mol \cdot rad^2}\right]$	θ_o $[^o]$
$U_{angle} = k_\theta(\theta - \theta_o)^2$	CG2D1-CG321-CC32A	40.751	110.091
	CG321-CC32A-HCA2A	53.619	109.514
	CG321-CC32A-OC30A	91.237	109.996
	HGA2-CG321-CC32A	49.856	112.011
	CG331-CG2D1-CG331	61.669	119.294
	OG2D1-CG201-NG2S1	110.295	122.580
	OG2D1-CG201-CG331	95.429	122.288
	CG201-NG2S1-CG331	80.188	120.362
	CG201-CG331-HGA3	53.234	110.899
	CG331-CG201-NG2S1	103.417	116.891
	CG331-NG2S1-CG331	56.119	115.717
	CG201-NG2S1-CG321	298.798	119.261
	CG331-NG2S1-CG321	277.040	118.049
	CG321-NG2S1-CG321	0.095	115.845

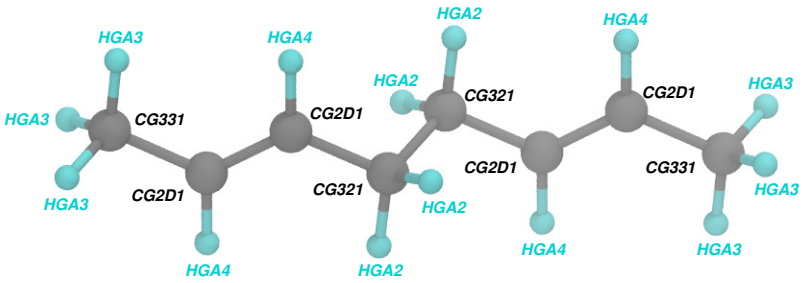
TABLE 3.3: Set 1 of 2 dihedral potential parameters generated with ffTK [42] and Gaussian [43] software. Parameters are organized by atom type which can be cross-referenced in Figure 3.3

Dihedrals	Atom Types	k_ϕ $\left[\frac{kcal}{mol}\right]$	n	δ [^{o}]
$U_{dihedral} = k_\phi[1 + \cos(n\phi - \delta)]$	HGA2-CG321-CC32A-OC30A	0.4470	1	180.00
	HGA2-CG321-CC32A-OC30A	0.5270	3	180.00
	HGA2-CG321-CC32A-OC30A	0.9900	4	180.00
	HGA2-CG321-CC32A-OC30A	0.6180	6	180.00
	CG2D1-CG321-CC32A-HCA2A	1.2000	3	0.00
	CG2D1-CG321-CC32A-HCA2A	0.9490	4	0.00
	CG2D1-CG321-CC32A-HCA2A	0.2810	6	0.00
	HGA4-CG2D1-CG321-CC32A	0.5860	1	0.00
	HGA4-CG2D1-CG321-CC32A	0.4330	3	0.00
	HGA4-CG2D1-CG321-CC32A	0.1010	4	180.00
	HGA2-CG321-CC32A-HCA2A	1.3830	1	180.00
	HGA2-CG321-CC32A-HCA2A	0.3010	2	0.00
	HGA2-CG321-CC32A-HCA2A	0.1830	6	0.00
	CG2D1-CG2D1-CG321-CC32A	0.2480	1	0.00
	CG2D1-CG2D1-CG321-CC32A	0.2090	2	0.00
	CG2D1-CG2D1-CG321-CC32A	0.1030	4	0.00
	CG321-CC32A-OC30A-CC33A	0.0840	1	180.00
	CG321-CC32A-OC30A-CC33A	0.0630	2	0.00
	CG321-CC32A-OC30A-CC33A	0.5800	3	0.00
	CG2D1-CG321-CC32A-OC30A	0.4670	1	180.00
	CG2D1-CG321-CC32A-OC30A	0.4510	2	180.00
	CG321-CC32A-OC30A-CC32A	0.0690	1	0.00
	CG321-CC32A-OC30A-CC32A	0.4530	3	0.00
	HGA3-CG331-CG2D1-CG331	0.1500	1	0.00
	HGA3-CG331-CG2D1-CG331	0.1500	3	0.00
	CG2D1-CG321-CG321-CG2D1	0.8830	1	180.00
	CG2D1-CG321-CG321-CG2D1	0.1170	2	180.00
	CG2D1-CG321-CG321-CG2D1	0.3180	3	180.00
	CG2D1-CG321-CG321-CG2D1	0.2260	4	0.00

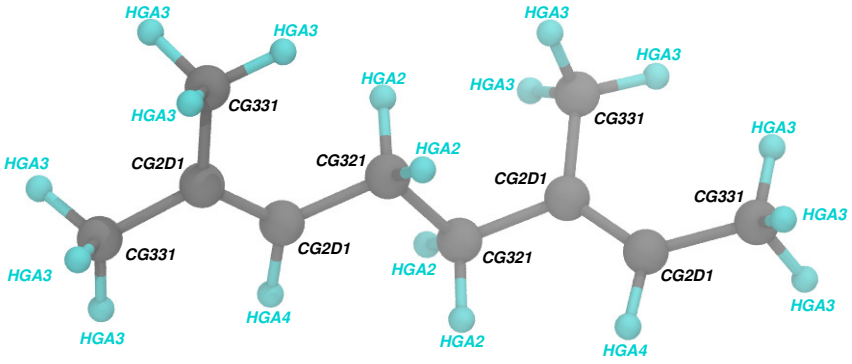
TABLE 3.4: Set 2 of 2 dihedral potential parameters generated with ffTK [42] and Gaussian [43] software. Parameters are organized by atom type which can be cross-referenced in Figure 3.3. *Parameters that could not be optimized.

Dihedrals	Atom Types	k_ϕ $\left[\frac{kcal}{mol}\right]$	n	δ $[^o]$
$U_{dihedral} = k_\phi[1 + \cos(n\phi - \delta)]$	NG2S1-CG201-CG331-HGA3	2.3500	1	180.00
	CG331-NG2S1-CG331-HGA3	2.2230	1	0.00
	CG201-NG2S1-CG331-HGA3	3.0000	1	0.00
	OG2D1-CG201-NG2S1-CG331	2.7310	1	180.00
	OG2D1-CG201-NG2S1-CG331	0.4680	2	180.00
	OG2D1-CG201-CG331-HGA3	2.9860	1	0.00
	CG331-CG201-NG2S1-CG331	3.0000	1	0.00
	CG331-CG201-NG2S1-CG331	2.5730	2	180.00
	CC32A-CG321-CG2D1-CG331	0.5550	1	180.00
	CG331-NG2S1-CG321-CG321	0.3390	4	0.00
	CG201-NG2S1-CG321-CG321	0.4810	2	0.00
	CG201-NG2S1-CG321-CG321	0.7830	3	0.00
	CG201-NG2S1-CG321-CG321	0.9230	4	0.00
	OG2D1-CG201-NG2S1-CG321	3.0000	1	180.00
	OG2D1-CG201-NG2S1-CG321	1.1130	6	180.00
	CG331-CG201-NG2S1-CG321	3.0000	1	0.00
	CG331-CG201-NG2S1-CG321	2.8190	2	180.00
	CG331-CG201-NG2S1-CG321	0.8870	3	0.00
	CG331-CG201-NG2S1-CG321	0.8210	4	180.00
	CG331-CG201-NG2S1-CG321	1.6890	6	0.00
	NG2S1-CG321-CG321-NG2S1	0.9800	1	180.00
	CG331-NG2S1-CG321-HGA2	0.2460	1	180.00
	CG331-NG2S1-CG321-HGA2	0.3100	3	0.00
	CG201-NG2S1-CG321-HGA2	0.6300	3	180.00
	CG201-NG2S1-CG321-HGA2	0.5030	4	0.00
	CG321-NG2S1-CG331-HGA3	2.9990	4	0.00
	CG321-NG2S1-CG331-HGA3	0.2290	6	180.00
	CG2D1-CG321-CG321-NG2S1*	-	-	-
	CG321-NG2S1-CG321-HGA2*	-	-	-
	CG321-NG2S1-CG321-CG321*	-	-	-

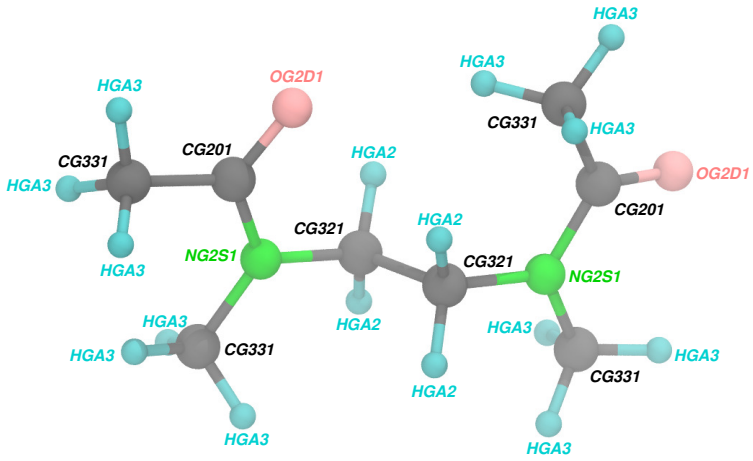
Polybutadiene



Polyisoprene



Polymethyloxazoline



Polyethyleneoxide

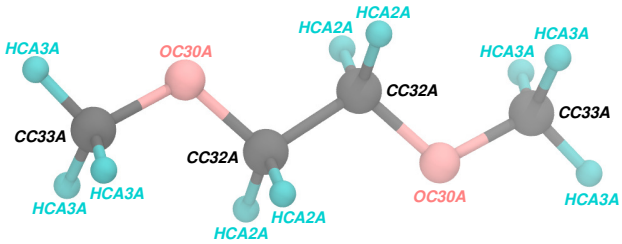


FIGURE 3.3: Dimers of the different polymers constructed and/or parameterized. Corresponding atom types are included and can be cross referenced with Tables 3.1 through 3.4.

Chapter 4

Simulations of Biomimetic Membranes

A significant amount of research done in the chemical engineering field is devoted to improving the efficiency and selectivity in separation processes [2, 5, 10, 11]. Nature provides examples of efficient systems that have been fine-tuned through many years of evolution [8, 9]. Cell membranes spontaneously form and incorporate membrane proteins in aqueous environments. Some of these membrane proteins are water-transport channels as they possess selectivity for water over other molecules. Aquaporins allow for the selective transport of water at a rate of $\sim 10^9$ molecules per second [3] while inserted in lipid bilayers. Though these biological systems can be replicated in the laboratory, they are not practical on an industrial scale. They lack the robustness and chemical stability necessary for applications in process engineering.

Experimental chemistry groups are currently exploring the use of block copolymers and synthetic carbon nanotube like structures as biomimetic membrane transport systems [3, 4, 15–17, 20, 22]. The peptide-appended pillar[5]arene (PAP) channel has been shown to insert into lipid bilayers and transport water at a rate of $\sim 10^8$ water molecules per second, approaching that of aquaporin [3]. Block copolymers can be used to build membranes with many advantages over lipid membranes, such as higher mechanical and chemical stability, lower gas and water permeability, and more customizable properties [11]. Biomimetic membranes such as these have shown promise in applications including water desalination, liquid and gas separations, drug delivery and screening, DNA recognition, and sensors [3, 13, 24, 25].

We have explored the conformational dynamics of the PAP channel in lipid and block copolymer membranes through the use of molecular dynamics (MD) simulations. In Section 4.1, we provide information about how we set up and performed these simulations. This is followed by Section 4.2 where we provide a breakdown of analysis results and a brief discussion of their interpretation. Our results suggest that the membrane environment can affect the channel dynamics and potentially its diffusive as well as transport characteristics.

4.1 System Preparation

Block copolymers built in Chapter 3, specifically PB-PEO and PI-PEO, were used to build pristine membrane structures. These structures were then equilibrated through the use of molecular dynamics (MD) simulations. In addition, we performed several independent long time-scale MD simulations of the PAP channel in three different membrane environments: POPC, $PB_{12}PEO_9$, and $PB_{23}PEO_{16}$. In most of these simulations, the PAP channel was embedded in the membrane, but we also carried out simulations where PAP was placed outside the membrane to observe spontaneous insertion. A description of each simulation set-up is contained in this section.

4.1.1 Pristine Membrane Construction and Equilibration

PB-PEO and PI-PEO Membranes

Block copolymer membranes were generated by first constructing a single polymer chain of desired length for PB and PEO blocks. The chains for each block were then connected using a patch as described in Section 3.2.2. The parameters for the resulting block copolymer were optimized as described in Section 3.3.2. The single PB-PEO chains were then replicated and placed in a regular 10 $\times$ 10 grid pattern over an area of 80 Å $\times$ 80 Å ($PB_{12}PEO_9$ and $PI_{12}PEO_9$) and a 9 $\times$ 9 grid pattern over an area of 80 Å $\times$ 80 Å ($PB_{23}PEO_{16}$ and $PI_{23}PEO_{16}$). All BCP membranes were arranged in a di-block configuration prior to equilibration (Figure 4.2). Pristine $PB_{12}PEO_9/PI_{12}PEO_9$ and $PB_{23}PEO_{16}/PI_{23}PEO_{16}$ membranes contained a total of 200 and 162 polymer chains respectively.

All BCP membranes were solvated using the autosolvate plugin in VMD [30]. Water molecules placed within the PB or PI layers were removed, while those that were placed in the PEO layer were retained. We conducted equilibration simulations of PB-PEO BCP membranes in three steps. In step 1, membranes were equilibrated in the NVT ensemble for 0.25 ns using a 1 fs time-step after minimizing the system for 4000 steps. All atoms were fixed during the first phase of simulation except for the hydrophobic chains. In step 2, the time-step was increased to 2 fs and the system was restarted after a brief minimization in the NPT ensemble using a constant area option for 0.50 ns. During the second step, external forces were applied using a `tcl`-script to keep water out of the hydrophobic layer. In step 3, all constraints on water molecules were removed and the system was restarted for a long timescale MD simulations. Trajectories were recorded every 20 ps. Simulations of PI-PEO membranes were carried out using the same method with one extra step. During the third stage of simulation, both PI-PEO membranes lost their planar geometry and began to bulge (Figure 4.1B). The PI chains contain an extra methyl group on every monomer (when compared to PB) and were unstable when constrained to the same area/BCP values as PB. To fix this, we stopped these simulations and restarted them using a constant ratio between the sides of the periodic cell. This allowed the membrane area to grow and the bulging to relax (Figure 4.1B). The membranes were equilibrated for 100 ns once the area had stabilized.

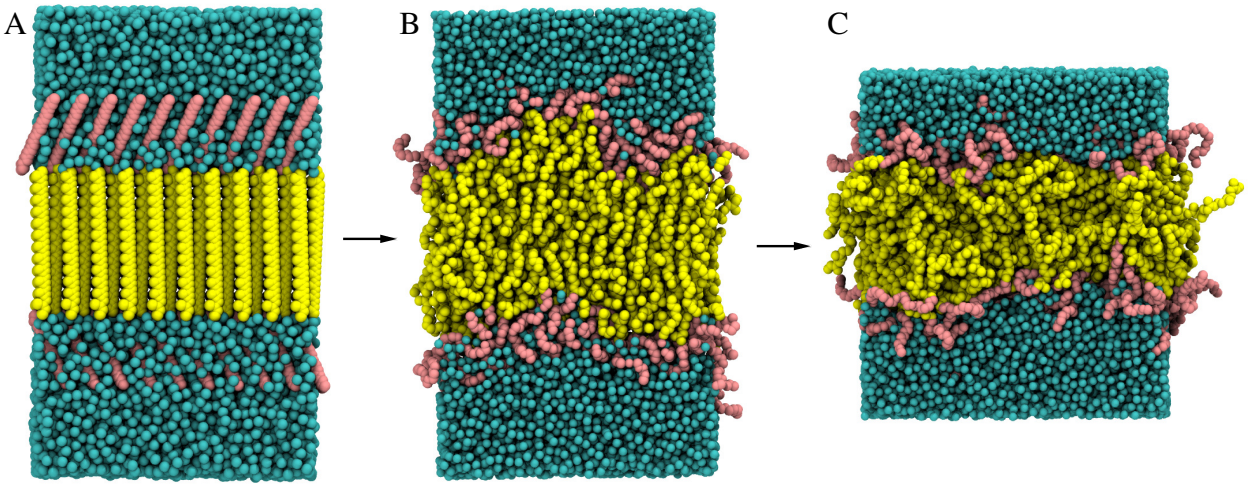


FIGURE 4.1: Snapshots of initial (A), bulging (B), and stable (C) membrane structures of $PI_{12}PEO_9$ prior to equilibration. PI chains are shown in yellow, PEO chains are shown in pink, and water is shown in cyan. All hydrogen atoms have been omitted.

POPC/PB-PEO Hybrid Membrane

To create the hybrid membrane, two separate pdb/psf pairs were combined. We first generated an 80 Å × 80 Å POPC membrane using the membrane builder plugin in VMD [30]. This membrane was merged with the equilibrated $PB_{12}PEO_9$ membrane structure obtained from pristine simulations. Both membranes were centered at the origin. The POPC membrane was displaced by 30 Å in the positive z-direction to create a space between PB and POPC chains. POPC and BCP chains from the opposite layers were identified and all other atoms were removed from the system. This procedure was carried out using a script called `merge.tcl` which can be found in Appendix B.6. The resulting structure can be seen in Figure 4.2. This structure was then deployed in a long scale MD simulation with the same settings as step 3 in the PB-PEO pristine membrane simulations. Due to the initial gap between the membrane layers, the periodic cell experienced large changes in the vertical direction. The system would become too small for the particle mesh Ewald (PME) grid and crash (This grid is used to calculate the electrostatic forces, more info on the PME grid can be found here [33]). The system was restarted with a new PME grid size in the z direction three times

before the simulation was stable. This method was necessary to avoid starting the simulation with overlapping structures as they could result in unphysical behavior or unstable simulations.

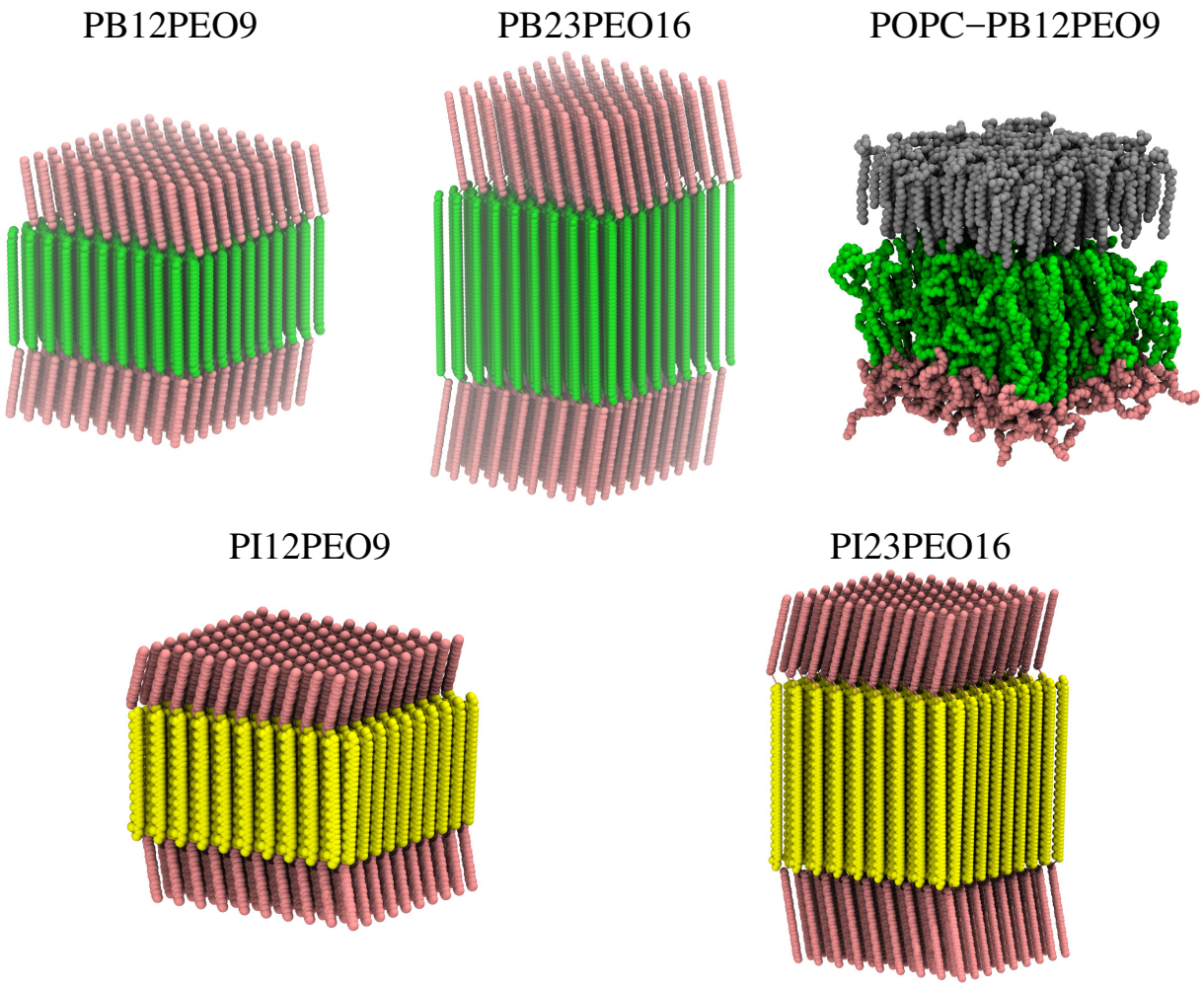


FIGURE 4.2: Snapshots of the initial $PB_{12}PEO_9$, $PB_{23}PEO_{16}$, POPC-$PB_{12}PEO_9$, $PI_{12}PEO_9$, and $PI_{23}PEO_{16}$ membrane structures prior to equilibration. PB chains are shown in green, PI chains are shown in yellow, PEO chains are shown in pink, and POPC chains are shown in gray. All hydrogen atoms have been omitted.

4.1.2 PAP-Embedded in POPC and BCP Membranes

We used the membrane builder plugin in VMD [30] to generate the POPC membrane. The POPC membrane used in all PAP/POPC simulations was 70 Å × 70 Å in size. The POPC membrane was

first aligned by moving its center of mass to the origin. The PAP channel was then added to the system and embedded into the membrane by aligning its center of mass to the origin as well. Any POPC molecules that were within 1.5 Å of the PAP channel were removed. After removing the unwanted lipids, the system was solvated using VMD's autosolvate plugin. Any water molecules placed within the lipid bilayer were removed from the system.

Simulations with PAP embedded in POPC membranes were carried out in three sequential steps. In the first step all atoms were fixed except those in the lipid tails. The system was then minimized and equilibrated in the NVT ensemble for 0.25 ns using a 1 fs time step. The restart files from step 1 were then used to initiate step 2. During step 2 the system was equilibrated in the NPT ensemble for 0.5 ns using a 2 fs time-step. Previous constraints placed on lipid and water molecules were released. During this simulation, the PAP channel was again constrained and a `tcl`-forces script was used to keep water out of the lipid membrane. In step 3, we released all constraints and carried out long timescale equilibration simulations in the NPT ensemble using a 2 fs time-step. A more detailed description of this procedure can be found in section 2.3. We carried out 4 independent trajectories of varying lengths in POPC membranes. We recorded trajectory data every 20 ps.

For PAP embedded in $PB_{12}PEO_9$ and $PB_{23}PEO_{16}$ membranes, we chose equilibrated configurations from pristine membrane simulations (See Sec. 4.1.1) for inserting PAP. A similar procedure to the POPC membrane was followed for the alignment of PAP in these membranes, where unwanted polymer chains were deleted. We carried out three long-time scale and PAP-embedded MD simulations for each BCP membrane.

4.1.3 PAP-Displaced Away From Lipid and BCP Membranes

To observe spontaneous insertion of a single PAP channel placed outside the membrane, we conducted two independent simulations each for the POPC/PAP system and the $PB_{12}PEO_9$/PAP system. The PAP channel was rotated by 90° such that its orientation was parallel to the membrane surface and then it was displaced by 45 Å and 55 Å from the origin in a direction normal to

the surface of the POPC and $PB_{12}PEO_9$ membranes respectively. We then followed the three step procedure described above and conducted long timescale MD simulations. We recorded trajectory data every 20 ps.

4.2 Results

We employed several techniques to analyze MD simulations described in Section 4.1. Membrane thickness and area per lipid/BCP were used to compare simulations with experimental observations/measurements. These provide a benchmark for physically realistic membrane behavior in our simulations. The diffusivity of PAP in all membranes was used to quantify mobility, and permeability measurements were taken to quantify performance. Measurements of the RMSD and orientation (angle) of the channel were used to quantify structural changes of PAP in different membrane environments.

4.2.1 Pristine Membrane Equilibration

Equilibration simulations of pure $PB_{12}PEO_9$ and $PB_{23}PEO_{16}$ membranes were 100.64 ns and 191.86 ns, respectively (Table A.1). The thickness of the hydrophobic layer for hybrid POPC-$PB_{12}PEO_9$, pure $PB_{12}PEO_9$, and $PB_{23}PEO_{16}$ membranes on equilibration were 35 ($\pm$1) Å, 41 ($\pm$1) Å and 51 ($\pm$1) Å, respectively (Figure 4.3). We used these equilibrated configurations of BCP membranes for simulations with the PAP channel. The simulations of pure $PI_{12}PEO_9$ and $PI_{23}PEO_{16}$ were each 100 ns and had area/BCP values of 0.88 ($\pm$0.04) $\frac{nm^2}{BCP}$ and 1.35 ($\pm$0.06) $\frac{nm^2}{BCP}$, respectively. The hydrophobic thickness of the $PI_{12}PEO_9$ and $PI_{23}PEO_{16}$ mebranes was 36 ($\pm$1) Å and 48 ($\pm$1) Å, respectively (Figure 4.4) All pristine membrane structures after 100 ns of equilibration can be seen in Figures 4.5 and 4.6.

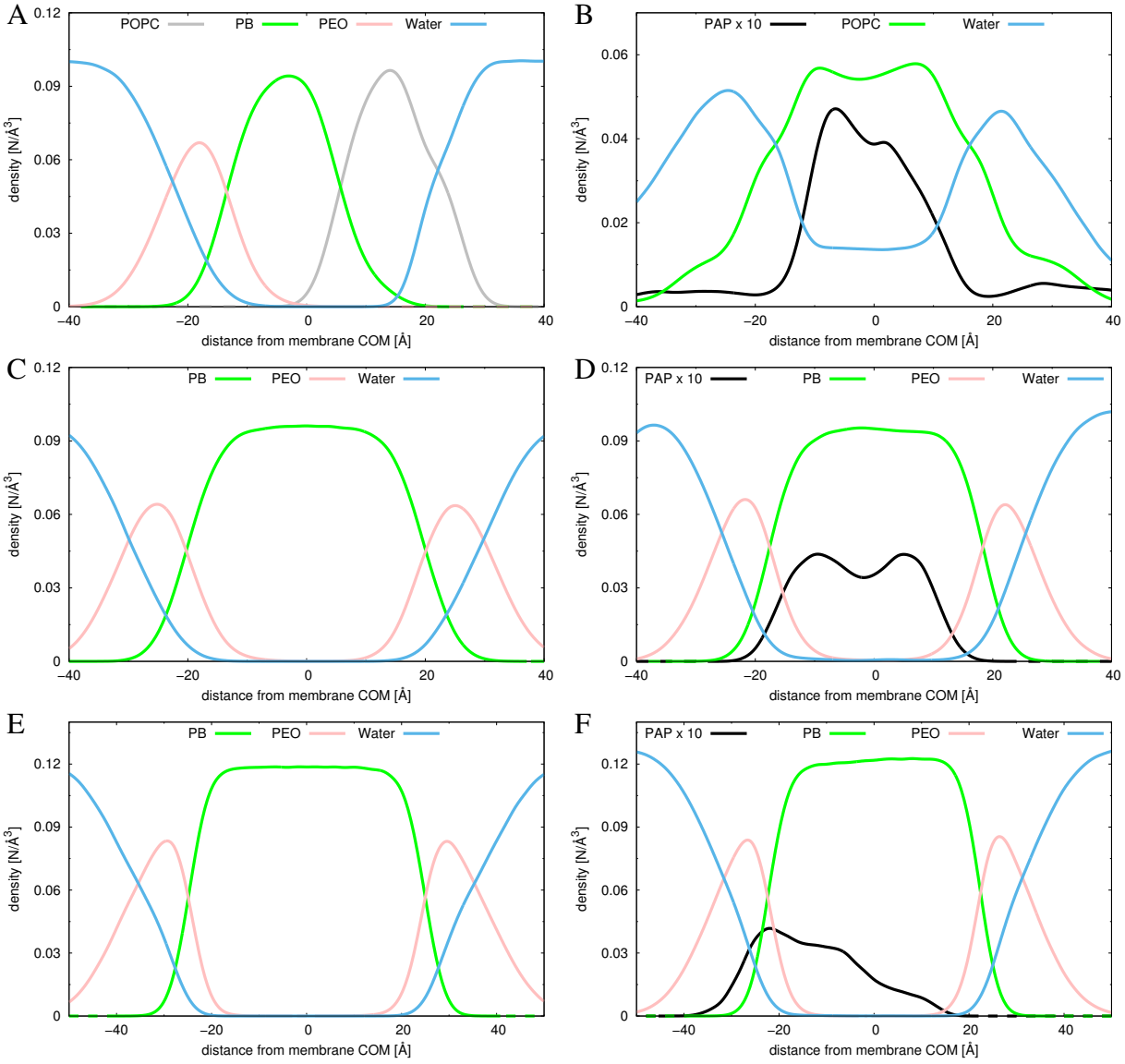


FIGURE 4.3: Average density (atoms/Å^3) by species vs. distance from membrane center of mass (COM) from pristine membrane simulations [POPC-$PB_{12}PEO_9$ (A), $PB_{12}PEO_9$ (C), and $PB_{23}PEO_{16}$ (E)] and PAP embedded simulations[POPC (B), $PB_{12}PEO_9$ (D), and $PB_{23}PEO_{16}$ (F)].

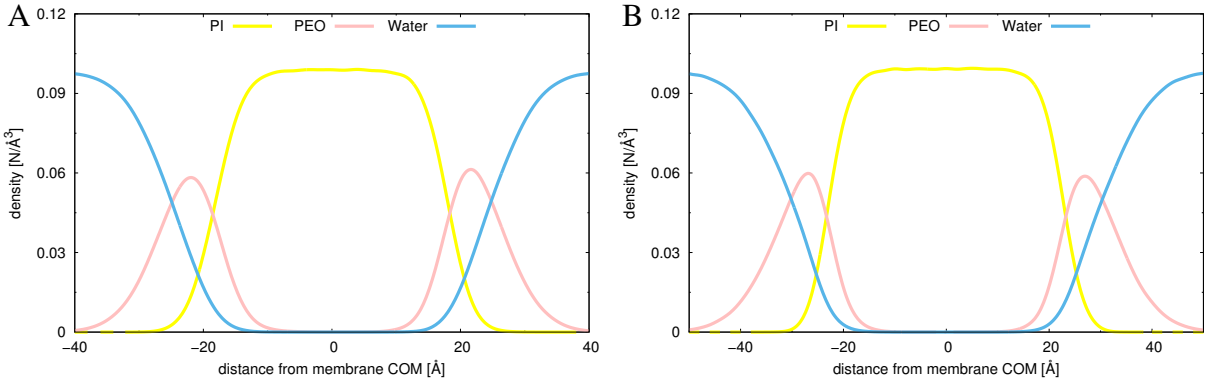


FIGURE 4.4: Average density (atoms/Å^3) by species vs. distance from membrane center of mass (COM) for pristine $PI_{12}PEO_9$ (A) and $PI_{23}PEO_{16}$ (B) membrane simulations.

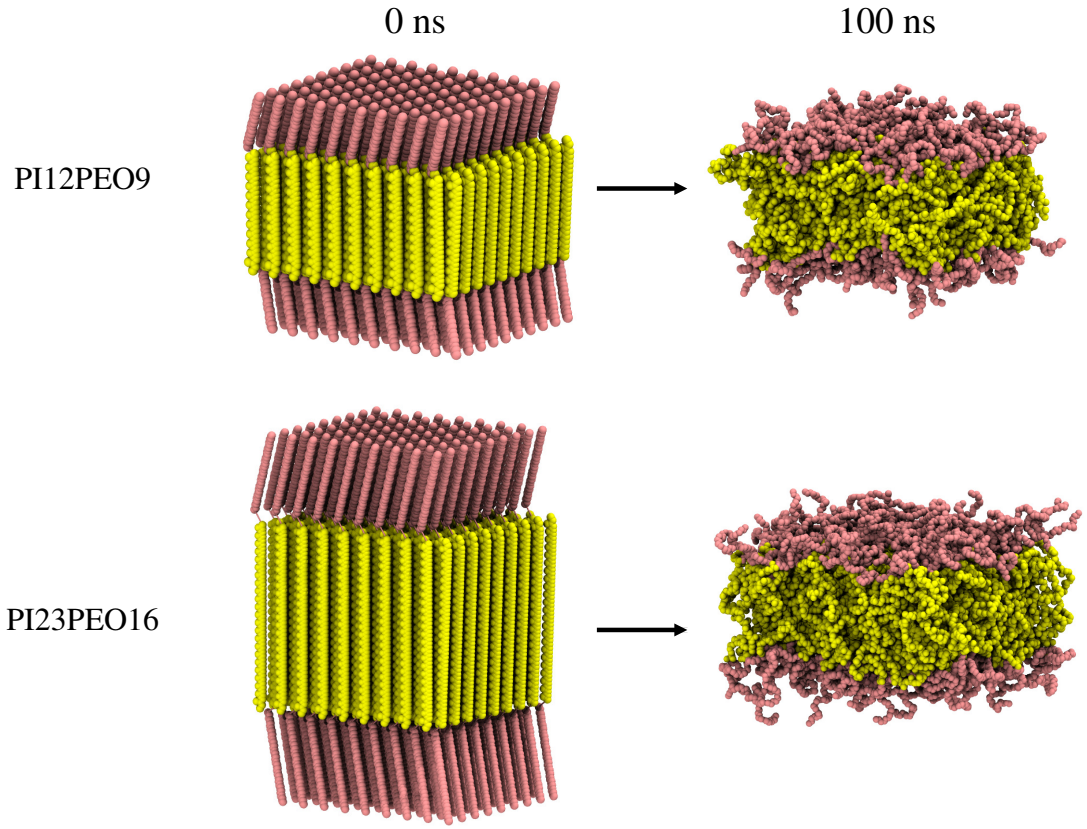


FIGURE 4.5: Snapshots of initial (0 ns) and final (100 ns) membrane ($PI_{12}PEO_9$ and $PI_{23}PEO_{16}$) structures during equilibration.

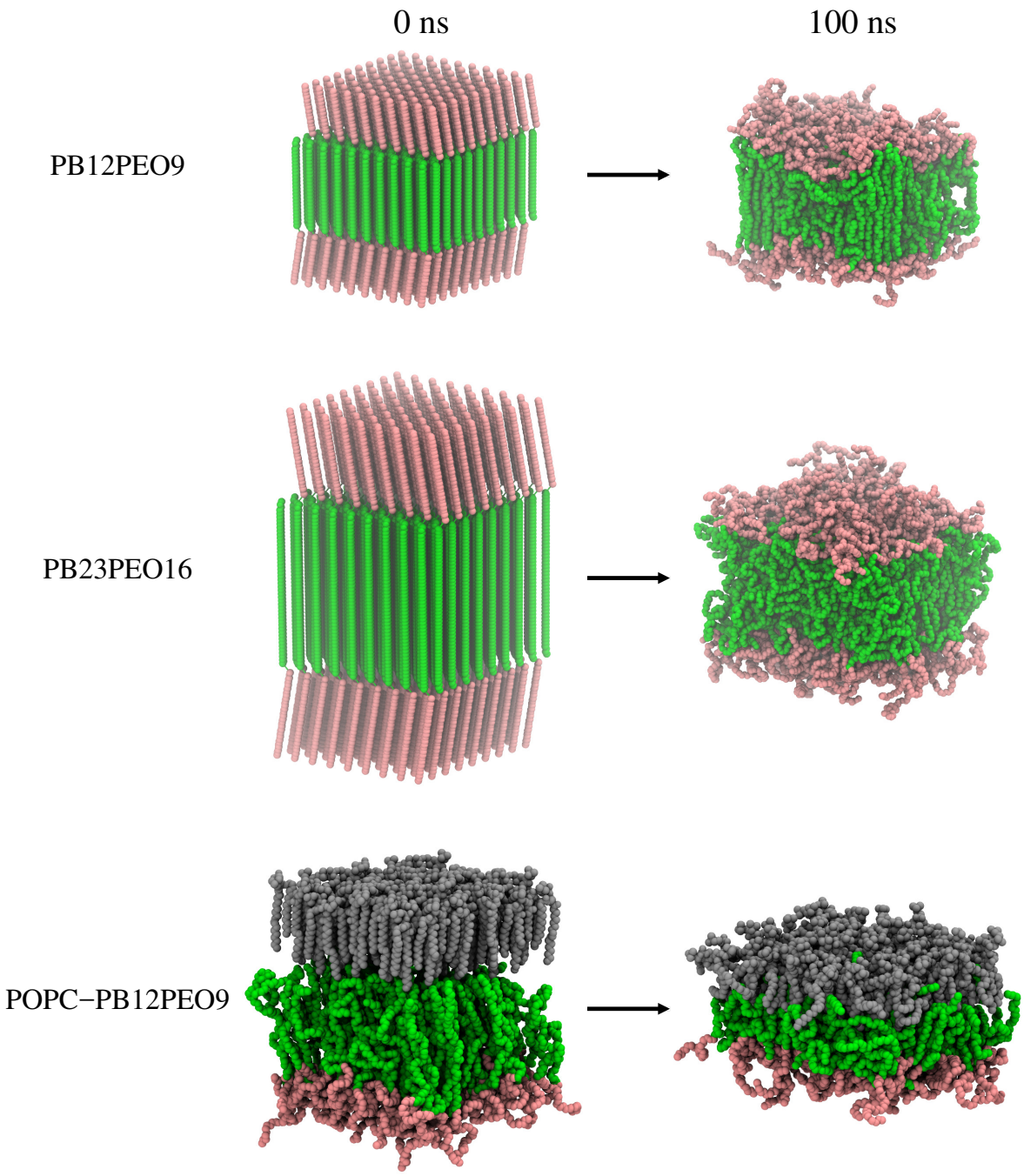


FIGURE 4.6: Snapshots of initial (0 ns) and final (100 ns) membrane ($PB_{12}PEO_9$, $PB_{23}PEO_{16}$, and POPC-$PB_{12}PEO_9$) structures during equilibration.

4.2.2 Permeability and Diffusivity

We measured the permeability of the PAP channel in POPC and $PB_{12}PEO_9$ membranes using a widely-used collective diffusion model of Zhu et al. [36]. A detailed description of this technique can be found in Section 2.4.5. No measurements are reported for the permeability of PAP in $PB_{23}PEO_{16}$ membranes as the channel did not span the membrane for the duration of any $PB_{23}PEO_{16}$ simulations (Figure 4.7). Therefore, water mobility in the channel could not result in net transport of water across the membrane. A decrease in permeability was observed on comparing PAP channels in POPC to those in $PB_{12}PEO_9$ (Figure 4.8D). The average permeability of the channel in POPC and $PB_{12}PEO_9$ was measured as 6.7 $(\pm 0.1) \times 10^{-14}$ $\frac{cm^3}{s}$ and 2.99 $(\pm 0.01) \times 10^{-14}$ $\frac{cm^3}{s}$, respectively.

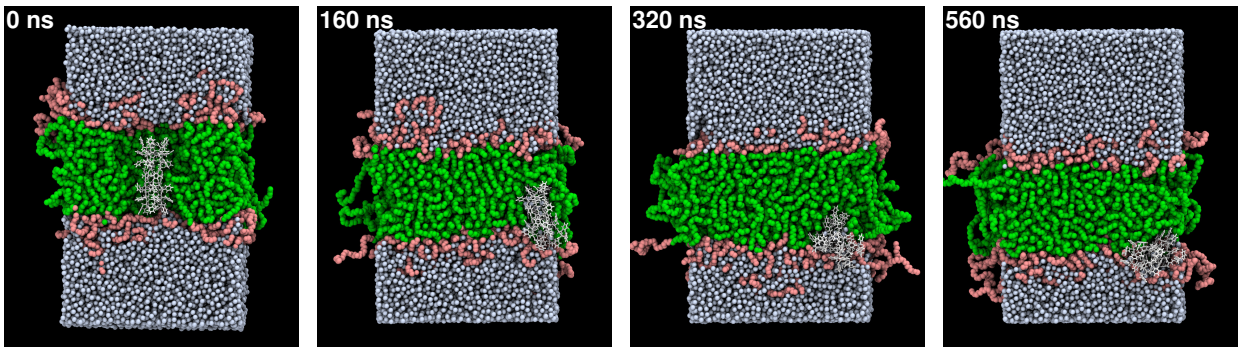


FIGURE 4.7: Snapshots showing diffusion of PAP out of the $PB_{23}PEO_{16}$ membrane. After ~60-100 ns, PAP localizes to one side of the $PB_{23}PEO_{16}$ membrane in all three $PB_{23}PEO_{16}$ PAP embedded simulations

The dominant contribution to the diffusivity of the PAP channel in each of the three membranes came from lateral movement in the plane of the membrane (Figures 4.8A-C). The lateral diffusivity of the PAP channel was measured by tracking the mean-squared-displacement (MSD) of the channel in each simulation. A detailed description of this technique can be found in Section 2.4.4. The lateral diffusivity of the PAP channel in POPC, $PB_{12}PEO_9$, and $PB_{23}PEO_{16}$ was measured as 5.35 (± 0.01) $\frac{\mu m^2}{s}$, 3.76 (± 0.01) $\frac{\mu m^2}{s}$, and 3.75 (± 0.02) $\frac{\mu m^2}{s}$ respectively.

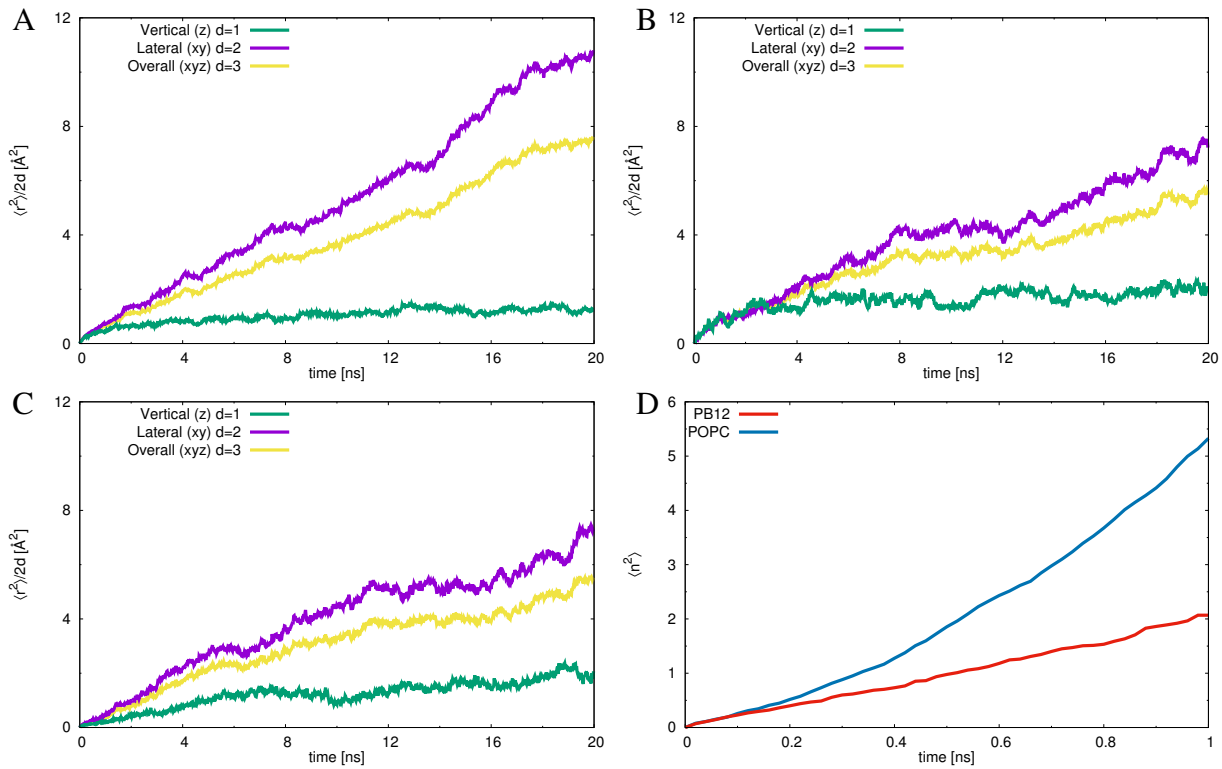


FIGURE 4.8: MSD (Å^2) vs. time (ns) plots of the pap channel in (POPC (A), $PB_{12}PEO_9$ (B), and $PB_{23}PEO_{16}$ (C)) membranes. The slope of each line is equal to the (overall (yellow), lateral (purple), and vertical (green)) diffusion coefficients of the PAP channel. (D) MSD vs. time (ns) of the collective diffusion coordinate n. The permeability of PAP in POPC (blue) and $PB_{12}PEO_9$ (red) membranes is proportional to the slope of each line.

4.2.3 RMSD and Orientation Angle of PAP

We report the root-mean-squared-deviation (RMSD) of PAP from all runs in Figure 4.9. In some of the simulations, the opening of the PAP channel at both ends was observed in a hydrophobic environment. When this happens, hydrophobic phenylalanine chains at the end of the channel create a hydrophobic cap over the entrance of the channel. This is likely a result of the hydrophilic ends of the channel arms rotating inward and collapsing toward the center of the channel. This type of structure can be seen in red conformation of PAP in Figure 4.9A. This conformation is seen in both BCP membranes as well as in one of the POPC runs in which the channel turned sideways

and rotated by ∼90°, thereby fully immersing in the hydrophobic layer of the POPC membrane (orange conformation in Figure 4.9B). This could be a possible explanation for decreased channel permeability and stability of PAP in BCP membranes. The orientation angle of PAP for all runs is reported in Figure 4.10. The angle traces show that the channel is oriented at an angle of ∼25° w.r.t. the membrane normal in most of the runs except for one $PB_{23}PEO_{16}$ run (cyan trace in Figure 4.10D) and in one POPC run, where it quickly adopts an angle of ∼90° w.r.t. the membrane normal (green trace in Figure 4.10B) thereby becoming parallel to the membrane surface while embedded in the membrane (see Figure 4.11 for additional snapshots for the green trace run shown in Figures 4.9B and 4.10B).

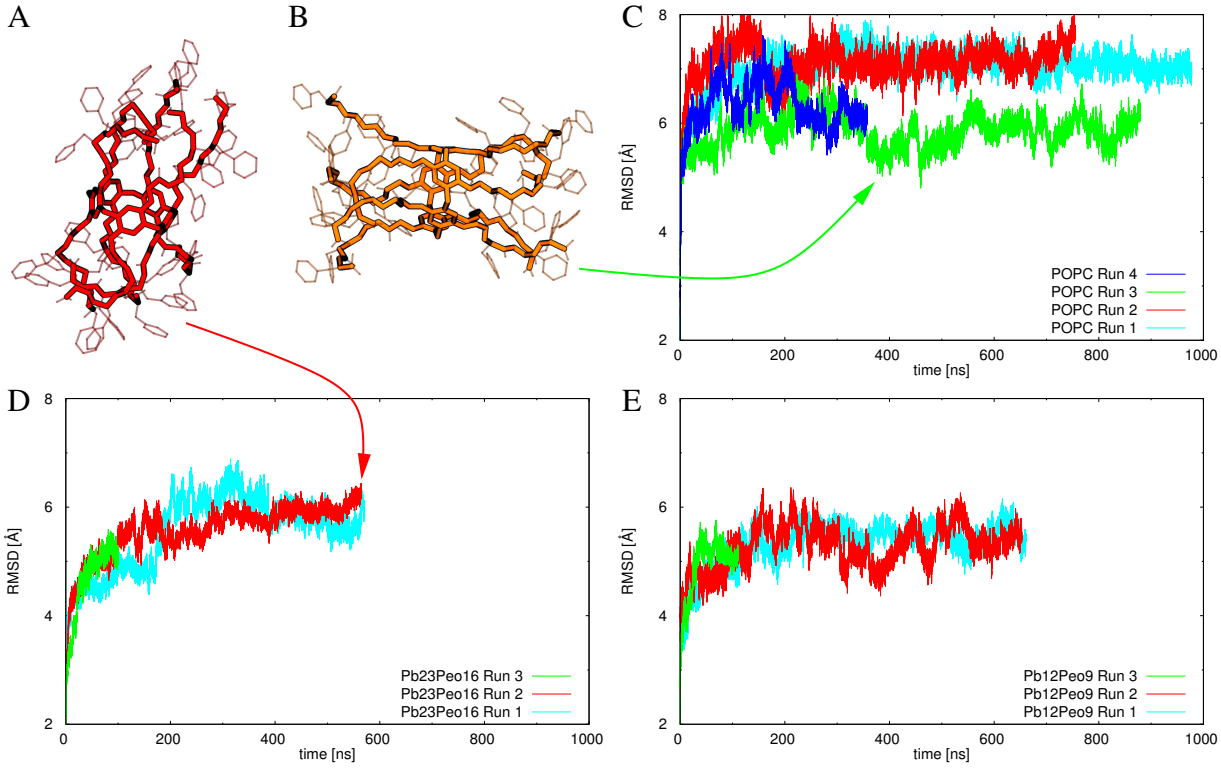


FIGURE 4.9: PAP structures observed in hydrophobic environments from $PB_{23}PEO_{16}$ (A) and POPC (B) simulations. RMSD (Å) vs. time (ns) of PAP in POPC (C), $PB_{23}PEO_{16}$ (D), and $PB_{12}PEO_9$ (E) membranes.

Additionally, the hydrophilic ends of the PAP channel cannot span the entire $PB_{23}PEO_{16}$

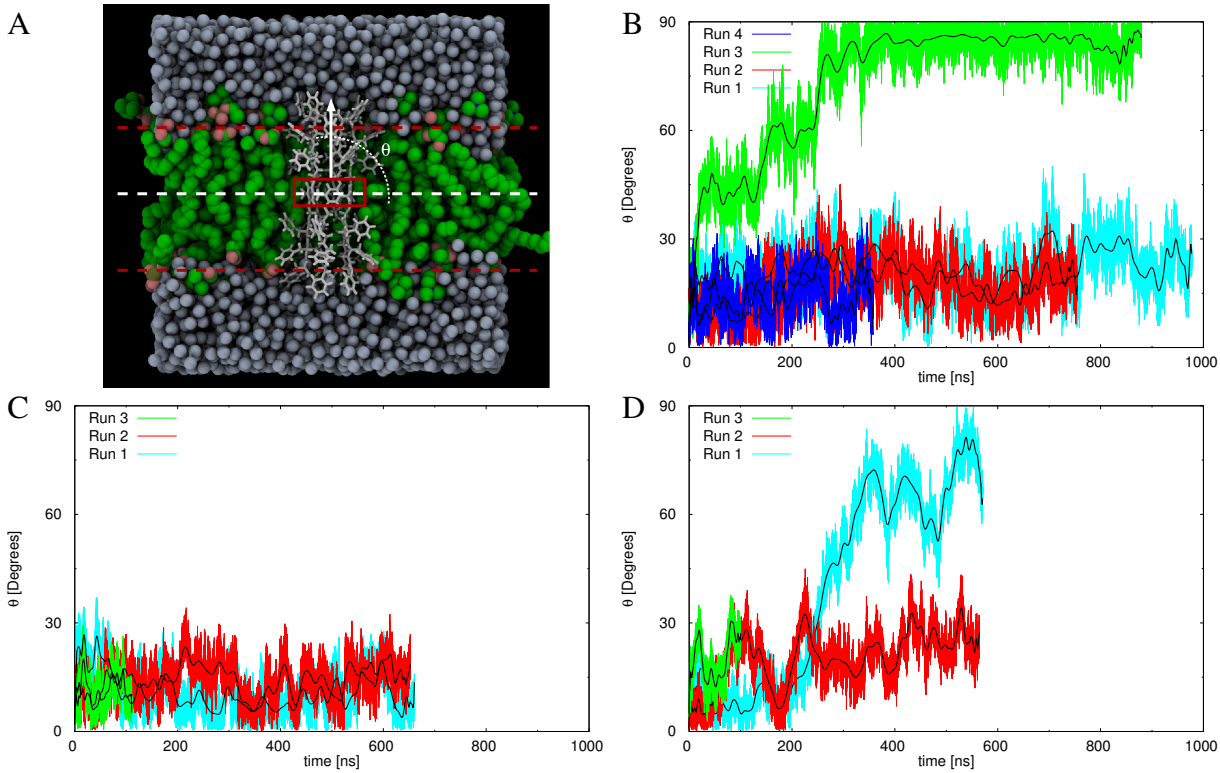


FIGURE 4.10: Diagram (A) showing the initial orientation angle (θ) of PAP defined as 0°. θ (°) vs. time (ns) of PAP in POPC (B), $PB_{12}PEO_9$ (C), and $PB_{23}PEO_{16}$ (D) membranes.

membrane to interact with PEO chains on both sides of the membrane at the same time (See Figure 4.3). As a result, the PAP channel in each $PB_{23}PEO_{16}$ run was observed to diffuse to one side of the $PB_{23}PEO_{16}$ membrane after ∼60-100 ns (Figure 4.7). In previous simulations of an array of PAP channels in POPC membranes [3], hydrogen bonding was observed between the carboxyl and amine groups among channels. It is possible that multiple channels interact via such specific hydrogen bonds which help not only in stabilizing the channel but also in spanning the $PB_{23}PEO_{16}$ membrane for successful water transport. This is a possible explanation for the channels ability to transport water in $PB_{23}PEO_{16}$ membranes as seen in experiments but not in $PB_{23}PEO_{16}$ simulations.

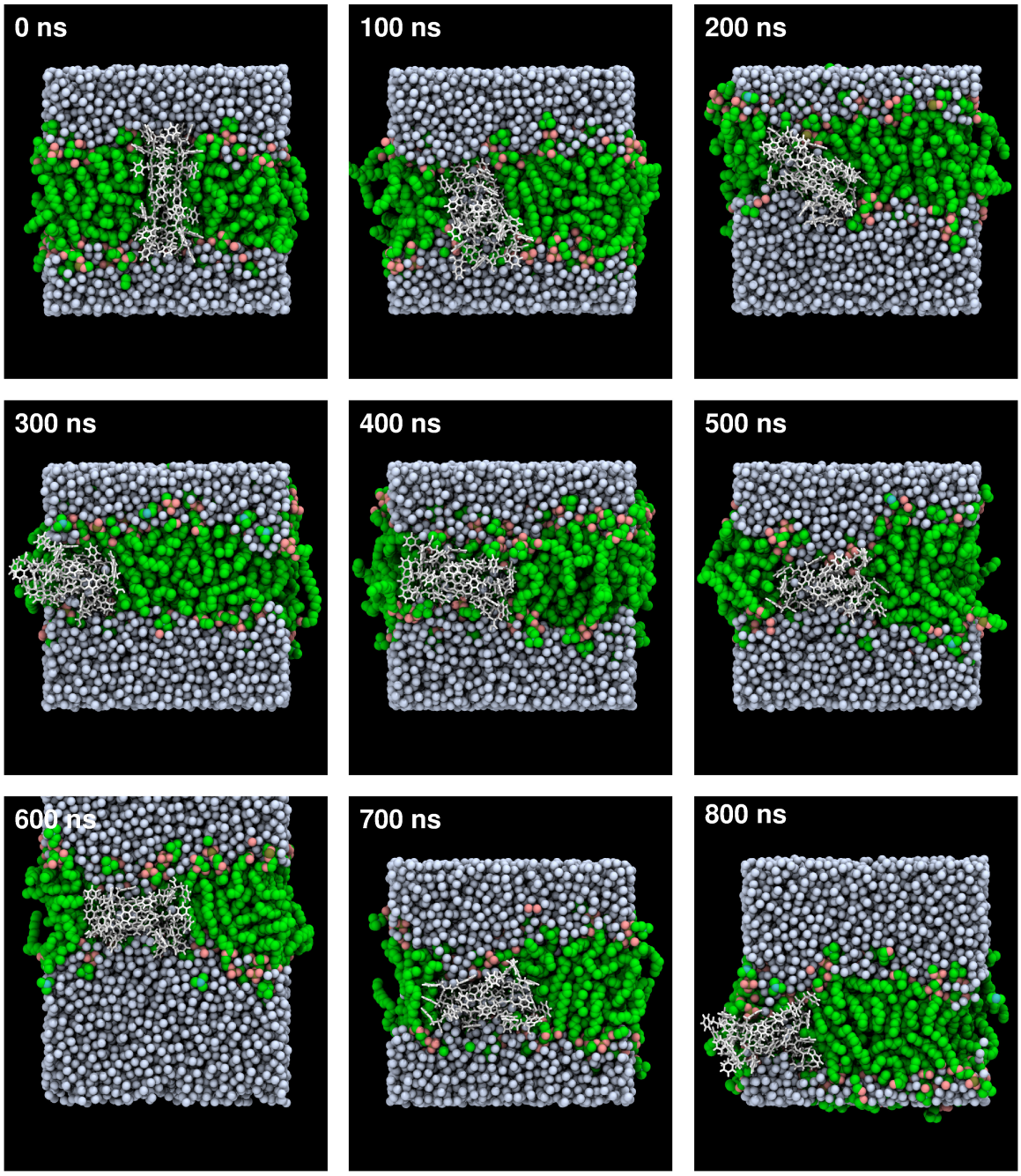


FIGURE 4.11: Snapshots spanning the entire PAP embedded in POPC Run 3 (Table A) simulation showing the channel adopt a ∼90° angle.

4.2.4 Summary and Conclusions

- Simulations containing a membrane system require special care and attention during set-up and initiation. When these steps are properly executed, simulations remain stable.
- Only water molecules should be used to specify and measure the area of periodic cells. Lipids and BCPs will not give measurements with the accuracy needed for membrane simulations.
- PI-PEO membranes were unstable when constrained to the same area/BCP values as PB-PEO membranes. When constraints on the area size were removed, $PI_{12}PEO_9$ and $PI_{23}PEO_{16}$ membranes reached equilibrium values of 0.88 ($\pm$0.04) $\frac{nm^2}{BCP}$ and 1.35 ($\pm$0.06) $\frac{nm^2}{BCP}$, respectively.
- The PAP channel was able to transport water and remain stable in POPC and $PB_{12}PEO_9$ membranes. The PAP channel was unstable in $PB_{23}PEO_{16}$ membranes, localizing to one side after $\sim$60-100 ns (Figure 4.7). This is likely due to the thickness of $PB_{23}PEO_{16}$ membranes relative to the length of PAP (Figure 4.3).
- Hydrophobic environments have an effect on the PAP channels average conformation (Figure 4.9). This may cause a reduction in transport ability (Figure 4.8D).

Chapter 5

Simulations of Coarse Grained Systems

The microscopic structures of block copolymer membranes remain unknown. Typical imaging techniques for these structures have a resolution capable of measuring thickness of the membranes but provide no insight as to the overall microscopic organization of the bilayer. The bilayers could form much like lipids with two chains interacting end to end (Figure 5.1A) or align in a cross linked pattern fully integrating each membrane layer into a single hydrophobic core (Figure 5.1B). Throughout the all atom simulations performed, we postulated initial structures of BCP membranes based upon geometric constraints on chain packing. This was necessary because performing self assembly simulations in all-atom detail is not feasible with reasonable computational resources. In an attempt to explore the possible membrane configurations, we used coarse grained simulations. In this chapter, we present our initial explorations of this method.

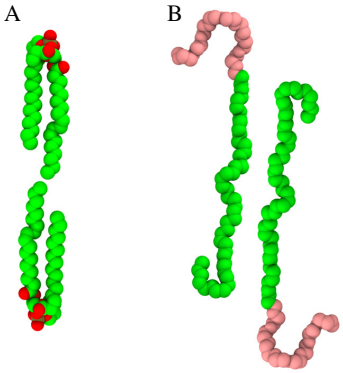


FIGURE 5.1: Snapshots illustrating two POPC lipids interacting end to end (A) and two $PB_{12}PEO_9$ molecules aligned in a cross linked pattern (B).

Coarse graining (CG) is a common technique used in molecular dynamics (MD) simulations. In general, this involves reducing the number of degrees-of-freedom present in a system, thereby reducing the computational complexity inherent to MD simulations. This increases the speed at which simulations can be performed, allowing the observation of behaviors that occur over longer time scales. This technique must be used judiciously as it may not faithfully reproduce all properties of a system. Coarse graining has been used extensively to study several biomolecular systems as well as the self assembly behavior of polymers [37, 38, 40, 46–54]. The MARTINI force-field [32] is a popular CG approach that was developed initially to simulate lipids and proteins and is optimized for these molecules. This force-field can be modified and optimized to simulate polymers and block copolymers including polyethylene oxide [48]. In Section 5.1, we test the original MARTINI [32] force-field parameters by simulating the self-assembly of a POPC lipid bilayer. We then use the MARTINI [32] force-field along with a VMD [30] plugin to build and simulate coarse grained models of $PB_{12}PEO_9$ in the Section 5.2. A brief description of the MARTINI [32] coarse graining method can be found in Section 2.2.3 and a more complete description is found in the published work on this force-field [32].

5.1 Self Assembly of Lipids

To familiarize ourselves with the technique of residue based coarse graining (RBCG) with the MARTINI force-field [32], we simulated the self-assembly of a POPC lipid bilayer. The steps for system preparation outlined in this section follow those presented in the RBCG tutorial [55] found here: `http://www.ks.uiuc.edu/Training/Tutorials`. This website also includes a link for the necessary files used to perform this simulation. We first constructed an all atom POPC membrane using the membrane builder plugin of VMD [30]. The membrane was 100 Å $\times$ 100 Å in size. All water molecules were removed and the membrane was dispersed in the direction normal to the membrane surface using the following code:

```
mol new membrane.psf
mol addfile membrane.pdb

for {set j 11} {$j < 29 } { incr j } {
for {set i 1 } {$i < 30 } { incr i } {
set sel [atomselect top "segid L$j and resid $i"]
set z [expr {45*(sin(($i)*15))}]
set offset [list 0 0 $z]
$sel moveby $offset
}
}
resetpsf
readpsf membrane.psf
set POPC [atomselect top "resname POPC and not (segid L11 and resid 1) and not
(segid L11 and resid 2) and not (segid L11 and resid 3) and not (segid L11 and
resid 4) and not (segid L11 and resid 5) and not (segid L11 and resid 6) and
not (segid L11 and resid 7)"]

$POPC writepdb self_popc.pdb
$POPC writepsf self_popc.psf

exit
```

The CG builder plugin [55] of VMD [30] was used to create the coarse grained model of this system. The necessary topology and .cgc files were provided in the tutorial files download link: `http://www.ks.uiuc.edu/Training/Tutorials`. A brief description of coarse grained topology and .cgc files can be found in Section 5.2. A more detailed description of these files can be found in the following tutorial [55]. The system was then solvated as described in the tutorial [55]. After a brief minimization, the coarse grained system was simulated in the NPT ensemble with a fixed area using a 20 fs time-step. All the necessary parameter files were provided in the tutorial files download link here: `http://www.ks.uiuc.edu/Training/Tutorials`. The simulation showed a successful self assembly of the lipid bilayer as shown in Figure 5.2

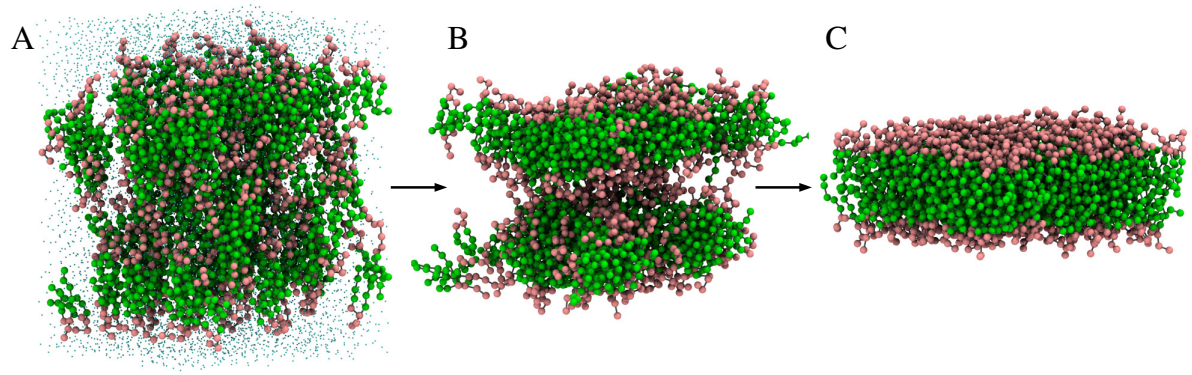


FIGURE 5.2: Snapshots of the successful simulation of a POPC lipid bilayer self assembly showing the initial (A), intermediate (B), and final (C) configurations.

5.2 $PB_{12}PEO_9$ Coarse Grained Simulations

After testing the MARTINI [32] force-field parameters for self assembly of a lipid bilayer, we developed necessary files to coarse grain one of our BCP structures. We did this by first building an all atom structure of $PB_{12}PEO_9$ as outlined in Section 3.2. The CG builder plugin was again used to create the coarse grained structure, however, we had to generate the .cgc and topology files needed. The .cgc file contains the mapping information to create a coarse grained PDB file. In the .cgc file, each super atom in the structure is defined by a list of the corresponding atoms in the all atom structure. The information in the file is organized as follows:

```
CGBEGIN
(RESNAME)(BEADNAME)0
(RESNAME)(ATOMNAME)0 (the first atom should be the central atom of the cluster)
(RESNAME)(ATOMNAME)0 (beyond that, the ordering is unimportant)
...
CGEND
```

This file is read by CG builder which then creates the desired coarse grained .pdb file. In our coarse grained model, it was natural to define each monomer in our BCP chains as one super atom. Each of the PB monomers were defined as a type C3T super atom. The PEO monomers were defined as a

type SN0 super atoms. These super atom types were based upon previous papers on the MARTINI force-field [32, 48]. The .cgc file used in all BCP RBCG simulations is included below.

```
#CG bead definitions for PBPEO

CGBEGIN
BDE             BCP     0
BDE     C2       0
BDE     H13      0
BDE     C1       0
BDE     C3       0
BDE     H11      0
BDE     H12      0
BDE     H21      0
BDE     H31      0
BDE     C4       0
BDE     H41      0
BDE     H42      0
CGEND

CGBEGIN
PEGM    BCP      0
PEGM    O1       0
PEGM    C1       0
PEGM    H1B      0
PEGM    H1A      0
PEGM    C2       0
PEGM    H2A      0
PEGM    H2B      0
PEGM    H2C      0
CGEND
```

To generate the corresponding .psf file using psfgen, we created a coarse grained topology file for the $PB_{12}PEO_9$ structure. This topology file differs from those described in Section 3.1.2. In these files, every super atom is given the generic name BCP. This simplifies defining all bonds and angles in the structure. Each super atom type is then determined by their residue name. The topology file and .cgc file must be consistent to avoid errors. A copy of the topology file we manually created is included below.

```
! PB
RESI BDE        0.0 !
ATOM BCP   C3T  0.0 !
BOND  BCP +BCP             !
ANGLE -BCP BCP +BCP !
! PEO
RESI PEGM            0.0 !
ATOM BCP   SN0  0.0 !
BOND  BCP +BCP      !
ANGLE -BCP BCP +BCP !
!
PRES PBPE

GROUP
ATOM BCP   C3T  0.0 !
GROUP
ATOM BCP   SN0  0.0 !

BOND  1BCP 2BCP     !

END
```

We note that there are no dihedral angles defined in the above topology file. This is not an oversight or error. The MARTINI [32] force-field does not use a dihedral term for lipids, therefore these simulations did not use any dihedral angles for the given atom types. Using these files, three different systems were prepared for simulation. The first system was a single CG $PB_{12}PEO_9$ chain inside a water box to test if the files and structure we created were compatible with the parameter files included with the tutorial [55]. This test was successful, so we then built two larger coarse grained models. These were longer simulations to test the self assembly behavior of the force-field.

The first of these longer simulations consisted of 72 coarse grained $PB_{12}PEO_9$ chains. These chains were placed in a water box with initial dimensions of 130 Å $\times$ 130 Å $\times$ 180 Å. After a brief minimization, this system was simulated in the NPT ensemble using a 20 fs time step. The BCP concentration inside the system was not large enough to capture membrane formation so there were no constraints placed on the cell area. In this simulation we wanted to observe micelle formation. The force-field was able to rapidly produce this behavior. Snapshots of the micelle formation can

be seen in Figure 5.3.

After the successful formation of micelles, we attempted to simulate the self assembly of a $PB_{12}PEO_9$ membrane. We first dispersed one of the all-atom membrane models of $PB_{12}PEO_9$ that we built in Section 4.1.1 using a code similar to the one included in Section 5.1. The all-atom model was then coarse gained and after a brief minimization, simulated in the NPT ensemble using a 20 fs time step. The area of the periodic cell was constrained to achieve a constant area/BCP value consistent with those in Section 4.1.1. The results of this simulation can be seen in Figure 5.4. The BCPs did assemble into a distinct phase, however, it is evident from the snapshot shown in Figure 5.4B that the hydrophobic and hydrophilic chains were not able to separate into distinct layers. This can also be seen in the density of species plot shown in Figure 5.4C. The original MARTINI force-field was unable to show the self assembly of $PB_{12}PEO_9$ and must therefore be further optimized to observe the desired behavior.

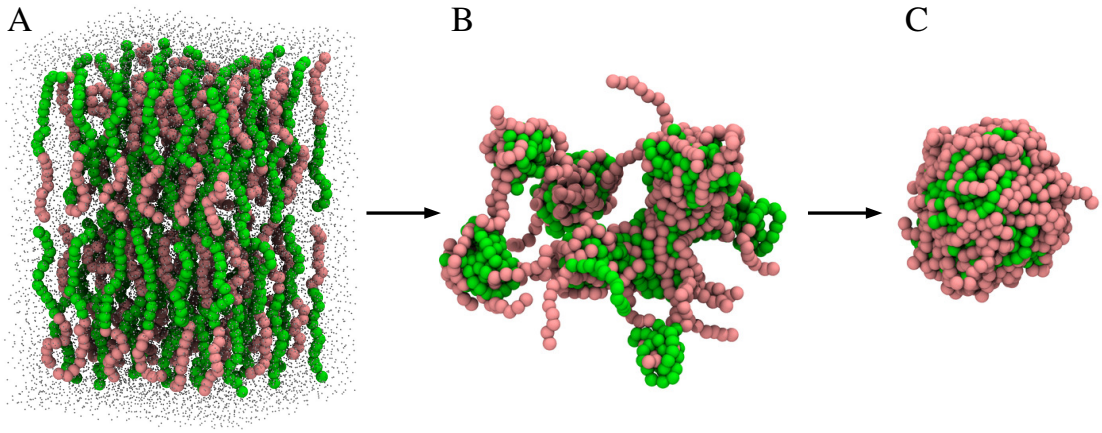


FIGURE 5.3: Snapshots from the first long time scale RBCG simulation of $PB_{12}PEO_9$ chains showing the initial configuration (A) followed by the formation of aggregates (B) and eventually a micellar structure (C).

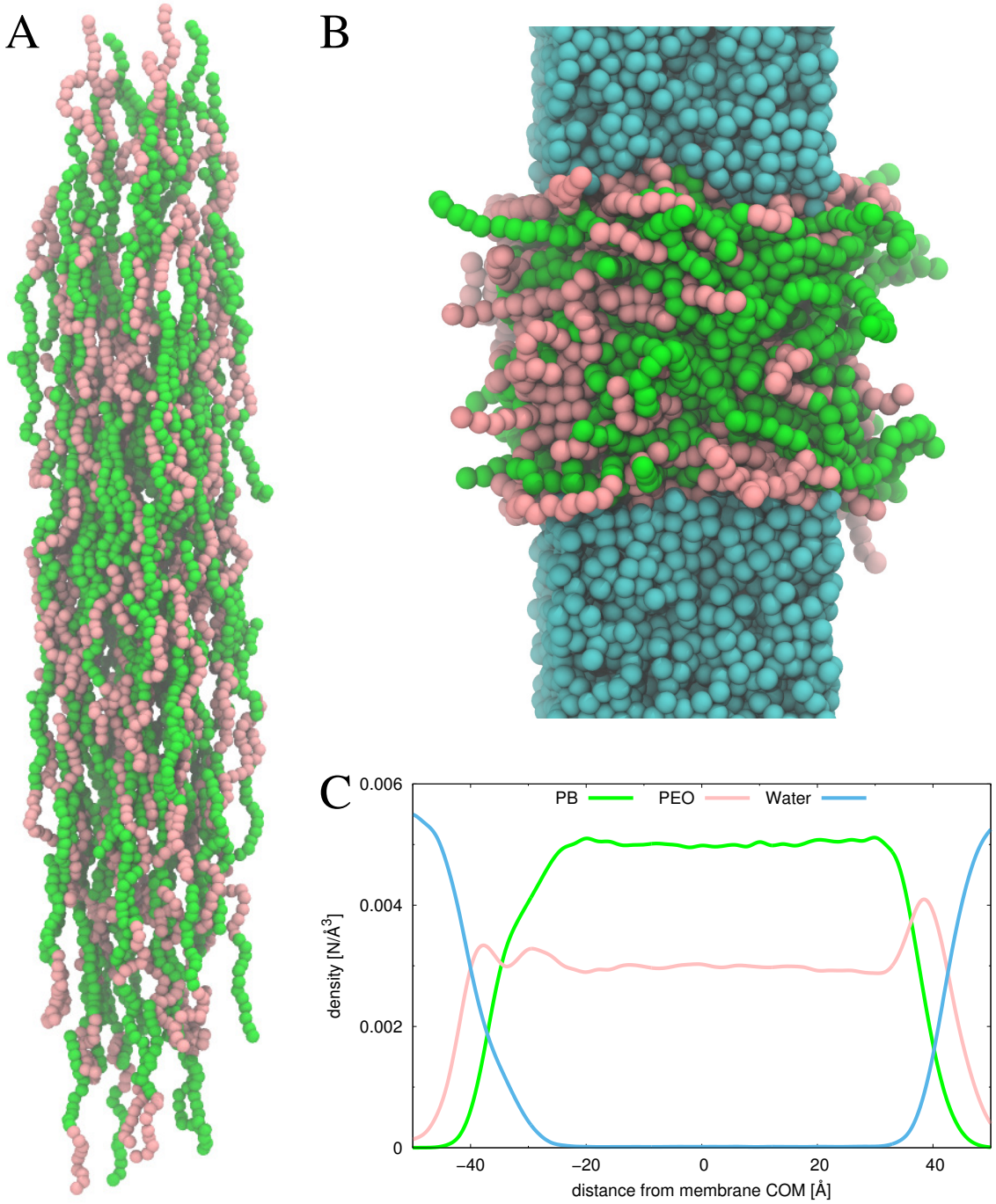


FIGURE 5.4: Snapshots of the initial (A) and final (B) configurations of the failed RBCG self assembly simulation of a $PB_{12}PEO_9$ membrane. The density of species plot (C) shows the BCP phase failed to form distinct hydrophilic and hydrophobic layers.

Chapter 6

Future Directions

The work presented in this thesis provides many opportunities for continued research. In this chapter, we offer suggestions on how one could proceed with further investigations. In Chapter 3 we were unable to obtain parameters to simulate PB-PMOXA and PI-PMOXA BCPs. New strategies to obtain these parameters must be explored. There are also a number of other polymer structures one could build and simulate. These polymers include 1,2-polybutadiene and polydimethylsiloxane (PDMS). Along with the missing parameters, new polymers would allow building and testing many new BCP membrane structures. This would significantly expand the work done in Chapter 4.

New simulations of pristine membranes with an induced pressure gradient could be useful to measure the surface tension of the membranes. Using simulations of this type, one could also test the rupturing pressure. These properties, along with hydrophobic thickness and area/BCP could be benchmarked against experimental measurements. This information could also asist in force-field refinement. In addition, larger membrane simulations containing multiple PAP channels could be done to investigate whether interactions among PAPs have an effect on their stability and transport ability. New transport channels can also be explored. For example, computer simulations of imidazole quartet (I-quartet) channels in lipids have been successfully completed [18]. These I-quartet channels can be incorporated and simulated in our BCP membranes to study the effect of membrane environment. Explorations of hybrid asymmetric membranes paired with asymmetric transport channels are also of interest for future work.

In Chapter 5, we were unsuccessful in our attempts to simulate a BCP membrane self assembly. To achieve this, the MARTINI force-field [32] requires further optimization. There are a number

of references which outline a procedure for extracting coarse grained (CG) parameters from all-atom simulations [47–49]. These methods can be used to develop CG parameters from all-atom simulations of a single BCP chain in a water box. These systems can be small and generate sufficient data in relatively short periods of time. Once the CG parameters have been extracted, self assembly simulations can be performed and mapped back to all-atom models. This will provide a rapid method of membrane building without postulating an initial structure. Furthermore, with CG models of BCPs and PAP, it may be possible to simulate the self assembly of entire vesicles containing PAPs.

References

[1] M. A. Shannon, P. W. Bohn, M. Elimelech, J. G. Georgiadis, B. J. Marinas, and A. M. Mayes, "Science and technology for water purification in the coming decades", *Nature*, vol. 452, no. 7185, pp. 301–310, 2008.

[2] M. Elimelech and W. A. Phillip, "The future of seawater desalination: Energy, technology, and the environment", *Science*, vol. 333, no. 6043, pp. 712–717, 2011.

[3] Y.-X. Shen, W. Si, M. Erbakan, K. Decker, R. De Zorzi, P. O. Saboe, Y. J. Kang, S. Majd, P. J. Butler, T. Walz, *et al.*, "Highly permeable artificial water channels that can self-assemble into two-dimensional arrays", *Proc. Natl. Acad. Sci. USA*, vol. 112, no. 32, pp. 9810–9815, 2015.

[4] M. Kumar, M. Grzelakowski, J. Zilles, M. Clark, and W. Meier, "Highly permeable polymeric membranes based on the incorporation of the functional water channel protein aquaporin z", *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 52, pp. 20 719–20 724, 2007.

[5] M. Kumar, J. E. Habel, Y.-X. Shen, W. P. Meier, and T. Walz, "High-density reconstitution of functional water channels into vesicular and planar block copolymer membranes", *J. Am. Chem. Soc.*, vol. 134, no. 45, pp. 18 631–18 637, 2012.

[6] P. Agre, "Aquaporin water channels", *Biosci. Rep.*, vol. 24, no. 3, pp. 127–163, 2004.

[7] B. L. de Groot and H. Grubmüller, "Water permeation across biological membranes: Mechanism and dynamics of aquaporin-1 and GlpF", *Science*, vol. 294, no. 5550, pp. 2353–2357, 2001.

[8] B. Bhushan, "Biomimetics: Lessons from nature–an overview", *Philos. Trans. A. Math Phys. Eng. Sci.*, vol. 367, no. 1893, pp. 1445–1486, 2009.

[9] A. B. Mann, R. R. Naik, H. C. DeLong, and K. H. Sandhage, "Biomimetic and bio-enabled materials science and engineering", *J. Mater. Res.*, vol. 23, pp. 3137–3139, 2008.

[10] C. Tang, Y. Zhao, R. Wang, C. Hélix-Nielsen, and A. Fane, "Desalination by biomimetic aquaporin membranes: Review of status and prospects", *Desalination*, vol. 308, pp. 34–40, 2013.

[11] Y.-X. Shen, P. O. Saboe, I. T. Sines, M. Erbakan, and M. Kumar, "Biomimetic membranes: A review", *J. Mem. Sci.*, vol. 454, pp. 359–381, 2014.

[12] B. M. Discher, Y.-Y. Won, D. S. Ege, J. C. Lee, F. S. Bates, D. E. Discher, and D. A. Hammer, "Polymersomes: Tough vesicles made from diblock copolymers", *Science*, vol. 284, no. 5417, pp. 1143–1146, 1999.

[13] G. H. Fredrickson and F. S. Bates, "Dynamics of block copolymers: Theory and experiment", *Annu. Rev. Mat. Sci.*, vol. 26, no. 1, pp. 501–550, 1996.

[14] A. González-Pérez, V. Castelletto, I. W. Hamley, and P. Taboada, "Biomimetic triblock copolymer membranes: From aqueous solutions to solid supports", *Soft Matter*, vol. 7, no. 3, pp. 1129–1138, 2011.

[15] A. Mecke, C. Dittrich, and W. Meier, "Biomimetic membranes designed from amphiphilic block copolymers", *Soft Matter*, vol. 2, no. 9, pp. 751–759, 2006.

[16] B. Corry, "Designing carbon nanotube membranes for efficient water desalination", *J. Phys. Chem. B*, vol. 112, no. 5, pp. 1427–1434, 2008.

[17] M. Majumder, N. Chopra, R. Andrews, and B. J. Hinds, "Nanoscale hydrodynamics: Enhanced flow in carbon nanotubes", *Nature*, vol. 438, no. 7064, pp. 44–44, 2005.

[18] E. Licsandru, I. Kocsis, Y.-x. Shen, S. Murail, Y.-M. Legrand, A. Van Der Lee, D. Tsai, M. Baaden, M. Kumar, and M. Barboiu, "Salt-excluding artificial water channels exhibiting enhanced dipolar water and proton translocation", *J. Am. Chem. Soc.*, vol. 138, no. 16, pp. 5403–5409, 2016.

[19] L. E. Cheruzel, M. S. Pometun, M. R. Cecil, M. S. Mashuta, R. J. Wittebort, and R. M. Buchanan, "Structures and solid-state dynamics of one-dimensional water chains stabilized by imidazole channels", *Angew. Chem. Int. Ed.*, vol. 42, no. 44, pp. 5452–5455, 2003.

[20] X.-B. Hu, Z. Chen, G. Tang, J.-L. Hou, and Z.-T. Li, "Single-molecular artificial transmembrane water channels", *J. Am. Chem. Soc.*, vol. 134, no. 20, pp. 8384–8387, 2012.

[21] M. Barboiu and A. Gilles, "From natural to bioassisted and biomimetic artificial water channel systems", *Acc. Chem. Res.*, vol. 46, no. 12, pp. 2814–2823, 2013.

[22] W. Si, L. Chen, X.-B. Hu, G. Tang, Z. Chen, J.-L. Hou, and Z.-T. Li, "Selective artificial transmembrane channels for protons by formation of water wires", *Angew. Chem. Int. Ed.*, vol. 123, no. 52, pp. 12 772–12 776, 2011.

[23] J. M. Prausnitz, R. N. Lichtenthaler, and E. G. de Azevedo, *Molecular thermodynamics of fluid-phase equilibria*, Third. Pearson Education, 1998.

[24] Y.-Y. Won, H. T. Davis, and F. S. Bates, "Giant wormlike rubber micelles", *Science*, vol. 283, no. 5404, pp. 960–963, 1999.

[25] E. Rakhmatullina and W. Meier, "Solid-supported block copolymer membranes through interfacial adsorption of charged block copolymer vesicles", *Langmuir*, vol. 24, no. 12, pp. 6254–6261, 2008.

[26] H. Bermudez, A. K. Brannan, D. A. Hammer, F. S. Bates, and D. E. Discher, "Molecular weight dependence of polymersome membrane structure, elasticity, and stability", *Macromolecules*, vol. 35, no. 21, pp. 8203–8208, 2002.

[27] F. Reif, *Fundamentals of statistical and thermal physics*. Waveland Press, 2009.

[28] D. Frenkel and B. Smit, *Understanding molecular simulation: From algorithms to applications*. Academic Press, 2001, vol. 1.

[29] J. C. Maxwell, *The Scientific Papers of James Clerk Maxwell*. University Press, 1890, vol. 2.

[30] W. Humphrey, A. Dalke, and K. Schulten, "Vmd: Visual molecular dynamics", *J. Mol. Graph.*, vol. 14, no. 1, pp. 33–38, 1996.

[31] B. R. Brooks, C. L. Brooks, A. D. MacKerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, *et al.*, "Charmm: The biomolecular simulation program", *J. Comp. Chem.*, vol. 30, no. 10, pp. 1545–1614, 2009.

[32] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. De Vries, "The martini force field: Coarse grained model for biomolecular simulations", *J. Phys. Chem. B*, vol. 111, no. 27, pp. 7812–7824, 2007.

[33] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten, "Scalable molecular dynamics with namd", *J. Comp. Chem.*, vol. 26, no. 16, pp. 1781–1802, 2005.

[34] A. Aksimentiev, M. Sotomayor, and D. Wells, "Membrane proteins tutorial", *University of Illinois at Urbana-Champaign*, 2012. [Online]. Available: `http://www.ks.uiuc.edu/Training/Tutorials/science/membrane/mem-tutorial.pdf`.

[35] A. Einstein, "On the movement of small particles suspended in stationary liquids required by the molecular-kinetic theory of heat", *Ann. D. Phys.*, vol. 17, pp. 549–560, 1905.

[36] F. Zhu, E. Tajkhorshid, and K. Schulten, "Collective diffusion model for water permeation through microscopic channels", *Phys. Rev. Lett.*, vol. 93, no. 22, p. 224 501, 2004.

[37] G. Srinivas, W. C. Swope, and J. W. Pitera, "Interfacial fluctuations of block copolymers: A coarse-grain molecular dynamics simulation study", *J. Phys. Chem. B*, vol. 111, no. 49, pp. 13 734–13 742, 2007.

[38] G. Srinivas, D. E. Discher, and M. L. Klein, "Self-assembly and properties of diblock copolymers by coarse-grain molecular dynamics", *Nat. Mat.*, vol. 3, no. 9, pp. 638–644, 2004.

[39] O.-S. Lee, V. Cho, and G. C. Schatz, "Modeling the self-assembly of peptide amphiphiles into fibers using coarse-grained molecular dynamics", *Nano Lett.*, vol. 12, no. 9, pp. 4907–4913, 2012.

[40] G. Srinivas, J. C. Shelley, S. O. Nielsen, D. E. Discher, and M. L. Klein, "Simulation of diblock copolymer self-assembly, using a coarse-grain model", *J. Phys. Chem. B*, vol. 108, no. 24, pp. 8153–8160, 2004.

[41] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank", *Nuc. Acids Res.*, vol. 28, no. 1, pp. 235–242, 2000.

[42] C. G. Mayne, J. Saam, K. Schulten, E. Tajkhorshid, and J. C. Gumbart, "Rapid parameterization of small molecules using the force field toolkit", *J. Comp. Chem.*, vol. 34, no. 32, pp. 2757–2770, 2013.

[43] M. Frisch, G. Trucks, H. Schlegel, G. Scuseria, M. Robb, J. Cheesemann, V. Zakrzewski, J. Montgomery Jr, R. Stratmann, J. Burant, *et al.*, *Gaussian software*, 2003.

[44] J. Phillips, T. Isgro, M. Sotomayor, and E. Villa, "NAMD tutorial", *Windows Version. University of Illinois. NIH Resource for Macromolecular Modelling and Bioinformatics Beckman Institute*, 2007.

[45] C. G. Mayne, M. Muller, and E. Tajkhorshid, "Parameterizing small molecules using the force field toolkit (fftk)", *University of Illinois at Urbana-Champaign*, 2015. [Online]. Available: `http://www.ks.uiuc.edu/Training/Tutorials/science/ffTK/fftk-tutorial.pdf`.

[46] K. A. Scott, P. J. Bond, A. Ivetac, A. P. Chetwynd, S. Khalid, and M. S. Sansom, "Coarse-grained md simulations of membrane protein-bilayer self-assembly", *Structure*, vol. 16, no. 4, pp. 621–630, 2008.

[47] D. Reith, M. Pütz, and F. Müller-Plathe, "Deriving effective mesoscale potentials from atomistic simulations", *J. Comp. Chem.*, vol. 24, no. 13, pp. 1624–1636, 2003.

[48] H. Lee, A. H. de Vries, S.-J. Marrink, and R. W. Pastor, "A coarse-grained model for polyethylene oxide and polyethylene glycol: Conformation and hydrodynamics", *J. Phys. Chem. B*, vol. 113, no. 40, pp. 13 186–13 194, 2009.

[49] B. Bayramoglu and R. Faller, "Coarse-grained modeling of polystyrene in various environments by iterative boltzmann inversion", *Macromolecules*, vol. 45, no. 22, pp. 9205–9219, 2012.

[50] A. J. Peters, R. A. Lawson, P. J. Ludovice, and C. L. Henderson, "Detailed mesoscale dynamic simulation of block copolymer directed self-assembly processes: Application of protracted colored noise dynamics", in *SPIE Advanced Lithography*, International Society for Optics and Photonics, 2012, 83231T–83231T.

[51] D. Reith, H. Meyer, and F. Müller-Plathe, "Mapping atomistic to coarse-grained polymer models using automatic simplex optimization to fit structural properties", *Macromolecules*, vol. 34, no. 7, pp. 2335–2345, 2001.

[52] D. Fritz, V. A. Harmandaris, K. Kremer, and N. F. van der Vegt, "Coarse-grained polymer melts based on isolated atomistic chains: Simulation of polystyrene of different tacticities", *Macromolecules*, vol. 42, no. 19, pp. 7579–7588, 2009.

[53] M. L. Klein and W. Shinoda, "Large-scale molecular dynamics simulations of self-assembling systems", *Science*, vol. 321, no. 5890, pp. 798–800, 2008.

[54] Z.-J. Wang and M. Deserno, "A systematically coarse-grained solvent-free model for quantitative phospholipid bilayer simulations", *J. Phys. Chem. B*, vol. 114, no. 34, pp. 11 207–11 220, 2010.

[55] R. Gamini and D. Chandler, "Residue-based coarse graining using martini force field in namd", in *University of Illinois at Urbana-Champaign, Computational Biophysics Workshop*, 2013, p. 3.

[56] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, *et al.*, "General atomic and molecular electronic structure system", *J. Comp. Chem.*, vol. 14, no. 11, pp. 1347–1363, 1993.

[57] F. Zhu, E. Tajkhorshid, and K. Schulten, "Pressure-induced water transport in membrane channels studied by molecular dynamics", *Biophys. J.*, vol. 83, no. 1, pp. 154–160, 2002.

[58] W. Li, Y. Kim, J. Li, and M. Lee, "Dynamic self-assembly of coordination polymers in aqueous solution", *Soft Matter*, vol. 10, no. 29, pp. 5231–5242, 2014.

[59] Y.-H. Tang, Z. Li, X. Li, M. Deng, and G. E. Karniadakis, "Non-equilibrium dynamics of vesicles and micelles by self-assembly of block copolymers with double thermoresponsivity", *Macromolecules*, vol. 49, no. 7, pp. 2895–2903, 2016.

Appendix A

Simulation Details

TABLE A.1: List of all-atom simulations conducted. Pristine runs are simulations of pure membrane structures, PAP embedded runs are simulations with a PAP channel inserted in a membrane, and free runs are simulations with the PAP channel displaced away from a pristine membrane.

Membrane Type	Run (Simulation Type)	Time-Step (fs)	Size (Atoms)	Length (ns)
POPC	Run 1 (PAP Embedded)	2	32,666	977.44
	Run 2 (PAP Embedded)	2	32,666	754.46
	Run 3 (PAP Embedded)	2	32,666	879.54
	Run 4 (PAP Embedded)	2	32,666	357.46
	Run 5 (PAP Free)	2	43,624	305.16
	Run 6 (PAP Free)	2	43,624	663.9
$PB_{12}PEO_9$	Pristine	2	75,124	100.64
	Run 1 (PAP Embedded)	2	71,408	660.46
	Run 2 (PAP Embedded)	2	71,408	652.72
	Run 3 (PAP Embedded)	2	71,408	110.78
	Run 4 (PAP Free)	2	75,001	237.82
	Run 5 (PAP Free)	2	75,001	209.76
$PB_{23}PEO_{16}$	Pristine	2	123,249	191.96
	Run 1 (PAP Embedded)	2	119,023	570.82
	Run 2 (PAP Embedded)	2	119,023	564.56
	Run 3 (PAP Embedded)	2	119,023	100
$PI_{12}PEO_9$	Pristine	2	87,796	100
$PI_{23}PEO_{16}$	Pristine	2	135,054	100

Appendix B

Input Files and Sample Scripts

B.1 Gaussian Geometry Optimization Input File

```
%chk=geomOpt.chk %nproc=1 %mem=1GB
# MP2/6-31G* Opt=(Redundant) SCF=Tight Geom=PrintInputOrient
<qmtool> simtype=‘‘Geometry optimization" </qmtool>
0 1
C1 0.0 0.0 0.0
H2 -0.32600000500679016 0.9459999799728394 -0.0
H3 -0.32600000500679016 -0.4729999899864197 0.8190000057220459
H4 -0.32600000500679016 -0.4729999899864197 -0.8190000057220459
C5 1.399999976158142 0.0 0.0
C6 2.130000114440918 1.3079999685287476 -0.06599999964237213
H7 1.9129999876022339 -0.8560000061988831 0.06599999964237213
H8 1.6299999952316284 2.1740000247955322 -0.06599999964237213
C9 3.630000114440918 1.3079999685287476 -0.06599999964237213
H10 4.001999855041504 1.2050000429153442 0.9760000109672546
H11 4.0289998054504395 0.4560000002384186 -0.6579999923706055
C12 4.182000160217285 2.5840001106262207 -0.6309999823570251
H13 4.570000171661377 3.2190001010894775 0.1940000057220459
C14 5.239999771118164 2.3269999027252197 -1.5110000371932983
H15 3.4030001163482666 3.1619999408721924 -1.1729999780654907
C16 6.060999870300293 3.4739999771118164 -2.0220000743865967
H17 5.466000080108643 1.3910000324249268 -1.781999945640564
H18 5.88700008392334 4.4070000648498535 -1.7070000171661377
C19 7.195000171661377 3.1989998817443848 -2.9639999866485596
H20 8.088000297546387 2.875999927520752 -2.388000011444092
H21 6.933000087738037 2.384999990463257 -3.6760001182556152
H22 7.4710001945495605 4.104000091552734 -3.546999931335449
```

B.2 Gaussian Charge Optimization Input Files

B.2.1 Acceptor Input

```
%chk=ODE-ACC-C2.chk
%nproc=1
%mem=1GB
# HF/6-31G* Opt=(Z-matrix,MaxCycles=100) Geom=PrintInputOrient

<qmtool> simtype=‘‘Geometry optimization" </qmtool>
ODE-ACC-C2

0 1
C1 -0.032023001462221146 0.18168500065803528 0.4885059893131256
H2 -0.28053799271583557 1.1330770254135132 0.9671869874000549
H3 -0.19148799777030945 -0.6192700266838074 1.2187880277633667
H4 -0.7419189810752869 0.015780000016093254 -0.3293190002441406
C5 1.380977988243103 0.1867399960756302 -0.012148999609053135
C6 2.232271909713745 1.2150779962539673 0.11348500102758408
H7 1.7262970209121704 -0.7211380004882813 -0.5093470215797424
H8 1.8892500400543213 2.1258859634399414 0.6093440055847168
C9 3.637691020965576 1.2289600372314453 -0.40772300958633423
H10 4.344244956970215 1.4332139492034912 0.4076260030269623
H11 3.892263889312744 0.238973006606102 -0.8071449995040894
C12 3.8399391174316406 2.2917490005493164 -1.501971960067749
H13 3.5858850479125977 3.2818078994750977 -1.1023969650268555
C14 5.245211124420166 2.3051509857177734 -2.023638963699341
H15 3.1330249309539795 2.0878140926361084 -2.3170900344848633
C16 6.096485137939453 3.3336689472198486 -1.899467945098877
H17 5.588099002838135 1.3938050270080566 -2.518605947494507
H18 5.751307010650635 4.24207878112793 -1.4031460285186768
C19 7.5093159675598145 3.338218927383423 -2.4006059169769287
H20 8.219548225402832 3.503972053527832 -1.5830349922180176
H21 7.7573652267456055 2.386698007583618 -2.8792760372161865
H22 7.668789863586426 4.1390509605407715 -3.1310179233551025
H1w       C5     rAH       C1   90.06       H7   90.09
  x      H1w     1.0       C5   90.00       C1    0.00
 Ow      H1w  0.9572        x   90.00       C5  180.00
H2w       Ow  0.9572      H1w  104.52        x     dih

rAH  2.0
dih  0.0
```

B.2.2 HF-Single Point Calculation File

```
%chk=ODE-sp-HF.chk
%nproc=1
%mem=1GB
# HF/6-31G* SCF=Tight

<qmtool> simtype="Single point calculation" </qmtool>
ODE-sp-HF

0 1
C1 -0.03200000151991844 0.18199999630451202 0.48899999260902405
H2 -0.2809999883174896 1.1330000162124634 0.9670000076293945
H3 -0.19099999964237213 -0.6190000176429749 1.218999981880188
H4 -0.7419999837875366 0.01600000075995922 -0.32899999618530273
C5 1.38100004196167 0.18700000643730164 -0.012000000104308128
C6 2.2320001125335693 1.215000033378601 0.11299999803304672
H7 1.7259999513626099 -0.7210000157356262 -0.5090000033378601
H8 1.8890000581741333 2.125999927520752 0.609000027179718
C9 3.638000011444092 1.2289999723434448 -0.40799999237060547
H10 4.343999862670898 1.4329999685287476 0.40799999237060547
H11 3.8919999599456787 0.23899999260902405 -0.8069999814033508
C12 3.8399999141693115 2.2920000553131104 -1.5019999742507935
H13 3.5859999656677246 3.2820000648498535 -1.1019999980926514
C14 5.244999885559082 2.305000066757202 -2.0239999294281006
H15 3.132999897003174 2.0880000591278076 -2.316999912261963
C16 6.0960001945495605 3.3340001106262207 -1.8990000486373901
H17 5.5879998207092285 1.3940000534057617 -2.5190000534057617
H18 5.750999927520752 4.242000102996826 -1.402999997138977
C19 7.508999824523926 3.3380000591278076 -2.4010000228881836
H20 8.220000267028809 3.503999948501587 -1.5829999446868896
H21 7.756999969482422 2.38700008392334 -2.878999948501587
H22 7.669000148773193 4.138999938964844 -3.13100004196167
```

B.2.3 MP2-Single Point Calculation File

```
%chk=ODE-sp-MP2.chk
%nproc=1
%mem=1GB
# MP2/6-31G* SCF=Tight Density=Current

<qmtool> simtype="Single point calculation" </qmtool>
ODE-sp-MP2

0 1
C1 -0.03200000151991844 0.18199999630451202 0.48899999260902405
H2 -0.2809999883174896 1.1330000162124634 0.9670000076293945
H3 -0.19099999964237213 -0.6190000176429749 1.218999981880188
H4 -0.7419999837875366 0.01600000075995922 -0.32899999618530273
C5 1.38100004196167 0.18700000643730164 -0.012000000104308128
C6 2.2320001125335693 1.215000033378601 0.11299999803304672
H7 1.7259999513626099 -0.7210000157356262 -0.5090000033378601
H8 1.8890000581741333 2.125999927520752 0.609000027179718
C9 3.638000011444092 1.2289999723434448 -0.40799999237060547
H10 4.343999862670898 1.4329999685287476 0.40799999237060547
H11 3.8919999599456787 0.23899999260902405 -0.8069999814033508
C12 3.8399999141693115 2.2920000553131104 -1.5019999742507935
H13 3.5859999656677246 3.2820000648498535 -1.1019999980926514
C14 5.244999885559082 2.305000066757202 -2.0239999294281006
H15 3.132999897003174 2.0880000591278076 -2.316999912261963
C16 6.0960001945495605 3.3340001106262207 -1.8990000486373901
H17 5.5879998207092285 1.3940000534057617 -2.5190000534057617
H18 5.750999927520752 4.242000102996826 -1.402999997138977
C19 7.508999824523926 3.3380000591278076 -2.4010000228881836
H20 8.220000267028809 3.503999948501587 -1.5829999446868896
H21 7.756999969482422 2.38700008392334 -2.878999948501587
H22 7.669000148773193 4.138999938964844 -3.13100004196167
```

B.3 Gaussian Hessian Input File

```
%chk=hess.chk
%nproc=1
%mem=1GB
# MP2/6-31G* Geom=(AllCheck,ModRedundant) Freq NoSymm IOp(7/33=1)
# SCF=Tight Guess=Read

B * * K
A * * * K
L * * * K
D * * * * K
B 2 1 A
B 3 1 A
B 4 1 A
B 5 1 A
B 6 5 A
B 7 5 A
B 8 6 A
B 9 6 A
B 10 9 A
B 11 9 A
B 12 9 A
B 13 12 A
B 14 12 A
B 15 12 A
B 16 14 A
B 17 14 A
B 18 16 A
B 19 16 A
B 20 19 A
B 21 19 A
B 22 19 A
A 6 5 1 A
A 7 5 1 A
A 3 1 2 A
A 4 1 2 A
A 5 1 2 A
A 4 1 3 A
A 5 1 3 A
A 5 1 4 A
A 8 6 5 A
```

```
A 9 6 5 A
A 7 5 6 A
A 10 9 6 A
A 11 9 6 A
A 12 9 6 A
A 9 6 8 A
A 13 12 9 A
A 14 12 9 A
A 15 12 9 A
A 11 9 10 A
A 12 9 10 A
A 12 9 11 A
A 16 14 12 A
A 17 14 12 A
A 14 12 13 A
A 15 12 13 A
A 15 12 14 A
A 18 16 14 A
A 19 16 14 A
A 17 14 16 A
A 20 19 16 A
A 21 19 16 A
A 22 19 16 A
A 19 16 18 A
A 21 19 20 A
A 22 19 20 A
A 22 19 21 A
D 6 5 1 2 A
D 7 5 1 2 A
D 6 5 1 3 A
D 7 5 1 3 A
D 6 5 1 4 A
D 7 5 1 4 A
D 8 6 5 1 A
D 9 6 5 1 A
D 8 6 5 7 A
D 9 6 5 7 A
D 10 9 6 5 A
D 11 9 6 5 A
D 12 9 6 5 A
D 10 9 6 8 A
D 11 9 6 8 A
D 12 9 6 8 A
D 13 12 9 6 A
D 14 12 9 6 A
```

```
D 15 12 9 6 A
D 13 12 9 10 A
D 14 12 9 10 A
D 15 12 9 10 A
D 13 12 9 11 A
D 14 12 9 11 A
D 15 12 9 11 A
D 16 14 12 9 A
D 17 14 12 9 A
D 16 14 12 13 A
D 17 14 12 13 A
D 16 14 12 15 A
D 17 14 12 15 A
D 18 16 14 12 A
D 19 16 14 12 A
D 18 16 14 17 A
D 19 16 14 17 A
D 20 19 16 14 A
D 21 19 16 14 A
D 22 19 16 14 A
D 20 19 16 18 A
D 21 19 16 18 A
D 22 19 16 18 A
```

B.4 Gaussian Torsion Scan Input File

```
%chk=BDE.scan1.pos.chk
%nproc=1
%mem=1GB
# opt=modredundant MP2/6-31g(d) Geom=PrintInputOrient

BDE Dihedral Scan at MP2/6-31G*

0 1
C1 -0.03200000151991844 0.18199999630451202 0.48899999260902405
H2 -0.2809999883174896 1.1330000162124634 0.9670000076293945
H3 -0.19099999964237213 -0.6190000176429749 1.218999981880188
H4 -0.7419999837875366 0.01600000075995922 -0.32899999618530273
C5 1.38100004196167 0.18700000643730164 -0.012000000104308128
C6 2.2320001125335693 1.215000033378601 0.11299999803304672
H7 1.7259999513626099 -0.7210000157356262 -0.5090000033378601
H8 1.8890000581741333 2.125999927520752 0.609000027179718
C9 3.638000011444092 1.2289999723434448 -0.40799999237060547
H10 4.343999862670898 1.4329999685287476 0.40799999237060547
H11 3.8919999599456787 0.23899999260902405 -0.8069999814033508
C12 3.8399999141693115 2.2920000553131104 -1.5019999742507935
H13 3.5859999656677246 3.2820000648498535 -1.1019999980926514
C14 5.244999885559082 2.305000066757202 -2.0239999294281006
H15 3.132999897003174 2.0880000591278076 -2.316999912261963
C16 6.0960001945495605 3.3340001106262207 -1.8990000486373901
H17 5.5879998207092285 1.3940000534057617 -2.5190000534057617
H18 5.750999927520752 4.242000102996826 -1.402999997138977
C19 7.508999824523926 3.3380000591278076 -2.4010000228881836
H20 8.220000267028809 3.503999948501587 -1.5829999446868896
H21 7.756999969482422 2.38700008392334 -2.878999948501587
H22 7.669000148773193 4.138999938964844 -3.13100004196167

D 6 9 12 14 S 18 10.000000
```

B.5 Sample psfgen Script

```
# Compilation of 4 Codes written by Harish Vashisth
# (c) Harish Vashisth-- October 2015--UNH
# Modified and compiled by D. Ryan Barden June 2016--UNH
# This script creates a polymer composed of PEO and PB subunits
# Only input needed from command line is number of each type. Run script as:
# vmd -dispdev text -e pb_peo_maker.pgn -args -pb [pb-blocks] -peo [peo-blocks]

#######################################
#Defining variable from arguments     #
#######################################

set nPB [lindex $argv 1]
set nPEO [lindex $argv 3]

set pbend  [lindex $argv 1]
set peoend [lindex $argv 3]

set resLast  [lindex $argv 1]
set resFirst [lindex $argv 3]

#######################################
#Generating PEO section               #
#######################################

package require psfgen
topology  ./top_patch/top_all35_ethers.rtf

segment E {

for {set i 1} {$i <= $nPEO} {incr i} {
    residue $i PEGM
}
}
coordpdb ./monomers/PEGM_noh.pdb E
regenerate angles dihedrals
guesscoord
writepdb polyPEO.pdb
writepsf polyPEO.psf
mol delete all
resetpsf
```

```
#######################################
#generating PB section                #
#######################################

package require psfgen
topology  ./top_patch/top_all36_cgenff.rtf
topology  ./top_patch/top_all35_ethers.rtf
topology  ./top_patch/BDE.top

segment B {

for {set i 1} {$i <= $nPB} {incr i} {
    residue $i BDE
}
}
coordpdb ./monomers/newBUD_carbon.pdb B

regenerate angles dihedrals
guesscoord
writepdb polyPB.pdb
writepsf polyPB.psf
mol delete all
resetpsf

#######################################
#Aligning PEO with PB                 #
#######################################

mol new     polyPB.pdb
mol addfile polyPB.psf

mol new     polyPEO.pdb
mol addfile polyPEO.psf

set pbC1  [atomselect 0 "segname B and resid 1 and name C1"]
set pbC4  [atomselect 0 "segname B and resid $pbend and name C4"]
set compb [atomselect 0 "all and segname B"]

set coor_pbC1  [measure center $pbC1  weight none]
set coor_pbC4  [measure center $pbC4  weight none]
set coor_compb [measure center $compb weight mass]
```

```
set peoC1  [atomselect 1 "segname E and resid 1 and name C1"]
set peoC2  [atomselect 1 "segname E and resid $peoend and name C2"]
set compeo [atomselect 1 "all and segname E"]

set coor_peoC1  [measure center $peoC1  weight none]
set coor_peoC2  [measure center $peoC2  weight none]
set coor_compeo [measure center $compeo weight mass]

set dist_com_c4_pb  [vecdist $coor_pbC4 $coor_compb]
set dist_c1_com_peo [vecdist $coor_compeo $coor_peoC1]

set req_dist [expr "$dist_com_c4_pb + $dist_c1_com_peo + 2.0"]
set vec_shift [vecscale $req_dist [vecnorm [vecsub $coor_pbC4 $coor_compb]]]

set new_com_peo [vecadd $coor_compb $vec_shift]
set offset [vecsub $new_com_peo $coor_compeo]

set sel_peo [atomselect 1 "all"]
$sel_peo moveby $offset

set moved_peo [atomselect 1 "all"]
$moved_peo writepdb moved_polyPEO.pdb

mol delete all
resetpsf

########################################
#Combining the two chains              #
########################################

package require psfgen
topology  ./top_patch/top_all36_cgenff.rtf
topology  ./top_patch/top_all35_ethers.rtf
topology  ./top_patch/BDE.top
topology  ./top_patch/patch.top

readpsf    polyPB.psf
coordpdb   polyPB.pdb

readpsf    polyPEO.psf
coordpdb   moved_polyPEO.pdb
```

```
patch PBEO B:${resLast} E:1
regenerate angles dihedrals
guesscoord

writepdb pb${resLast}peo${resFirst}.pdb
writepsf pb${resLast}peo${resFirst}.psf

########################################
#Keep it clean                         #
########################################

mv pb${resLast}peo${resFirst}.pdb ./single_chains_pb_peo/
mv pb${resLast}peo${resFirst}.psf ./single_chains_pb_peo/

rm moved_polyPEO.pdb
rm polyPB.pdb
rm polyPB.psf
rm polyPEO.pdb
rm polyPEO.psf

exit
```

B.6 merge.tcl

```
##########################################################################
# This script generates a hybrid membrane of pb12peo9 BCPs and POPC      #
# lipids by combining two pristine membrane structures with the same area. #
##########################################################################

#define variables from the membrane file and pap channel
set popc popc_mem
set bcp pb12peo9_eq

#call popc files in vmd and center
mol new $popc.psf
mol addfile $popc.pdb
set lip [atomselect top all]

#Center and identify desired lipids
$lip moveby [vecinvert [measure center $lip]]
$lip moveby {0 0 30}
$lip writepdb popc_TEMP_center.pdb
$lip set beta 1
set lip_keep [atomselect top "segid L11 or segid L12 or segid L13 or segid L14"]
$lip_keep set beta 0
set badlipid [atomselect top "beta > 0"]
set seglistlipid [$badlipid get segid]
set reslistlipid [$badlipid get resid]

#Loop generating file with desired lipid membrane half
mol delete all
package require psfgen
resetpsf
readpsf $popc.psf
coordpdb popc_TEMP_center.pdb

foreach segid $seglistlipid resid $reslistlipid {
delatom $segid $resid
}

writepsf popc_half.psf
writepdb popc_half.pdb
```

```
#call pb files in vmd and center
mol delete all
mol new $bcp.psf
mol addfile $bcp.pdb
set pb [atomselect top all]

#Center and identify desired pb
$pb moveby [vecinvert [measure center $pb]]
$pb writepdb bcp_TEMP_center.pdb
$pb set beta 1

set top {}
set B [atomselect top "chain B"]

for {set j 0 } { $j < 99 } { incr j } {
    set seltext "segname B${j} or segname E${j}"
    set pb_keep [atomselect top $seltext]
    $pb_keep set beta 0
}

set badpb [atomselect top "beta > 0"]
set seglistpb [$badpb get segid]
set reslistpb [$badpb get resid]

#Loop generating file with desired bcp membrane half
mol delete all
package require psfgen
resetpsf
readpsf $bcp.psf
coordpdb bcp_TEMP_center.pdb

foreach segid $seglistpb resid $reslistpb {
delatom $segid $resid
}

writepsf bcp_half.psf
writepdb bcp_half.pdb
```

```
#combining the temporary files
mol delete all
package require psfgen
resetpsf
readpsf popc_half.psf
coordpdb popc_half.pdb
readpsf bcp_half.psf
coordpdb bcp_half.pdb

writepsf popc_pb12_peo9_raw.psf
writepdb popc_pb12_peo9_raw.pdb

exit
```

B.7 Density of Species Script

```
# This script was used to perform the density of species calculation
# for PB-PEO BCP membranes.

mol load dcd chop.dcd psf solvated.psf

set nf [molinfo top get numframes]

set dz 2
set ddz [expr $dz./2.]
set zt 120
set lw 80
set vol [expr $lw*$lw*$dz]

set jt [expr $zt+1]
set off [expr $zt./2.]
set norm [expr $nf*$vol]

#########################################################
#### Loop over all frames and align COM.         ####
#########################################################

for {set i 0} {$i < $nf} {incr i} {

    puts "frame $i of $nf"
    set all [atomselect top "all" frame $i]
    set mem [atomselect top "lipid or chain B or chain E" frame $i]
    $all moveby [vecinvert [measure center $mem weight mass]]

#########################################################
#####             Loop over segments            #####
#########################################################

    for {set j 0} {$j < $jt} {incr j $dz} {

        set zcen [expr $j-$off]
        set zmin [expr $zcen-$ddz]
        set zmax [expr $zcen+$ddz]

        puts "j is $j z is $zcen bin is $zmin < z < $zmax"
```

```
        set sellip [atomselect top "lipid and z<$zmax and z>$zmin" frame $i]
        set selpb  [atomselect top "chain B and z<$zmax and z>$zmin" frame $i]
        set selpeo [atomselect top "chain E and z<$zmax and z>$zmin" frame $i]
        set selwat [atomselect top "water and z<$zmax and z>$zmin" frame $i]
        set selpap [atomselect top "chain A and z<$zmax and z>$zmin" frame $i]

        set denlip($i.$j) [llength [$sellip get index]]
        set denpb($i.$j)  [llength [$selpb get index]]
        set denpeo($i.$j) [llength [$selpeo get index]]
        set denwat($i.$j) [llength [$selwat get index]]
        set denpap($i.$j) [llength [$selpap get index]]

    }
}

#########################################################
#####               Average segments            #####
#########################################################

set outfile1 [open den_lipid.dat w]
set outfile2 [open den_pb.dat w]
set outfile3 [open den_peo.dat w]
set outfile4 [open den_wat.dat w]
set outfile5 [open den_pap.dat w]

for {set j 0} {$j < $jt} {incr j $dz} {

    puts "averaging bin $j"

    set lipsum($j) 0
    set  pbsum($j) 0
    set peosum($j) 0
    set watsum($j) 0
    set papsum($j) 0

    for {set i 0} {$i < $nf} {incr i} {

        set lipsum($j) [expr $lipsum($j) + $denlip($i.$j)]
        set  pbsum($j) [expr $pbsum($j) + $denpb($i.$j)]
        set peosum($j) [expr $peosum($j) + $denpeo($i.$j)]
        set watsum($j) [expr $watsum($j) + $denwat($i.$j)]
        set papsum($j) [expr $papsum($j) + $denpap($i.$j)]

    }
```

```
    puts $outfile1 "[expr $j-$off] [expr $lipsum($j)/$norm.]"
    puts $outfile2 "[expr $j-$off] [expr $pbsum($j)/$norm.]"
    puts $outfile3 "[expr $j-$off] [expr $peosum($j)/$norm.]"
    puts $outfile4 "[expr $j-$off] [expr $watsum($j)/$norm.]"
    puts $outfile5 "[expr $j-$off] [expr $papsum($j)/$norm.]"
}

close $outfile1
close $outfile2
close $outfile3
close $outfile4
close $outfile5

exit
```

B.8 Root Mean Squared Displacement

```
# This script was used to calculate the RMSD of the PAP channel in all
# membrane environments.

mol load dcd ../pap_vac.dcd psf ../channelA.psf

set outfile [open rmsd_PAP.dat w];
set nf [molinfo top get numframes]
set frame0 [atomselect top "chain A" frame 0]
set sel [atomselect top "chain A"]
set dt 0.02
# rmsd calculation loop
for {set i 0 } {$i < $nf } { incr i } {
    $sel frame $i
    $sel move [measure fit $sel $frame0]
    puts $outfile "[expr "$i*$dt"]  [measure rmsd $sel $frame0]"
}
close $outfile
mv rmsd_PAP.dat ../data_files/
exit
```

B.9 PAP Orientation

```
# This script was used to measure the orientation angle of the PAP channel
# in all membrane environments.

mol load dcd ../pap_vac.dcd psf ../channelA.psf
set outfile [open angle_pap.dat w];
set nf [molinfo top get numframes]

set seltop "chain A and (name C32 or name C36 or name C31 or name C35 or name C4
or name C30 or name C34 or name C29 or name C33 or name C5)"

set selbot "chain A and (name C1 or name C2 or name C17 or name C21 or name C18
or name C22 or name C19 or name C23 or name C20 or name C24)"

set top [atomselect top $seltop]
set bot [atomselect top $selbot]
set PI 3.14159265359
set dt 0.02

for {set i 0 } {$i < $nf } { incr i } {

    set xy {}
    $top frame $i
    $bot frame $i
    set tCOM [measure center $top]
    set bCOM [measure center $bot]
    set norm [vecsub $tCOM $bCOM]
    set x [lindex $norm 0]
    set y [lindex $norm 1]
    lappend xy $x
    lappend xy $y
    lappend xy 0
    set n [veclength $norm]
    set ij [veclength $xy]
    set sin [expr $ij/$n]
    set theta [expr asin($sin)]
    puts $outfile "[expr "$i*$dt"]  [expr $theta*180/$PI]"
}
close $outfile
exit
```

B.10 PAP Diffusivity

```
# This script was used to measure the displacement of the PAP channel
# in all membrane environments.

mol new ../unwrap.dcd waitfor all autobonds off
mol addfile ../solvated.psf

set nf [molinfo top get numframes]
set seltxt "chain A"
set sel1 [atomselect top $seltxt]
set outfile1 [open pap_trajectory.dat w];
set slist1 {}
set system [atomselect top "all"]
set dt 0.02

for {set i 0 } {$i < $nf } { incr i } {

       $sel1 frame $i
       $system frame $i

       set com_system [measure center $system weight mass]
       set com1_c [measure center $sel1 weight mass]
       set com1_r [vecsub $com1_c $com_system]

       puts $outfile1 "[expr "$i*$dt"]  $com1_r"
   }
close $outfile1

mol delete all
mv pap_trajectory.dat ../data_files/

exit
```

B.11 Channel Permeability

```
# This script was used to compute permeability using collective diffusion
# model of Zhu et al. (2004) Physical review letters 93.22: 224501

mol load psf ../solvated.psf
mol addfile ../realigned_aligned_npt03.dcd waitfor all autobonds off

set nf [molinfo top get numframes]
set outfile1 [open n_trajectory_hv.dat w];
set outfile2 [open n_water.dat w];

set seltxt "chain A and (name C35 or name C31 or name C27 or name C23 or name C19 or name C39 o

set vol [atomselect top $seltxt]
set all [atomselect top all]

set COM [measure center $vol weight mass]

puts "Center of chosen volume: $COM"

set x [lindex $COM 0]
set y [lindex $COM 1]
set z [lindex $COM 2]

set zmax [expr $z+4]
set zmin [expr $z-4]

puts "x:$x y:$y z:$z zmax:$zmax zmin: $zmin"

set seltxt1 "water and (sqrt(sqr(x-0.0)+sqr(y-0.0))<3 and z<$zmax and z>$zmin)"
puts "seltxt1: $seltxt1"

set sel [atomselect top $seltxt1]
set nt 0
```

```
################################################
# n(t) code                                    #
################################################

for {set i 1 } {$i < $nf } { incr i } {


    $sel frame $i
    $sel update

# Get indices of all water atoms in selection, their resids and segnames
    set sel1 [$sel get index]
    puts "sel1 contains: $sel1"
    puts "sel1: [llength $sel1]"
    set wresids   [$sel get resid]
    set wsegnames [$sel get segname]


#This block of code finds unique resids and segnames of each
#whole water molecule in volume

    set pair {}
    set newpair {}
    set counter 1

   for {set w 0 } {$w < [llength $sel1] } { incr w 1} {

       set r  [lindex  $wresids $w]
       set s  [lindex  $wsegnames $w]

      # grow the list pair with resid and segname grouped for each index
       set temp {}
       lappend temp $r
       lappend temp $s
       lappend pair $temp
      # Compare if each group in the list pair is repeated at least
      # three times. This ensures that only those water molecules whose
      # three atoms were found are counted.
       if {$w != 0} {
           if {[lindex $pair $w] == [lindex $pair [expr $w-1]] } {
               set counter [expr $counter+1]
           }
       }
```

```
       if {$counter == 3 } {
         lappend newpair [lindex $pair $w]
         set counter 1
       }
   }

##sort the list newpair for unique ids although it is already unique, i think-----
   puts "pair: $pair "
   puts "unique pairs: [lsort -unique $pair]"
   puts "new pair: $newpair "
   puts "new unique pairs: [lsort -unique $newpair]"

# Separate each group in the list newpair to get resids and segnames
   set uniquewresids {}
   set uniquewsegnames {}

   foreach pr [lsort -unique $newpair] {
      lappend uniquewresids   [lindex  $pr 0]
      lappend uniquewsegnames [lindex  $pr 1]
    }

# We need to cycle only through unique whole waters for dz, so get a counter m

    set m [llength [lsort -unique $newpair]]

    puts "No. of full water molecules in volume at frame $i: $m"
    puts "resids of full waters found are: $uniquewresids"
    puts "segnames of full waters found are: $uniquewsegnames"
    puts "Check on list sizes  resids: [llength $uniquewresids] segnames:
    [llength $uniquewsegnames]"


    set sum 0
    set dt 0.02
```

```
 if 1 {

    for {set n 0 } {$n < $m } { incr n} {

        set watersel [atomselect top "water and resid [lindex $uniquewresids $n]
        and segname [lindex $uniquewsegnames $n]"]
        set waterresid   [lsort -unique [$watersel get resid]]
        set watersegname [lsort -unique [$watersel get segname]]
        puts "At frame $i info of water [expr $n+1] resid: $waterresid  segname:
        $watersegname"

        set watcurr [atomselect top "water and resid $waterresid and segname
        $watersegname" frame $i]
        set watprev [atomselect top "water and resid $waterresid and segname
        $watersegname" frame [expr $i-1]]

        set wcomcurr [measure center $watcurr weight mass]
        set wcomprev [measure center $watprev weight mass]

        puts "wcomcurr: $wcomcurr wcomprev: $wcomprev"

        set zcurr [lindex $wcomcurr 2]
        set zprev [lindex $wcomprev 2]

        puts "zcurr: $zcurr zprev:$zprev"

# Check if any chosen water within our volume has current z outside the range
# -4.0 to +4.0

        if { ($zcurr >=$zmax || $zcurr <=$zmin) } {
         puts "CATASTROPHE: Problem reported at frame $i for $waterresid segname
         $watersegname zcurr: $zcurr"
        }


        set tempdist [expr $zcurr-$zprev]
        set dzmax [expr $zcurr-$zmax]
        set dzmin [expr $zcurr-$zmin]
```

```
        if {$zprev >= $zmax} {
             puts "water on top was outside"
            set dz $dzmax
        } elseif {$zprev <= $zmin} {
            puts "water on bottom was outside"
            set dz $dzmin
        } else {
             puts "water is still inside"
             set dz $tempdist
        }

        set sum [expr $sum+$dz]

    }

      puts "At frame $i sigma dz is: $sum"
      set dn [expr $sum/8.0]
      puts "At frame $i dn is: $dn"
      set nt [expr $nt + $dn]
      puts "At frame $i nt is: $nt"
      puts $outfile1 "[expr "$i*$dt"]  $dn  $nt  0  0"
      puts $outfile2 "[expr "$i*$dt"]  $m"

      puts "Finished frame $i of $nf"

}

}


 close $outfile1
 close $outfile2

exit
```

Appendix C

Mini Tutorials

C.1 Molefacture

The most efficient way to build polymers of any size is to first build the corresponding monomer in molefacture. Molefacture is better suited to building relatively small and simple molecules. It is natural then to first build the monomer, which will act as a building block for generating polymer chains. This monomer can then be used with `tcl` scripts to automatically generate polymer chain lengths of any size. This is much faster than building a complete polymer in molefacture every time a new chain is desired. It will become apparent that using molefacture can be time-consuming.

To begin Molefacture, launch VMD [30] and navigate to the *Extensions* tab. Under the *Modeling* subsection, find *Molefacture*. Click *Start Molefacture* to continue. This will bring the main graphical user interface(GUI) for Molefacture. Start your molecule by navigating to *build* $\rightarrow$ *New molecule from fragment*, and select a base molecule from the drop down menu. For example, to build the monomer sub-unit of polybutadiene, we will select butane. The new structure will appear in the VMD [30] display. The monomer for polybutadiene is essentially 2-butene. This makes the build relatively straight forward. Now that we have our base structure, we need to delete one hydrogen atom from the carbon atoms at positions 2 and 3. This can be done by clicking on the hydrogen atom in the VMD display, or highlighting it in the atom list and then pressing "Delete Selected Atom". When an atom is selected, it is highlighted with an orange sphere. If you are building a complex molecule, you can replace hydrogens with small organic groups. After selecting the hydrogen you wish to replace, simply select a fragment from the list located under *build* $\rightarrow$ *Replace hydrogen with fragment*. If the wrong hydrogen has been deleted, the user can replace it by selecting the atom it was bonded to and clicking "Add hydrogen to selected atom" from the build menu or the GUI. This may be helpful when building larger and more complex molecules.

Next, we need to convert the single bond between the second and third carbon to a double bond. Under the Bonds section of the GUI, select the bond for carbon atoms 2 and 3. The two orange spheres signify the bonded atoms, while the two yellow spheres signify the other two atoms that make up the dihedral group. After raising the bond order, the purple markers signifying empty orbitals should be gone, while the bond should appear to have a cylinder surrounding a portion of it representing the new double bond. The last thing we should do before moving on is to delete one hydrogen atom from each terminal carbon. We are doing this because we are modeling a single unit of a polymer. The empty orbitals will be addressed in section C.1.1 when we build a patch.

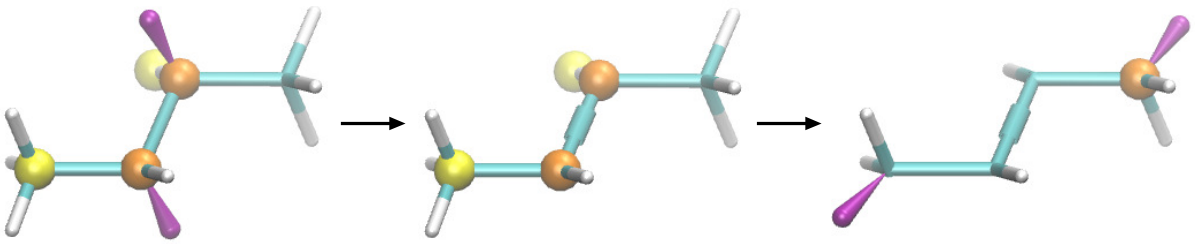


FIGURE C.1: Snapshots of modifications to the 2-butene fragment in Molefacture

The most important part of building a molecule in molefacture is manually editing each atom in the atom list. This is done by highlighting an atom and clicking the "Edit selected atom" button on the GUI. This will open a small window containing 4 entries. We will need to change the name, type, and charge for each atom. A good strategy from naming atoms is to work left to right naming all non-hydrogen atoms with their periodic table symbol followed by a unique number. If only a single atom of a given element appears in the structure, the number is unnecessary. Larger and more complex structures may require a different strategy but this approach is effective for small monomers. For our monomer, we will name our carbons C1, C2, C3, and C4. Hydrogens are named using their periodic table symbol and 2 additional numbers. The first corresponds to the number of the carbon its bonded to, while the second is a unique identifying number. For example, the three hydrogens bonded to carbon C1 will be named H11, H12, and H13. This method will have problems if there are hydrogens bonded to elements other than carbon in your structure. A suggested fix is to replace the first number with the name of the atom to which it is bonded. For example, if there was a nitrogen named N1, name the hydrogen bonded to it as HN11. Every atom in your structure needs a unique name for the purposes of applying patches and generating PSF files.

Atom types are determined by the orbital bond structure. When editing the atom type and charge, you will need to browse structures previously defined in existing topology files. It is important to stay consistent when assigning atom types as they are not arbitrary. Atoms types are what specify force-field parameters. In the `top_all_36cgenff.rtf` topology file, there is an entry for 2-hexene which is included below. This molecule has all the information we will need to define our monomer as well as the patches we will build in Section C.1.1.

```
RESI HXE2         0.00 ! C6H12 2-hexene, yin/adm jr.
GROUP
ATOM  C1  CG331  -0.27 !  H12 H13
ATOM  H11 HGA3    0.09 !    \ |
ATOM  H12 HGA3    0.09 ! H13-C1    H31 H51 H52
ATOM  H13 HGA3    0.09 !       \    /    \ /
GROUP                  !        C2=C3    C5    H61
ATOM  C2  CG2D1  -0.15 !       /     \  /  \  /
ATOM  H21 HGA4    0.15 !     H21      C4    C6-H62
GROUP                  !             / \      \
ATOM  C3  CG2D1  -0.15 !           H41 H42     H63
ATOM  H31 HGA4    0.15 !
GROUP
ATOM  C4  CG321  -0.18
ATOM  H41 HGA2    0.09
ATOM  H42 HGA2    0.09
GROUP
ATOM  C5  CG321  -0.18
ATOM  H51 HGA2    0.09
ATOM  H52 HGA2    0.09
GROUP
ATOM  C6  CG331  -0.27
ATOM  H61 HGA3    0.09
ATOM  H62 HGA3    0.09
ATOM  H63 HGA3    0.09
```

Atoms C2 and C3 will be assigned atom types CG2D1 with a charge of -0.15 as they are defined above. The hydrogens bonded to those atoms should be assigned types consistent with those described as well. We need to keep in mind that we are defining atom types for a segment of a polymer. When assigning atom type to the C1 and C4 atoms in our structure, we will use the information for the C4 atom in 2-hexene. We are assigning these atom types because this structure should be modeled after a single unit appearing within a polymer. Once all of the atoms have been properly identified, click on the "Edit segname/resname/chain" button. We will change the entry for resname to "BDE" and chain to "B". Click the "Apply" button followed by "Done". Navigate to $File \rightarrow write\ top\ file$ and $File \rightarrow write\ psf\ and\ pdb\ files$ to generate the necessary outputs. I suggest using the resname of the structure as the prefix for all of the output files. An example of the resulting topology file can be found in Section C.1.1.

C.1.1 Patches and Topology File Modifications

Patches are entries that appear at the end of topology files. They contain information that allows larger structures to be built from multiple residues or create modified residues. This is done with the psfgen tool included in VMD [30]. A brief description of how to utilize the psfgen feature is outlined in Section C.2, while a more comprehensive tutorial can be found online. In this section,

we describe how to manually build a patch and edit the topology file we created using molefacture. A copy of the original topology file generated by Molefacture is presented below. This is followed by the full modified file and a numbered list containing detailed information of all the changes made to the file. The modified file contains numbered entries next to the new/modified lines. The numbers correspond to the detailed description in the list of changes.

Original Topology File

```
*>>>>>> CHARMM topology file generated by Molefacture <<<<<<
27 1

MASS     1 CG321  12.01100  C
MASS     1 HGA2   1.00794  H
MASS     1 CG2D1  12.01100  C
MASS     1 HGA4   1.00794  H

AUTO ANGLES DIHE

RESI  BDE     0.00
GROUP
ATOM   C1 CG321 -0.18000
ATOM  H11 HGA2  0.09000
ATOM  H12 HGA2  0.09000
ATOM   C2 CG2D1 -0.15000
ATOM  H21 HGA4  0.15000
ATOM   C3 CG2D1 -0.15000
ATOM  H31 HGA4  0.15000
ATOM   C4 CG321 -0.18000
ATOM  H41 HGA2  0.09000
ATOM  H42 HGA2  0.09000
BOND C1 H11  C1 H12  C1 C2
BOND C2 H21  C2 C3  C3 H31  C3 C4
BOND C4 H41  C4 H42

END
```

Modified Topology File

```
*>>>>>> Modified CHARMM topology file generated by Molefacture <<<<<<
27 1

MASS     1 CG321  12.01100  C
MASS     1 CG331  12.01100  C  ! 1. New mass entry for atom type CG331.
MASS     1 CG2D1  12.01100  C
MASS     1 HGA2   1.00794  H
MASS     1 HGA4   1.00794  H
MASS     1 HGA3   1.00794  H   ! 2. New mass entry for atom type HGA3.

AUTO ANGLES DIHE
DEFAULT FIRST HBD1 LAST HBD4   ! 3. Line specifying the default patches
                               ! applied to the residue.
RESI  BDE     0.00
GROUP                          ! 4. Map of the atoms.
ATOM  C1  CG321  -0.18000 !   H11 H12
ATOM  C3  CG2D1  -0.15000 !     \ |
ATOM  H11 HGA2    0.09000 ! (C4)-C1      H31
ATOM  H12 HGA2    0.09000 !        \     /
ATOM  C2  CG2D1  -0.15000 !         C2=C3
ATOM  H21 HGA4    0.15000 !        /     \
ATOM  H31 HGA4    0.15000 !      H21      C4-(C1)
ATOM  C4  CG321  -0.18000 !               | \
ATOM  H41 HGA2    0.09000 !             H41 H42
ATOM  H42 HGA2    0.09000 !
BOND C1 H11  C1 H12  C1 C2
BOND C3 C2  C3 H31  C3 C4  C2 H21
BOND C4 H41  C4 H42
BOND C1 -C4                    ! 5. Bond connecting two monomers
                               ! 6. Internal coordinates table
IC -C3 -C4 C1  C2     0.000  0.000  180.0  0.000  0.000
IC -C4  C1 C2  C3     0.000  0.000  180.0  0.000  0.000
IC  C1  C2 C3  C4     0.000  0.000  180.0  0.000  0.000
IC  C2  C3 C4 +C1     0.000  0.000  180.0  0.000  0.000
```

```
                                ! 7. Two new patches for terminal residues
PRES HBD4            0.00  ! Complete terminal methyl at C4
                           !
ATOM H43   HGA3     0.09   !
ATOM C4    CG331   -0.27   !
ATOM H41   HGA3     0.09   !
ATOM H42   HGA3     0.09   !
BOND H43 C4

PRES HBD1            0.00  ! Complete terminal methyl at C1
                           !
ATOM H13   HGA3     0.09   !
ATOM C1    CG331   -0.27   !
ATOM H11   HGA3     0.09   !
ATOM H12   HGA3     0.09   !
BOND H13 C1

END
```

List of Modifications

1. New mass entry for atom type CG331: Patches are essentially used to change existing atom types, add new atoms/bonds, and delete existing atoms/bonds. Our terminal carbon atoms are a different atom-type than those we defined for our interior. We need to define mass for this terminal carbon atom-type (CG331). This atom type can be seen in the topology for 2-hexene included in Section C.1.

2. New mass entry for atom type HGA3: Similar to the carbon atom, we need to define mass for the hydrogens with atom-type HGA3. This atom type can be seen in the topology for 2-hexene included in Section C.1.

3. Line specifying the default patches applied to the residue: The psfgen tool in VMD [30] builds structures in multiple segments, where several residues make up each segment. This line will instruct psfgen to automatically apply patch HBD1 to a BDE residue if it is the first residue in the segment. The line also instructs psfgen to apply patch HBD4 to a BDE residue if it is the last residue on a segment. Both of these patches are defined in item 7.

4. Map of the atoms: This addition is optional and will not be read by any software. It is advisable as it will give anyone who is viewing the file a clear picture of which atoms are connected.

5. Bond connecting two monomers: We created our monomer with the intent of making a larger polymer. Adding this bond will instruct psfgen to connect the C4 atom of a BDE residue with the C1 atom of the next BDE residue. The "-" appearing prior to C4 informs psfgen that

these atoms are not on the same residue. If the "-" was not present, psfgen would erroneously create a bond between the C1 and C4 atom of the same residue.

6. Internal coordinates table: As residues are added to a segment, psfgen will guess their positions. You will likely end up with an initial residue that looks good, while the rest will have many overlapping atoms. The internal coordinates table will give psfgen instructions on how to place atoms relative to one another. This requires some trial and error to get it right, but the result does not have to be perfect. Atoms just need to be separated from one another. Energy minimization prior to simulation will adjust any unrealistic bond lengths.

7. Two new patches for terminal residues: Patches are identified by the heading PRES, short for patch residue. On the same line as the heading is the patch name and the overall charge of the patch. Our patches are named HBD4 and HBD1. The HBD4 patch will be applied to the C4 carbon while the HBD1 will be applied to the C1 carbon. The patches will override the information contained in the main RESI entry. In the HBD4 patch, we can see that the C4, H41, and H42 atoms will be assigned new atom types. There is also a new atom, H43, defined to cap the terminal carbon. The last line of the patch defines the bond between the C4 atom and the new H43 atom. These patches were modeled after the C1 atom in the topology for 2-hexene included in section C.1.

Patches Connecting Two Different Polymer Segments

To combine two different types of polymer segments, a unique patch must be created for that combination. In this section, we use the example of a patch created to combine a segment of polybutadiene (PB) with a segment of polyethyleneoxide (PEO). It is convenient to store patches of these type together in a separate topology file. The topology file can be appended as new polymers are added to your structure library. These patches start just as before with the heading PRES followed by the unique patch name and charge. In this example, our patch is given the name PBEO and carries a charge of 0.17. An important difference to note in these patches is the number in front of the atom name. It is likely that two different residues will contain atoms with the same name. To ensure the patch works as intended, each residue is identified by a number. In this example, our PB monomer is denoted with a 1 while the PEO monomer is denoted with a 2. After the new atom types are specified, the next two lines instruct psfgen to delete a hydrogen atom from each of the residues to make space for the new bond. The new bond is defined in the next line followed by an internal coordinates table.

```
PRES PBEO    0.17

GROUP                        !    H41  H1A
ATOM 1C4  CG321 -0.18        !     |    |
ATOM 1H41 HGA2   0.09        !-C3--C4---C1-O1
ATOM 1H42 HGA2   0.09        !     |    |
GROUP                        !    H42  H1B
ATOM 2C1  CC32A -0.01        !
ATOM 2H1A HCA2A  0.09        !
ATOM 2H1B HCA2A  0.09        !

DELE ATOM 1H43
DELE ATOM 2H1C

BOND 1C4 2C1

IC 1C2  1C3  1C4  2C1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C3  1C4  2C1  2O1   0.0000  0.0000  180.0000  0.0000  0.0000
IC 1C4  2C1  2O1  2C2   0.0000  0.0000  180.0000  0.0000  0.0000
```

C.2 psfgen

Now that we have made the necessary changes to our topology file and created the proper patches, we can use psfgen to create polymer chains of any desired length. In this section we will outline how to use psfgen to create our polymers. A `tcl`-script that will generate a polybutadiene and polyethyleneoxide block copolymer can be found in appendix B.5. Examples contained in this section are excerpts from that psfgen script.

The first step in creating a block copolymer is to create each polymer separately. The first line in the example below tells VMD [30] that we wish to use the psfgen package. The lines beginning with `topology` instruct psfgen to read the topology file listed. These files contain the information required to build the desired structure. The next line instructs psfgen that we would like to build a new segment named "B". The brackets should contain a numbered list of each of the residues in the segment defined by a residue name. This example has been adopted from an automated script so that the list is replaced with a `for` loop. In the `for` loop, the variable `nPB` denotes the length of the polymer chain. The command `coordpdb` tells VMD [30] to use the pdb file specified as a template to set coordinates for the segment specified. The command `regenerate angles dihedrals` tells psfgen to generate all the angle and dihedral terms in the structure from the bond information contained within the topology file. The last lines instruct psfgen to guess the coordinates of missing atoms in the structure, write the specified pdb/psf pair for the new structure, remove the structure from VMD [30], and clear all topology information from psfgen.

```
#######################################
#generating PB section                #
#######################################

package require psfgen
topology  ./top_patch/top_all36_cgenff.rtf
topology  ./top_patch/top_all35_ethers.rtf
topology  ./top_patch/BDE.top

segment B {

for {set i 1} {$i <= $nPB} {incr i} {
    residue $i BDE
}
}
coordpdb ./monomers/newBUD_carbon.pdb B

regenerate angles dihedrals
guesscoord
writepdb polyPB.pdb
writepsf polyPB.psf
mol delete all
resetpsf
```

Once we have created two different polymers of the desired length, we combine them into a single block copolymer. Before doing this, it is necessary to load the two structures into VMD [30] and move them so that they are properly aligned. It is best if the two residues which are to be patched together are relatively close to one another. This will guarantee that the new bond connecting the structure can quickly be fixed during minimization. A new PDB of the polymer that has been moved should be generated before moving forward. An example of how to automate the alignment process can be seen in Appendix B.5. To combine the structures, the user should first call psfgen as before. The user then needs to load the topology files containing all structure and patch information needed. The user then instructs psfgen to read the PSF file of each of the polymers and store the coordinates from the desired PDB files. The command `patch`instructs psfgen to apply the desired patch. In the example below, psfgen is instructed to apply the patch PBEO to the last residue of segment B with the first residue of segment E. Note that this example was taken from an automated script, `${resLast}` calls the stored variable `reslast`. For instance, if we wanted to connect a PB segment with 12 units to a PEO segment, reslast would be 12. The remaining lines instruct psfgen to generate all the angle and dihedral terms in the structure, guess any unspecified atomic coordinates, and write the PDB/PSF pair of the finished block copolymer structure.

```
#####################################
#Combining the two chains           #
#####################################

package require psfgen
topology  ./top_patch/top_all36_cgenff.rtf
topology  ./top_patch/top_all35_ethers.rtf
topology  ./top_patch/BDE.top
topology  ./top_patch/patch.top

readpsf    polyPB.psf
coordpdb   polyPB.pdb

readpsf    polyPEO.psf
coordpdb   moved_polyPEO.pdb

patch PBEO B:${resLast} E:1
regenerate angles dihedrals
guesscoord

writepdb pb${resLast}peo${resFirst}.pdb
writepsf pb${resLast}peo${resFirst}.psf
```

C.3 Force Field Toolkit (ffTK)

As mentioned in the beginning of Section 3.3, the size of structures should be kept to a minimum when generating new parameters. Dimers should be sufficient for most polymers, however there are exceptions. In Sections C.3 through C.3.4, we will walk through an example of generating the missing parameters for a polybutadiene dimer. Before launching VMD [30], create a new directory with the following sub-directories: 1-sysPrep, 2-geomOpt, 3-chargeOpt, 4-baOpt, and 5-dihOpt to keep files organized. To access Force Field Toolkit [42] (ffTK), launch VMD [30] and navigate to *Extensions* → *Modeling* → *Force Field Toolkit*. This will bring up the ffTK [42] graphical user interface (GUI). Under the "BuildPar" tab, expand the "Identify Missing Parameters" section. Add the PDB/PSF pair and any existing parameter files using the appropriate file dialogs. Once you have done this, click on "Analyze". This will populate the four sections below with any missing parameters. For the polybutadiene dimer, we are only missing one dihedral angle (see Fig. D.2).

All other bonds, angles and non-bonded parameters are defined in the CHARMM36 [31] force field. This will not be typical of all dimers. There will often be missing bond and angle parameters. Non-bonded parameters should always be defined as they are unique to each atom type. Instructions on how to generate and optimize bond/angle parameters are included in this section even though we will not need to do them for this particular example. The ffTK tutorial [45] located at `www.ks.uiuc.edu/Training/Tutorials/` contains detailed instructions on how to complete those steps.

Now that we have identified our missing parameters, we need to generate an output parameter file that will contain our newly developed parameters. Click "SaveAs" next to the "Output PAR File" file dialogue box. Choose a location for the parameter file and give it a name of your choice with the extension .par.

C.3.1 Geometry Optimization

In this section we will use Gaussian [43] to optimize the geometry of our structure. This is an important step as it corrects any unrealistic structures created by psfgen. If this step is skipped, any Gaussian [43] runs to calculate bonds/angles and dihedrals will likely fail. Give ffTK [42] the location of your original PDB file and create a new Gaussian [43] output file by clicking "SaveAs" next to the "Output GAU File" file dialogue box. Save this file as geomOpt.gau in the 2-geomOpt directory. An example of the guassian input file generated by ffTK [42] is included Appendix B.1.

All the necessary output information, including the optimized geometry, will be written to the log file geomOpt.log. Populate the file dialogue boxes in the "Write Updated PDB" section with the necessary file locations. Click "Load Gaussian LOG File" to see the geometry optimized structure in the visualization window. After inspection, click "Write Optimized Geometry to PDB" to create the new PDB file. Use the geometry optimized PDB file in the remaining steps.

C.3.2 Charge Optimization

Under the "Water Int." tab (Fig. D.4), populate the necessary file dialogue boxes in the "Input/Output" section. Click "Load PSF/PDB" followed by "Basename From TOP" to load the PDB/PSF pair into ffTK [42] and set the base-name. After this is done, click "AutoDetect Indices" under the "Hydrogen Bonding Atoms" section. This will generate a list of donor and acceptor atoms in your structure. Click the "Write Gaussian Input Files" button to generate all of the charge optimization files. There will be a file for each of the atoms in the donor and acceptor lists as well as files called wat-sp.gau, ODE-sp-HF.gau, . If you wish to visualize the water interaction sites, click "Load GAU Files". An example of each of these files can be seen in Appendix B.2. Each of these files were run in Gaussian [43], again directing the output to a log file with the same prefix as the input. If you wish to visualize the results, click "Load LOG files", otherwise continue on to the next step on charge optimization.

Under the "Opt. Charges" tab, expand the "Input" section. Specify the location of your PDB/PSF files. Click "Load PSF/PDB" followed by "Resname From TOP". Add any parameter files you may have, including the new file generated for the molecule. Under the "Charge Constraints" section (Fig. D.6), click the "Guess" button to load all of the charge groups. Highlight any groups of hydrogens with a charge of +0.09 and remove them by pressing "Delete". Set the "Net Charge" to 0 and press "Calculate from PSF". Under the "QM Target Data" section (Fig. D.7), populate the file dialogue boxes with all the necessary files.

There is no need to make changes under the "Advanced Settings" section. More information about this section can be found here `http://www.ks.uiuc.edu/Research/vmd/plugins/fftk`. Navigate to the "Results" section and start the optimization process by clicking "Run Optimization". To run successive iterations, after the optimization is complete, highlight the results and

click "Set As Initial". Continue doing this until there is a negligible difference between the "Prev. Charge" and "Final Charge" values. Once the charges have converged, create a new PSF file containing the optimized charges with the "Update PSF with new charges" file dialogue box. If the charges are significantly different than those in the initial PSF, it is a good idea to update the original topology file with the new charges. For the polybutadiene dimer, there is a negligible change in the charges so this is unnecessary.

C.3.3 Bond Angle Optimization

Under the "Calc. Bonded" tab (Fig. D.8), populate the file diologue boxes under the "Input/Output Settings" section. Generate the required Gaussian [43] input file by clicking "Write Gaussian Input File". This will generate the file hess.gau, which is included in Appendix B.3. Run this file in gaussian and direct the output to a file called hess.log.

The remainder of section C.3.3 contains examples of optimizing the bond and angle parameters of a polymethyloxazoline dimer. After the program has completed, navigate to the "Opt. Bonded" tab and expand the "Input" section (Fig. D.9). Populate the file dialogues with the appropriate file locations and add any additional parameter files. ffTK [42] requires the use of NAMD [33] to optimize bonds and angles. A copy of the NAMD [33] binary file must be located locally on your workstation. Specify the location of the NAMD [33] binary file in the proper file dialogue box.

Open the "Parameters to Optimize" section (Fig. D.10). Click the "Guess" button to add the missing parameters along with initial guesses provided by ffTK [42]. All of the initial guesses can be edited by the user. Entries can also be manually added or removed. This will not be necessary to complete the work presented in Chapter 3.

Under the "Advanced Settings" tab (Fig. D.11), make the following changes: Set "Geom. Weight" to 2.0 and "Angles–Eq.Deviation" to 5.0. This is done at the suggestion of the ffTK tutorial [45]. These settings were used to perform all bond and angle optimizations in Chapter 3. Click "Run Optimization" to perform the first iteration of the optimization process.

Once the first optimization has finished, open the "Results" section (Fig. D.12). Here you will find a list of the updated parameters. Notice the entries for "Current Final Obj. Value" and "Previous Final Obj. Value" located beneath the parameter list. To run a second iteration, click the "Set As Initial" button. Open the "Input" section and change the name of the "Output LOG" by adding a number. After you have made these two changes, click the "Run Optimization" button. When the second optimization has completed, you should see new parameters. There should now be values for both "Current Final Obj. Value" and "Previous Final Obj. Value". The user should perform successive iterations as described above until the "Current Final Obj. Value" is greater than the "Previous Final Obj. Value".

Once you have performed enough iterations, we need to save the optimized parameters to our new parameter file. Return to the "BuildPar" tab and open the section marked "Update Parameter File with Optimized Parameters" (Fig. D.13). Specify the location of the initial parameter file created in the "Input Parameter File" dialogue box. In the dialogue box next to "Optimization LOG File", provide the location of the LOG output file from the last bond optimization iteration performed. Then specify a location for a new parameter file, we suggest you call this `bondOpt.par`

and save it in your `4-BaOpt` directory. Click "Write Updated Parameter File" to save the optimized parameters to your new parameter file.

C.3.4 Dihedral Optimization

Open the "Scan Torsions" tab (Fig. D.14). Provide the file locations of your PDB and PSF as well as the location where you would like the Gaussian [43] input files stored. Once you have done this, click "Load PSF/PDB" and then "Basename from TOP". Next, click the "Read from PAR" button and specify the location of your initial parameter file. ffTK [42] will automatically populate the "Dihedrals to Scan" window with the missing parameters. ffTK [42] will exclude dihedral angles that terminate with hydrogens. Dihedrals that terminate with hydrogens do not need to be scanned explicitly unless the hydrogens carry a charge other than +0.09. If your molecule contains missing dihedral angles that terminate with hydrogens carrying a charge other than +0.09, they will need to be added to the list manually by clicking the "Add" button.

Make the following changes to all entries in the "Dihedrals to Scan" list: set "Scan +/- (°)" to 180 and "Step Size (°)" to 10. Do this by highlighting each entry, modifying the values in the "Edit Entry" boxes, and clicking the "$\surd$" button. These modifications are made at the suggestion of the ffTK tutorial [45]. These settings were used for all dihedral scans performed in Chapter 3. Press "Generate Dihedral Scan Input" to generate all of the Gaussian [43] input files. Each dihedral angle will result in two scan input files, one positive scan and one negative scan. Run each of these in Gaussian [43] saving the outputs to a log file of the same name. An example of a positive scan input is included in Appendix B.4.

Once Guassian [43] has completed all the necessary scans, open the "Opt. Torsions" tab. Under the "Input" section (Fig. D.15), provide the required file locations in the dialogue boxes as in previous steps. Be sure to include the in-progress parameter file that contains any optimized bond parameters. Open the "QM Target Data" section (Fig. D.16) and click the "Add" button. Select all of the Gaussian [43] log files.

Open the "Dihedral Parameter Settings" section (Fig. D.17) and click "Read from PAR". Select the bond optimized parameter file to load the missing dihedral list. Entries can then be added, removed, duplicated, or edited from this window. When optimizing a small number of dihedral angles, you will likely not need to change anything. For the polybutadiene dimer, we only have one dihedral angle to optimize so the procedure will be relatively straightforward and simple. A strategy to optimize molecules that have a large number of missing dihedral parameters is presented at the end of this section. For now, leave the "Advanced Settings" alone and click "Run Optimization".

Once the first optimization has completed, open both the "Visualize Results" and "Refine" sections (Fig. D.18). Highlight the initial run, uncheck "include MMEi" and hit plot selected. The target energetic profile is shown in black while the energetic profile produced by the initial optimization is shown in blue. This is a relatively simple energetic landscape and we have already reproduced the shape very well (Fig. C.2A). To further refine these parameters, we need to perform another iteration. Click the "Set As Refit Input" button. This will populate the "Modify Dihedral Parameters for Refitting/Refinement" list in the "Refine" section. To get the best fit possible, duplicate this term 3 times by hitting the "Duplicate" button. Change the "Periodicity (n)" value in each of the three duplicates to 2, 3, and 4. Run successive iterations using the "Run

Refitting/Refinement" button and "Set As Refit Input". Do not use the "Run Optimization" button as this will start the process over.

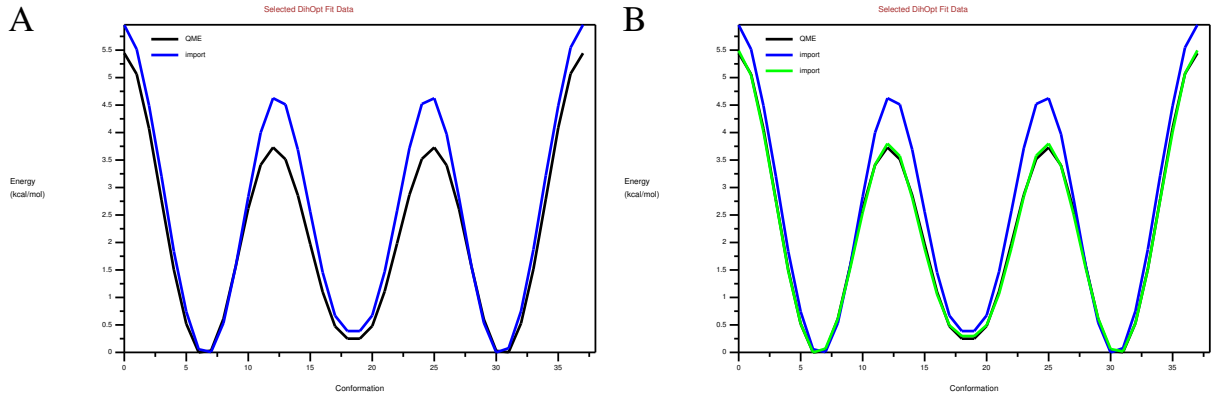


FIGURE C.2: Energy profile of the missing polybutadiene dihedral angle (black) as well as the A. initial (blue) and B. final (green) fittings.

Once you have reproduced the general shape of your energy profile, the ffTK tutorial [45] suggests the following changes be made: switch "Mode" to downhill and "Tol" to 0.0001. Running the refitting algorithm in this mode helps better match the amplitude of your energetic profile. Once you are satisfied with your fit, save it by highlighting the most recent run and clicking the "Write Selected to LOG" button. This can be done at any point during the refitting. You also have the option of loading previously saved log file so you can start refitting where you left off if you need to close VMD [30]. To save the new parameters, go to the "BuildPar" tab and open the "Update Parameter File with Optimized Parameters" section. Specify the location of the bond optimized parameter file in the "Input Parameter File" dialogue box. In the dialogue box next to "Optimization LOG File" provide the location of the LOG output file from the the latest run of dihedral refitting. Choose a name and location for your completed parameter file and click "write Updated Parameter File".

The simple example presented above may not be as helpful when trying to parameterize a large amount of dihedral angles at once. When parameterizing a polymethyloxazoline dimer, there were 8 missing dihedral angle terms. The energetic profile for this molecule is much more complicated than the polybutadiene dimer. It would be almost impossible to guess the right combinations of periodicities for all 8 of the angles in order to best fit the energy profile. The following strategy helped identify the right combinations of terms.

1. Before running the initial optimization, duplicate every dihedral angle 4 times and change the "Periodicity (n)" value in each of the three duplicates to 2, 3, 4, and 6. Now every angle should have 5 terms with different periodicity values. Run the initial optimization. The first optimization will likely give you a very poor fit. In this example the initial optimization had an RMSE of 5.241.

2. Use "Set As Refit Input" and "Run Refitting/Refinement" to perform successive iterations. The RMSE will initially decrease by large amounts. The decreases will continue to become

smaller and smaller as you perform more iterations. Continue performing iterations until there is little to no change in RMSE or the RMSE begins to increase.

3. Make sure the lowest, most recent refitting is loaded in the "Refine" section. Carefully go through the list of parameters and identify the smallest "Force Constant (k)" value. Remove the term with the smallest "k" value as this essentially weights the terms overall contribution to the energy profile. Be sure when you are removing terms that you always keep at least one term for every dihedral. Run the refitting and refinement and continue to repeat steps 2 and 3 until the smallest "k" value is larger than 0.01.

4. Change the "Mode" to downhill and "Tol" to 0.0001 and run the refitting optimization.

5. Delete the smallest "k" value and run the refitting/refinement in "Simulated Annealing" mode. The RMSE will increase, set this as the refit input and run the refitting/refinement in "Downhill" mode. Compare the final RMSE with the RMSE prior to deleting the last term and refitting in "Simulated Annealing". Repeat this step as long as the RMSE continues to decrease. When the RMSE no longer decreases, you have likely found the best possible fit.

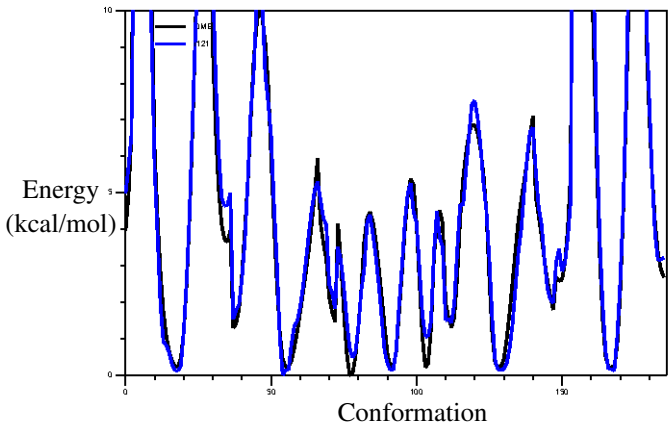


FIGURE C.3: Energy profile of the missing polymethyloxazoline dihedral angles (black) as well as the final (blue) fittings.

Using the steps outlined above, we were able to produce the energetic profile (blue) shown in Figure C.3 after approximately 60 iterations. This is time consuming but it takes away a lot of guess work. The fit has an RMSE of 0.524 which is high but acceptable given the complexity of the profile. This fit also closely reproduces all energetic minimas which is important.

Appendix D

Molefacture and ffTK GUI Images

This appendix contains images of the molefacture and ffTK GUIs referenced in Chapter 3 and Appendix C.

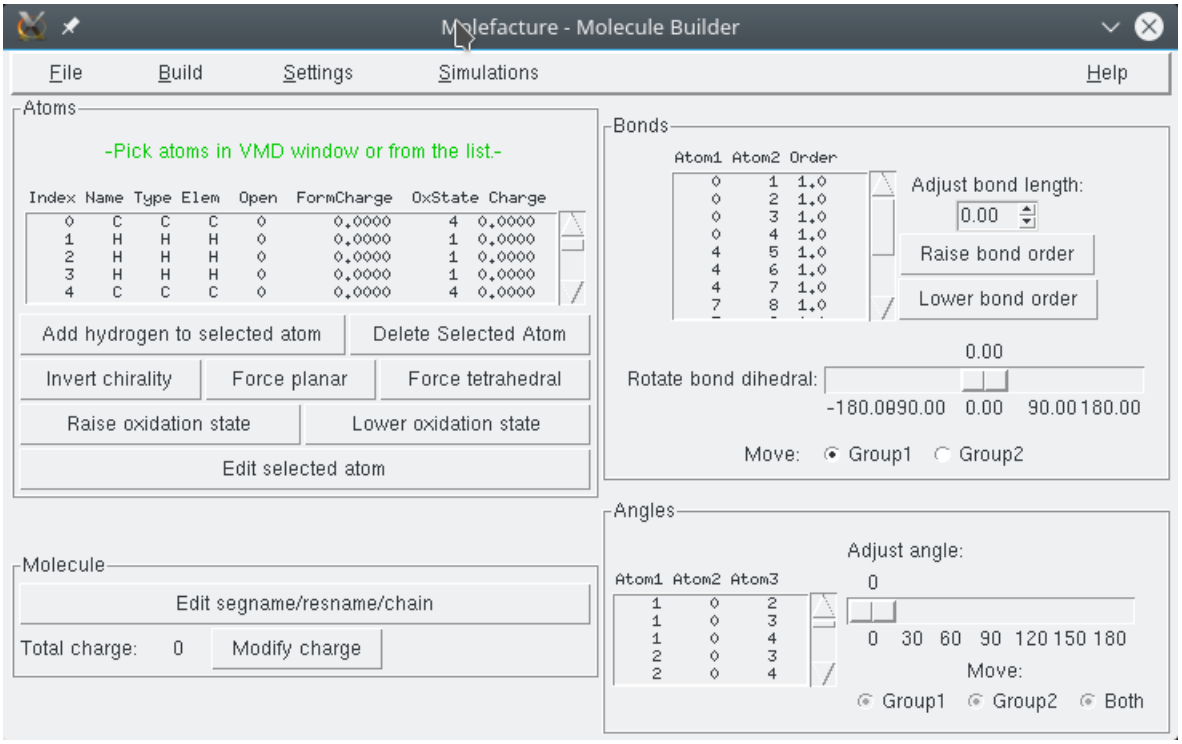


FIGURE D.1: Snapshot of the Molefacture GUI.

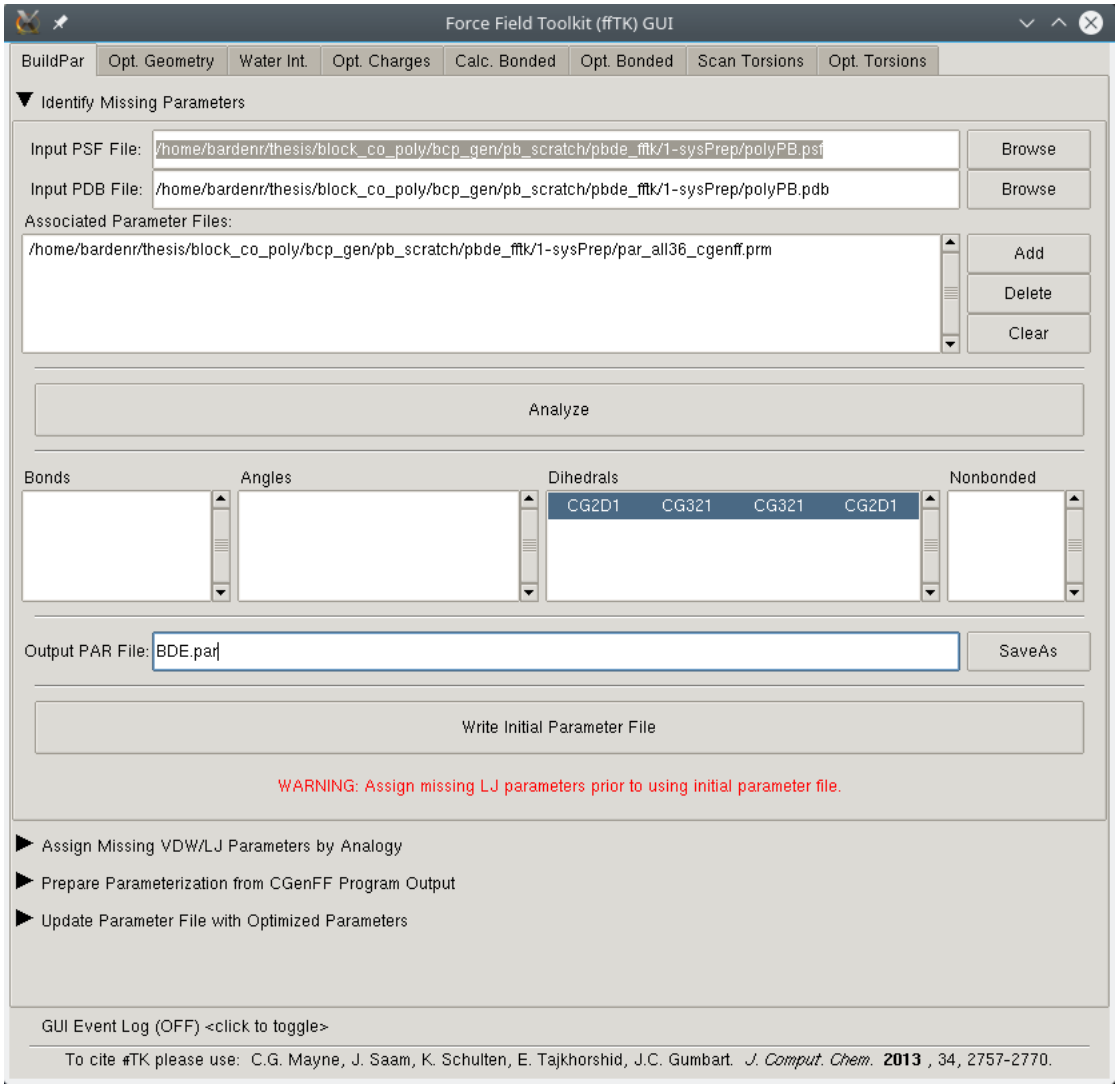


FIGURE D.2: Snapshot of the "BuildPar" tab in the ffTK[42] GUI.

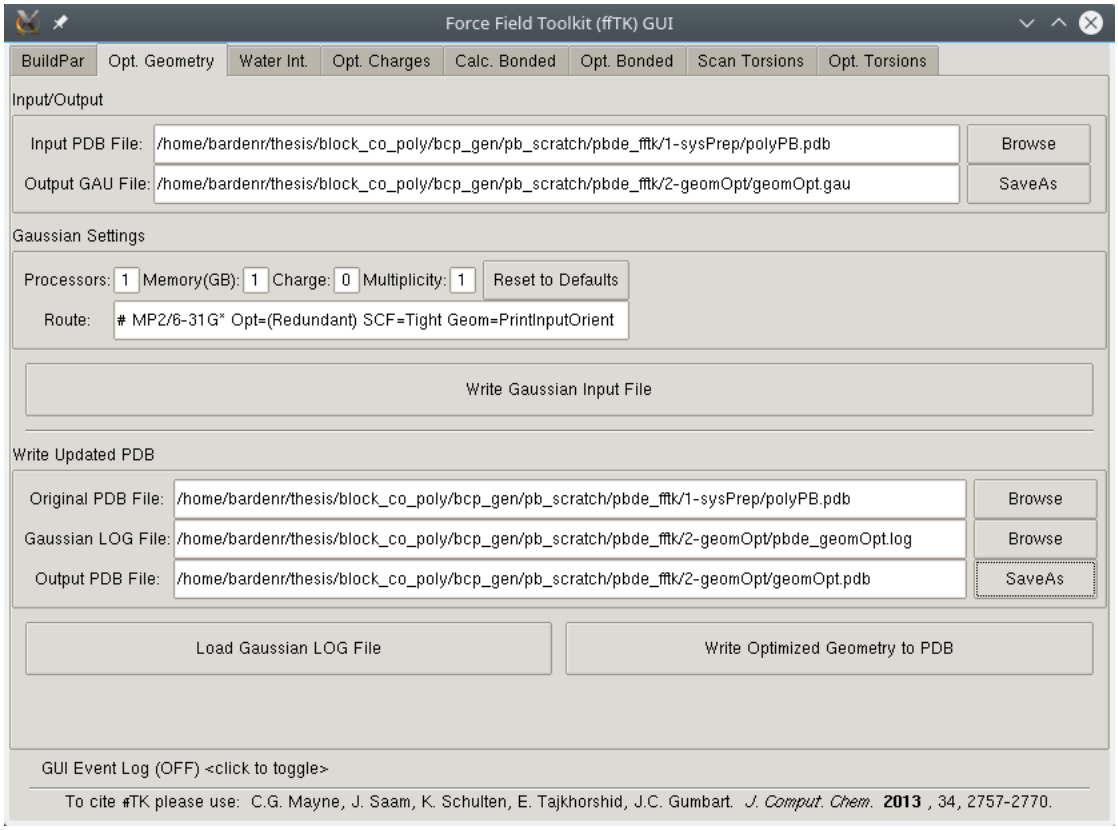


FIGURE D.3: Snapshot of the "Opt. Geometry" tab in the ffTK[42] GUI.

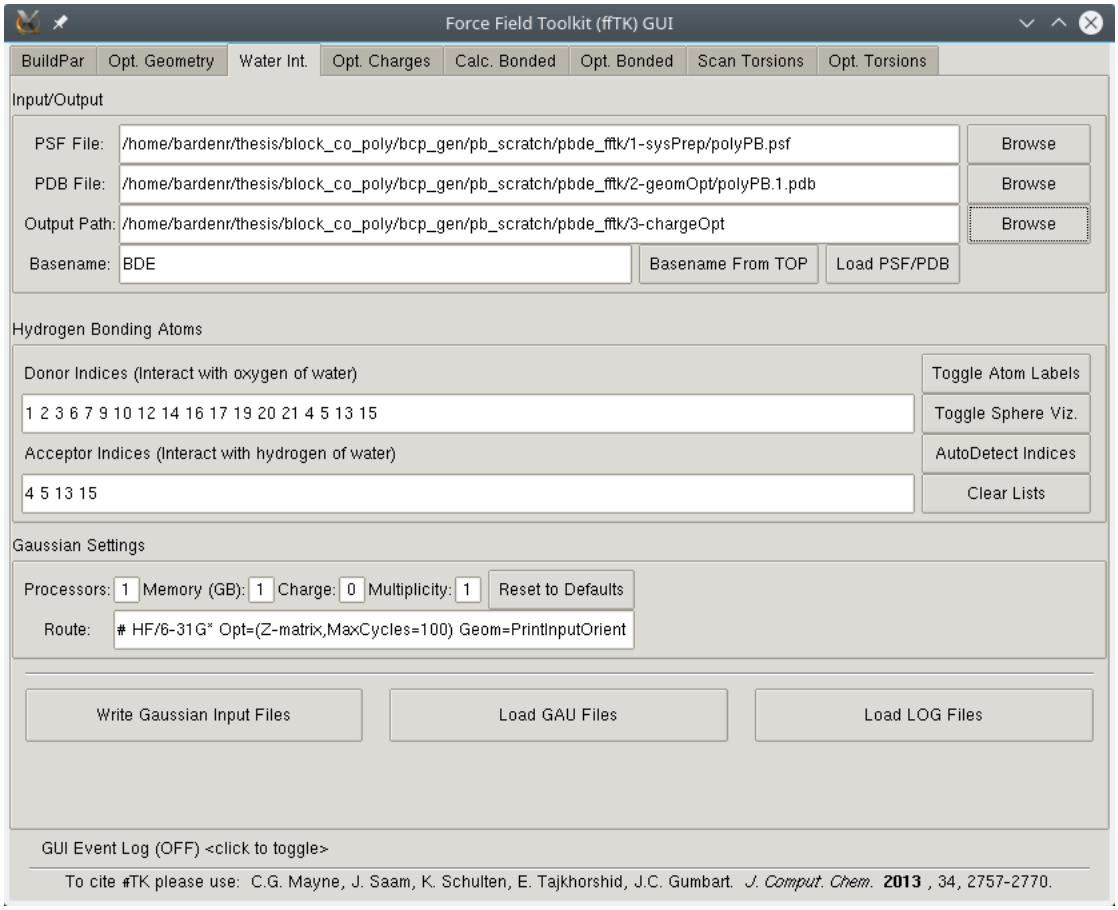


FIGURE D.4: Snapshot of the "Water Int." tab in the ffTK[42] GUI.

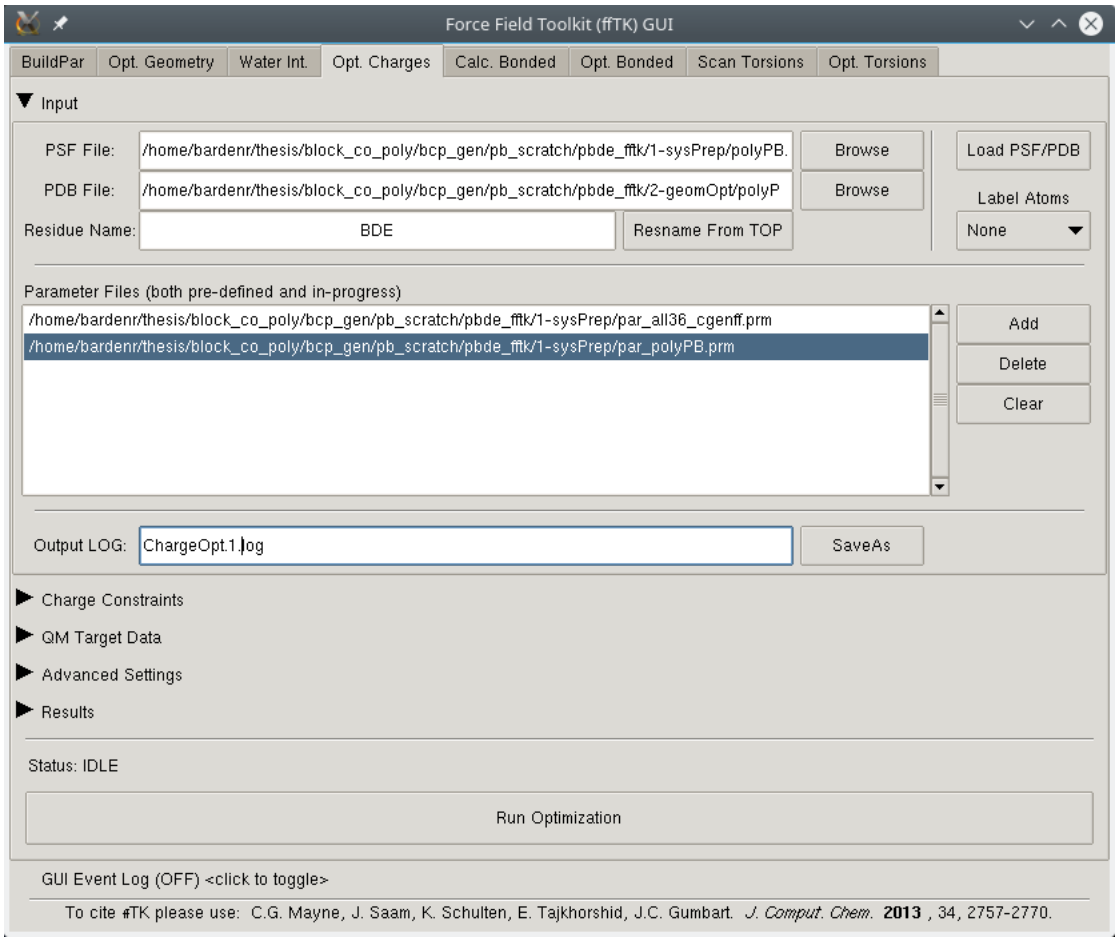


FIGURE D.5: Snapshot of the "Input" section of the "Opt. Charges" tab in the ffTK[42] GUI.

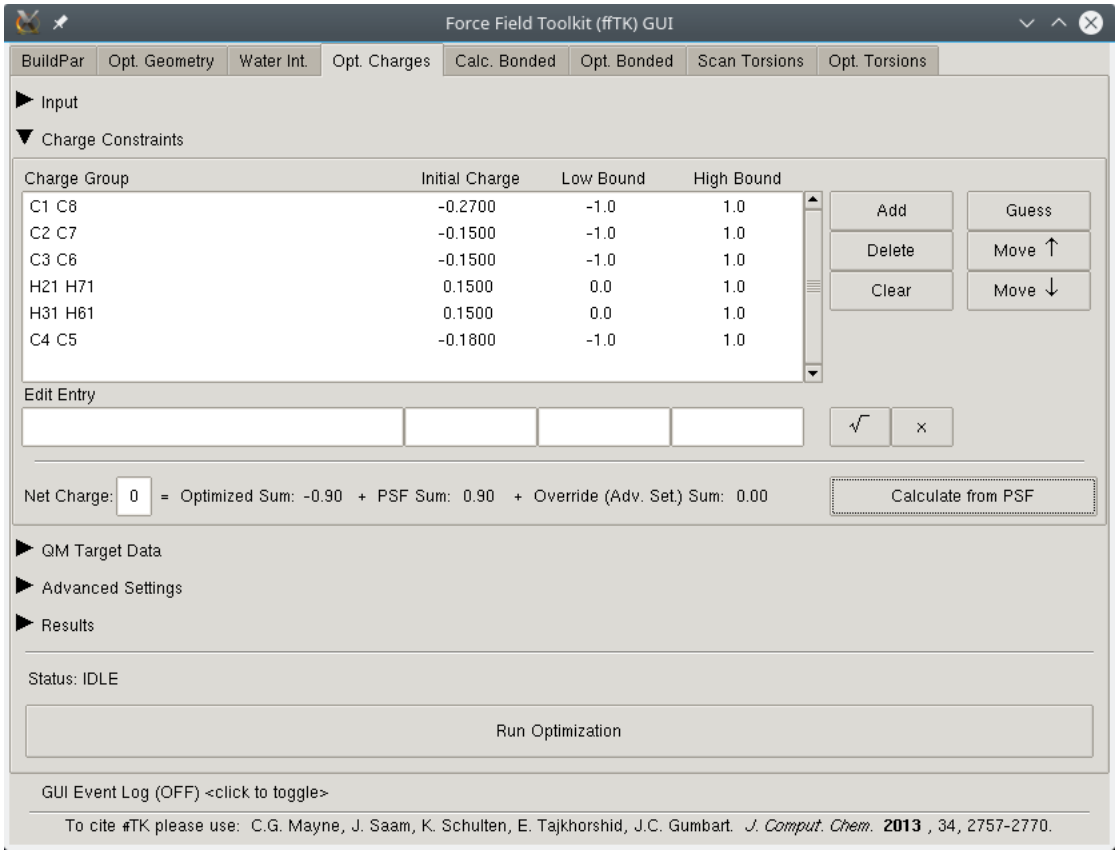


FIGURE D.6: Snapshot of the "Charge Constraints" section of the "Opt. Charges" tab in the ffTK[42] GUI.

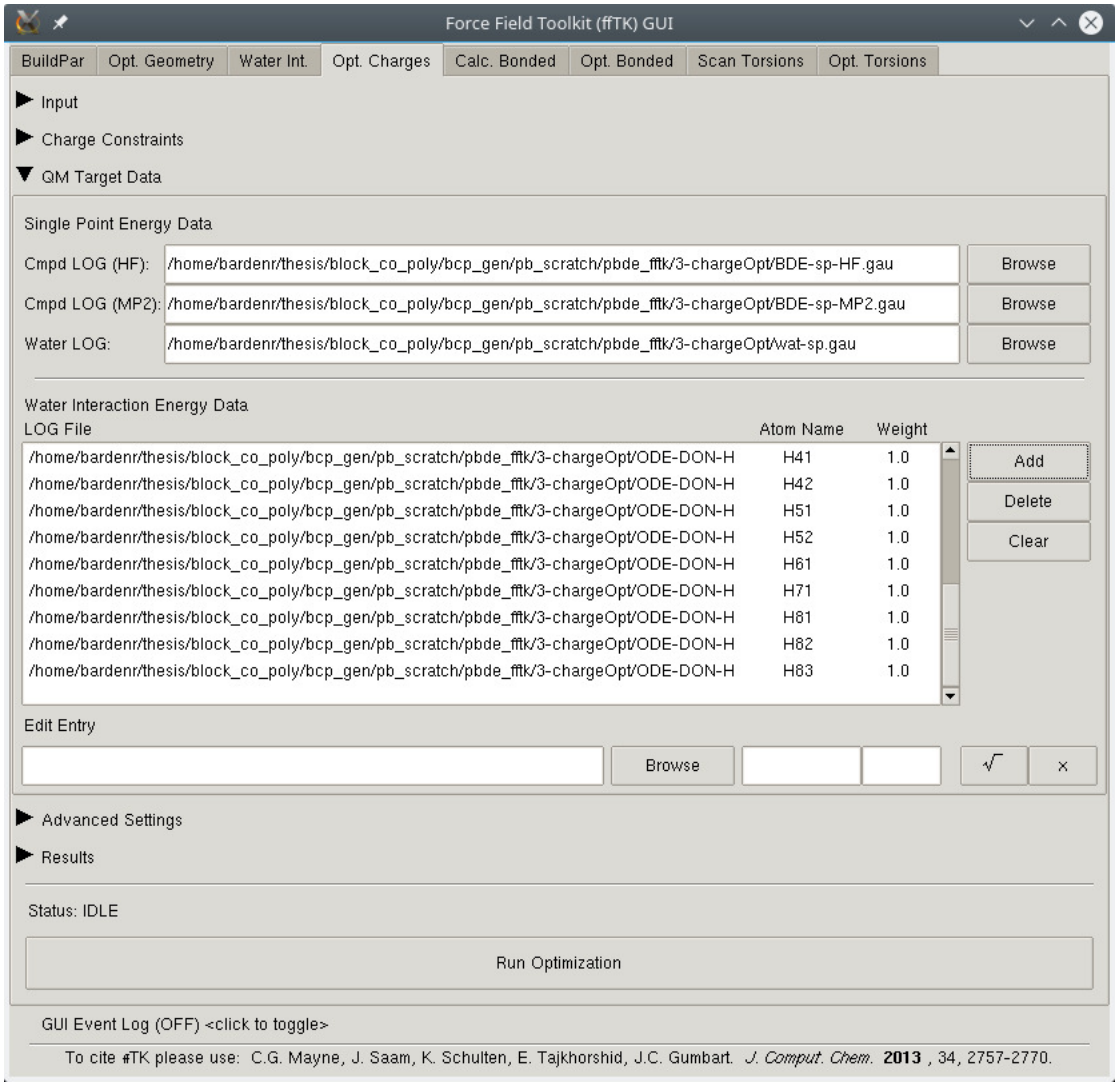


FIGURE D.7: Snapshot of the "QM Target Data" section of the "Opt. Charges" tab in the ffTK[42] GUI.

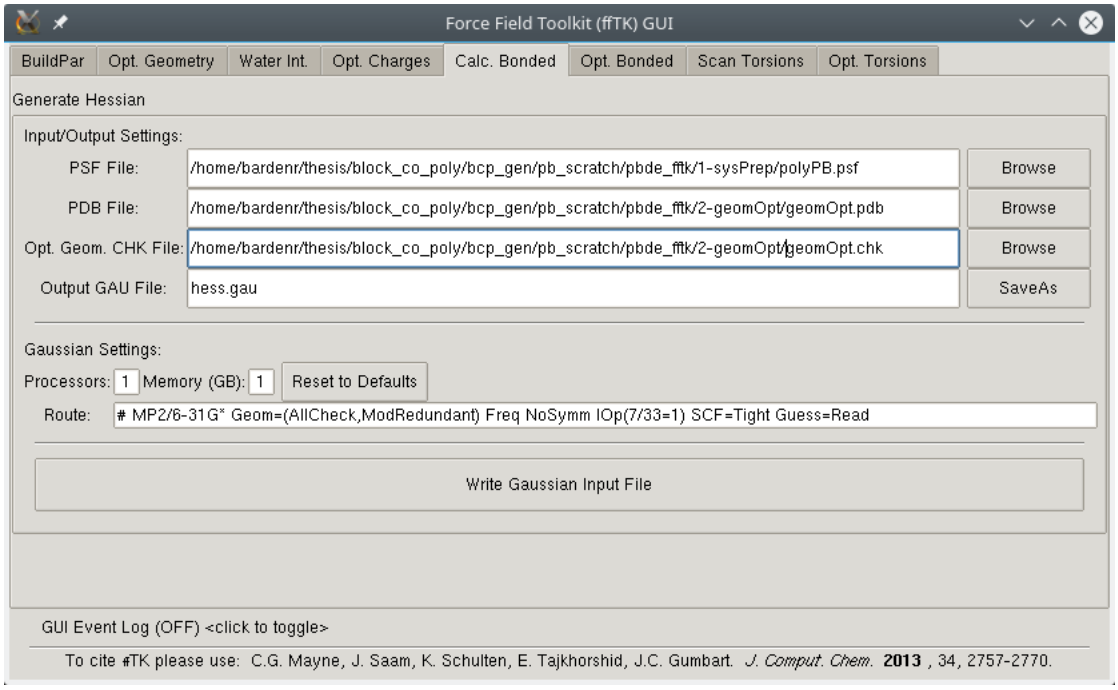


FIGURE D.8: Snapshot of the "Calc. Bonded" tab in the ffTK[42] GUI.

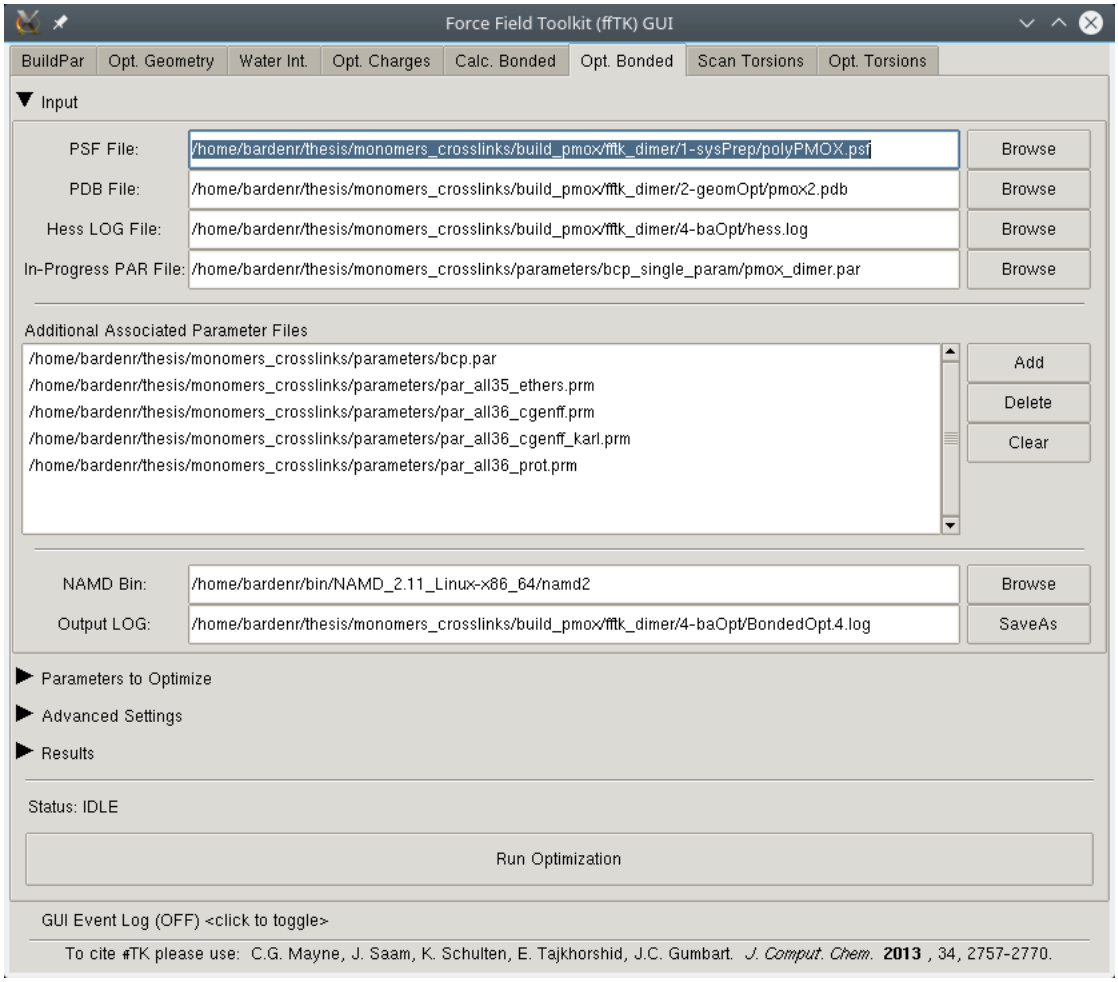


FIGURE D.9: Snapshot of the "Input" section of the "Opt. Bonded" tab in the ffTK[42] GUI.

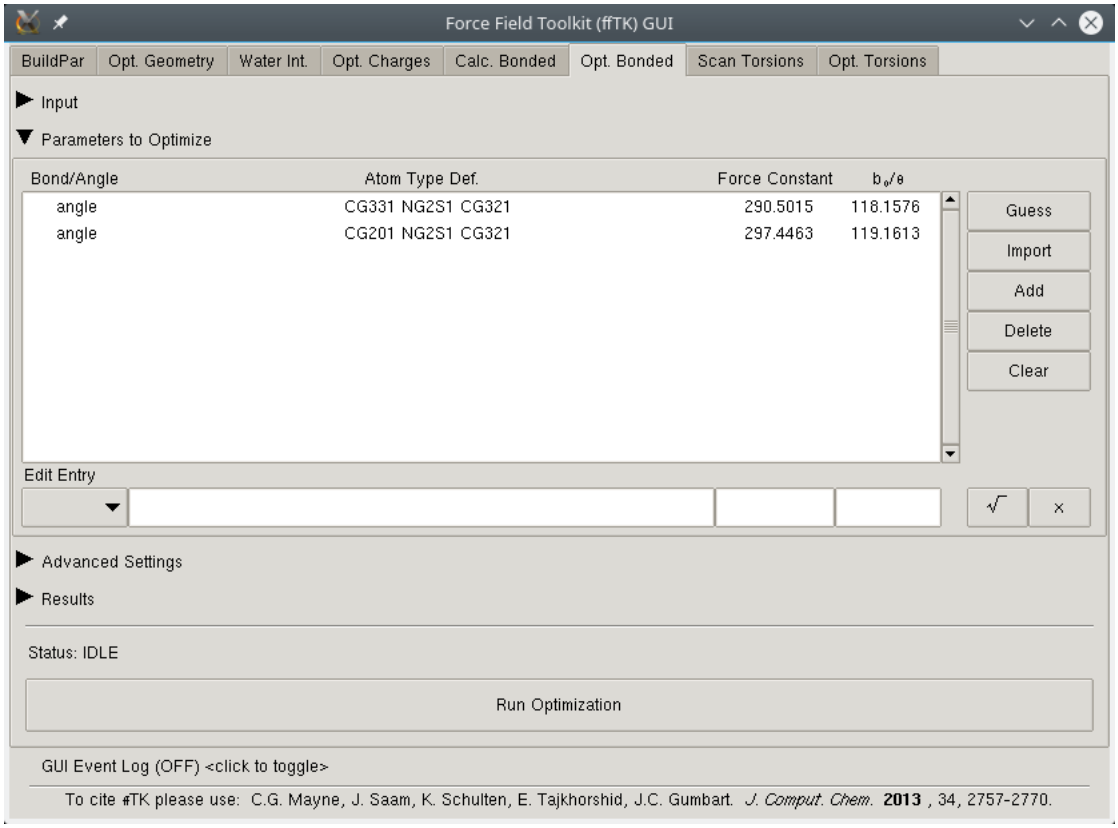


FIGURE D.10: Snapshot of the "Parameters to Optimize" section of the "Opt. Bonded" tab in the ffTK[42] GUI.

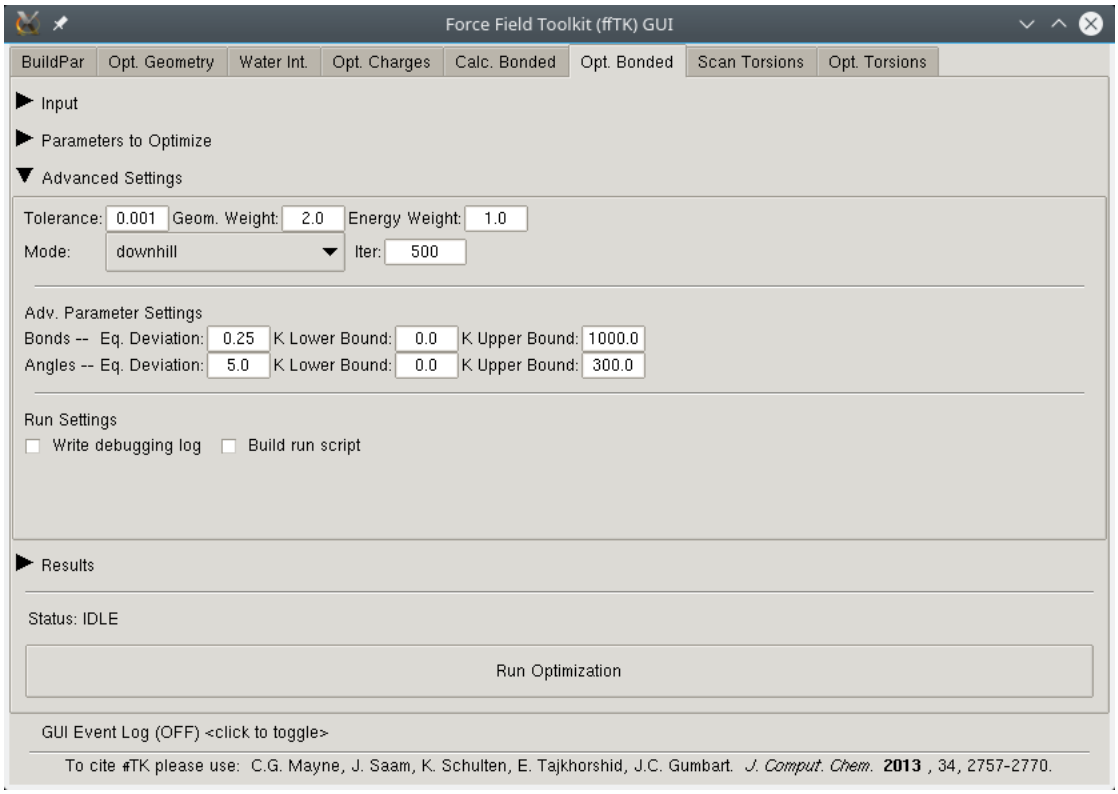


FIGURE D.11: Snapshot of the "Advanced Settings" section of the "Opt. Bonded" tab in the ffTK[42] GUI.

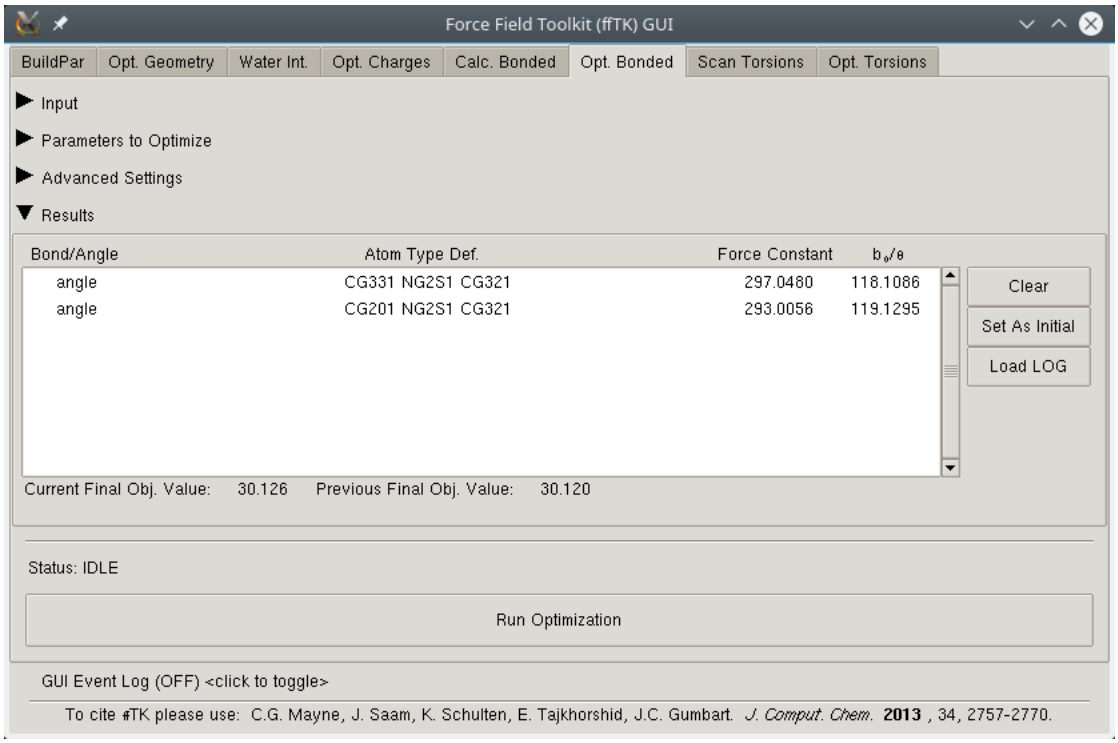


FIGURE D.12: Snapshot of the "Results" section of the "Opt. Bonded" tab in the ffTK[42] GUI.

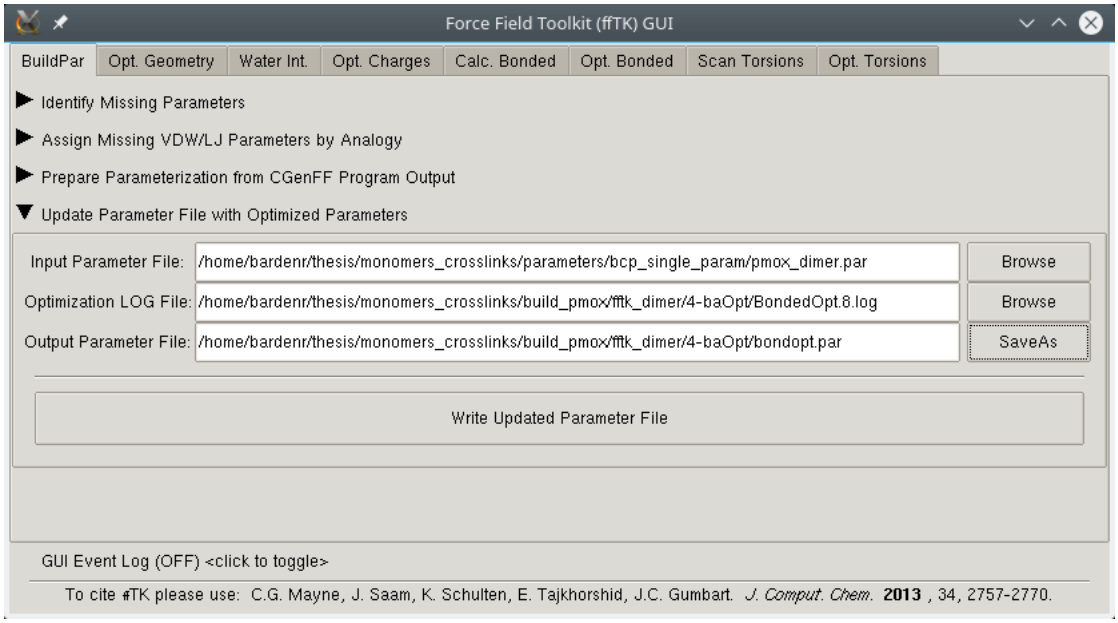


FIGURE D.13: Snapshot of the "Update Parameter File with Optimized Parameters" section of the "BuildPar" tab in the ffTK[42] GUI.

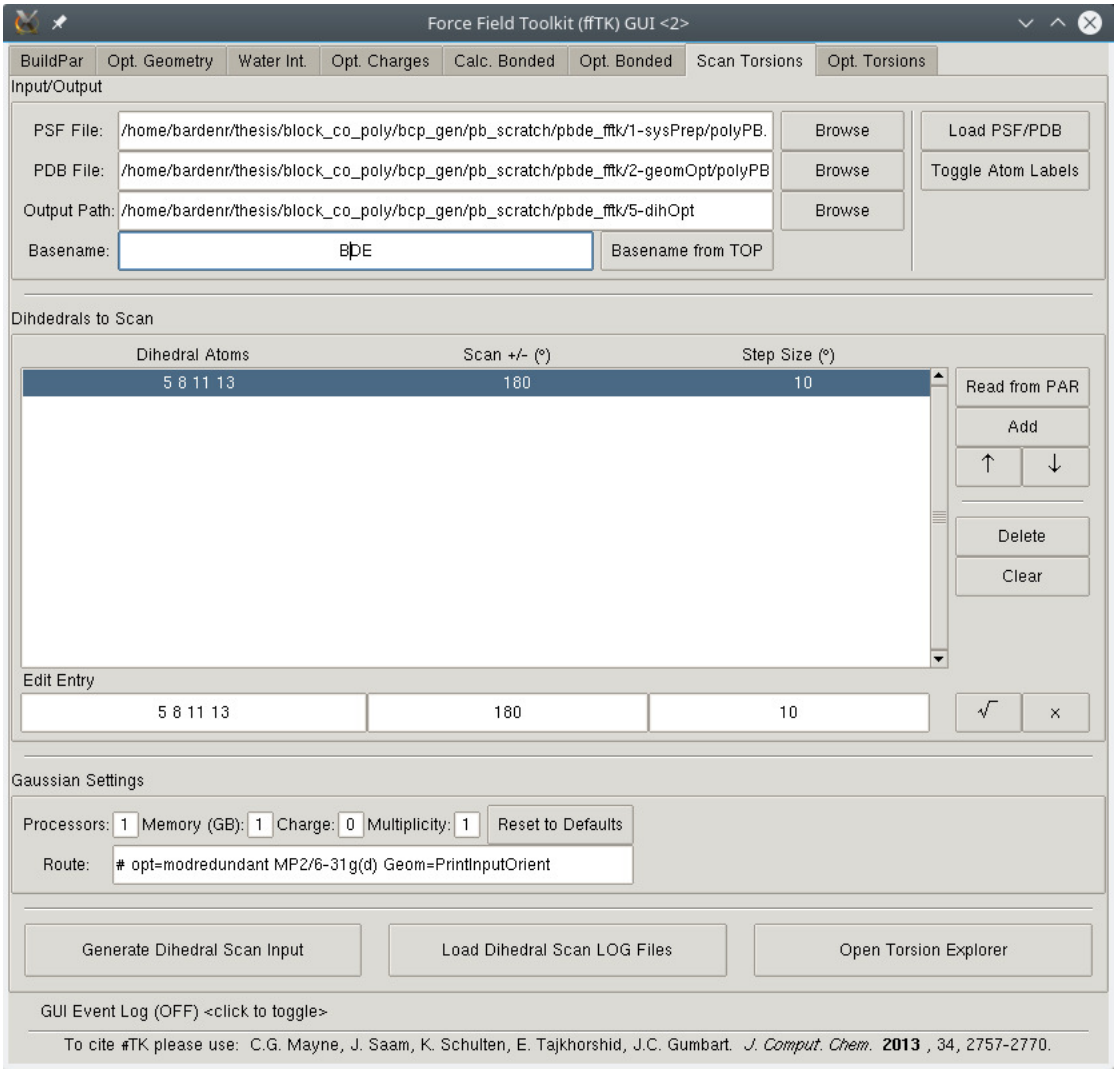


FIGURE D.14: Snapshot of the "Scan Torsions" tab in the ffTK[42] GUI.

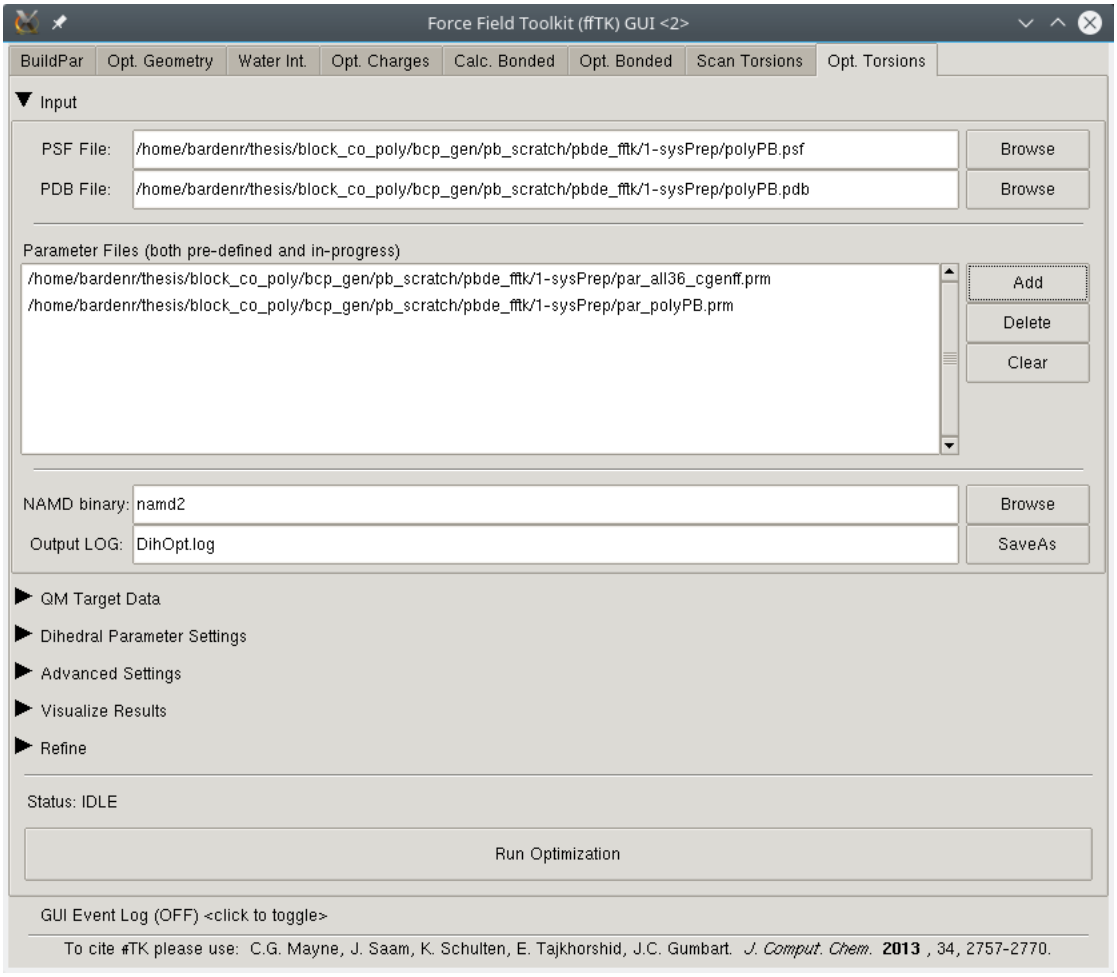


FIGURE D.15: Snapshot of the "Input" section of the "Opt. Torsions" tab in the ffTK[42] GUI.

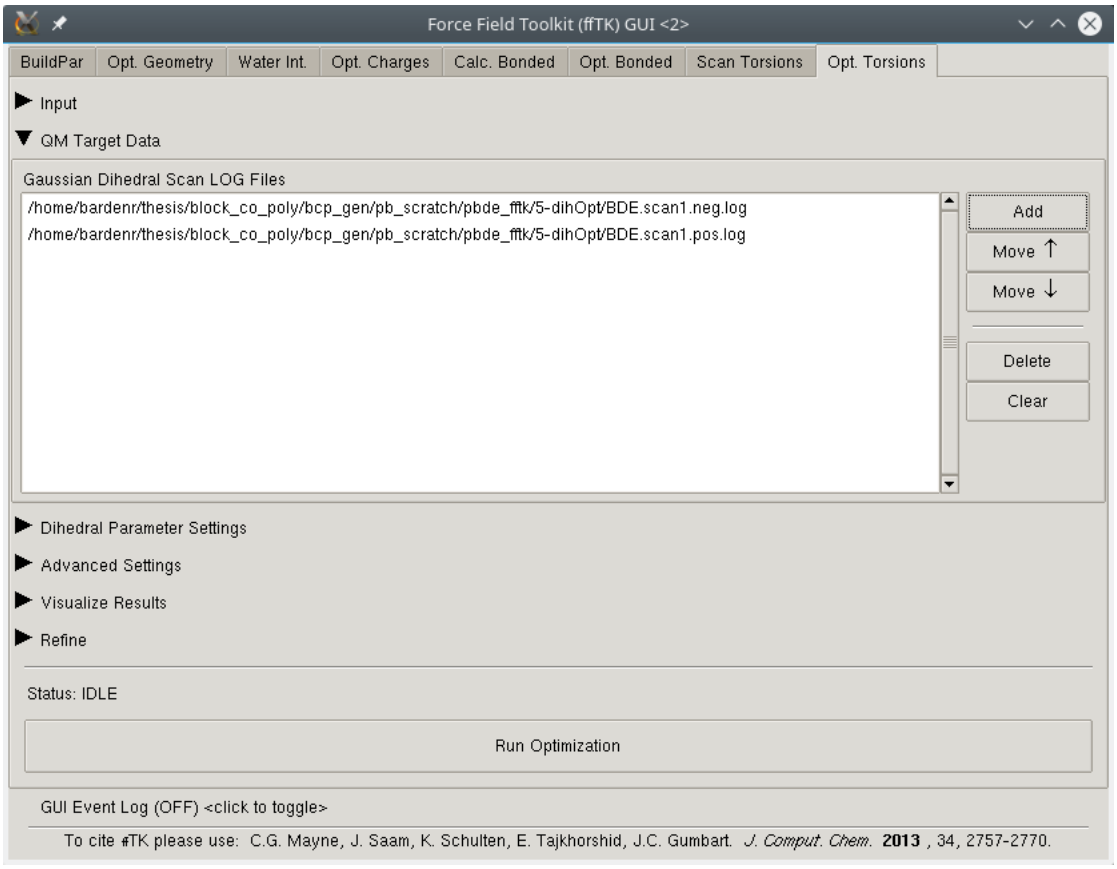


FIGURE D.16: Snapshot of the "QM Target Data" section of the "Opt. Torsions" tab in the ffTK[42] GUI.

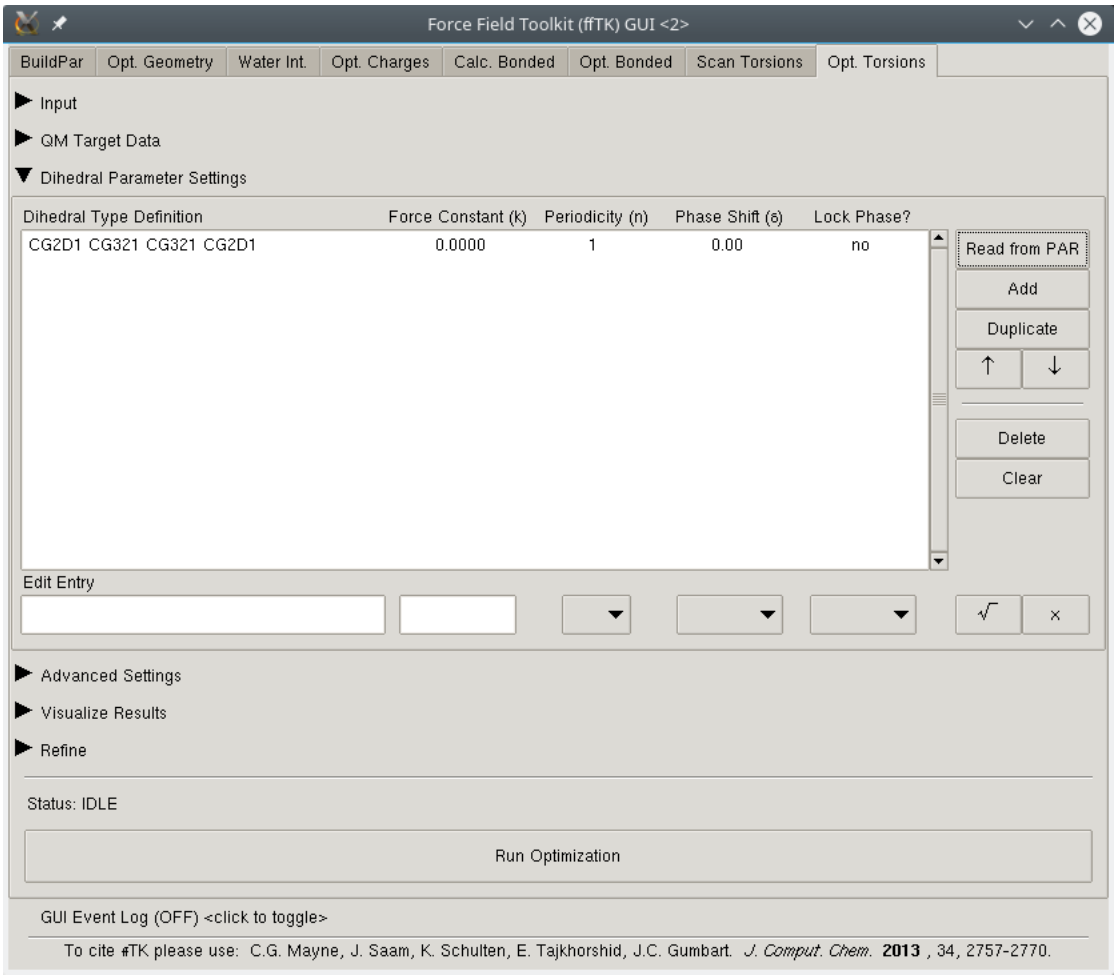


FIGURE D.17: Snapshot of the "Dihedral Parameter Settings" section of the "Opt. Torsions" tab in the ffTK[42] GUI.

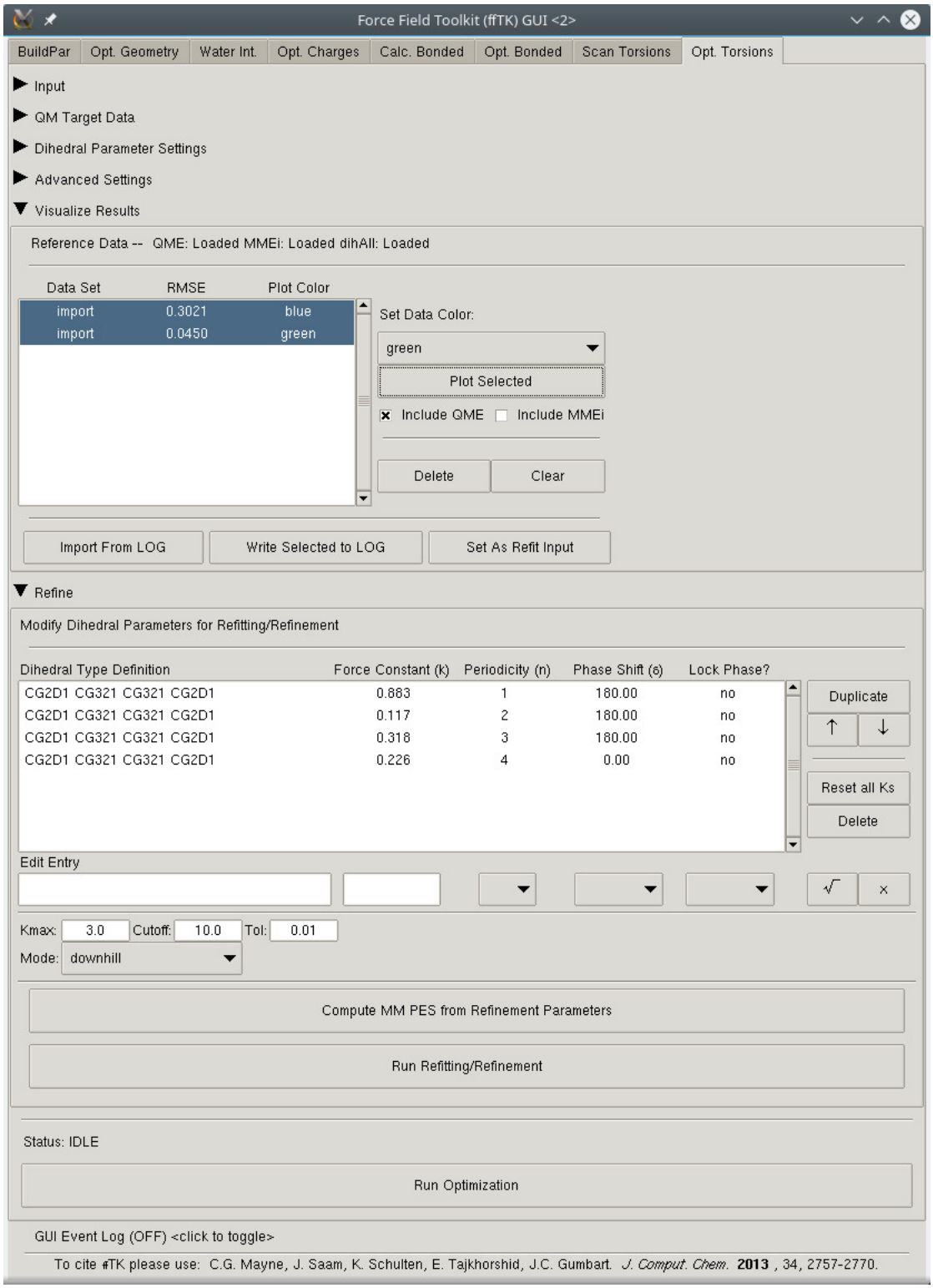


FIGURE D.18: Snapshot of both the "Visualize Results" and "Refine" sections of the "Opt. Torsions" tab in the ffTK[42] GUI.