Atomistic and Coarse-Grained Investigation of Membrane-Protein Interactions

by

Carlos Osorio

A dissertation submitted to the Department of Mechanical Engineering

Cullen College of Engineering

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Mechanical Engineering

Chair of Committee:	Ashutosh Agrawal
Committee Member:	Tanmay Lele
Committee Member:	Theocharis Baxevanis
Committee Member:	Dong Liu
Committee Member:	Shailendra P Joshi

University of Houston

August 2021

ACKNOWLEDGEMENTS

I would first like to thank my supervisor, Dr Agrawal, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to acknowledge my group mates. Mehdi Torbati, Ehsan Irajani, Nidin Thomas and Maziyar Bazmara. I would also like to thank my lab mates.

In addition, I would like to thank my parents and my brother for their wise counsel and sympathetic ear. You are always there for me.

ABSTRACT

Gaining mechanistic insights into membrane-protein interactions is vital for comprehending cellular processes and health disorders. In this work, we pursued two sets of studies to elucidate such interactions. First, we used atomistic studies to investigate increased binding affinity of a membrane-remodeling protein (epsin) to membranes subjected to high tension. We performed umbrella sampling calculations to provide the first quantitative evidence that tension energetically promotes the insertion of a transmembrane helical domain of epsin. Next, we embarked on an ambitious journey to build a coarse-grained Monte Carlo computational framework to simulate membrane-protein interactions at the mesoscale. This framework utilizes building blocks such as body, bonded and non-bonded components and discretized differential geometry operators necessary to create interactions of customizable molecules with a highly versatile membrane. We provide a concise overview of the key notions underlying this framework. Next, we demonstrate the unique abilities of this framework via several toy problems inspired from biological systems. We simulated, to the best of our knowledge, the highest genus membrane structure in the literature till date. This structure resembles a nuclear envelope and consists of two adjacent spherical membranes fused at hundreds of sites. Next, we demonstrated the formation of coexistent domains on an ellipsoidal vesicle generated by heterogeneous lipid composition. Next, we demonstrated the assembly/disassembly of a tri-legged protein (called clathrin) on a vesicle and the ability of the polymerized protein to form cargo-carrying vesicles. Finally, we show extreme tubulation caused by aggregation of banana-shaped proteins on a spherical membrane. These test cases confirm the potential of our new framework to model complex membrane geometries and membrane-protein interactions.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

Biological phenomena are arguably one of the most difficult physical processes to understand, as they entail multiple dynamic entities with varying degrees of complexity. Biologists have used experimentation and observation to comprehend the roles of biological entities in vital cellular processes. These entities, such as lipids and proteins, undergo transformation, binding, and self-assembly in order to execute their biological functions. The behavior of such elements is sophisticated enough that they appear almost sentient at a first glance. However, the are governed by the same laws of physics that control our every day lives. Hence, by analyzing the biological observations through the lens of physics, we can obtain a deeper understating into to the working of cells.

One of the active areas of research in cellular biology relates to cellular interfaces. Fig. 1.1 shows a schematic of the key cellular interfaces. The left panel shows the interfaces of the cell, called the plasma membrane, and the internal organelles and structures such as nucleus, endoplasmic reticulum, Golgi apparatus, endosomes, and vesicles. The right panel shows the electron micrographs of a Golgi apparatus, an endosome and HIV-1 virus budding from the plasma membrane.

The cellular interfaces are primarily made of lipids and proteins. Lipids are amphiphic molecules with hydrophilic headgroup and hydrophobic tails. Because of these contrasting properties, lipids self-assemble into a double layer structure, called a bilayer, to shield the tails from the surrounding environment. This tendency to self-assemble acts as a glue to hold the proteins inside the membranes. While lipids move freely on the surface of the bilayers, they resist bending and stretching of the bilayers. As a result, the bilayers behave as liquid crystals. This unique property allows the cellular interfaces to possess complex topologies and undergo significant shape changes without compromising structural integrity.

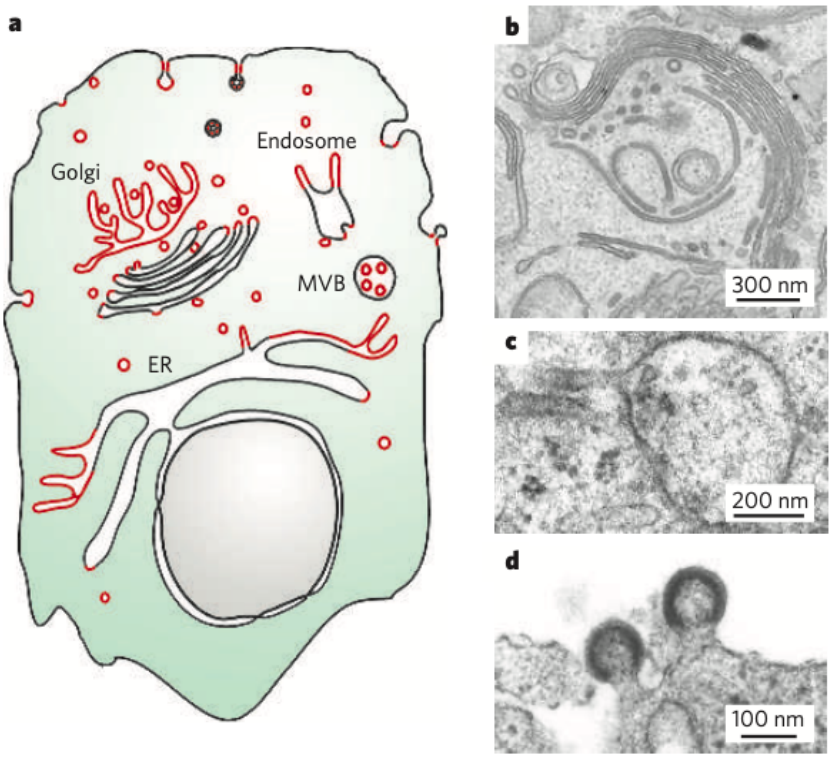

Figure 1.1: Cellular interfaces. Left: Schematic shows membrane interfaces of the cell and its organelles such nucleus, endoplasmic reticulum, Golgi apparatus etc. Right: Electron micrographs of a Golgi apparatus, an endosome and HIV-1 virus. Figure taken from [1].

Fig. 1.2 shows a schematic of the complex 3D architectures of the nuclear membranes and the endoplasmic reticulum. The nuclear interface, called the nuclear envelope, has an ultradonut topology consisting of two hollow thin shells fused at hundreds of sites with donut-shaped holes. The endoplasmic reticulum exhibits shapes such as helicoidal ramps and tubules. Fig. 1.3 shows a schematic of a mitochondrion, the powerhouse of the cell, undergoing large deformations during the fission process. Fig. 1.4 shows formation of cargo-carrying vesicles generated by local bending of plasma membrane during cellular transport processes.

These cellular membranes are far from being passive entities. The creation, maintenance and remodeling of these interfaces is actively regulated by numerous proteins and constituent lipids. Fig. 1.5 shows the key mechanisms by which proteins and lipids remodel cellular

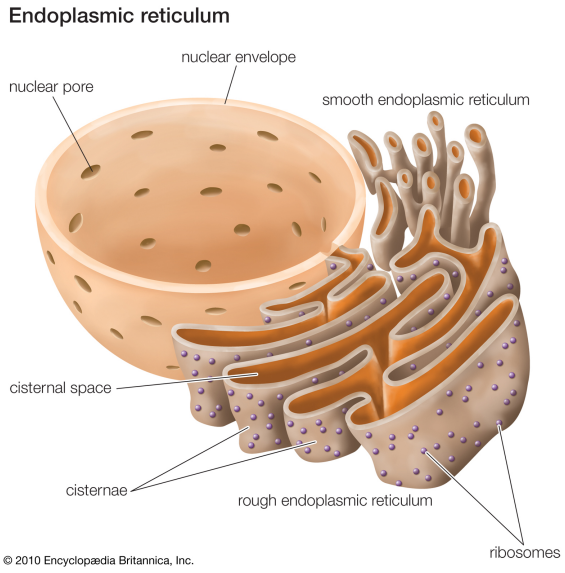

Figure 1.2: Schematic showing the complex 3D topology of the nuclear membranes and the endoplasmic reticulum. Figure taken from [2].

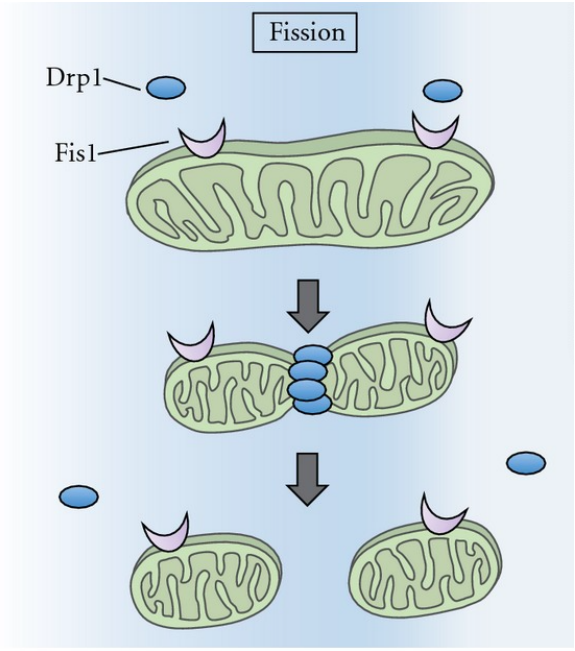

Figure 1.3: Schematic shows the protein-mediated extreme deformation undergone by the mitchondrial membranes during the fission process. Figure taken from [3].

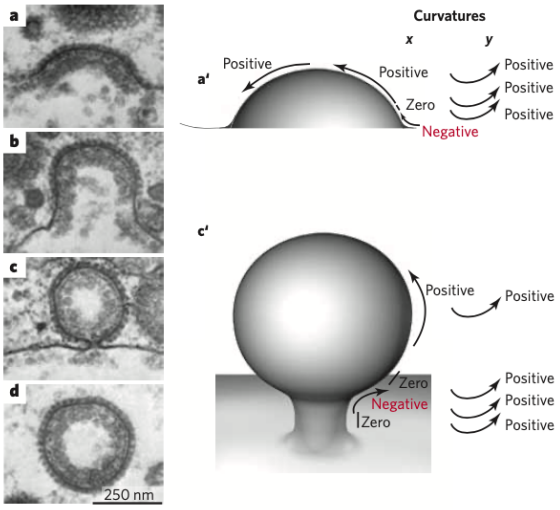

Figure 1.4: Clathrin-mediated endocytosis. The left column shows the sequence of shape transformation leading to vesicle formation. The right column shows the nature of curvatures present in a vesicle. Figure taken from [1].

membranes. Proteins can deform the membranes by i) embedding into the membrane, ii) applying pulling or pushing forces onto the membrane, iii) forming scaffolds on top of the membrane, and iv) inserting a subdomain into a membrane. Lipids on the other hand can deform the membranes if they have a conical structure, and they are present in unequal quantities in the two layers of the bilayer. These membrane-deforming interactions are often dynamic. The proteins come to a specific site at a specific time during a process, interact with the membrane, and then return back to their source. The same can happen to lipids, which can localize at appropriate times to deform the membrane or regulate the proteins. While experiments play a critical role in revealing the remodeling of cellular membranes, the specific roles of proteins and lipids often remain a mystery. One fundamental reason for this is that the experiments often lack the spatio-temporal resolution to capture the molecular interactions and dynamics. In addition, this problem is further worsened by the fact that biological systems contain many redundant regulating mechanisms. Thus, computational sciences can play an important role in delving into these mysteries and unraveling the physics of lipid-protein interactions.

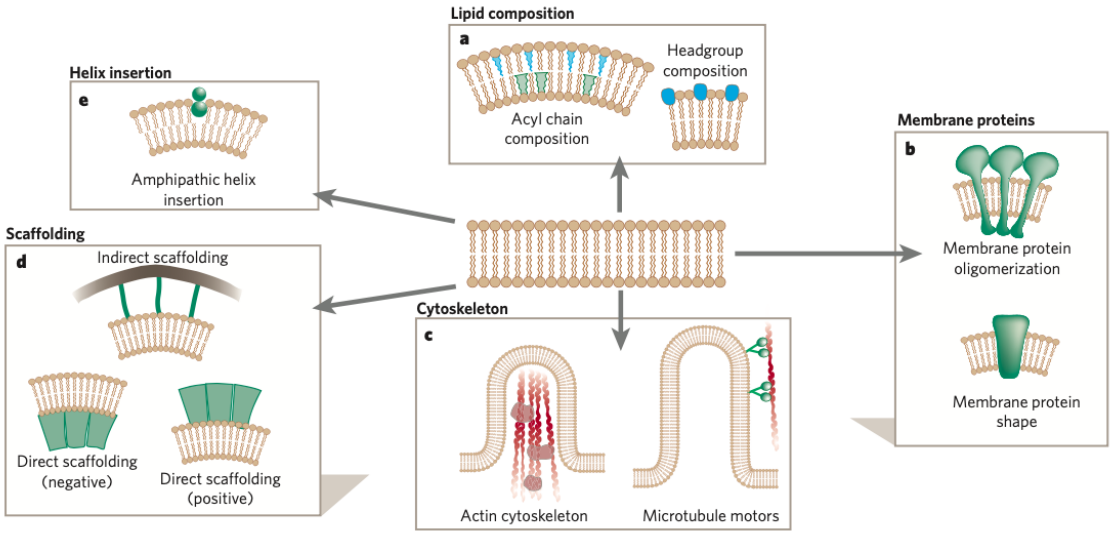

Figure 1.5: Cellular membranes are deformed by a) conical lipids, b) embedded proteins, c) cytoskeletal proteins, d) scaffolding proteins, and e) alpha helix-inserting proteins. Figure taken from [1].

1.1 Computational modeling

The use of computers in the simulations of physical systems is almost as old as the idea of the modern computer. A few years after Alan Turing published his seminal paper titled "Computable Numbers" [4] on the concept of *Universal computing machine* in 1937, mathematicians Jon Von Neumann and Stanislaw Ulam used the first machines based on Turing's concept to solve quantum mechanics problems on neutrons [5]. Since then, the application of computers in all branches of physics have become ubiquitous, ranging from finding the trajectory of galaxies to finding the wave functions of electrons. One of the key areas where computer simulations have a potential of creating the most impact is in improving human life. Computer simulations can model systems at the micro and nano scales and shed new light into molecular mechanisms behind the fundamental processes that govern life.

The key tools for understanding nanoscale and microscale phenomena are the atomistic and coarse-grained molecular dynamics simulations. They have the ability to provide insights into the behavior of lipid-protein interactions by invoking the well known forces of nature at the fundamental level. One of the potential applications of these tools is in the de-

sign of drugs to cure diseases. Depending on the length- and time-scale of the interactions, one can pick between the atomistic and the coarse-grained approaches. We use atomistic simulations to study the role of specific atoms or a group of atoms in small systems. It suffers from the disadvantage that the amount of variables that we need to handle are very large. On the other hand, coarse-grained simulations combine many atoms into one molecule to simulate larger systems. Various seminal studies have invoked the two approaches to investigate lipid-protein interactions. For example, atomistic simulations have been used to study interactions of membranes with neuropeptides, ion channels, endocytic proteins, cytoskeletal proteins, fission, and fusion proteins etc. Coarse-grained simulations have been performed to investigate many of these systems as well. While there exist standard atomistic simulation platforms used worldwide such as GROMACS [6, 7, 8, 9] and LAMMPS [10, 11], a generalized coarse-grained computational framework is not yet available to the best of our knowledge.

In the context of membrane-protein simulations, Gommper, Knoll and collaborators created a Monte-Carlo based coarse-grained membrane model and made a seminal contribution to the field [12, 13, 14]. Later, Kumar and collaborators enriched the Gommper and Knoll framework to include heterogeneous properties of lipids in the simulations [15]. Hirochi Noguchi used the Grommper and Knoll model to study the behavior of high genus vesicles [16, 17] and erythrocyte membranes [18, 19]. Noguchi also modeled the interactions of banana-shaped proteins with the membranes [20]. Spakowitz and co-workers studied the stability of clarthin network in the presence of membrane fluctuations and membrane curvatures [21, 22, 23]. In addition, Van der Otter and Giani created a novel coarse-grained model to study clathrin assembly and interaction as with adaptor proteins [24, 25, 24]. Voth and co-workers used coarse-grained models to investigate membrane-protein interactions [26, 27, 28]. Ipsen and collaborators used a coarse-grained model to study the effects of curvature-inducing nematogens [29]. Finally, Radhakrishnan and co-workers performed computational studies to elucidate membrane-protein interactions [30, 31].

In this thesis, we first use atomistic simulations to gain insight into mechano-sensitivty exhibited by an endocytic protein. Next, we build upon the above-mentioned pioneering computational studies to develop a new Monte Carlo-based computational framework to

enable coarse-grained simulations of membrane-protein interactions. The main motivation to pursue a new approach was to develop a flexible and generalized, modular framework that could be customized to simulate a diverse set of membrane-protein and protein-protein interactions. The detailed outline of the thesis is discussed below.

1.2 Thesis outline

In Chapter 2, we investigate the tension-sensitivity exhibited by an endocytic protein called epsin. This protein is capable of anchoring itself into the membrane through an amphiphilic helix that causes the membrane to bend. We use atomistic simulations to provide the first evidence that tension can lower the free energy of the protein in the embedded state. In Chapter 3, we discuss the basic building blocks of our new Monte Carlo framework to emulate lipid-protein interactions. In Chapter 4, we describe the 2D discretized model of lipid membranes. We discuss the relevant geometric quantities and operators required to model lipid membranes. In Chapter 5, we discuss the basic mechanisms that govern the basic architecture of the program. In Chapter 6, we apply the computational framework to simulate a range of toy problems inspired from biological processes. First, we simulate high genus vesicles, including structures that possess ultradonut topology similar to that of nuclear envelope. Next, we simulate domain formation on an ellipsoidal membrane with multiple lipid species. Next, we study the ability of an endocytic protein, called clathrin, to form nanovesicles from a giant spherical vesicle. Next, we simulate the assembly and disassembly of clathrin molecules. Finally, we explore the interactions of orthotropic proteins (banana-shaped proteins) with spherical vesicles. In Chapter 7, we summarize our findings and discuss potential future work.

Chapter 2

Atomistic study reveals tension promotes protein insertion

[1] Clathrin mediated endocytosis (CME) involves the internalization of cargo by sculpting plasma membrane into 60–120 nm-sized buds, supported by a clathrin protein coat [33]. CME is a well-studied endocytic pathway present in organisms at all developmental stages [33, 34, 35, 36]. It plays a critical role in nutrient uptake, intracellular trafficking, and signal transduction [33, 34, 35]. CME is a multistep process involving (i) initiation of membrane budding with adapter proteins and membrane bending proteins [37, 38, 39], (ii) clathrin coat formation [40, 41], (iii) maturation of coated pits [42, 35, 36], and (iv) dynamin mediated scission of the buds [1, 35, 39]. Progression of CME involves extensive deformation of the flat plasma membranes to Ω-shaped pits [33, 43]. Given CME is a mechanical process, membrane tension has been shown to play an inhibitory role during the membrane deformation process, preventing the transition from a flat membrane to hemispherical domes [42, 44] and the transition from hemispherical domes to Ω-shaped pits [42, 44, 43]. Yet, CME is observed ubiquitously in cells under different membrane tension regimes, and this points to the existence of tension-sensitive molecular mechanisms supporting CME [43, 45, 46]. The actin mediated transition of hemispherical domes to Ω-shaped pits at high tension was established by Boulant et al. [43]. However, how membrane-associated proteins aid to overcome the elevated energy barrier needed to initiate budding remains an open question. We hypothesize that endocytic membrane bending proteins possess the ability to sense and counteract membrane tension to facilitate clathrin coat budding at elevated tension. Epsin/AP180 family is a major family of proteins involved in membrane bending during

[1] This is an adaptation of the paper [32]

the initiation of CME [37, 47, 48]. Epsin, a prominent member of the family, is known to insert its N-terminus amphipathic helix (H_0 helix in epsin N-terminus homology (ENTH) domain) into the bilayer with a wedging effect after binding to PIP2 to initiate membrane bending [47]. Epsin also has a C-terminus intrinsically disordered protein (IDP) region containing multiple binding sites to endocytic constituent proteins like clathrin and AP2, which stably dock epsin in the endocytic pit [49, 50, 51, 52]. An alternate hypothesis proposed recently posits the C-terminus IDP domain of epsin initiates membrane bending via steric crowding [53, 54]. In vitro studies have shown that the insertion of purified ENTH into giant unilamellar vesicles (GUVs) reduces membrane rigidity and area compressibility modulus of the lipid bilayer [55]. Further, the recruitment of ENTH softens the bilayer at high tension and initiates tubulation at low tension [56]. ENTH is shown to recruit selectively to the highly curved surface of cylindrical membrane tethers held at different tensions [57]. An increase in lipid packing defects at high tension may be key in aiding helix insertion at high tension from theoretical studies [58, 59]. This evidence points to the existence of a tension-sensitive recruitment mechanism of ENTH domain-containing proteins. It exists an interplay between membrane tension and peripheral protein density that mediates membrane deformation [60]. However, there is still ambiguity in the exact mechanism of ENTH binding at different tensions, and a lack of experimental evidence for tension-mediated recruitment of epsin to clathrin coat nucleation sites in cells. Here, we used molecular dynamics (MD) simulations to investigate the tension-dependent recruitment of ENTH domain into membranes. We deciphered the role of the H_0 helix in tension sensing and the molecular mechanism of tension-mediated recruitment of epsin.

2.1 Atomistic insights into ENTH-membrane interactions at zero tension

2.1.1 Method

We performed MD simulations on a membrane-protein system composed of an ENTH domain of epsin (PDB number $1H_0$A16), a PIP2 molecule, and 99 POPC lipids in the top

leaflet and 100 POPC molecules in the bottom leaflet. In addition, the solution consisted of TIP3 (an all atom model of water) water molecules with 0.15 mM KCl. The simulations were performed in GROMACS using 303.15 K and 1 bar using CHARMM36 force field [61]. The file taken from the protein data bank contained an ENTH domain of epsin and a PIP2 head in the inserted configuration. The head group was replaced with the entire lipid, preserving the position and original orientation of the head group. Then, the POPC molecules were inserted in a grid pattern in order to construct the bilayer. Finally, the solution with the corresponding KCl concentration was added. All the files detailing the molecule and the equilibration procedure were obtained from CHARMM GUI website [62]. In order to compute different positions of ENTH with respect to the membrane, we pulled the protein sequentially starting from the inserted configuration by imposing a restraining force in the center of mass of the protein. Additional restraints were applied to the PIP2 lipid to prevent it from leaving the membrane. The pulling proceeds until the protein are not in contact with the membrane. All protein configurations were given an initial equilibration time of 70ns and a production run of 150ns. The secondary structure analysis tool of VMD [63] was applied on the last 100ns of the production run to obtain the secondary structure for each residue in a given frame. The helicity plot was subsequently created by calculating the ratio of frames classified as a helix divided by the total number of frames for each residue. The area plot for the inserted proteins were obtained using g-lomepro [64]. We used a 100ns production run after an initial equilibration of 100ns. Since the protein has no restrain of movement along the plane of the bilayer during the simulation, the frames had to be centered around the H_0 helix center of mass.

2.1.2 Results

Our simulations reveal an instantaneous interaction of the H_0 helix with PIP2 and the subsequent insertion of the H_0 helix into the membrane. The ENTH domain was then pulled away from the protein. As a consequence, the inserted H_0 helix first transitioned to an adsorbed state, and on further pulling was removed from the membrane into the solution. The aforementioned three stages of ENTH-membrane interaction are shown in Fig. 2.1a. It is notable that the secondary structure of the H_0 helix undergoes a transformation based on

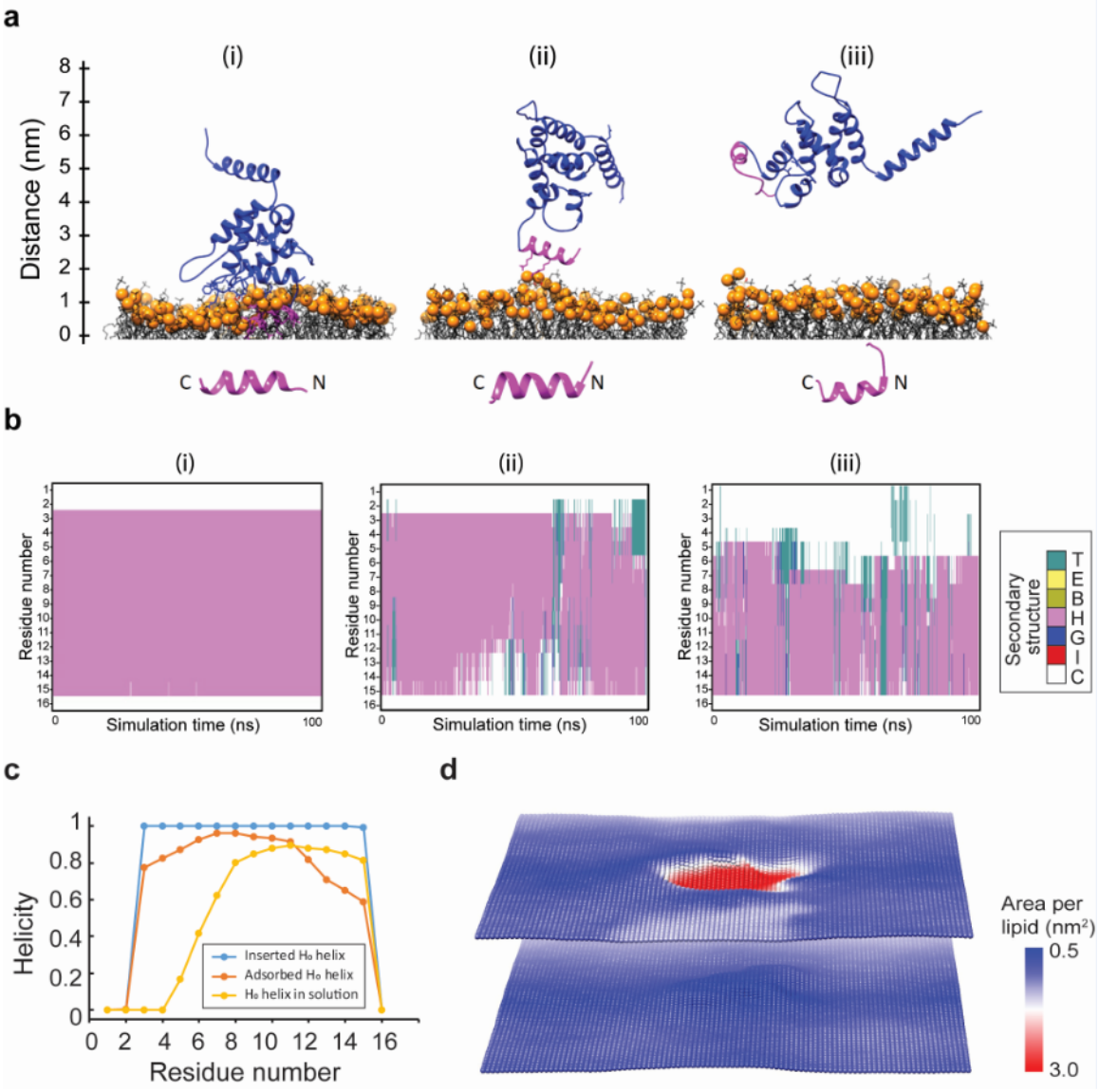

Figure 2.1: Atomistic insights into ENTH-membrane interactions. a) The H_0 helix shown in magenta is in the inserted (i), adsorbed (ii) and solvent (iii) states. b) Secondary structure analysis of each residue of H_0 in the three states shown in (a). c) The summary of the helicity analysis in (b). d) The area per lipid plot corresponding to the inserted state shown in (a).

the degree of lipid interactions (Fig. 2.1a bottom panel). Figure 2.1b shows the secondary structure analysis as a function of the three stages of the H_0 helix. In the inserted state, the H_0 has an alpha-helix structure (pink color). In the adsorbed state with reduced lipid interactions, the H_0 begins to become disordered from the N-terminus (teal color). In the final state when H_0 is in the solution, the alpha-helix domain shrinks further, transforming into the disordered domain (teal color). The extent of helicity of H_0 helix residues in the three states is summarized in Fig. 2.1c. Figure 2.1d shows the areal footprint of the H_0 helix inside the membrane. The plot shows the area per lipid in the two leaflets of the membrane. The POPC lipid area is around 0.64 nm^2 (blue color). Because of the H_0 helix insertion, the lipids are moved out of the H_0-occupied domain. This displacement of lipids effectively increases lipid–lipid separation, which in turn results in an increase in the area per lipid (red color). Since the protein sits primarily in the top leaflet, the change in area per lipid is minimal in the bottom leaflet. This areal footprint plot suggests a potential mechanism for tension sensitivity exhibited by epsin. A single H_0 helix occupies an area of 2 nm^2 (red region) and displaces lipids in the membrane. If the membrane has zero resting tension and the lipids are allowed to move freely, there would be no energetic advantage to displacing the lipids. However, if the membrane has a nonzero resting tension (σ), the displacement of lipids would be associated with an energetic incentive of $-\sigma\delta A$, where δA is the area occupied by the H_0 helix. This idea is similar to the notion that explains the tension sensitivity of mechano-sensitive channels in bacterial membranes [65, 66].

2.2 Atomistic insights into H_0-membrane interactions at non-zero tension

2.2.1 Methods

We again used the H_0 structure derived from the crystallography structure ($1H_0$A). The $1H_0$A pdb file contains an entire ENTH domain with 84 residue domains that are in contact with the membrane, attached via a PIP2 head. We used CHARMM-GUI to embed this structure in a POPC bilayer and construct the initial structure for the molecular dynamics

simulations. To this end, we positioned the ENTH domain in the middle of the bilayer with 200 lipids and an area of 64.1 nm^2. We then replaced the PIP2 head with a full PIP2 molecule. The top leaflet comprised of 99 POPC plus one PIP2 molecule and the bottom leaflet comprised of 100 POPC molecules. Finally, we solvated the system with a 50 solvation number and neutralized the system with 15 Mm concentration of K. Simulations were performed in GROMACS 2018 using CHARMM36m force field.

In order to investigate the effect of tension, we varied the area per lipid to yield tension values of 0 mN/m, 1 mN/m and 5 mN/m. The tension in a membrane is given by

$$\sigma = \frac{A - A_0}{A_0} K_A \tag{2.1}$$

where $A_0 = 64.1$ A^2 and $K_A = 270.6$ mN/m. In order to prescribe desired tension values, we rearranged the above equation to obtain

$$A = \sigma \frac{A_0}{K_A} + A_0. \tag{2.2}$$

This yields the values of 64.1 A^2, 64.34 A^2 and 65.28 A^2 for 0 mN/m, 1 mN/m and 5 mN/m tension values, respectively. We prescribed these area per lipid values to simulate the three tension values.

We performed umbrella sampling free energy simulations to estimate the free energy of H_0 inside and outside the membrane. We picked the vertical distance from the center of mass of the bilayer to the center of the mass of the protein as the reaction coordinate. This allows the protein to freely rotate and orient during the simulation. We pulled the protein from its equilibrium configuration inside the bilayer to the equilibrium configuration in the solution. Since there is a high binding affinity between H_0 and PIP2, we prescribed a restraint to avoid pulling of the lipid from the bilayer. We used the gmx wham tool to compute the free energy and the respective histograms. We employed 12 windows to map the free energy profile. Each window was simulated for 300ns with 0.002fs time step. We then filled the gaps between histograms and added new windows using the existing frames.

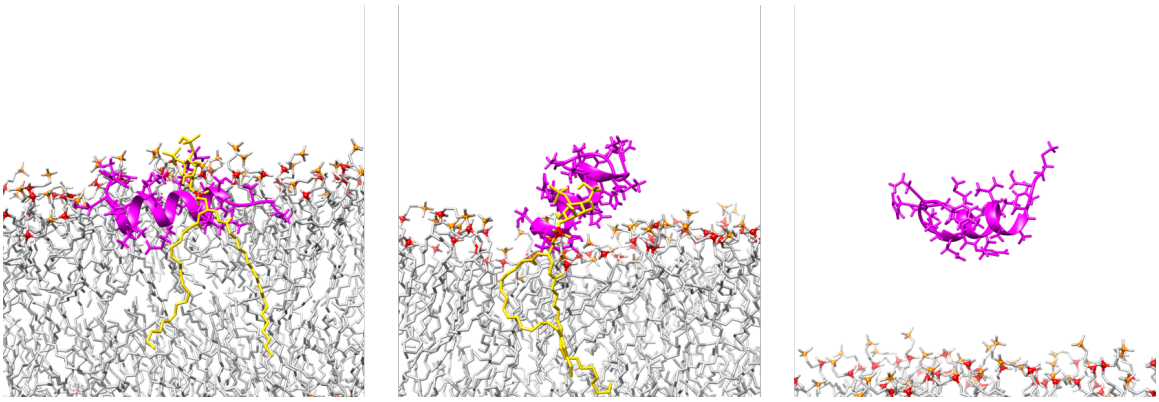

Figure 2.2: Three stages of H_0 helix (in magenta) during a pulling simulation. PIP2 lipid is shown in yellow.

2.2.2 Results

Fig. 2.2 shows H_0 helix in three different stages during a pulling simulation. The bilayer is subjected to 1 mN/m tension. In the first stage, the helix is embedded into the bilayer. In the second stage, the helix is adsorbed onto the bilayer. In the third stage, the helix detaches from the bilayer and goes into the solution.

To investigate the propensity of the helix to be buried into the membrane, we computed the free energy of the helix during the pulling simulation for the three tension values (Fig. 2.3). The plots reveal several critical features of membrane-protein interaction. First, the helix prefers to interact with the membrane, as the energy of the helix in the solution is much higher than that in the adsorbed or the embedded state. Second, there are two energy wells corresponding to the adsorbed state (right wells) and the embedded state (left wells). Third, in the absence of tension, both the embedded and the adsorbed states have similar energies. Thus, the protein can be expected to have equal likelihood of being in either of the states. Fourth, in the presence of tension, the energy of the embedded state decreases by 2.5 k_BT compared to the adsorbed state. This implies that the propensity of the helix to be embedded into the membrane increases with the prescribed tension. Fifth, the energy of the embedded state remains nearly identical for the two tension values. This suggests that the effect of tension is non-linear, and an increase in tension does not necessarily translate into further lowering of the energy of the embedded state. Sixth, the two tension curves have the minima at the same reaction coordinate from the bilayer center. This suggests

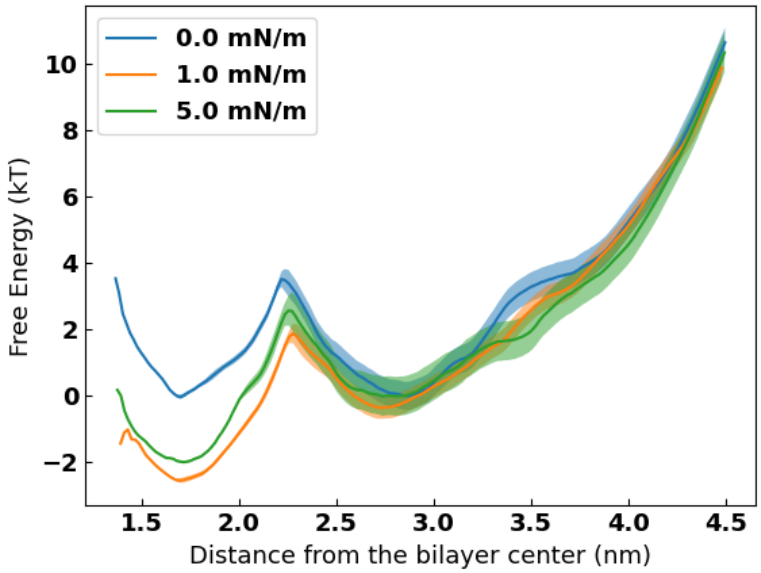

Figure 2.3: The free energy profiles of the H_0-membrane system for the three tension cases of 0 mN/m, 1 mN/m and 5 mN/m.

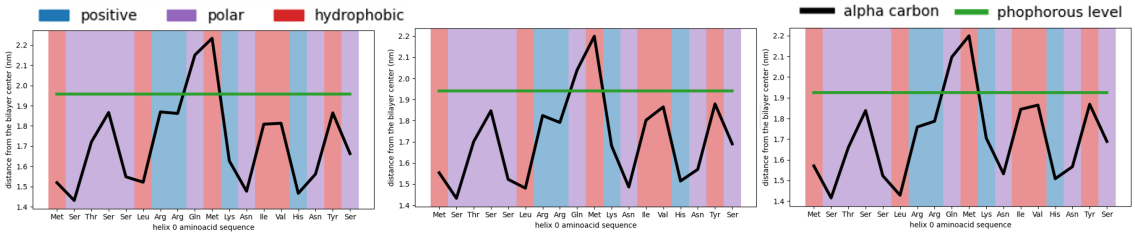

Figure 2.4: Height of the centroid of the resultant carbon atom density distribution from the center of mass of the bilayer.

that the level of insertion of the helix into the membrane is unaffected by increased tension in the bilayer.

We further investigated the above finding by computing the density distribution of the alpha carbon of each amino acid and the resulting height of the centroid of each resultant density distribution from the center of mass of the bilayer. Fig. 2.4 shows the height plots for the embedded helix for the three tension values. The height of the carbon atoms remain unchanged for the three tension cases. This confirms that the depth of the helix in the bilayer and also the orientation of the helix in the bilayer does not change with the prescribed tension values.

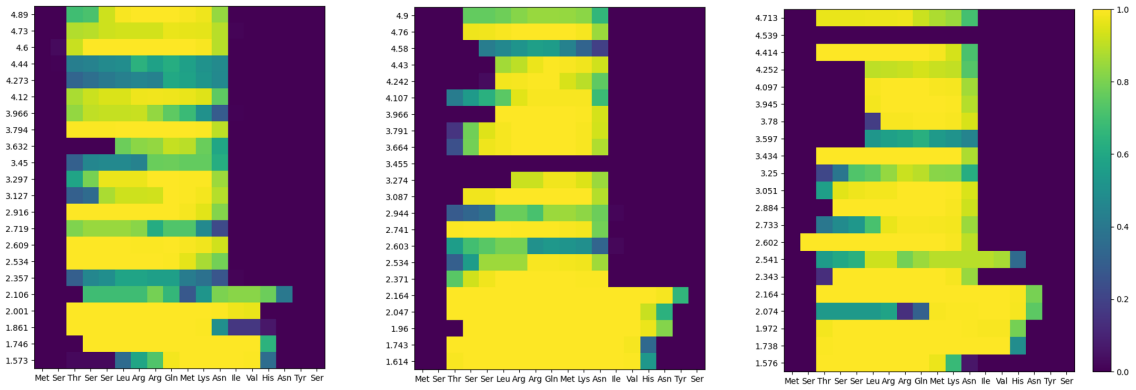

Figure 2.5: Helicity plot as a function of the residue number and the reaction coordinate for the three tension values (left: 0 mN/m; middle: 1 mN/m; right: 5 mN/m).

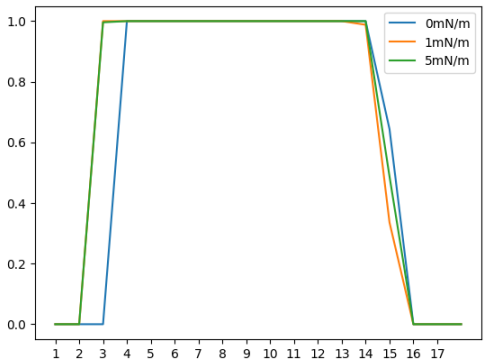

Figure 2.6: Helicity as a function of the residue number in the embedded state for the three tension values.

To further investigate the effect of tension on helix insertion, we examined the secondary structure of the H_0 helix for the three tension cases. Fig. 2.5 shows the helicity index as a function of the residue number in the embedded state of the helix. In the absence of tension, a residue called threonine is in the coiled state. However, for the two prescribed tension values, this residue transitions to an alpha-helical structure. This change might be related to the observed change in the free energy of the embedded helix in the bilayer.

2.3 Discussion

In this study, we investigated the effect of tension on the insertion of the H_0 helix of the epsin protein into a bilayer. Epsin is a membrane-bending protein that participates

in cellular trafficking. We computed the free energy of the membrane-protein system for three tension values. The free energy simulations reveal that the embedded state of the helix becomes energetically favorable in the presence of membrane tension. We observed that the two minima for the tension cases have 2.5 k_BT lower energy compared to the corresponding minimum at zero tension. Remarkably, the free energies at these minima for the 1 mN/m and 5 mN/m tension values are almost identical and are located practically at the same reaction coordinate. This indicates that the effect of tension on protein insertion in a membrane is non-uniform. It is conceivable that the tension effects could be similar to an ON and OFF switch. Below a certain threshold value, the effect of tension is non-existent, and above the threshold value, tension affects the secondary structure of the protein leading to stronger protein-lipid interactions.

To the best of our knowledge, this is the first energetic evidence of tension-dependent insertion of a protein in a membrane. This finding is of a general nature, and could help initiate future studies to investigate tension sensitivity observed in other membrane-remodeling proteins. It would be insightful to delve deeper into residue interactions and protein-lipid interactions to identify the origin of the energetic advantage seen in non-zero tension cases.

Chapter 3

Monte Carlo framework: Basic architecture

One of the main strategies for modeling physics of systems is using the Monte Carlo method. Every step of this algorithm starts with an unperturbed state. We then change the configuration of the system randomly. If the energy of the new state is less than the unperturbed state, we accept the new state. If the energy is more, we compute the ratio of the Boltzmann probability of the new state with respect to the unperturbed one. If the result is larger than a random number generated by a uniform distribution, we accept the transition, else we revert back to the original system. Because this method is based on an equilibrium distribution of states, the quantities obtained from analyzing this ensemble during the simulation are also equilibrium quantities.

In spite of its promising use in investigating molecular systems, the Monte Carlo method is not a silver bullet for modelling any system. If atomistic resolution is maintained in a model, one can only capture events that happen at most on a length scale of a micrometer and a timescale of microseconds. In order to go beyond these length and time scales, one can use a coarse grained system, where the number of degrees of freedoms can be reduced in favor of diminishing computational load. As we increase the scale of the system, the complexity of the inter-molecular interactions and the behavior they exhibit also undergoes enrichment. For atomistic studies, an atom is modeled as a point particle that interacts through bonded and non-bonded interactions irrespective of the system being studied. However, a model that simulates a coarse-grained polymer, needs to capture conformational changes and interactions between the coarse-grained beads that represent a polymer. Similarly, a

coarse-grained model that describes a lipid membrane needs to simulate its liquid crystal behavior. While all these systems can be satisfactorily modeled with Newton's law at the atomistic scale, they require a more detailed model, often guided by continuum theories at the macromolecular scale.

For this reason, there is a lack of coarse-grained computational platforms for simulating biological systems. While atomistic simulations have been standardized globally via computational software like GROMACS and LAMMPS, a similar mesoscale framework that is widely used by the scientific community is not available to the best of our knowledge. A challenge for such a computational framework is to deal with the multiplicity of interactions between a diverse set of entities. Another issue is defining a relevant behavior when multiple elements with very complex characteristics interact. Finally, entities in a mesoscale system, in contrast with atomistic systems, have an evolving complex 3D morphology. As a result, we are left with a daunting task of implementing computational models that accommodate multiple kinds of behaviors and interactions. Many times this entails re-writing large parts of the code every time we are required to add a new particle or a feature to a system. Also, the process of changing system morphology and editing entity interactions is very cumbersome and ad hoc.

In order to make progress and address this issue, we embarked on an ambitious journey to establish a coarse-grained framework to model membrane-protein interactions. In the subsequent sections and chapters, we describe the philosophy and the building blocks of this novel framework. We have refrained from going into minute details of the code with the objective of providing a concise yet complete overview to the reader.

3.1 Model Construction

The main priority of the framework is to allow flexibility for implementing any specific model while minimizing the computational cost for running and editing the framework. First, we define the rules that govern any model. These rules are independent of the objects being simulated, the space where the simulation is taking place, and the conditions under which the simulation is taking place. Second, we use a modular approach to build the

elements of a model. Instead of coding the behavior of a complex object like a protein and a membrane, we identify the physical features they have in common and incorporate them into the code. As a consequence, building a model simply requires assembly of these features in the configuration that most closely resembles the real phenomenon under investigation. In the following sections, we describe the key ingredients of the framework in greater detail.

3.1.1 Components

The components are the building blocks of a system that define its core physical features. They are based in the composition design pattern. The framework has six main components, namely, the protein body components, the protein binding sites, the non-bonded components, the bonded components, the membrane body components, and the membrane binding sites. In this section, we will explain the first four components in detail. We defer the description of the last two components to the next chapter, which is dedicated to the membranes.

Protein body components:

The protein body components represent the geometry of a molecule or an aggregate of molecules. Its primary function is to serve as a geometric scaffold to build the rest of the components. For this reason, a body component does not contribute to the Hamiltonian directly. But its configuration is used by the other components for computing the state and the energy of the molecules.

The available shapes for the body components are cylinders and spheres. In Table 1, we list them along with the physical parameters required to define these components. A sphere component is defined by its radius. A cylinder component is defined by its radius and length. Finally, an oriented sphere component consists of a regular sphere with a coordinate axis attached to its center to map its orientation.

Table 3.1: Body components and their physical parameters.

Shapes	**Parameters**	**Configuration**
Sphere	Radius	Center
Cylinder	Radius and length	Circular face centers
Oriented sphere	Radius	Center and axis

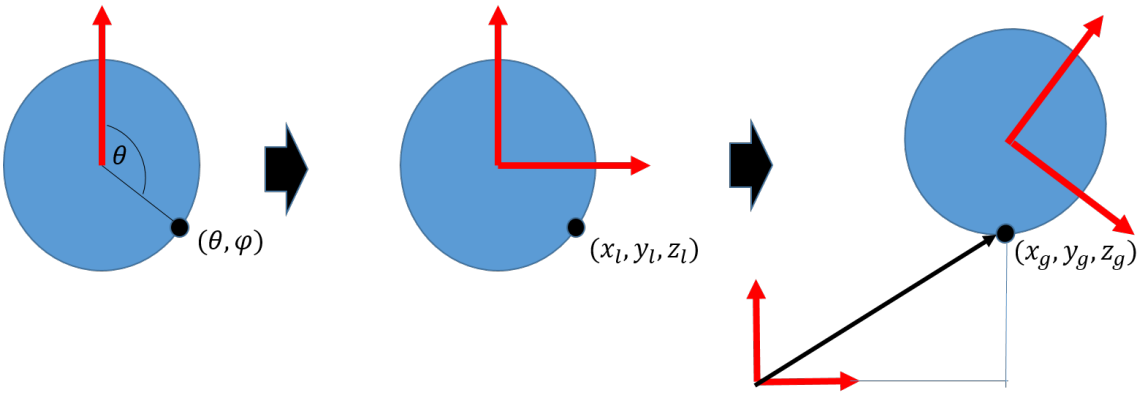

Figure 3.1: Transformation of the parametric coordinates of a binding site on a sphere to the global coordinates.

Protein binding sites: These components represent a point on a body component surface where bonded interactions can be established. They use a shape-specific parametric coordinate used to find the position, normal vector, and other properties in the local reference frame of the body component they are embedded on. Like the body components, their states do not contribute directly to the Hamiltonian, but are used by the bonded components to calculate the energy contributions. Fig. 3.1 illustrates an example of a binding component located on an oriented sphere. The binding site position is located by the local parametric coordinates $\{\theta, \phi\}$ defined on a sphere (left schematic). These spherical coordinates are then used to obtain local Cartesian coordinates $\{x_l, y_l, z_l\}$ of the binding site (middle schematic). These local Cartesian coordinates are then used in conjunction with the configuration of the oriented sphere to transform the local coordinates of the binding component to the global coordinates $\{x_g, y_g, z_g\}$ (right schematic). Because the local parametric coordinates of the binding component do not change during a simulation, we only need to invoke the last step to update the location of the binding site during a simulation.

In addition to the parametric coordinates, we characterize a binding site by its type. This parameter is used at the bond generation step to determine if a bond is possible between any two sites and to find the type of the bond that exists between these sites. A binding component can be occupied by a bond in two potential ways. First, one can a priory assign a bond to a binding component. Or second, a Monte Carlo move can create a bond based on system energetics. Likewise, bonds can also be vacated by the Monte Carlo

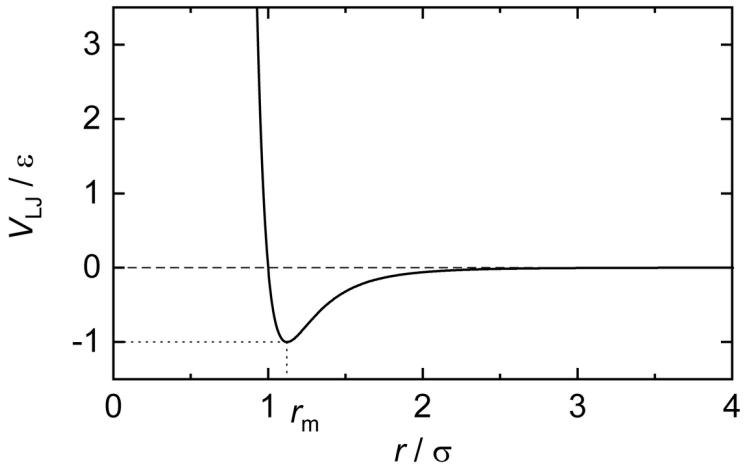

Figure 3.2: Non-bonded interactions between body components are modeled via Lennard-Jones potential.

binding step.

Non-bonded components

Non-bonded components are responsible for calculating the non-bonded energy of a molecule. They are always attached to a body component and use the relative position and orientation of body components to compute the interaction energy. The non-bonded interactions between any two molecules in this framework decay with the distance between them. Hence, beyond a certain threshold distance, called cutoff length, the interactions between two molecules become negligible. We therefore make the computation more efficient by calculating energy contributions located within the cutoff distance. For instance, Fig. 3.2 shows the Lennard-Jones potential between two molecules. As seen, the potential becomes practically zero beyond 2.5 times the radius of the molecule (σ).

One common strategy we used to take advantage of the locality of non-bonded interactions was to divide the domains and maintain a record of the components located inside each sub-domain. Therefore, a non-bonded component only needs to compute its energy contribution from the body components located in the same neighboring sub-domain. Fig. 3.3 shows a schematic of such neighboring sub-domains (in red) for a molecule (in yellow). In this framework, we use a space partitioning scheme called uniform grids, in which a domain is divided into equally-sized cubic bins. This scheme enables locating a molecule inside a bin in a constant amount of time.

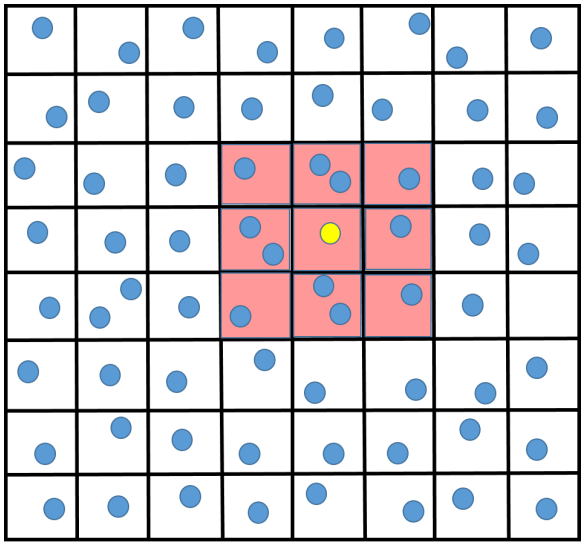

Figure 3.3: Space partitioning scheme used to define the neighborhood (in red) of a particle (in yellow) for computing non-bonded interactions.

In order to enable a switch between different potential types found in the literature, we created the concepts of non-bonded groups and non-bonded objects. A non-bonded group represents the potential used to calculate the non-bonded energy. Inside each group, we define the non-bonded objects that represent the potential function applied to a specific body type. For instance, in a general context, as seen in electrostatics, one can invoke different potentials based on sphere-sphere, cylinder-cylinder, or sphere-cylinder interactions. The objects refer to such entities that govern the mathematical expressions of the non-bonded potentials.

When we compute the interaction energy between two different non-bonded objects, the framework will search for the function that corresponds to the two object types. For instance, we have defined functions to calculate interactions between two spheres, between a sphere and a cylinder, and between two cylinders. Next, we use these functions along with individual parameters for each object to obtain the parameters for the non-bonded interactions between two different objects. For instance, if we have two spherical objects interacting via Lennard-Jones potential, we use the following combination rule to arrive at the effective parameters:

$$\sigma_{ij} = \frac{\sigma_i + \sigma_g}{2}, \text{ and} \tag{3.1}$$

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j}. \tag{3.2}$$

Above, $\{\sigma_i, \epsilon_i\}$ and $\{\sigma_j, \epsilon_j\}$ are the parameters associated with two spherical molecules with different sizes and chemical properties. To compute the non-bonded interaction energy between them, we use the effective parameters σ_{ij} and ϵ_{ij} along with the distance between these molecules.

Rigid Body interactions

In addition to the regular non-bonded interactions, this framework implements collision detection at the body component level. When a body tagged with collision detection undergoes movement, it browses its neighborhood for other bodies enabled with collision detection. When such bodies are detected, the framework performs a test that returns 'true' if the molecules intersect. In this case, the current Monte Carlo move is aborted and all the changes made are reverted. Similar to the non-bonded interactions, the possible collision partners of a body are identified within its neighborhood. Hence, we use the same space partitioning scheme described previously in order to reduce the computational cost for executing collision detection.

In this framework, we divide the bodies on which we perform collision detection in groups. These groups are used to define the collision partners. For instance, we can define a collision group for a membrane and another group for molecules in the solution. We can then tell the framework to track collisions between the membrane and the particles found in the solution. In addition, we can also tell the framework to detect self-collision within the membrane. We would like to note that molecule-molecule collisions are automatically avoided by volume exclusion implemented via non-bonded components.

Bonded components

Bonded components implement interactions that depend on the configuration of two or more binding sites. According to the number of binding sites participating in an interaction, these components can be divided into pair bonds with two binding sites, angle bonds with three binding sites, and dihedral bonds with four binding sites.

The pair bond uses the global coordinates of the binding sites to compute the interaction energy. In the simplest case of a harmonic potential, we use the distance between two

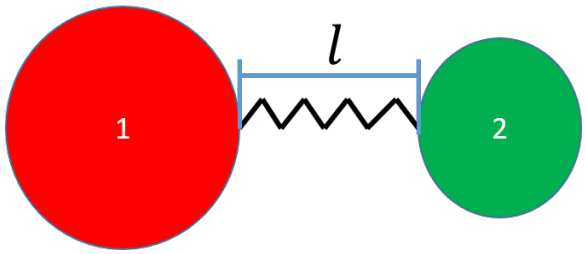

Figure 3.4: The bonded interactions between two body components is modeled via harmonic potential.

interacting binding sites to compute the energy. Fig. 3.4 shows a typical harmonic potential between two molecules, represented by a spring. The potential is given by

$$e = k(l - l_0)^2 - e_0. \tag{3.3}$$

where k is the stiffness of the spring, l_0 is the equilibrium distance between the molecules, l is the current distance between the molecules during a simulation, and e_0 is the biding energy that is released when the bond is created.

Another example of bonded interaction is the hinge potential, where we penalize the bending of the line that connects two binding sites. Fig. 3.5 shows a typical hinge potential, for which the energy depends on the angle defined by the bond vector (black arrow) and the normal vectors of the binding sites ($\vec{n}_1$ and $\vec{n}_2$). Any change in the angle between $\vec{b}$ and $\vec{n_1}$ or $\vec{n_2}$ from the reference configuration ($\{\theta_1^0, \theta_2^0\}$) to the current configuration ($\{\theta_1, \theta_2\}$) implies bending of the bond vector in the plane defined by the bond vector and the normal vectors. In a biological context, such bending could be used to model bending of multi-legged protein structures such as clathrin. A simple way to express the energy of such a bond is

$$e = k(l - l_0)^2 - e_0 + k_{\theta_1}(\theta_1 - \theta_1^0)^2 + k_{\theta_2}(\theta_2 - \theta_2^0)^2. \tag{3.4}$$

where k_{θ_1} and k_{θ_2} are the module associated with the rotational springs.

The angle bond uses the angle between two pair bonds $\{\vec{b}_1, \vec{b}_2\}$ emanating from a common molecule, as shown in Fig. 3.6. We use a simple harmonic potential in the framework to penalize the deviation of the angle θ from the preferred angle in the reference configuration. The dihedral bond accounts for the interaction between six binding sites. It penalizes

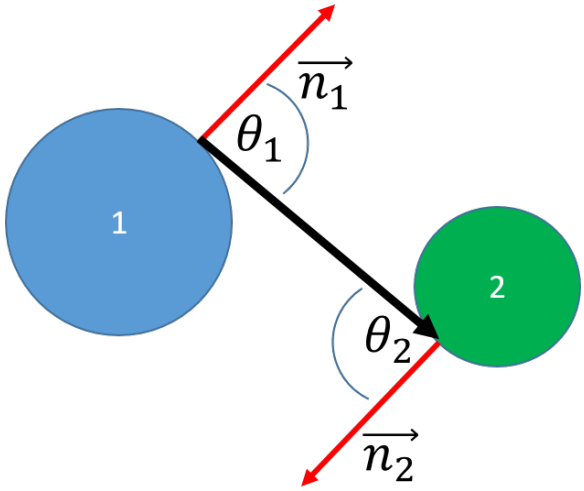

Figure 3.5: Schematic showing a hinge bond between two body components and the relevant parameters used to define the potential.

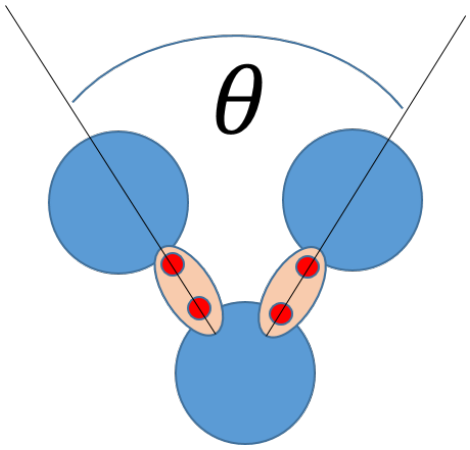

Figure 3.6: The angle bond between three body components is characterized by the angle θ shown in the schematic.

the dihedral angle θ formed from the three bond vectors $\{\vec{b}_1, \vec{b}_2, \vec{b}_3\}$ as shown in Fig. 3.7 via a harmonic potential.

3.1.2 Groups

After declaring the components, we join the components in a specific way to create groups that represent the objects we plan to simulate in the biological phenomenon under investigation. Currently, the framework supports two types of groups: molecules and membrane sites. Each type of the group follows a fixed recipe. For instance, a molecule

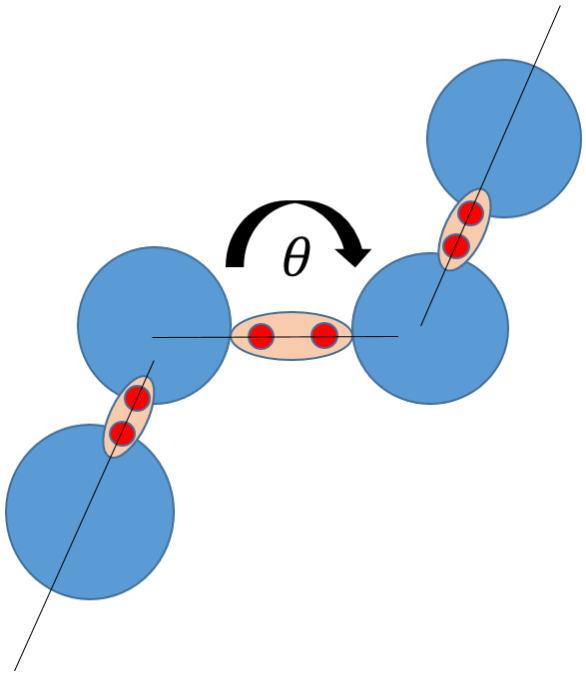

Figure 3.7: A schematic showing a dihedral bond between four body components.

always starts with a body component, to which a non-bonded component and/or a binding site may be attached. Fig. 3.8 shows a typical molecule constructed as a group from three components: the spherical body component (in green), a non-bonded component (in blue) and a binding sites (in red).

After declaring a group, the framework stores the information of the components and their connections. Whenever we create an instance of a group, each component is created individually and connected according to the group type. The way the group is connected in the framework is shown in the Fig. 3.9. The connection in turn defines the flow of the information. For example, since a body component contains information only about the position and the orientation, it does not care about the state of a binding site present on it. However, the binding site is required to know any change in the state undergone by a body component in order to recompute its new state.

3.1.3 Reactions

Reactions are transitions from one group called the reactant to another group called the product. They are executed by mapping the reactant components to the product

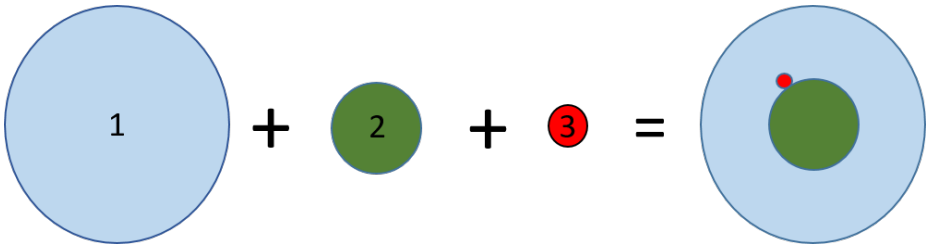

Figure 3.8: A schematic showing a group composed of a body component in green, a non-bonded component in blue and a binding site in red.

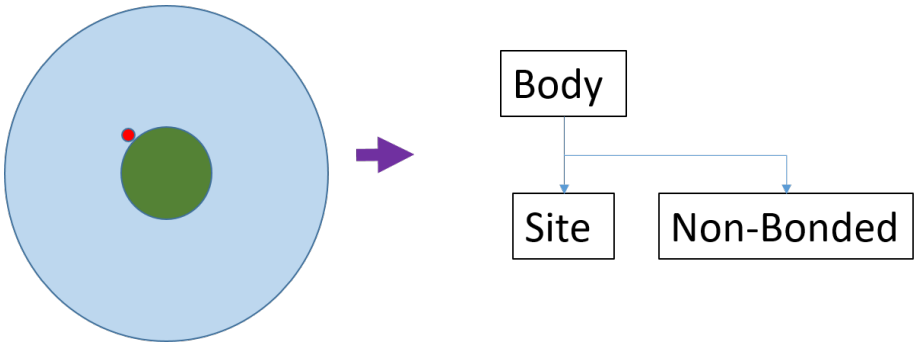

Figure 3.9: Flowchart showing information flow in a group.

components. All the components from the reactant that are not mapped to a product component are deleted. Likewise, all the product components that do not originate from a reactant component are created.

Reactions are activated through a combination lock system. Reactants internally have a set of flags, each representing a specific feature of one of the reaction component state. For instance, the most commonly used flag describes the occupancy of a binding site component. For every change to a reactant component, the relevant flag is updated. Then all the flags are submitted to a Boolean operation. If the output is true, then a reaction is activated. For example, Fig. 3.10 shows a reaction where the molecule on the left changes to the molecule on the right. A spherical body component could undergo a potential change in size (from blue to green sphere). Similarly, the non-bonded component could undergo a change in the potential. The reaction leads to the deletion of the purple binding site and creation of the yellow binding sites. The red binding site remains, but it gets assigned to a different group after the reaction has taken place.

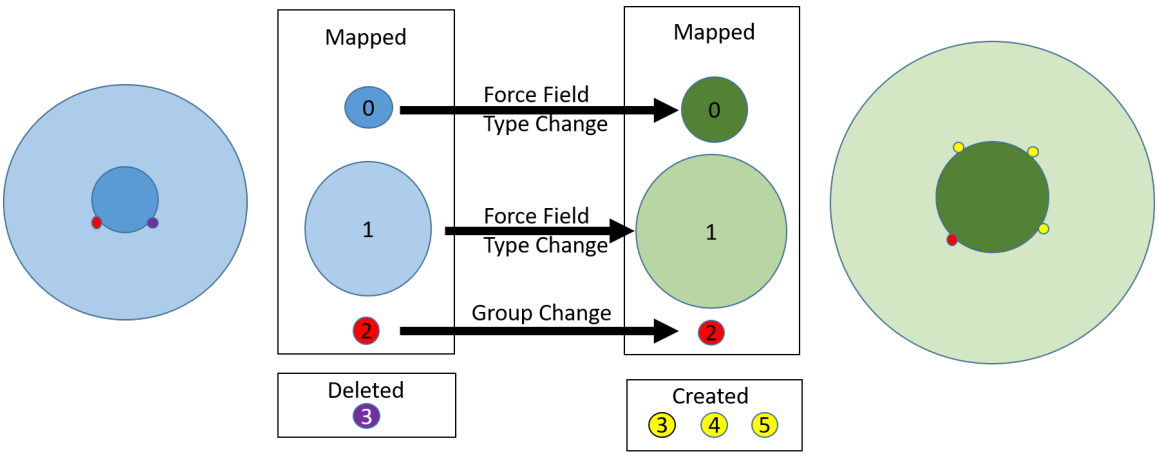

Figure 3.10: A schematic showing a typical reaction which transforms a group into another group. An old binding site (in purple) is deleted and three new binding sites (in yellow) are created.

3.2 Domain

Once we define the components and create the groups that they are going to be placed in, we build our system and notify the framework about the location of the molecules. If we start a simulation and permit unrestricted movement of molecules in the 3D space, the system will soon diffuse out because of entropic forces. Therefore, we need a method to keep the molecules constrained inside a subset of R^3.

In order to solve this issue, we introduce the concept of a boundary condition. It is a function that takes the coordinates in R^3 and maps them to a subset of R^3 in which the simulation takes place. The simplest implementation of this concept is that whenever an object crosses a boundary box, we set its new position at the boundary. However, this type of mapping might generate artifacts as particles will start to accumulate at the boundary. The most wide spread and commonly used mapping that avoids this limitation is called the periodic boundary condition. It consists of two surfaces separated by a fixed distance vector ($\vec{d}$) (Fig. 3.11). Whenever a molecule crosses one of the surfaces, it is mapped back to the simulation domain by subtracting the fixed distance vector ($\vec{d}$) from its position vector. The rationale behind this concept comes from the structure of crystalline materials. It consists of unit cells which are copies of each other and are tiled next to one another so that they occupy the whole extent of the material volume. In this case, the simulation domain can

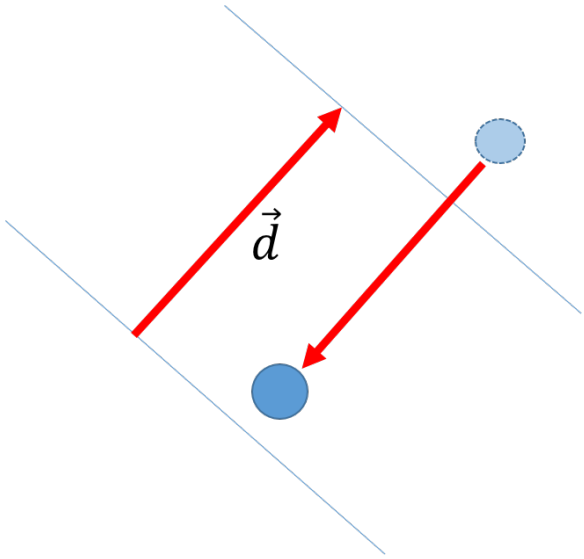

Figure 3.11: A schematic showing the generalized notion of periodic boundary condition. The boundaries are defined by the blue lines and a sample particle by a blue circle.

be thought of as a unit cell and when a particle crosses the boundary, it can be considered to move to the neighboring unit cell. As the unit cells are copies of the same system, a molecule enters from the opposite direction from an adjacent unit cell into the simulation domain.

3.3 Monte Carlo steps

Once we create a domain and put all the copies of the groups in place, we need to define the internal variables and the degrees of freedom of the system. A degree of freedom is a quantity that is part of the state of a component and is independent of other parameters. For instance, the position of a molecule is a set of degrees of freedom. In contrast, the position of a binding site is determined from the global position of a molecule and the local parametrization, and hence is not a degree of freedom. Another common example of a degree of freedom is a bond that can form or break independently. For example, a pair bond between two binding sites is a degree of freedom, whereas an angle bond is dependent on the pair bond and is therefore not a degree of freedom. The number of degrees of freedom depends on the type of molecule. The centroid of a body has three degrees of freedom,

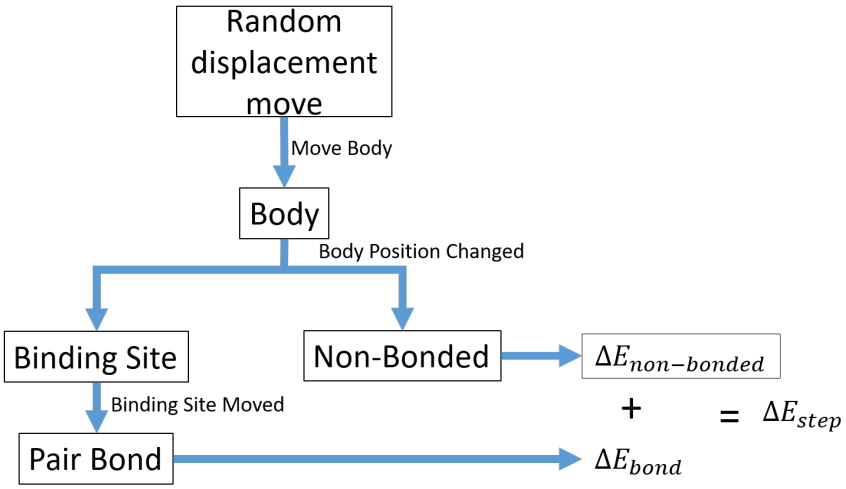

Figure 3.12: A typical Monte Carlo position step and its downstream events related to the constituent components.

corresponding to each component in the Cartesian system. The body orientation also has three degrees of freedom, corresponding to the Euler angles. A pair bond between two binding sites has only one degree of freedom.

3.3.1 Monte Carlo Positional Step:

Here, we perturb our degrees of freedom in order to minimize the system energy. We calculate the perturbation distance by multiplying a random vector computed inside a unit sphere and multiply it by the maximum permissible step size. Next, we select a random molecule and move it by the distance we previously calculated. Then, we calculate the new states of the affected components and find the resultant energy from this positional change. Finally, we apply the metropolis algorithm to check if the current state is to be accepted or has to be reverted back. Similarly, if the molecule has a rotational degree of freedom, we select a random axis and a random angle of rotation to compute a rotation matrix in order to reorient the molecule. Fig. 3.12 shows one typical Monte Carlo position step and its consequence downstream on related components.

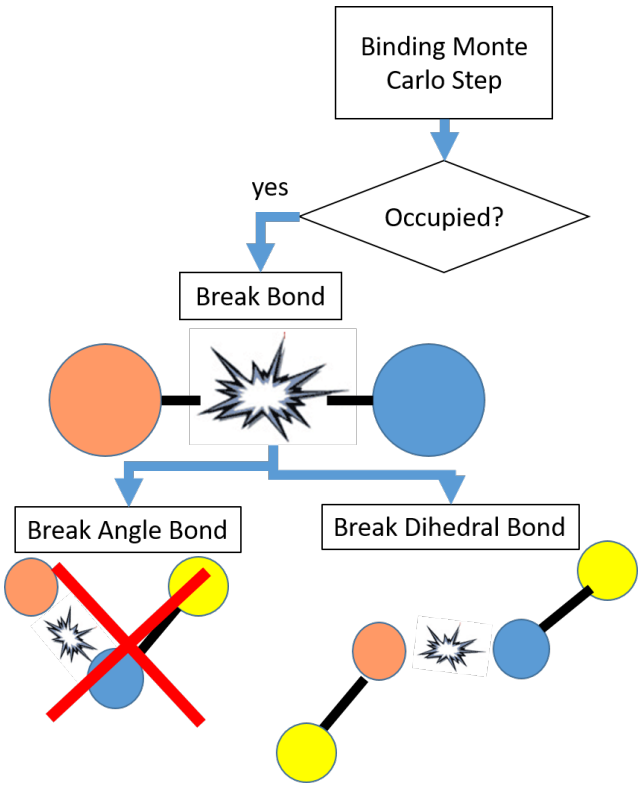

Figure 3.13: A typical Monte Carlo binding step that shows disruption of bonds.

3.3.2 Monte Carlo binding step

Here, we select a random binding site and check whether it is occupied or not. If it is occupied, we break the pair bond and destroy all the angle bonds and the dihedral bonds that depend on it (Fig. 3.13). Next, we subtract all the energies from the destroyed components and apply the metropolis algorithm to check if the current step is to be accepted or rejected. If the site is empty, we begin to search its neighborhood for another empty site. Once a permissible site is found, we create a pair bond and the resulting angle and dihedral bonds (Fig. 3.14). Next, we add all the energies generated by the creation of these components and apply the metropolis algorithm to accept or reject the move.

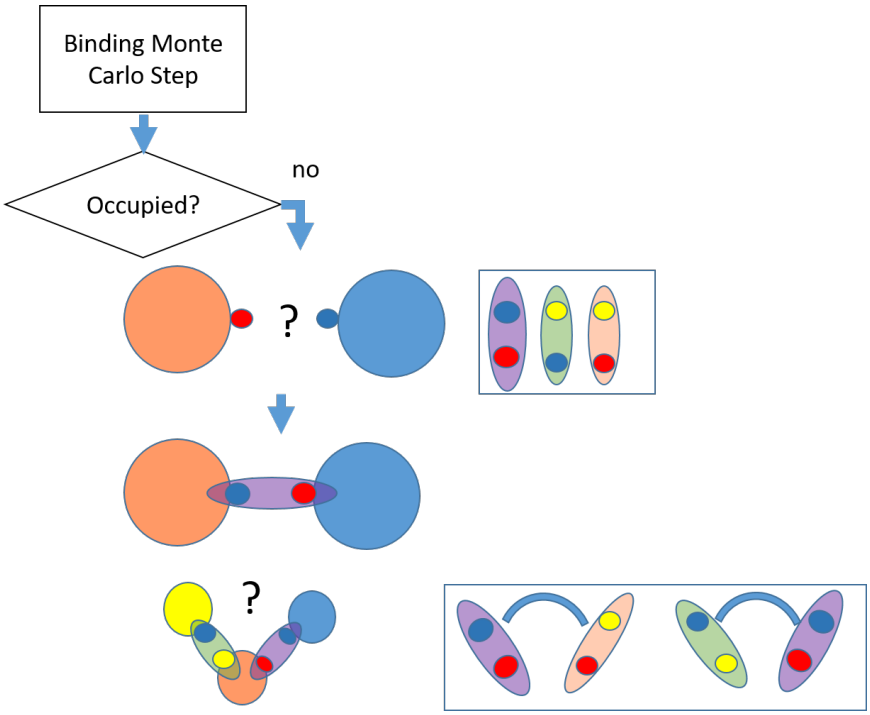

Figure 3.14: A typical Monte Carlo binding step that shows creation of new bonds.

Chapter 4

Monte Carlo framework: Membrane model

Lipid membranes have a thickness on the order of 4-5 nm, whereas the in-plane dimensions can range from hundreds of nanometers to microns [67]. As a consequence, membrane are represented as 2D surfaces embedded in a 3D space. These membranes are made of amphiphilic lipid molecules that resist bending and stretching deformations as they change their exposure to the surrounding aqueous medium. In contrast, the lipid molecules can move freely on the membrane surface. Because of these contrasting characteristics, a lipid membrane behaves as a 2D elastic film in the out-of-plane deformations and a 2D fluid in the tangential plane.

The Hamiltonian for such a membrane with N domains is given by [68, 15]:

$$H_{memb} = \sum_{i=1}^{N} \int_{\Omega_i} (c - c_{0i})^2 \kappa_i \; dA + K_A \frac{(A - A_0)^2}{2A_0} + K_V \frac{(V - V_0)^2}{2V_0} + \sum_{i=1}^{N} \sum_{j=1}^{N} \oint_{\Gamma_{ij}} d_{ij} dl. \quad (4.1)$$

Above, the first term is a summation of the bending energies over all lipid domains. In this term, c is the mean curvature at any point on the 2D surface, c_{0i} is the spontaneous curvature which captures the preferred mean curvature of the ith domain, and κ_i is the material parameter, called the bending modulus, of the ith domain. This term penalizes any deviation of the mean curvature of the surface from the preferred curvature. A membrane can have a non-flat geometry as a preferred shape, either due to the molecular structure of the lipids or due to its interactions with the proteins. Since each domain can have a different lipid type, the spontaneous curvature and the bending modulus could be different for each domain on the surface. The second term is the global area constraint where A is the

area of the membrane in the current configuration, A_0 is the reference area, and K_A is the stretching modulus of a membrane. This constraint is imposed because lipid membranes are only able to undergo 2-3% areal dilation before undergoing rupture. Thus, for all practical purposes, the membrane can be assumed to possess a constant surface area. The third term is the global volume constraint where V is the volume enclosed by the membrane in the current configuration, V_0 is the equilibrium volume in the reference configuration, and K_V is the compressibility modulus of a membrane. This constraint is imposed because lipid membranes are non-leaky, and hence they maintain their internal volume while undergoing shape transformations. Finally, the last term accounts for the interaction energy between any two neighboring domains that might arise due to different lipid compositions via a line energy d_{ij}. The total energy from this term is obtained by performing summation over all the edges Γ_{ij} of the N domains.

4.1 Discrete surface differential operators

For a 2D surface, the mean curvature of a point can be calculated by applying the Laplace-Beltrami operator to its position vector [69]

$$c\vec{n} = \nabla\nabla\vec{x} \tag{4.2}$$

where ∇ represents the surface gradient operator and $\vec{n}$ is the normal vector and c is the mean curvature.

For a triangular mesh, we can discretize the operator and apply it in the follow way: To calculate the mean curvature at node i, we first iterate through every neighbor j and compute a parameter σ_{ij}, where

$$\sigma_{ij} = \frac{l_{ij}}{2}(\cot\alpha_{ij} + \cot\beta_{ij}). \tag{4.3}$$

Above, l_{ij} is the length of the vector that connects nodes i and j, and α_{ij} and β_{ij} are the angles opposite to l_{ij} in the two adjacent triangles (Fig. 4.1). The resultant quantity σ_{ij} represents the length of a side that intersects l_{ij} perpendicularly in a dual mesh, shown in

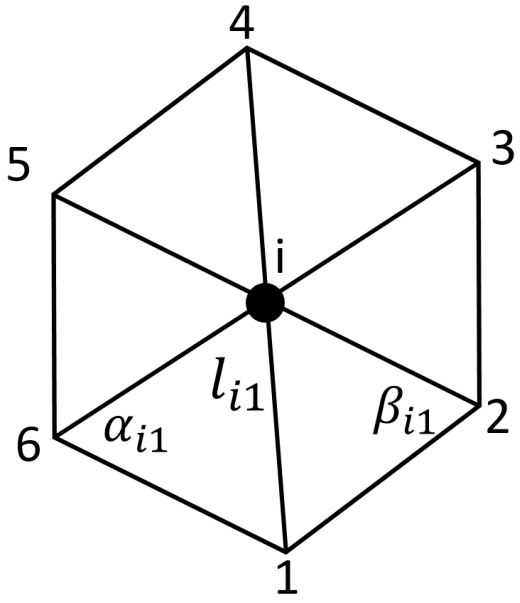

Figure 4.1: A local patch of triangulated mesh around a point (dark circle) on a membrane surface. The schematic shows the edge and the angles involved in the discretized calculation of the mean curvature at the point under consideration.

blue in Fig. 4.2. Next, we calculate the mean curvature by

$$c\vec{n} = |\sum_{n=1}^{N_{tri}} \frac{\sigma_{ij}}{l_{ij}}(\vec{x_i} - \vec{x_j})| \tag{4.4}$$

where $\vec{x}_i$ and $\vec{x}_j$ are the position vectors of point i and point j, respectively [12].

In order to calculate the curvature tensor at any point on the membrane surface, we follow the numerical approach of the least square fitting [69]. To do so, we first define a coordinate frame at each mesh point. We create a coordinate triad by defining the tangent, the cotangent and the normal vectors. The tangent vector is a unit vector parallel to one of the sides of the triangle passing through the mesh point. The cotangent vector is then given by the cross product of the normal vector with the tangent vector. Next, we define a distance vector between mesh points i and j lying on the tangent plane. This is the projected vector of the $(\vec{x}_i - \vec{x}_j)$ on the tangent plane and can be expressed as

$$\vec{d} = \{\vec{t} \cdot (\vec{x}_i - \vec{x}_j)\}\vec{t} + \{\vec{c} \cdot (\vec{x}_i - \vec{x}_j)\}\vec{c}. \tag{4.5}$$

The normal curvature in a direction of $(\vec{x}_i\vec{x}_j)$ is given by

$$c_{ij} = 2\frac{(\vec{x}_i - \vec{x}_j)\cdot(\vec{n}_j)}{|(\vec{x}_i - \vec{x}_j)|^2}. \tag{4.6}$$

Next, we compute the curvature tensor

$$\vec{B} = \begin{pmatrix} c_t & \tau \\ \tau & c_c \end{pmatrix} \tag{4.7}$$

with respect to the tangent and cotangent vectors. Above, c_t and c_c are the normal curvatures along the tangent and the cotangent vectors, and τ is the twist. The curvature tensor yields the tensor invariants, namely, the mean curvature

$$c = (c_t + c_c)/2 \tag{4.8}$$

and the Gaussian curvature

$$\kappa = c_t c_c - \tau^2. \tag{4.9}$$

The normal curvature in the $(\vec{x}_i\vec{x}_j)$ direction can also be computed with the help of curvature tensor using

$$c_{ij} = \vec{d}^T\vec{B}\vec{d}. \tag{4.10}$$

We can then pose a least-square approximation problem and define an error

$$E(c_t, c_c, \tau) = \sum_{j=1}^{j=N}(c_{ij} - \vec{d}^T\vec{B}\vec{d})^2. \tag{4.11}$$

subject to the constraint

$$c = \frac{c_t + c_c}{2}. \tag{4.12}$$

The minimization of the error yields the three components of the curvature tensor.

Next, we compute the area enclosed within the dual mesh σ_i given by,

$$\sigma_i = \frac{1}{4}\sum_{i=1}^{N}\sigma_{ij}l_{ij} \tag{4.13}$$

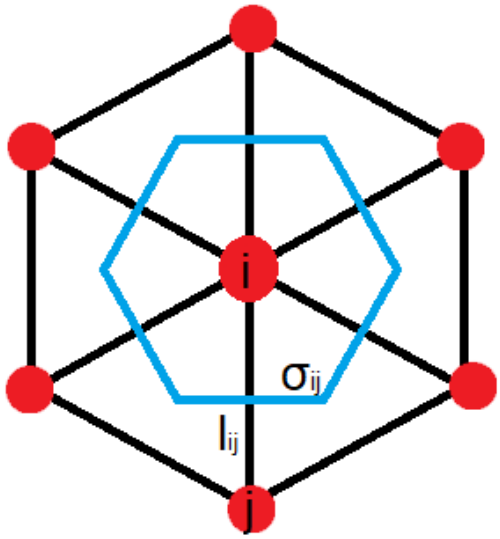

Figure 4.2: The schematic shows the dual mesh (in blue) generated on a local patch of membrane to compute the local differential geometric parameters such as curvatures, surface area and perimeters.

where j is a neighbor index and N is the number of neighbors. Finally, we calculate the bending energy of the membrane enclosed by the dual mesh by evaluating

$$dE = (c - c_0)^2 dA = (c - c_0)^2 \sigma_i. \tag{4.14}$$

Next, we compute the total area of the membrane as a sum of the area of each individual triangle given by

$$A = \sum_{i}^{N_{tri}} A_i = \sum_{i}^{N_{tri}} \frac{|\vec{l_{ij}} \times \vec{l_{jk}}|}{2} \tag{4.15}$$

where $\vec{l_{ij}}$ and $\vec{l_{jk}}$ are the sides of a triangle. Then, we find the volume inside the membrane by adding the signed volume of each tetrahedron generated by each triangle of the mesh and the origin. The volume of a tetrahedron is easily obtained from the triple scalar product formula

$$V = \sum_{i}^{N_{tri}} V_i = \sum_{i}^{N_{tri}} \frac{\vec{l_{ij}} \times \vec{l_{jk}} \cdot \vec{x_i}}{6}. \tag{4.16}$$

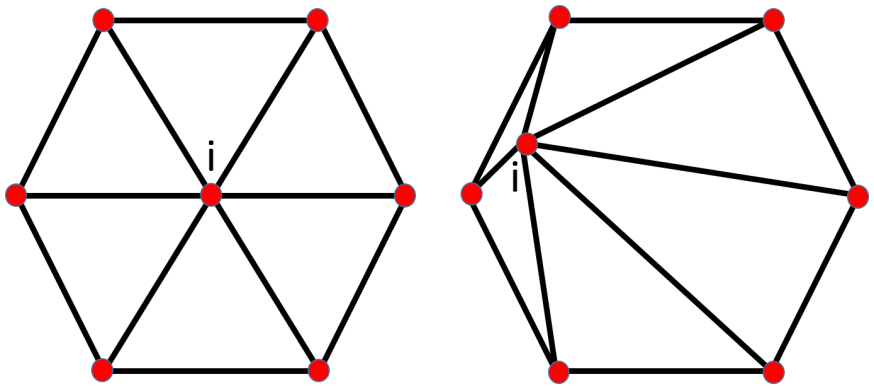

Figure 4.3: The schematic shows the distortion of a triangular mesh upon in-plane deformation undergone by a membrane due to lack of shear resistance. Such deformation often leads to very skewed triangles.

finally, we compute the boundary energy by using the lengths of the dual mesh as

$$E_{bound} = \sum_{i}^{N_{l_{ij}}} \sigma_{ij} d_{ij}. \tag{4.17}$$

4.2 Membrane model implementation

The main feature that sets apart a fluid membrane from a solid membrane is the lack of an energy term in the Hamiltonian that penalizes any in-plane distortion. For example, in Fig. 4.3 a patch on the left can undergo a fluid-like movement in which the central node relocates giving rise to the patch on the right. Since the movement happens in a plane, the curvature of the patch remains unchanged. Therefore, the bending energy in the Hamiltonian before and after the node movement remains the same. As a result, all in-plane movements in a Monte Carlo simulation will be automatically accepted. This in turn can lead to a computational challenge due to a lack of shear resistance. Furthermore, a simulation with almost perfect equilateral triangles may end up with a heavily distorted mesh with extremely skewed triangles with huge variations in sizes, as seen in the right schematic of Fig. 4.3. This will inevitably lead to a degradation of the numerical method with every step, eventually culminating in a failed simulation.

We circumvent this challenge by implementing the following strategies [14]. First, we avoid skewed triangles by limiting the range of their possible shapes. In particular, we

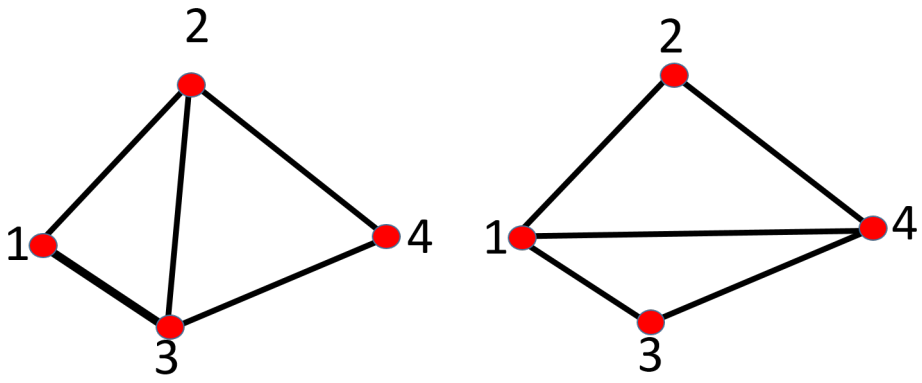

Figure 4.4: A schematic showing the flipping of vertices in a discretized mesh that models the in-plane fluid nature of the surface.

restrict the length of the sides of the triangles between a minimal and a maximal value. If the length of any side goes out of this prescribed range, we reject the change. Next, we add an extra step to allow unrestricted movement of mesh points to capture in-plane fluid behavior of lipids. To accomplish this, we flip the sides that connect any two triangles, as shown in Fig. 4.4. As a result, points on the surface can have unrestricted movement in a plane as is expected in a 2D fluid material without generating any energetic cost.

Barycentric coordinates:

Barycentric coordinates [70] are used to define the location of a point P within a triangle (Fig. 4.5). They are defined by

$$b_i = \frac{t_i}{A} \tag{4.18}$$

where t_i is the area of the triangle formed by the point P and the opposite vertices to the point i, and A is the total area. These coordinates obey the relation:

$$t_1 + t_2 + t_3 = 1. \tag{4.19}$$

Also, for any point inside the triangle, the following inequality holds $0.0 < t_i < 1.0$. These coordinates are widely used in many fields that deal with triangular meshes because of their superior interpolation capabilities. Any property defined at a mesh node can be interpolated

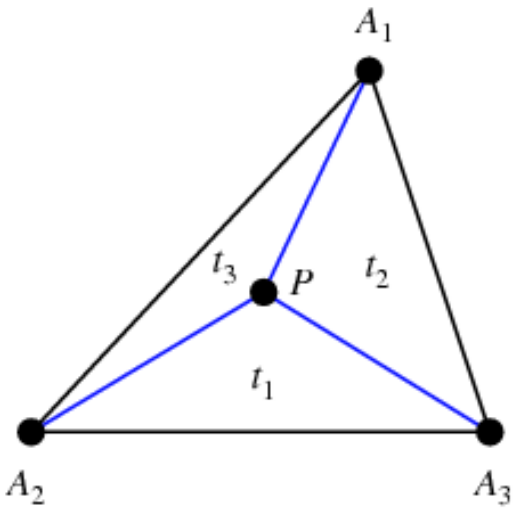

Figure 4.5: A schematic showing the areas of the discretized triangle used for computing the barycentric coordinates. These coordinates are used to interpolate physical quantities inside the triangle $A_1A_2A_3$.

using these coordinates by invoking

$$a_p = b_1a_1 + b_2a_2 + b_3a_3 \tag{4.20}$$

where a_i is the property at vertex i and a_p is the interpolated property at node P.

Membrane coordinates:

The membrane coordinates define a point embedded on the 2D surface. They are used by other components to find the position vector, the normal vector, the curvatures, specie concentrations and other attributes at that point. These coordinates consist of an integer that represents the index of the triangle where the point is located, and three floating point numbers that represent the barycentric coordinates of the triangle.

Point transport:

Often we need to move a point represented by membrane coordinates along a given 2D random vector $\vec{d}$ lying on the surface in a Monte Carlo move. Because our surface is represented by a triangular mesh, we have to map the movement along this 2D vector inside a triangle in a global frame of reference. In order to move the point inside a triangle, we first define an internal frame of reference of the triangle. To define this frame, we first select

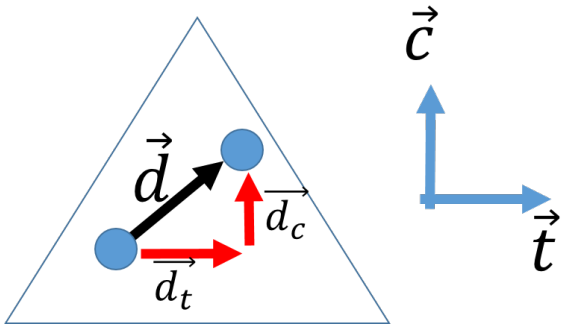

Figure 4.6: The schematic shows the local frame used to transport points on the discretized membrane. $\vec{d}$ is the displacement vector, $\vec{t}$ is the tangent vector, and $\vec{c}$ is the cotangent vector.

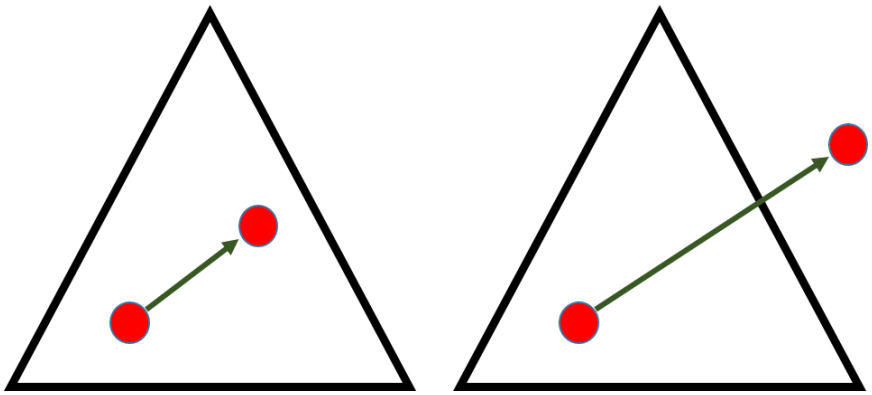

Figure 4.7: A schematic showing the potential movement of a point outside the triangle due to a Monte-Carlo move.

one of the triangle sides (side ij in Fig. 4.6) and normalize it to obtain a tangent vector $\vec{t}$. Next, we take the cross product of this tangent vector with the triangle normal $\vec{n}$ to compute a cotangent vector $\vec{c}$ (Fig. 4.6). The internal reference frame is then given by the triad consisting of the tangent, the cotangent and the normal vectors. Once the frame is defined, we use the projections of the vector $\vec{d}$ to compute the global displacements along the reference axes. We then use these values to arrive at the new membrane coordinates of the point.

When moving a point inside a triangle, two things can happen, either it crosses one of the sides and goes to another triangle, or it remains within the bounds of the triangle (Fig. 4.7). In order to determine which scenario is applicable, we have to find the intersection of the displacement vector with each of the triangle sides. To achieve this, we define a segment which starts with the initial location of the point under consideration and is oriented along

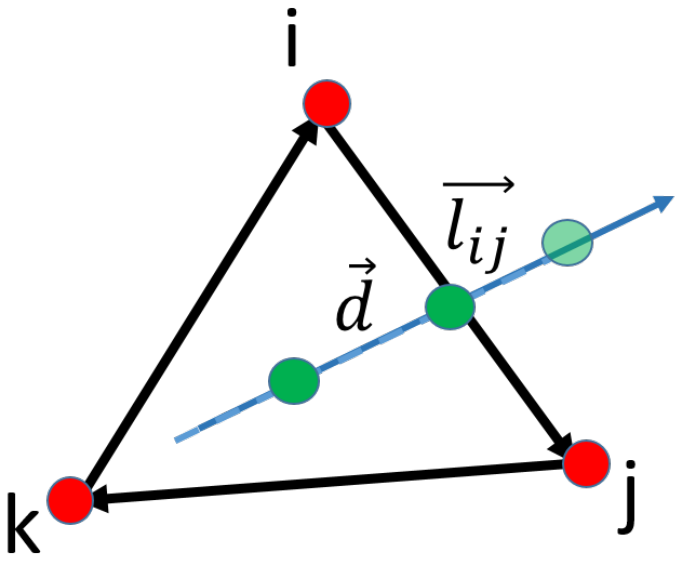

Figure 4.8: The schematic shows the layout of the numerical scheme used to reassign a point to a new triangle once it crosses the boundary of the original triangle. The scheme checks for intersection of $\vec{d}$ with the side vectors of the triangle.

the vector $\vec{d}$. This segment can be expressed in parametric form as

$$\vec{x}_t = t\vec{d} + \vec{x}_0 \tag{4.21}$$

where $\vec{x}_0$ is the starting point, $\vec{d}$ is the distance vector, and t is a parametric coordinate ($0 \leq t \leq 1$). The extreme values of $t = 0$ and $t = 1$ correspond to the initial position $\vec{x}_0$ and the final position $\vec{x}$, respectably.

To check intersection with a side $\vec{l}_{ij}$ of the triangle, we define another segment

$$\vec{x_s} = s\vec{l}_{ij} + \vec{x}_i \tag{4.22}$$

along this side. Any two lines are either parallel or they intersect with each other in the Euclidean space. In our case, intersection occurs if the two segments defined above intersect along the side $\vec{l}_{ij}$. For example, in Fig. 4.8, the $\vec{x}_t$ segment intersects with the side $\vec{l}_{ij}$ inside the domain and with the side $\vec{l}_{ki}$ outside the domain. In mathematical terms, we have 6 equations from the two vector equations above with 5 unknowns. We discard the trivial

equation along the triangle normal, and compute the unknown parameters. If t and s are both found to be negative and/or greater than 1, there is no intersection. Otherwise, there is intersection.

In the case of an intersection, we update the position of the point and subtract the position vector from the intersection point. We then map this new displacement vector to the new triangle using Rodrigues rotation. It consists of dividing the initial displacement vector into parallel and perpendicular parts with respect to the side along which the intersection occurred. Next, we rotate the perpendicular component using the intersected side as an axis and the angle between the normals of the two triangles connected by the intersected side. Once we get the new distance vector, we apply the same process for the new triangle, except that we skip checking for an intersection with the side that was already intersected. We repeat this process until no more intersection occurs.

Membrane Site

A membrane site component is in charge of two possible actions. First, it can be used to define additional energetic penalty associated with membrane deformation based on its composition. Second, it can be used to attach a molecule to the membrane. For every membrane move, we track the movement of the affected membrane sites. If a vertex of the triangle where a site resides changes, we update the state of the site and compute its energy contribution. As a result, a membrane has a record of all the sites that are on top of a specific triangle. When a membrane site moves, we just remove the site from the record of that triangle and add it to the list of the new triangle.

There are two types of changes that a site has to handle according to the location of the site with respect to the affected vertex. First, if a site (site A in Fig. 4.9) is inside a triangle which is located in the first ring (blue triangles in Fig. 4.9), we need to update the position and the surface attributes like the mean curvature. If the affected site (site B in Fig. 4.9) lies in the second ring (gray triangles in Fig. 4.9), we only update the surface attributes. The main difference is that the position of each vertex is independent of the vertex, but the membrane attributes are a function of the position of that vertex and its immediate neighbors.

The type of binding site depends on the attributes it reads from the membrane surface.

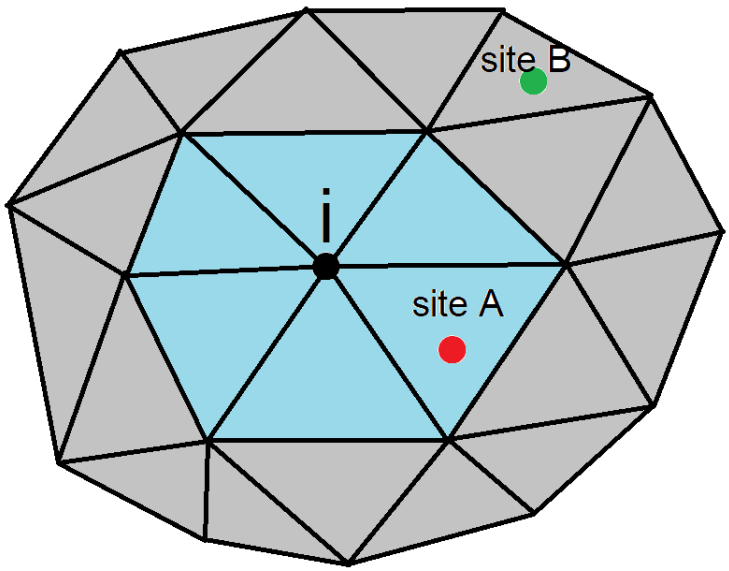

Figure 4.9: A schematic showing the two-ring structure employed to update membrane site information. Site A lies in the inner ring shown in blue, and site B lies in the outer ring shown in gray.

The framework has the following membrane site types:

- Hub site: It reads the membrane position and the membrane normal. It does not contribute to the Hamiltonian and is used as a reference for other components to compute their state.
- Curvature sensor site: It reads the membrane position, the membrane normal and the mean curvature, and contributes to the Hamiltonian.
- Diffusion sensor site: It reads the lipid concentration at the membrane site.

In addition, a membrane site possesses a coordinate axis where the z axis is oriented along the normal vector of the membrane at the location of the site, and the other two axes are oriented along the orthogonal tangent and the cotangent vectors. Hence, the membrane site possesses an additional degree of freedom related to the in-plane rotation of the site about the normal vector.

Membrane site group

It uses a membrane site component and builds a molecule on top of it (Fig. 4.10). The main idea is that the body component is not independent, but it relies on the location,

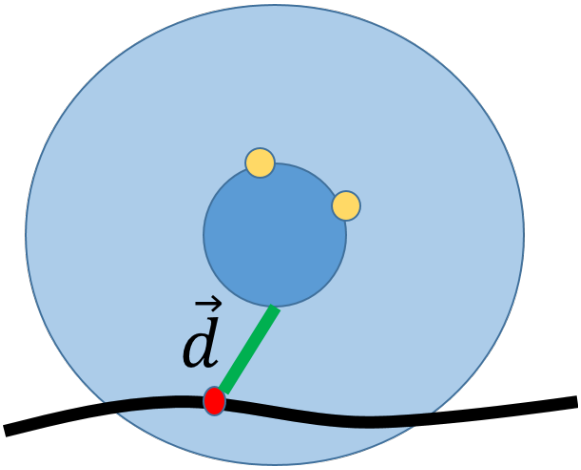

Figure 4.10: The schematic shows a typical group assembled at a membrane site. The vector $\vec{d}$ shows the location of the body component with respect to the membrane site.

the normal and the tangent vector of the membrane site to compute its state. The body has a prescribed location on the local reference frame of the membrane ($\vec{d}$) and every time the site moves, we update the new site position and compute the global coordinate of the body. Other than this, the other components of the molecules such as binding sites and non-bonded interactions work as described in Chapter 3.

Chapter 5

Monte Carlo framework: Programming

In this chapter, we explain the details of the component architecture and the basic ideas underlying the Monte Carlo-based programming.

5.1 The BasicPhysObj class and the event pipeline

Monte Carlo algorithms present many challenges compared to other types of simulations. The first issue is about the communication of the change of a degree of freedom to the rest of the affected components. For example, in molecular dynamics, all the degrees of freedom get updated simultaneously. In contrast, in a Monte Carlo simulation, the computational framework has to remember all the components and their attributes that get affected by a Monte Carlo move. We solve this problem by applying the 'observer pattern', in which each component keeps a list of its dependents. As a result, every time a component is perturbed, it will communicate its dependents of the change of its state. We manage this component-dependent communication using events. An event encapsulates a change of state by providing information of changes undergone by components.

The second issue is that we would like to reduce the number of changes incurred by a component during a Monte Carlo move. For instance, if a component 'A' depends on two or more components that were affected by the same Monte Carlo step, we end up recomputing the state of the component 'A' multiple times. We solve this problem by delaying the update of a component until all the other components it depends on are fully updated. The first step in achieving this objective is to assign a number to every component called precedence.

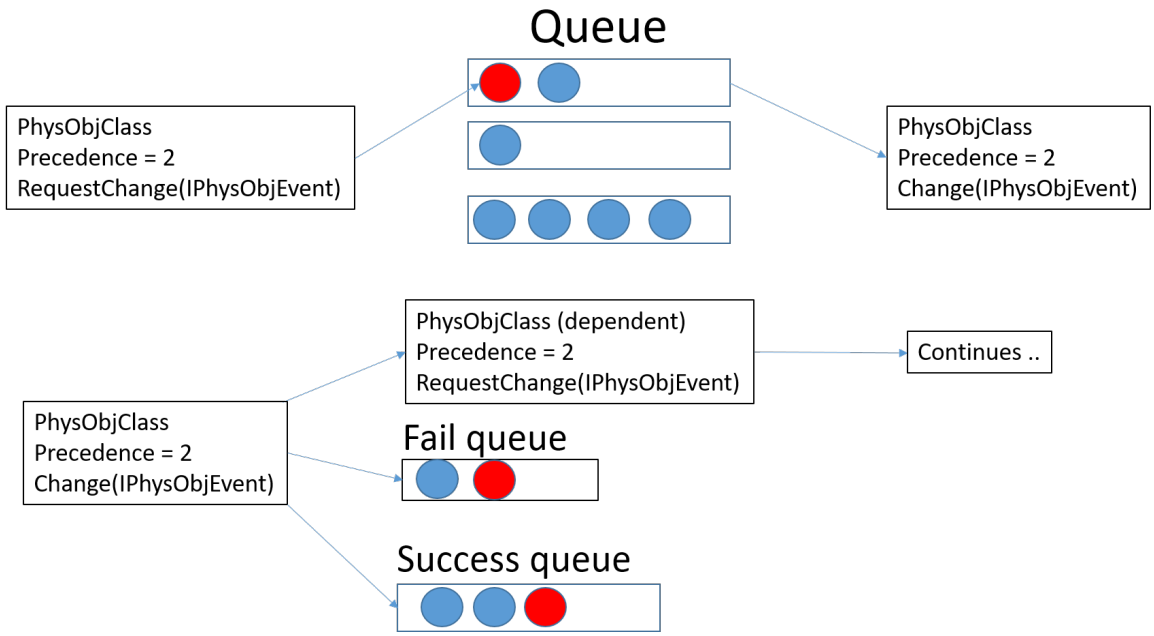

Figure 5.1: The schematic shows the queuing system executed in the computation framework. For each Monte Carlo step, the program iterates through the queues in the order of the precedence values to update the components.

If it is independent (like the body in a molecule), we assign a precedence value equal to zero. On the other hand, if an object depends on two or more objects, we assign a precedence value which is greater than the largest precedence value of the component it directly depends on by one. Then, we create a multi-level queuing system where a component subscribes to the queue according to its precedence level. Afterwards, when a component receives the notification that one of the components it depends on has changed, a request is submitted to the queue corresponding to its precedence value (Fig. 5.1). Finally, when a Monte Carlo step occurs, the program iterates through the queues and updates the components.

The third issue arises due to the stochastic nature of the Monte Carlo method. At the moment of executing a Monte Carlo step, we do not know if the changes are done permanently or if they are going to be reverted. Hence, after applying the Metropolis algorithm, we need to go through the modified components in the same order in which the perturbation took place. The solution uses a queue of requests, and a component, after modification, may send two requests to the queues. One, where the request is executed if the Monte Carlo move fails and the other, where the request is executed when the Monte

Carlo move is successful (Fig. 5.1). Because the requests are submitted in the sequence they are modified, all the components will be updated or reverted to the previous configuration respecting their precedence value.

Events encapsulate the change of state of a component [71]. Its main purpose is to decouple the sender component from the receiver component. In consequence, the sender component creates events related to the type of change, and the receiver component reads the types of events it cares about and handles them according to their types. There are three types of methods in the BasicPhysObj class responsible for handling events. The first one is the OnChange method, that handles the perturbation and returns the energy change derived from the change of a state. The other two methods Restore and Update are used when a Monte Carlo step is accepted or rejected, respectively.

5.2 The BasicPhysObj lifetime

The lifetime of a component is managed by the Domain and the physObjManager class. The life of a component starts either at the beginning of the simulation or as a result of a Monte Carlo move. In either case, the components are built in order and joined with their respective parents. Next, for each created component we submit an event called OnGroupCreated, which has the function of coordinating the existing components of the system and perform necessary tasks for computing the initial state. Next, if the Monte Carlo move that created a component succeeds, we call the Update and Register methods for each of the created components. The latter is in charge of adding the component to the list of the components it depends on, and to the Domain. In case of the biding site components, collision bodies, and nonBonded components, the Register method also adds these components to the uniform grid.

When a component expires, it receives the OnDestroyed event. It then has the opportunity of coordinating with the rest of the components of the domain and send a message to its dependents. Finally, if the change is accepted, the object will call the method Unregister, which will remove any reference to the component in the related domain.

5.3 Reactions

There are three faces of a reaction:

- Reaction definition
- Creation of a group
- Execution of the reaction

Reaction definition consists of three things. First, how the reactant components will be mapped to the product components. Second, which components of the reaction are going to be destroyed, and which are going to be created. Third, a definition of the reaction condition used to determine when a reaction is triggered. This object encapsulates a key that opens when the state of a specific component matches certain conditions. We also define a lock that triggers a reaction. It consists of a user defined Boolean operation that reads the state of the conditions as an input (Fig. 5.2). If the operation yields true, it triggers the reaction.

Creation of a group entails assigning a number to a component, ranging between zero and the number of components belonging to the group. This number is used as an identifier and as an index for accessing component information inside the group. Whenever an instance of a group is created, if it is a reactant of a reaction, a special component called the reaction component is created. It is responsible for tracking the states of the relevant components and for getting a reaction started by reading the events that are sent to their dependents. These events are submitted to the reaction condition objects, which process the events, update the locks and check if a reaction has to happen (Fig. 5.2).

Instead of continuing in the same event pipeline, we execute the reactions in a parallel pipeline. Otherwise, it will conflict with the rules of precedence and the framework will have an undefined behavior. When the root component receives the reaction event, it reads the kind of reaction that has been triggered and executes the changes accordingly. The kind of changes due to a reaction are open-ended, but most of the time, changes include notifications about force fields, group types, dependent components, and created/destroyed

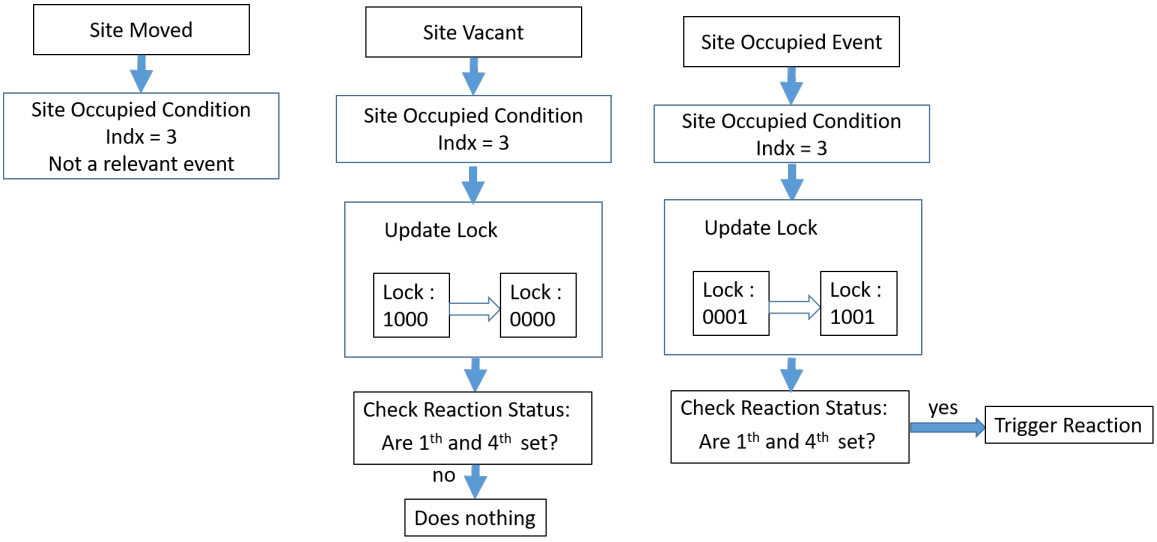

Figure 5.2: The schematic shows the key and lock mechanism executed to program reactions. The keys are the events and the locks, contained inside the components, trigger the reactions if the conditions are met.

components. Next, we send the events to the dependent components, which then process the events in a similar fashion. Finally, when we reach the bonded components (since they do not belong to a group) we have to check if the binding sites have changed their types. If there is no change, we just update the position of the binding site. If they do change, we find a new bond definition matching the new binding sites type. This rule applies to all the supported pair, angular and dihedral bonds.

Chapter 6

Toy problems

In this chapter, we validate the membrane model and apply the computational framework discussed in the previous chapters to solve some sample problems.

6.1 Membrane validation

First, we performed a fluctuation analysis to verify the membrane model. We ran a simulation of a spherical with a normalized radius of 30 and a bending modulus of $20k_bT$ (Fig. 6.1). The undulation power spectrum of a spherical vesicle is given by[72]:

$$\langle |a_{lm}| \rangle^2 = \frac{\kappa_B T}{\kappa(l^2(l+1)^2 - 2l(l+1))} \tag{6.1}$$

where a_{lm} are the spherical harmonic coefficients, κ is the bending modulus and l is the mode number. We compute the fluctuation profile of the vesicle and compute the spherical harmonic coefficients for each mode. We then plot the undulation spectrum computed numerically and predicted by the equation above in Fig. 6.2. The two curves show excellent overlap for lower modes and deviation for higher modes. The latter happens because of membrane discretization. These two curves and the overall trend is consistent with earlier results reported for spherical vesicles in the literature [72]. Thus, our framework is able to capture the expected response of lipid membranes.

6.2 High genus vesicles and nuclear envelope-like vesicles

A majority of the work in the field of membrane biophysics has dealt with analyzing single layer spherical vesicles. However, many organelles in cells possess complex membrane

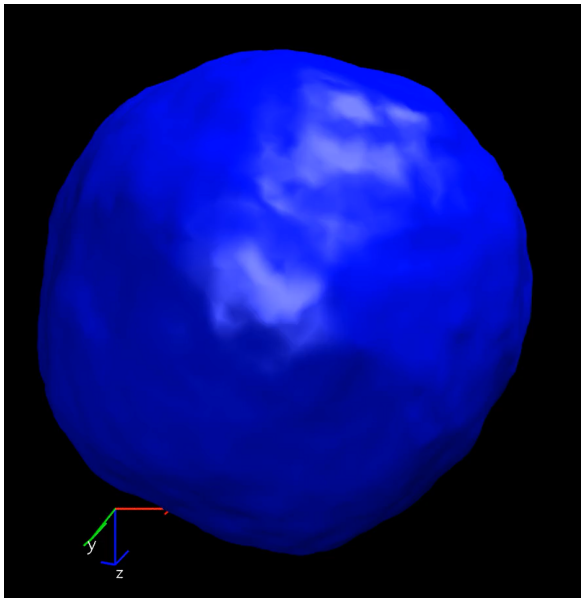

Figure 6.1: Spherical vesicle simulated for validation.

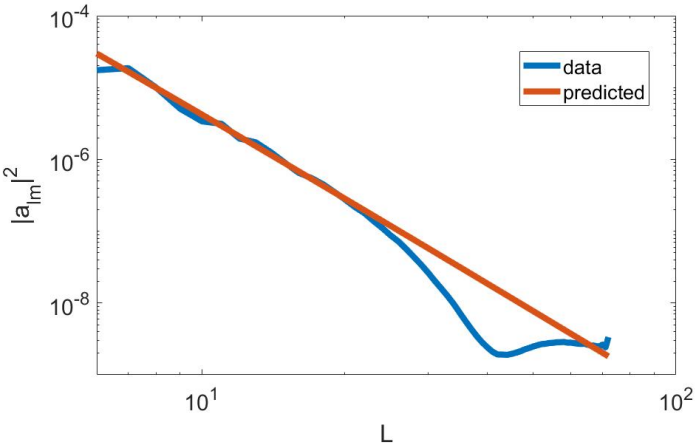

Figure 6.2: Undulation spectrum. The blue curve is the numerical prediction from the simulation and the red curve is the theoretical prediction.

structures with multiple bilayers and holes [73]. One excellent example of such a geometry is the nuclear envelope [74]. The nuclear envelope is the physical barrier that guards the genome. It is made of two lipid bilayers that are fused at hundreds of sites with donut-like holes. Because of this unique architecture, the nuclear envelope has a very high genus equivalent to the number of holes present in it. There is thus a need to investigate such a complex topological structure and understand its mechanical response.

Recent biomechanical studies on simplified nuclear membrane geometry indeed show

that unique properties and phenomenon may arise because of its topology. For example, analysis of the double bilayer structure around a single donut-like hole revealed that compressive forces can cause buckling instabilities, which can determine the sites of membrane fusion and the distribution of holes in the nuclear envelope [75]. Using a similar approach, it was recently shown that the nuclear membranes can possess an almost one order of magnitude higher flexural stiffness compared to a single membrane [76]. This idea is similar to the 1D notion of I-beams, where creation of flanges increases the moment of inertia, leading to high flexural resistance. The work by Noguchi [17] on high genus vesicles show a rich interplay between hole geometry and hole distribution.

While these studies are beginning to provide new insights into the effect of topology and mechanical properties, there is a need to go beyond unit cell geometries to elucidate more realistic and physiologically relevant effects. To this end, we use the computational framework discussed in the previous chapters to simulate some of the most complex topological membranes structures in the field of membrane biophysics. We begin by analyzing simple toy problems with double layered vesicles with donut-like fused holes. The dimensional ratios and pore densities are not reminiscent of nuclear envelope. The purpose was to simulate and understand the basic physics of high genus membrane structures.

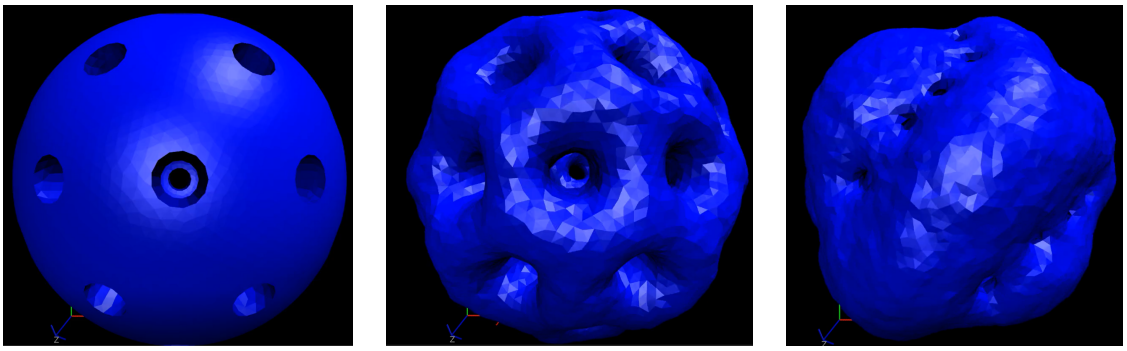

Figure 6.3: Initial, intermediate and final geometry of a vesicle with a genus of 24. The normalized outer and inner radii are 20 and 10, respectively.

Fig. 6.3 shows in order the initial, intermediate and final structure of a vesicle with a genus of 24. In normalized units, the outer radius of the vesicle is 20 and the inner radius is 10. This implies that the spacing between the membranes is quite significant (same as the inner radius). We create a uniform distribution of holes. The bending modulus of the

bilayer is 20 k_BT, and the stretch modulus is 200 k_BT. We have constrained the total area of the bilayers and the volume enclosed within the two bilayers. We can make several important observations. First, the initial structure has sharp edges because of the methodology adopted to create the structure. However, as the structure begins to equilibrate, the edges round up and adopt a donut-like shape. This is because the sharp edges lead to a very high bending energy and therefore, energy minimization smoothens the sharp boundaries. Second, remarkably, the holes rearrange and redistribute from a symmetric pattern to a highly non-symmetric pattern. The holes in the equilibrated structure appear to arrange in a ring pattern. Third, the holes shrink in size compared to the initial geometries. This entails reduction in bilayer spacing at the hole sites. However, this change cannot be homogeneous throughout the vesicles because of the prescribed volume constraint. As a result, the outer domains of the vesicle (outside of the hole rings) appear to have smaller bilayer separation, whereas the central domain (between the hole rings) appears to have a bulge and a larger bilayer separation. Next, we varied the separation between the two fused bilayers to gauge

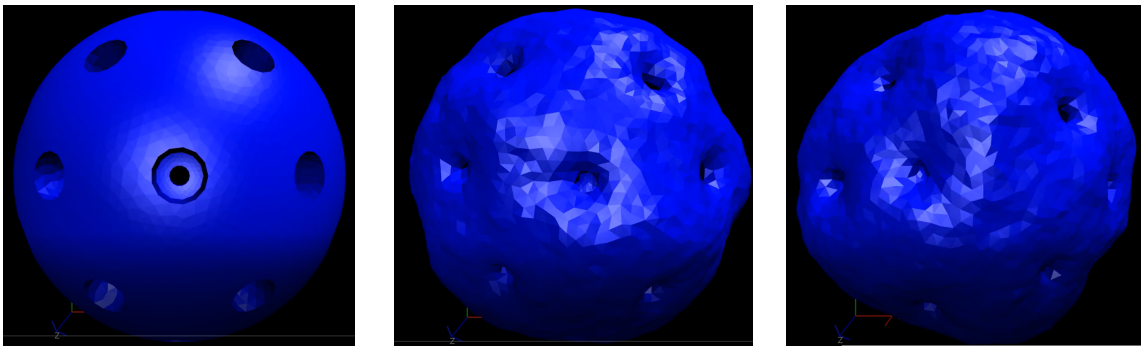

Figure 6.4: Initial, intermediate and final geometry of a vesicle with a genus of 24. The normalized outer and inner radii are 20 and 16, respectively.

the impact on hole distribution. Figs. 6.4, show the initial, intermediate and final geometries of a vesicle where the normalized outer radius is 20 and inner radius is 16. These simulations also reveal several features observed in Figs. 6.3, such as rounding of the edges and redistribution of the holes. However, the extent of redistribution and shape asymmetry developed is much less pronounced. To explore this further, we simulated another vesicle with a normalized outer radius of 20 and inner radius of 18. Figs. 6.5 show the initial, intermediate and final geometries. These shapes show a reduced rearrangement of the holes

and a reduced loss of symmetry compared to the previous simulations. Collectively, the three simulations appear to suggest that the inter-bilayer spacing has a critical impact on the overall geometry of the vesicle and the distribution of the holes. A larger spacing seems to result in more dynamic holes and more asymmetric vesicle morphology.

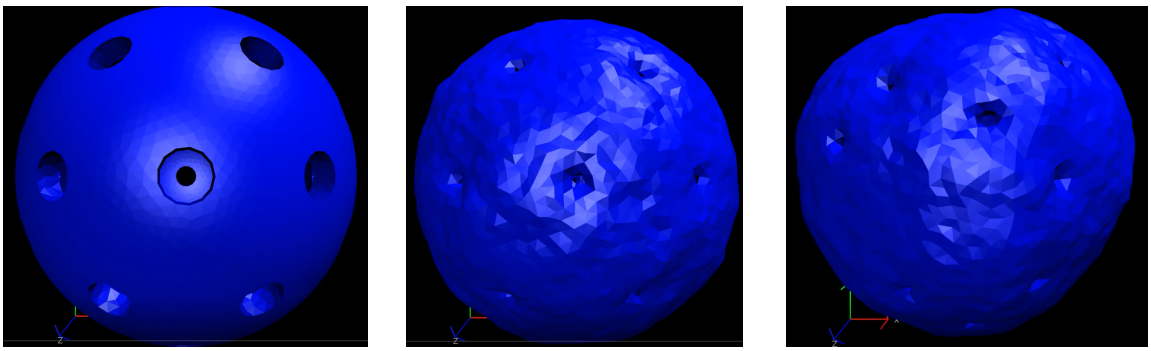

Figure 6.5: Initial, intermediate and final geometry of a vesicle with a genus of 24. The normalized outer and inner radii are 20 and 18, respectively.

To further investigate the dynamic nature of holes, we created an initial vesicle with random distribution of holes (Fig. 6.6 left). We kept the normalized inner radius to be 18, and the outer radius to be 20. Figs. 6.6 and 6.6 right show the intermediate and final shapes of the vesicle, respectively. Despite an initial random distribution, the holes do not appear to redistribute upon equilibration. They undergo local shape changes and become smoother, as in the previous cases. Altogether, this suggests that the bilayer separation seems to be the dominant factor regulating hole distribution. If the bilayer separation is small, holes can survive in close proximity and do not undergo redistribution due to hole-hole interactions, as one would presume. If we extend this finding to the context of holes in the nuclear envelope, this suggests that the holes can remain stationary in a nucleus without assistance from protein structures. However, this can change in scenarios where the nucleus undergoes extreme deformations.

Next, we simulated a double layer structure with closer resemblance to the nuclear envelope. The normalized outer radius of the vesicle was set to 50 and the inner radius was set to 49. The number of holes were roughly 525. A membrane structure with this order of genus has not been simulated till date. If we equate the normalized length of 1 to 50 nm, which is roughly the spacing between the bilayers in the nuclear envelope, a

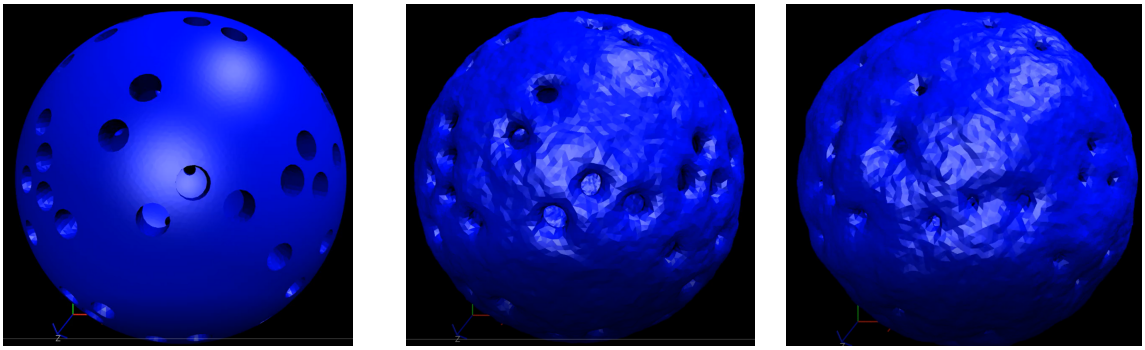

Figure 6.6: Initial, intermediate and final geometry of a vesicle with a genus of 24. The normalized outer and inner radii are 20 and 18, respectively.

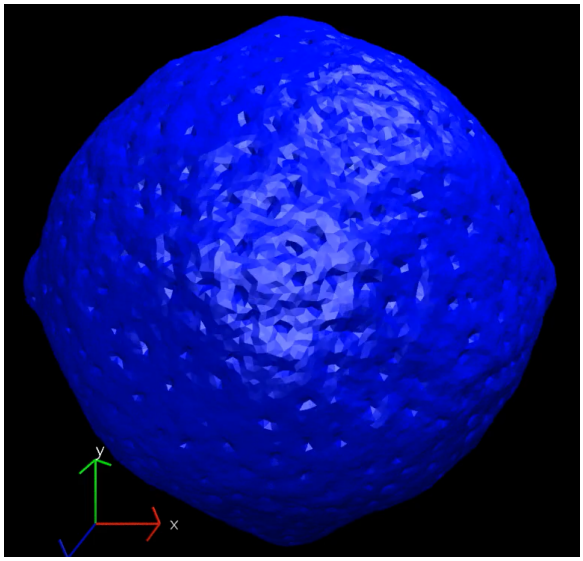

Figure 6.7: Final geometry of a spherical vesicle that resembles a nuclear envelope after equilibration. The normalized outer and inner radii are 50 and 49, respectively. The structure has an approximate genus of 525.

normalized radius of 50 would correspond to a nucleus of 2.5 micron radius. This size falls in the experimental domain. Fig. 6.7 shows the geometry of the spherical nuclei after equilibration. The simulation reveals that this shape with extraordinary genus is stable. Also, the holes maintain their distribution and do not undergo any noticeable dynamics. The hole sizes also remain stable during the simulation. These results show that membrane mechanics is capable of maintaining the complex topology on its own.

Finally, we investigate the shape of a flattened nucleus, which is often encountered in experimental conditions. Since cells are cultured on flat substrates, cells and the nuclei withing them become flattened. To study such a shape, we simulated an oblate ellipsoid-

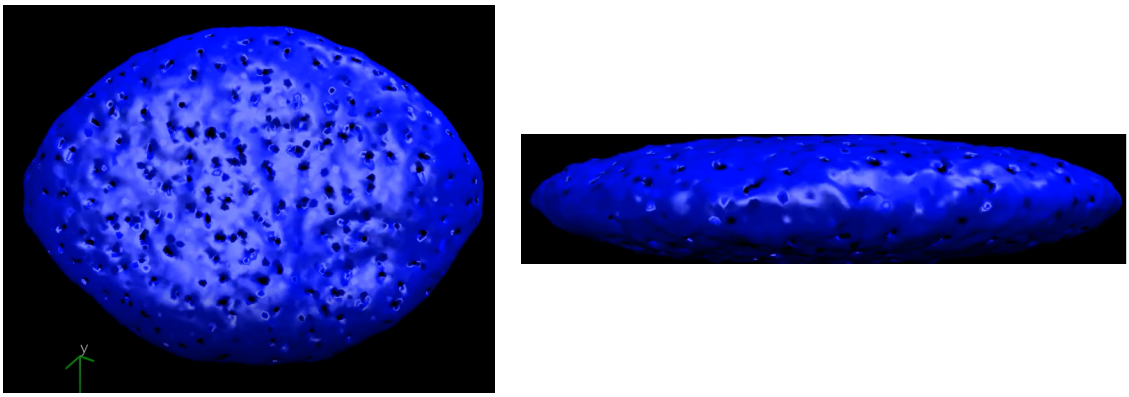

Figure 6.8: Top and side view of the final equilibrated geometry of a flattened nuclear envelope-like structure. The initial aspect ratio of the oblate ellipsoid is 50:30:10. The structure has an approximate genus of 500.

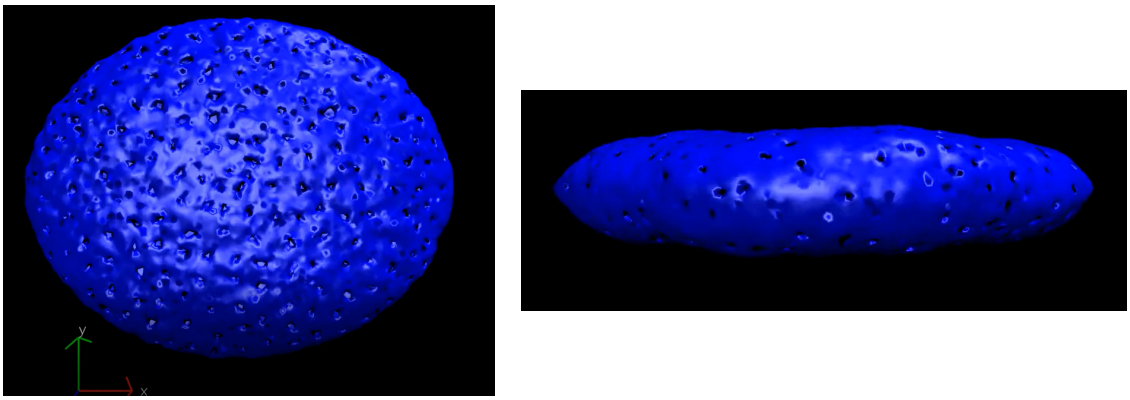

Figure 6.9: Top and side view of the final equilibrated geometry of a flattened nuclear envelope-like structure. The initial aspect ratio of the oblate ellipsoid is 50:30:10. The structure has an approximate genus of 500.

shaped nucleus with two aspect ratios. Fig. 6.8 shows the side view and the top view of the structure with an aspect ratio of 50:30:10. The first number corresponds to the major axis of the ellipse, the second number corresponds to the minor axis of the ellipse, and the last number corresponds to the height. Fig. 6.9 shows the side view and the top view of the structure with an aspect ratio of 50:30:20. All the four figures show that both the overall shape and the hole size and distribution are stable during equilibration. Despite a large difference in the aspect ratios of the oblate ellipsoids compared to the sphere, the two morphologies do not have a propensity to turn into spherical geometry.

6.3 Mixed membrane with curvature inducing lipids

The next problem we simulated to test our novel computational framework was to investigate lipid dynamics and domain formation in an ellipsoidal vesicle with a single bilayer consisting of two lipid species. In the first simulation, we studied the effect of line tension occurring at the interface of two lipid species. The two lipids had zero spontaneous curvature, bending module of 20 k_BT and stretch modulus of 200 k_BT. At the interface of the two different lipid types, we prescribed a line tension of 15 k_BT/length. We started with an initial configuration shown in Fig. 6.10. As the simulation progressed we arrived at Fig. 6.11 and then finally, we equilibrated to Fig. 6.12. Because of line tension, the system progressed to minimize the interfaces and the small domains coalesced to form bigger domains as seen in Fig. 6.12. However, as the system further equilibrates, domains undergo out-of-plane bending in order to minimize the free energy. This behavior is qualitatively aligned with the theoretical predictions made by Phillips and co-workers in the context of multi-domain bilayers [77] and the experimental observations on multi-lipid spherical vesicles [78]. So, between the initial stage and the intermediate stage, the micro-domains coalesce in order to reduce the net interfacial length. However, once this process stagnates, the membrane switches to the alternative mechanism and undergoes out-of-plane bending to reduce the net interfacial length. This results in a driving force that pinches the domains and generates out-of-plane bending.

Next, we analyzed the effect of having two lipid species with opposite spontaneous curvatures in the bilayer in equal concentration. We prescribed one species (in blue) a normalized spontaneous curvature of 0.3 and the other species (in red) a normalized spontaneous curvature of -0.3. Fig. 6.13 shows the initial geometry and Fig. 6.14 shows the final geometry. Intriguingly, contrasting spontaneous curvatures do not appear to lead to a coalescence of domains. The results only reveal development of local curvatures aligned with the preferred spontaneous curvatures in microdomains. As a result, the blue domains exhibit ridge-like shapes, whereas the red domains exhibit saddle-like shapes. It appears that contrasting spontaneous curvatures inhibit mixing of lipids and lipid domains. However, changing the relative concentration of these lipids, or relative strengths of the spontaneous curvatures,

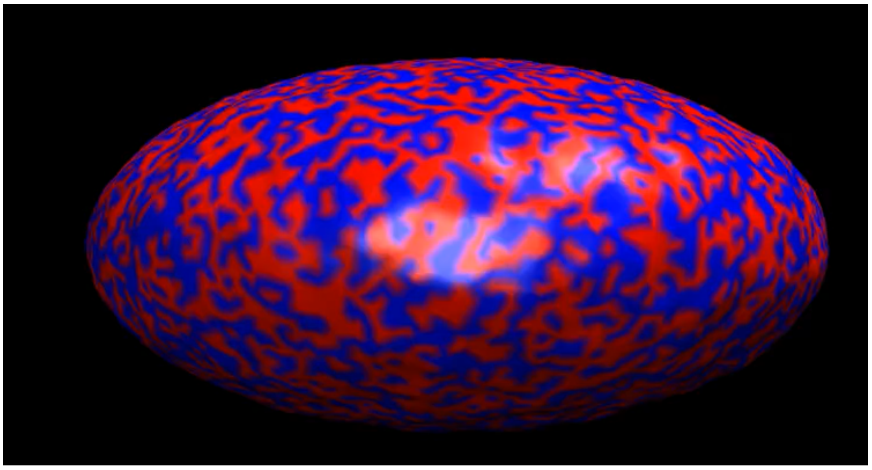

Figure 6.10: Initial configuration of an ellipsoidal vesicle with two lipid species (in red and blue). The interfaces of the two lipid types are penalized by a line tension energy.

can lead to different outcomes. Such studies will be pursued in the future.

6.4 Membrane-clathrin interactions

Next, we used our framework to simulate the process of vesicle formation during cellular transport by a protein called clathrin. The plasma membrane acts as a barrier that blocks the entry and exit of components into and out of the cells. While small molecules are able to pass through the cellular membrane via pores, channels and pumps, large molecules get transported by a set of processes called endocytosis. In this process, the plasma membrane starts creating an invagination that keeps growing until it engulfs the cargo molecules inside the vesicle. These vesicles are then separated from the membrane, which then go inside the cytoplasm and release the cargo. This process is executed by a large set of proteins that act in a well orchestrated spatio-temporal order [79].

One of the most important proteins involved in endocytosis is clathrin. Clathrin is a tri-legged protein that gets recruited from the cytoplasm with the help of accessory proteins at the start of endocytosis [35]. This protein assembles on to the membrane in a network with a chicken-wire shape. Once it reaches a critical size, the clathrin-coated membrane invaginates, initiating the formation of a vesicle [35]. While this is accepted as the current working model, there are many open questions related to this process. For example, the

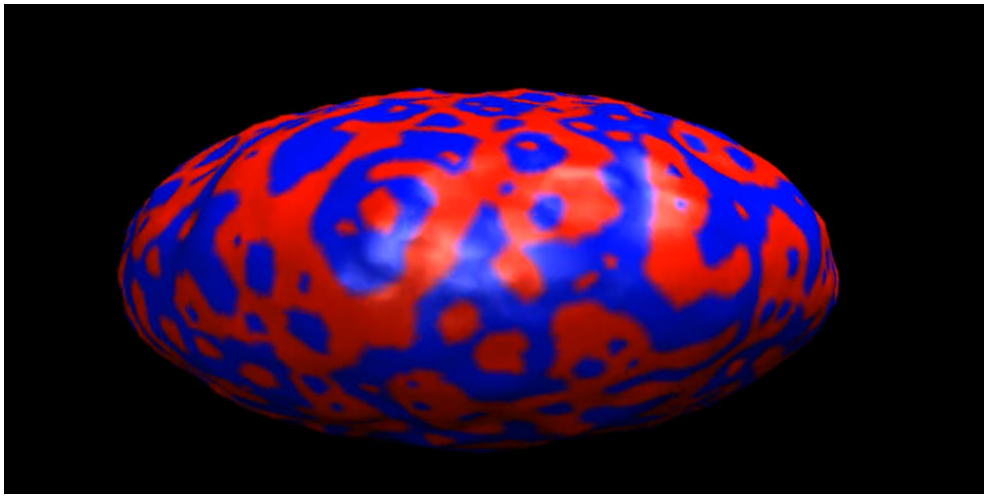

Figure 6.11: Intermediate configuration of an ellipsoidal vesicle with two lipid species (in red and blue) during the equilibration process. The two lipid species begin to show domain formation.

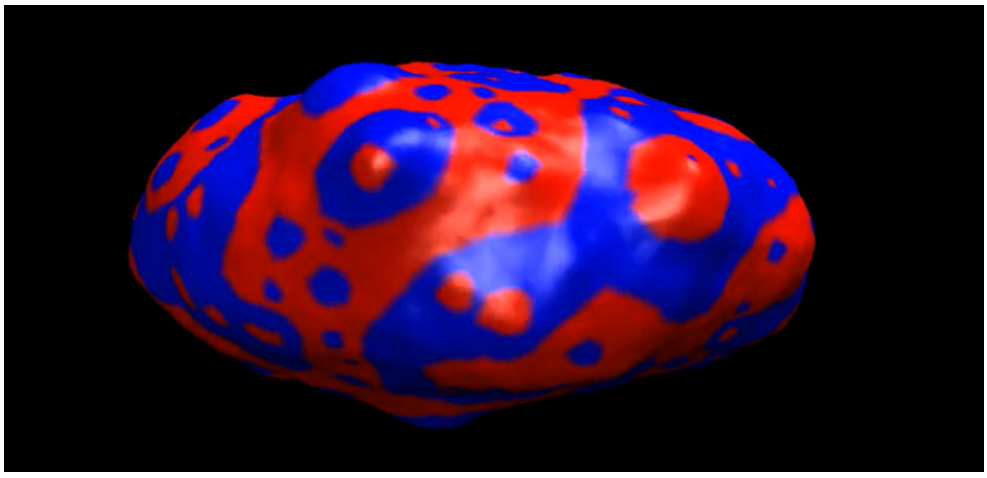

Figure 6.12: Final configuration of an ellipsoidal vesicle with two lipid species (in red and blue) after equilibration. The two lipid species redistribute into domains which undergo out-of-plane bending deformation in order to minimize the free energy.

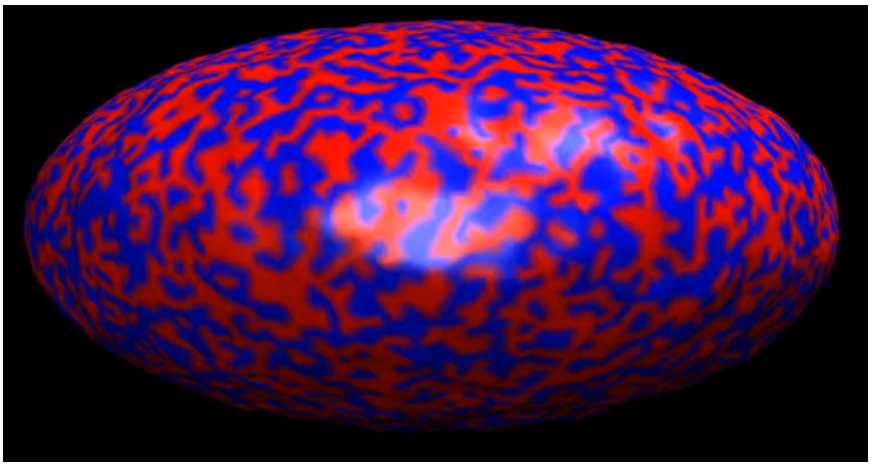

Figure 6.13: Initial configuration of an ellipsoidal vesicle with two lipid species (in red and blue). One lipid type has positive spontaneous curvature and the other lipid type has negative spontaneous curvature.

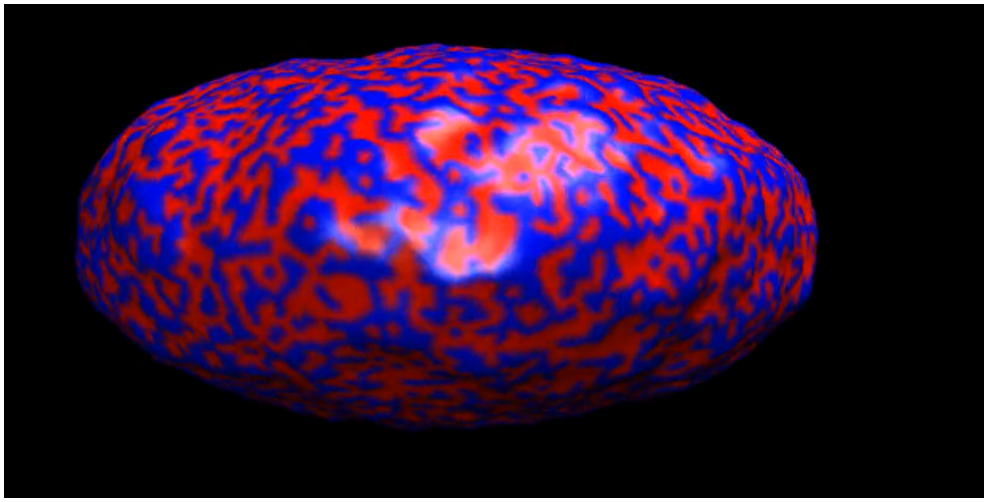

Figure 6.14: Final configuration of an ellipsoidal vesicle with two lipid species (in red and blue) after equilibration. The two lipid species undergo minimal redistribution.

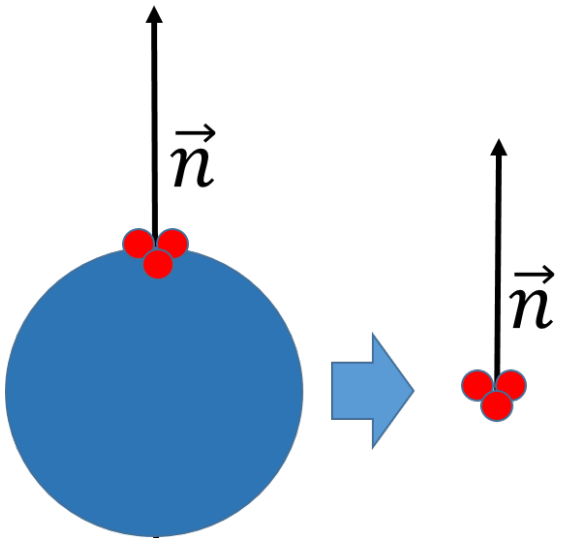

Figure 6.15: The body component used to model clathrin. The component has zero radius and three binding sites corresponding to the three legs of clathrin. The vector $\vec{n}$ is the normal of the membrane where the clathrin is located.

evolution of the polymerized clathrin network and the initiation of the invagination are not well understood [33]. While efforts have been made to understand clathrin-membrane interactions, the majority of the work invokes simple membrane geometries and indirect role of clathrin molecules via spontaneous curvature field. The work by Spakowitz group has used Monte Carlo simulations to investigate mechanics of polymerized clathrin network[21]. Here, motivated by these studies, we performed two sets of analysis. In the first set, we studied the interaction of polymerized clathrin network on large spherical vesicles. In the second set, we allowed clathrin to undergo assembly and disassembly as a spherical vesicle.

We adapted the clathrin model from [21] into our framework to simulate membrane-clathrin interactions. We first defined the body of the clathrin molecule as an oriented sphere with zero radius. We added three binding sites at the north pole as shown in Fig. 6.15. This might appear to be an odd choice at the first sight. However, it allows us to control the out-of-plane deformation of the clathrin network at the bond level. Next, we added the bond definition. We used a hinged bond with a bond vector $\vec{b}$ of unit length and angles θ_1 and θ_2 as shown in Fig. 6.16. We then add a twisting term that penalizes deviation of dihedral angles generated by the normal of the clathrin molecule with the bond. The energy associated with the clathrin molecules is given by

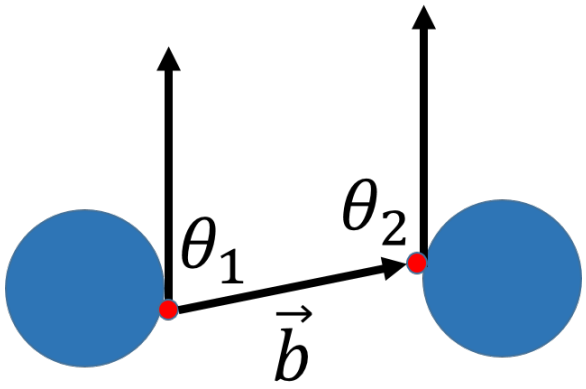

Figure 6.16: The schematic shows a hinge bond between two clathrin proteins. The angles between the two normals and the bond vector $\vec{b}$ are represented by θ_1 and θ_2.

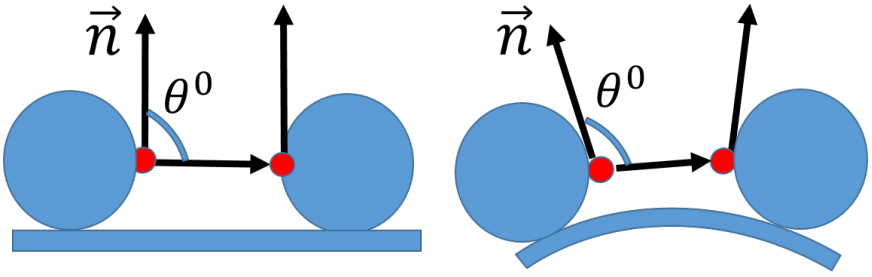

Figure 6.17: The preferred angle between the normal and the bond vector is called the pucker angle. It controls the ability of the clathrin molecule to generate out-of-plane bending of the membrane.

$$e = k(l - l_0)^2 - e_0 + k_{\theta_1}(\theta_1 - \theta^0)^2 + k_{\theta_2}(\theta_2 - \theta^0)^2 + k_\gamma(\gamma - \gamma_0)^2 \tag{6.2}$$

where

$$\gamma = cos^{-1}\{(\vec{n}_1 \times \vec{b}) \cdot (\vec{n}_2 \times \vec{b})\}. \tag{6.3}$$

Above, the first term corresponds to the stretching of the bond, the second term is the binding energy, the next two terms are the energies of the angular bonds, and the last term is the energy associated with the torsional bond. γ is the torsional angle obtained from the two normal vectors at the membrane sites and the bond vector. θ^0 is called the pucker angle, and it accounts for the preferred angle between the normal and the bond vector at the membrane site (Fig. 6.16). Physically, it determines the out-of-plane orientation of the clathrin legs. Therefore, it controls the ability of clathrin molecules to deform the membrane. Fig. 6.17 shows a schematic of two scenarios with pucker angles of 90^0 and 100^0. Then we attach the molecule to the membrane hub with a vector $\vec{d}$ equal to [0.0, 0.0, 1.0]. As a result, the clathrin molecule is always oriented parallel to the local normal at the membrane site.

We simulated the effect of clathrin molecules on the shape of a spherical vesicle with bending modulus of 5 k_BT (simulation parameters presented in Table 6.1). We added 1000 clathrin molecules on the membrane surface in a random fashion. In order to investigate the effect of clathrin molecule structure on vesicle shape, we varied the pucker angle θ^0 and performed simulations with three pucker angles equal to 100^0, 105^0, and 110^0. Figs. 6.18, 6.19 and 6.20 show the configuration of the clathrin molecules with a pucker angle of 100^0 in the initial, intermediate and final stages of the simulation. The figures reveal the polymerization of clathrin molecules into a polygonal network structure. In the final stage, one can see the emergence of a small partial vesicle because of clathrin induced bending. We would like to note that the figures only show the clathrin molecules and not the underlying membrane. Figs. 6.21, 6.22 and 6.23 show the evolution of clathrin molecules with a pucker angle of 105^0 in the initial, intermediate and final stages of the simulation. Here, the polymerized clathrin domains appear to be smaller in size compared to the previous case. However, the vesicle generated in the final stage is a mature vesicle with nearly a

complete spherical geometry. This suggests that a pucker angle of 100^0 is more effective in bending the membrane and generating vesicles. Finally, Figs. 6.24, 6.25 and 6.26 show organization of clathrin molecule with a pucker angle of 110^0 in the initial, intermediate and final stages of the simulation. Similarly, clathrin domains seem to be small in size. Interestingly, the final shape appears to have many more invaginations compared to the previous two cases. However, the extent of vesicle formation appears to be compromised. We do not see a nearly spherical vesicle for this scenario. Rather, the mature vesicle we see seem to have an elongated shape. Thus, this angle might not be conducive for vesicle formation. Overall, these studies collectively suggest that there might be an optimal pucker angle for generating clathrin-mediated vesicles. This is a new finding which could give insights into the architecture of clathrin molecules. It is possible that the optimal clathrin architecture is compromised in a health disorder, which would then impact the metabolic processes in cells.

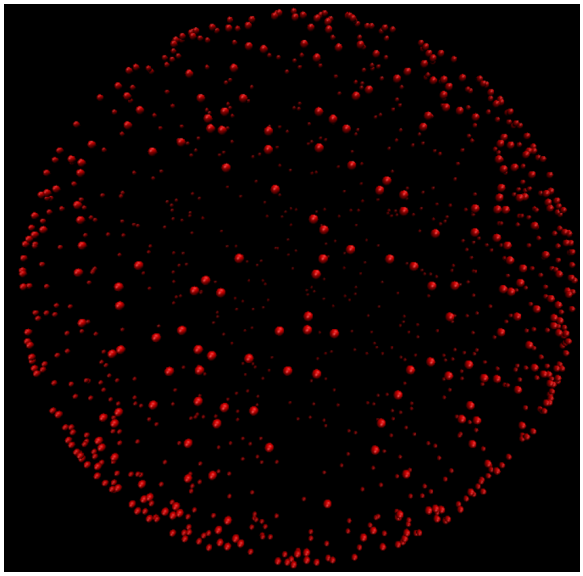

Figure 6.18: Initial random distribution of the clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 100^0.

6.5 Clathrin assembly and disassembly

In the previous section, we explored the assembly of clathrin molecules on the membrane. All the molecules were restricted to lie on the membrane surface. They had the flexibility

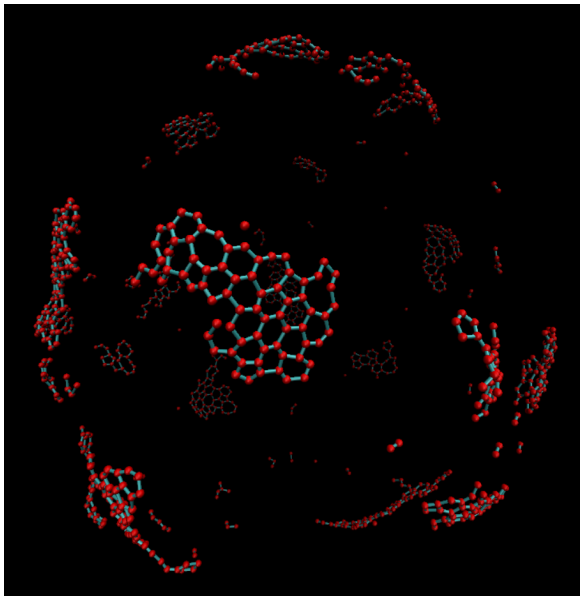

Figure 6.19: Intermediate configuration of clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 100^0. Clathrin molecules begin to polymerize.

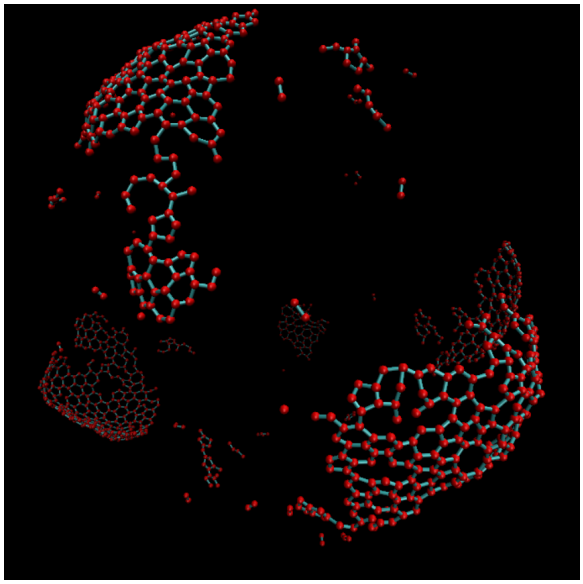

Figure 6.20: Final configuration of clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 100^0. The polygonal clathrin network grows in size, but fails to generate partial or mature vesicles.

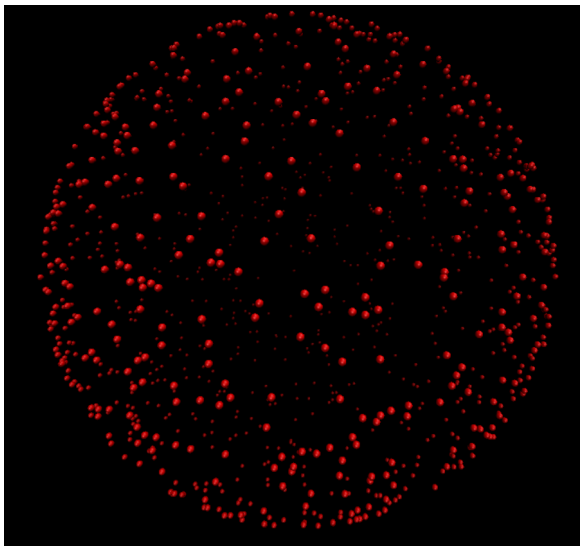

Figure 6.21: Initial random distribution of the clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 105^0.

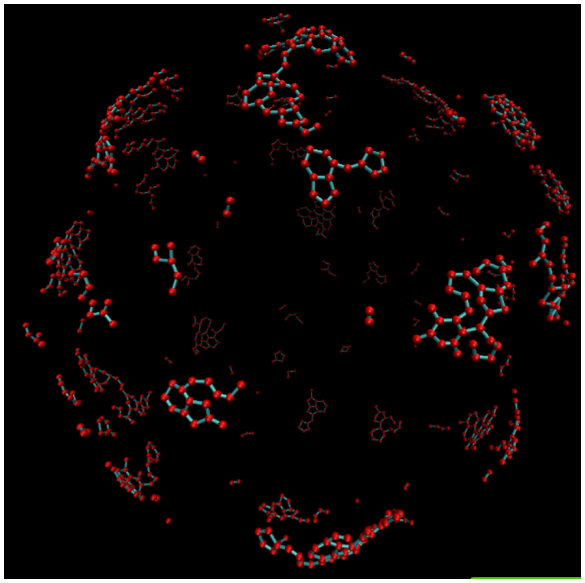

Figure 6.22: Intermediate configuration of clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 105^0. Clathrin molecules begin to polymerize into a network at various locations.

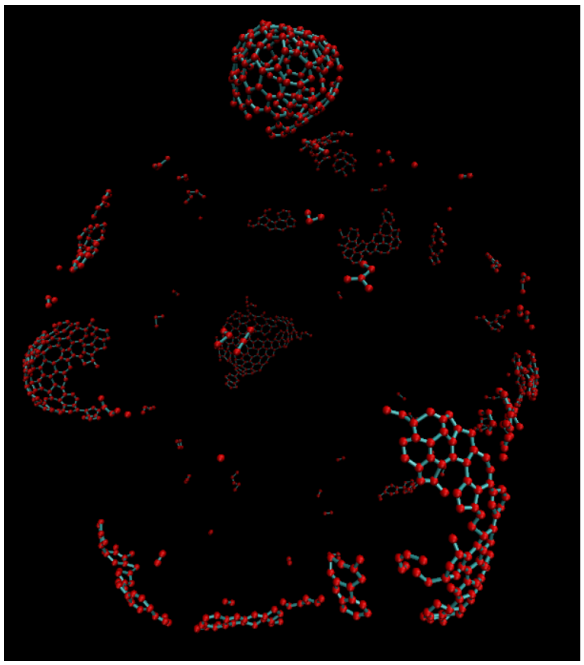

Figure 6.23: Final configuration of clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 100^0. The polygonal clathrin network succeeds in forming a smaller spherical vesicle from the original bigger vesicle.

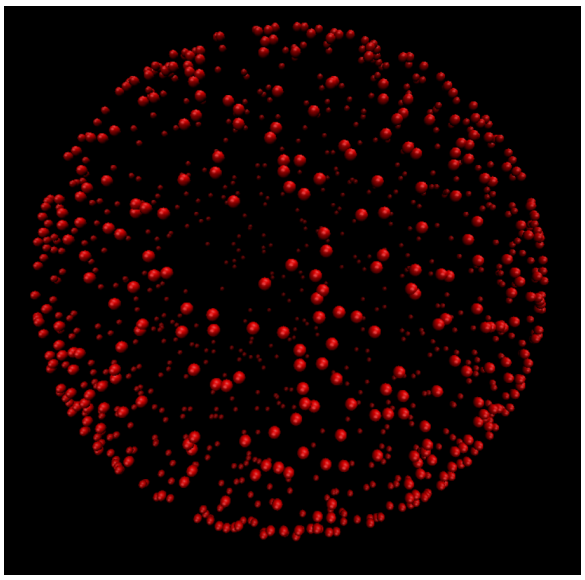

Figure 6.24: Initial random distribution of the clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 110^0.

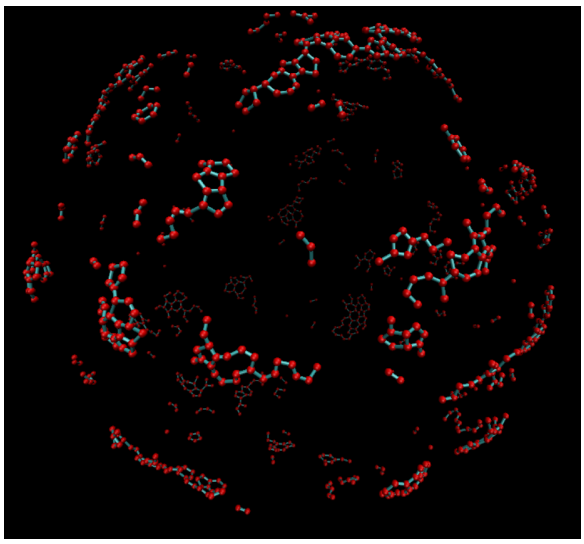

Figure 6.25: Intermediate configuration of clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 110^0. Clathrin molecules polymerize into small networks at various locations.

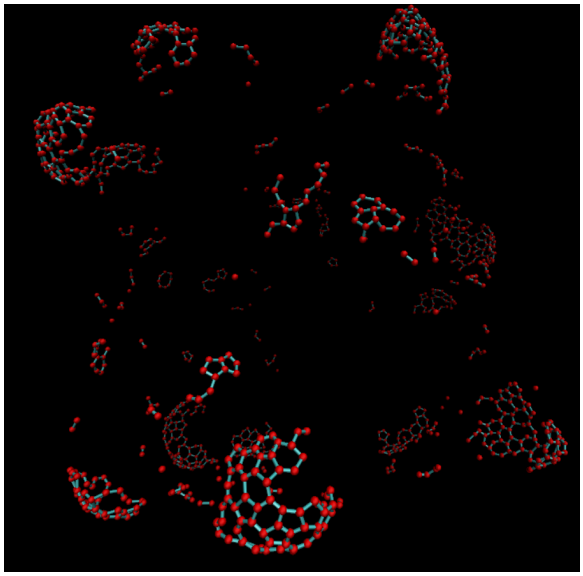

Figure 6.26: Final configuration of clathrin molecules (red particles) on a spherical vesicle (not shown) with a pucker angle of 100^0. The polygonal clathrin networks form partial non-spherical vesicles, but do not succeed in forming a mature vesicle.

to undergo in-plane movement and rotation, but were not allowed to leave the membrane. In this section, we extended this model to incorporate assembly and disassembly of clathrin molecules from the solution on to the membrane. Here, a clathrin molecule can bind to the membrane, unbind from the membrane, move on the membrane and reorient on the membrane. To accomplish this goal, we started by creating membrane sites with a membrane hub that carries a molecule with a single binding site at its north pole. The normal at this binding site is the normal to the membrane at the membrane site. We then added a binding site at the south pole of the clathrin molecule to complement this binding site. We again created spherical vesicles with a bending modulus of 5 k_BT and 1000 clathrin molecules in the solution and 1000 binding sites on the membrane (simulation parameters presented in Table 6.1). Fig. 6.27 shows the initial frame of the simulation. The gray particles are the clathrin molecules and the red particles are the binding sites. As the simulation evolves, clathrin molecules self-assemble on to the membrane at the binding sites (Fig. 6.28). As the simulation progresses further and reaches steady state, we see that the bound clathrin molecules polymerize into a polygonal network (Fig. 6.29). What is also seen is that the majority of the clathrin molecules are bound to the membrane and are not present in the solution. This is likely due to the choice of the binding energy in the Hamiltonian. While clathrin formed a network, we did not see the formation of a vesicle as we saw in the previous section. This likely occurred due to a choice of 90^0 pucker angles. At this pucker angle, clathrin has minimal incentive to bend the membrane. It would be insightful to repeat this simulation with a pucker angle of 105^0 which was found to be the optimal angle for creating vesicles.

Next, we wanted to gauge the effect of membrane curvature on clathrin assembly and disassembly. So, we created an ellipsoidal vesicle and repeated the simulation. We again had 1000 clathrin molecules in the solution and 1000 binding sites on the membrane. Fig. 6.30, 6.31 and 6.32 show the initial, intermediate and final frame of the simulation. As before, the majority of the clathrin molecules bind on to the membrane. They also polymerize to form a polygonal network. What is intriguing is that the network has a larger areal footprint, and it forms in the cylindrical domain of the ellipsoid as opposed to the hemispherical cap domains. This is an interesting finding because clathrin is known to prefer and generate

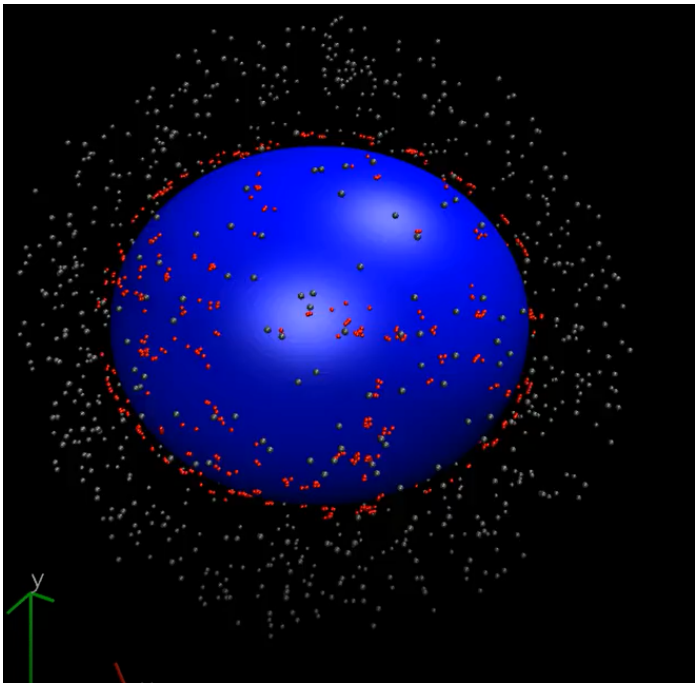

Figure 6.27: Initial random distribution of the clathrin molecules (red particles) in the solution with a pucker angle of 90^0. The spherical vesicle is shown in blue and the membrane binding sites are shown in red.

spherical curvature. However, we have to be cautious with this observation because it could be a consequence of 90^0 pucker angles. With a higher pucker angle, the network might form in the hemispherical domains.

Table 6.1: Clathrin parameters.

Parameter	**Value**
κ	5.0 k_BT
k	42.5 k_BT
e_0	6.5 k_BT
γ_0	0.0
$k_{\theta 1}$	4.25 k_BT
$k_{\theta 2}$	4.25 k_BT
k_γ	4.25 k_BT

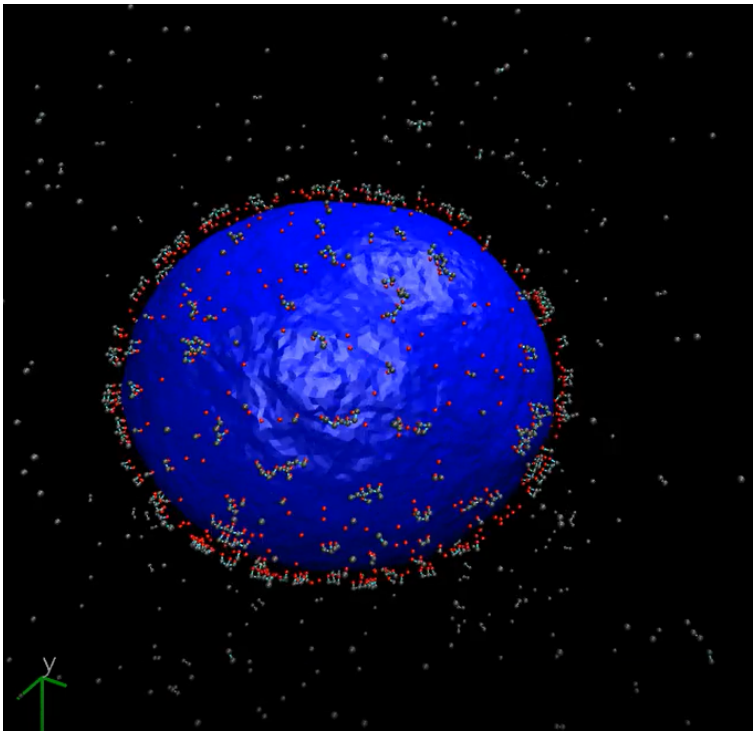

Figure 6.28: Intermediate distribution of clathrin molecules (red particles) with a pucker angle of 90^0. The clathrin molecules begin to assemble on the spherical vesicle at the membrane binding sites.

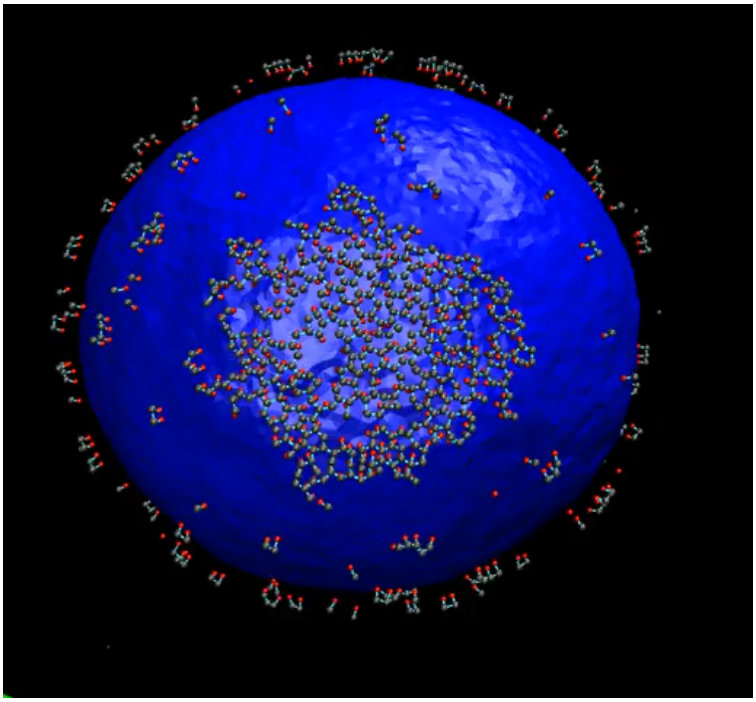

Figure 6.29: Final distribution of clathrin molecules (red particles) with a pucker angle of 90^0. The clathrin molecules polymerize to form a network on the spherical vesicle, but do not show any vesiculation.

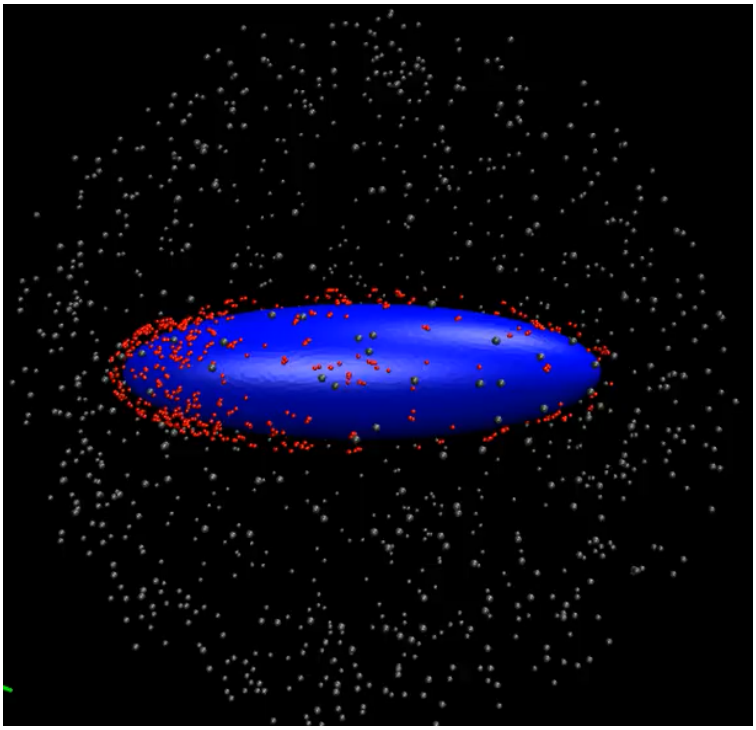

Figure 6.30: Initial random distribution of the clathrin molecules (red particles) in the solution with a pucker angle of 90^0. The ellipsoidal vesicle is shown in blue and the membrane binding sites are shown in red.

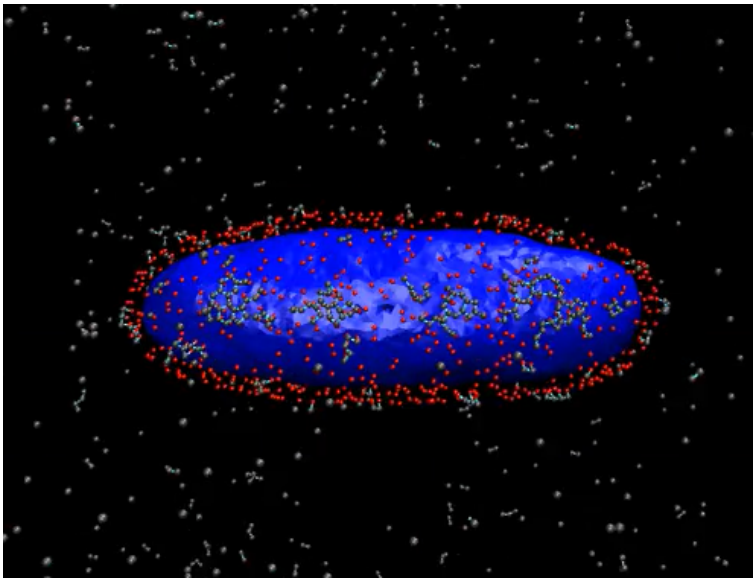

Figure 6.31: Intermediate distribution of clathrin molecules (red particles) with a pucker angle of 90^0. The clathrin molecules begin to assemble on the ellipsoidal vesicle at the membrane binding sites.

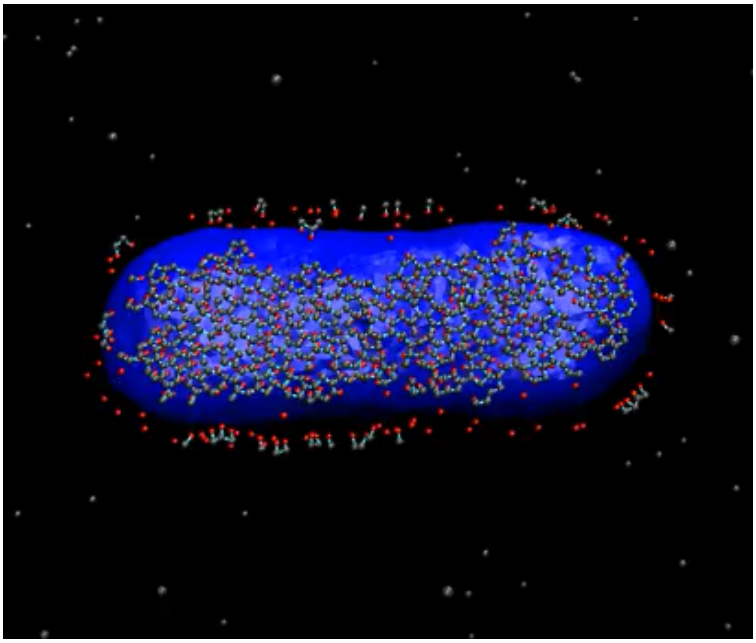

Figure 6.32: Final distribution of clathrin molecules (red particles) with a pucker angle of 90^0. The clathrin molecules polymerize to form a network in the cylindrical domain of the vesicle, but do not show any vesiculation.

6.6 Orthotropic proteins

Some protein motifs like bar domains which possess banana-like shapes impose anistropic curvatures on the membrane. This means that they prefer different curvatures in perpendicular directions. For example, Fig. 6.33 shows a schematic of a banana-shaped protein that prefers a circular curvature in the $\vec{t}$ direction and zero curvature in the orthogonal contangent direction. As a result, these proteins prefer tubular membrane geometries. This feature is very different from that of clathrin protein, which more or less prefers uniform curvature in all the directions. Hence, clathrin proteins prefer spherical curvatures. Orthotropic proteins are involved in many biological processes, such as endocytosis, trafficking, motility, cell division and organelle division.

In this final toy problem, we simulated the effect of anisotropic proteins on the shape of a spherical vesicle. To account for their anisotropic curvatures, we have to modify the Hamiltonian. For clathrin molecules, we were required to impose the spontaneous curvature associated with the mean curvature in order to prescribe a preferred spherical curvature. However, we need to invoke spontaneous curvatures along the tangent and the cotangent

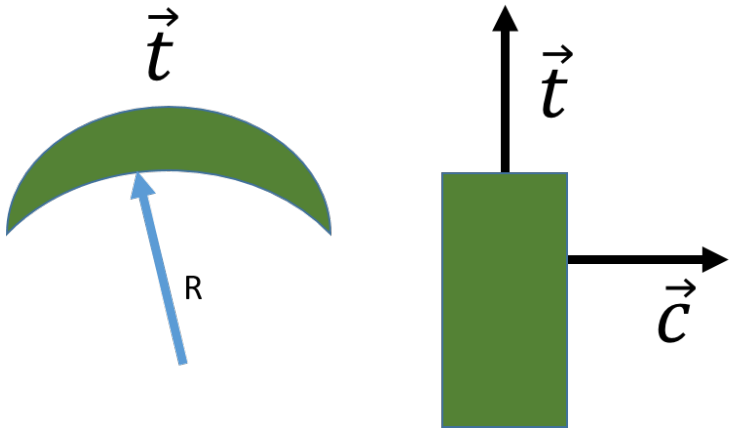

Figure 6.33: Side view of a banana-shaped protein with a preferred radius of curvature R and tangent vector $\vec{t}$. Top view of the protein, showing the tangent vector and the cotangent vector $\vec{c}$.

vectors associated with the orthotropic protein. This effect is captured by the revised Hamiltonian:

$$E = k_1(k_t - c_1)^2 + k_2(k_c - c_2)^2 \tag{6.4}$$

where (k_1, k_2) are the bending module, $(c_1\ , c_2)$ are the spontaneous curvatures and (k_t, k_c) are the normal curvatures along the tangent and the cotangent directions. Each time the membrane strain energy is needed, we use the curvature tensor defined in Chapter 4 and the following identities to compute the normal curvatures:

$$k_t = \vec{t} \cdot \vec{B}\vec{t} \tag{6.5}$$

and

$$k_c = \vec{c} \cdot \vec{B}\vec{c}. \tag{6.6}$$

We created a spherical vesicle with a bending modulus of 10 k_BT (simulation parameters presented in Table 6.2). We randomly distributed 1000 orthotropic proteins on the membrane surface. These proteins had the freedom to move on the membrane and reorient, but were not allowed to leave the surface. Figs. 6.34, 6.35 and 6.36 show the initial, intermediate and final configuration of the vesicle. It is remarkable that the proteins aggregate with time

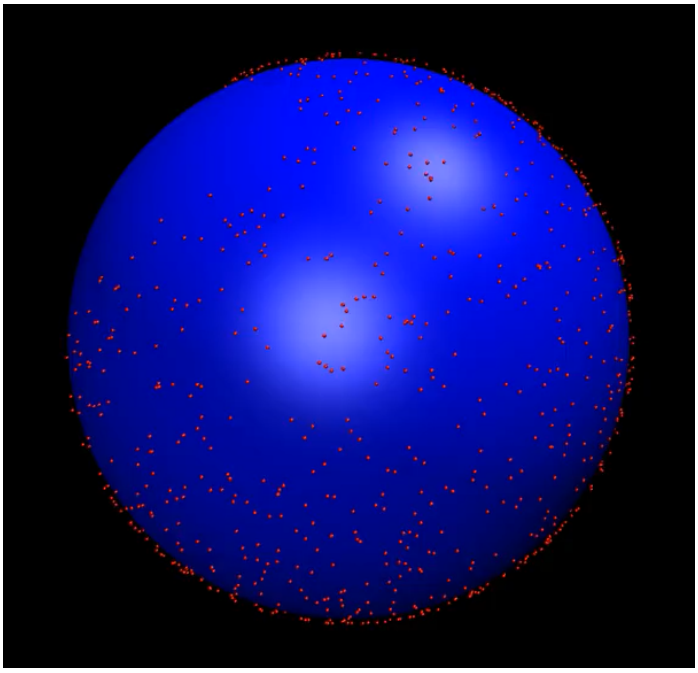

Figure 6.34: Initial random distribution of banana-shaped molecules (in red) around a spherical vesicle (in blue).

and generate tubular domains in the spherical vesicle. This is aligned with the intuition, since these proteins have zero spontaneous curvature along the cotangent direction.

Table 6.2: Orthotropic protein parameters.

Parameter	**Value**
k_1	10.0 k_BT
k_2	10.0 k_BT
c_1	2.0 (normalized)
c_2	0.0

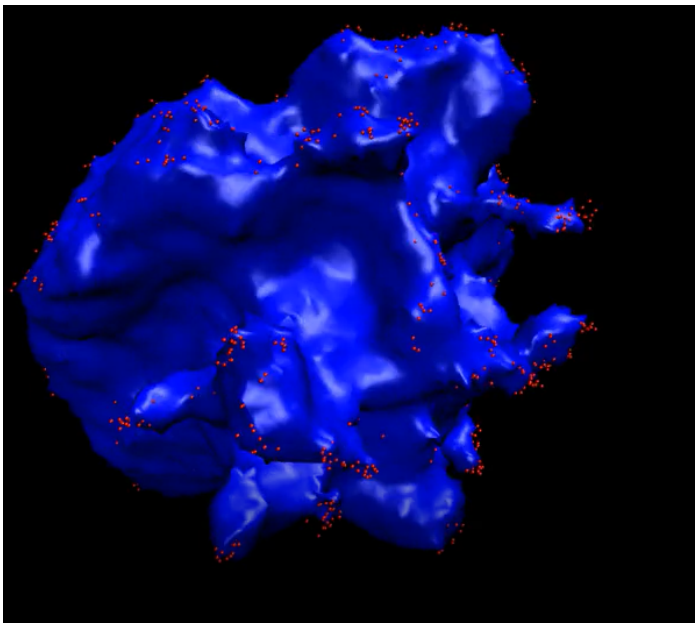

Figure 6.35: Intermediate configuration of the vesicle during equilibration process. The aggregation of the proteins (in red) lead to invaginations.

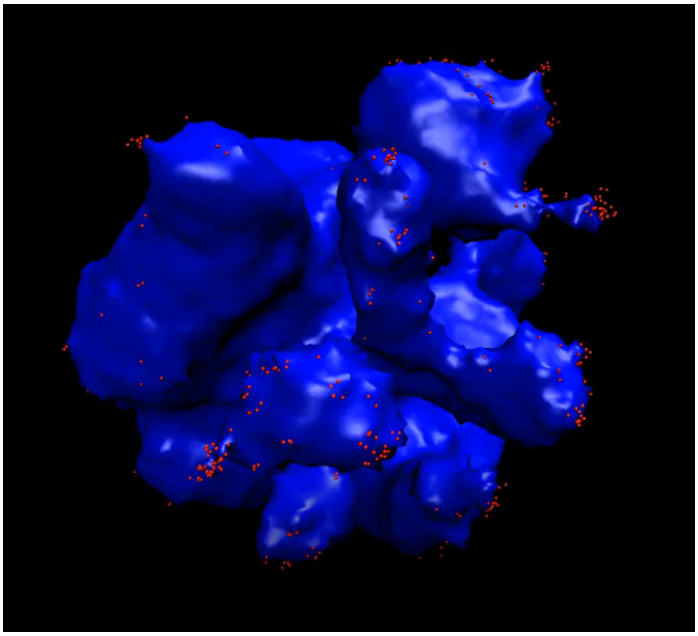

Figure 6.36: Final configuration of the vesicle. Continued aggregation of proteins (in red) lead to significant deformation and tubulation in the vesicle.

Chapter 7

Concluding remarks and future works

In this thesis, we studied membrane-protein interactions via two approaches. In the second chapter, we used molecular dynamics to characterize the mechano-sensitivity exhibited by an endocytic protein called epsin. Our simulations reveal that membrane tension energetically favors the embedded state of epsin helix. This could explain the role of this protein in vesicle formation in high tension environments. In the next three chapters, we explained the new Monte Carlo framework we developed to simulate dynamics of membrane-protein interactions. Then we showed how the features of the framework can be used to simulate some important membrane-protein systems in chapter 6. We used the membrane model to simulate the most complex topology of the nuclear envelope. Our preliminary study suggests that the inter-bilayer spacing can play an important role in regulating hole geometry and distribution. Next, we investigated the mixing of two lipid species. Our simulations revealed lipid segregation and domain formation in the presence of line tension acting at the interface of two lipid species. However, intriguingly, the simulations did not reveal a similar phenomenon when the two lipids were assigned opposite spontaneous curvatures. In the next problem, we studied polymerization of clathrin molecules on the surface of a spherical vesicle. We varied the pucker angle and computed the shape transformations. Our simulations suggest that there exists an optimal pucker angle for creating spherical vesicles. Next, we expanded the analysis to included binding and unbinding of clathrin molecules from the surrounding solution. The simulations demonstrate the assembly of clathrin molecules onto the vesicle and subsequent polymerization into a polygonal network. Finally, we modeled the effect of banana-shaped proteins on vesicle geometry. Our simulations show that polymerization of these orthotropic proteins lead to formation of tubular domains in the spherical vesicle.

Going forward, there is immense potential for refining and employing the computational framework to rigorously analyze biological phenomena. As we stated in the beginning of the thesis, one of the primary reasons that biological problems are difficult to study is the fact that the number of entities involved in a process are ginormous. Hence, we aimed to create a computational framework that can emulate and incorporate the key physical properties of these entities with minimal coding effort. We have achieved this objective by disintegrating the physical properties from the physical bodies. As a result, one can invoke different fundamental features of the geometry and the energetics, and create customized coarse-grained molecules. In principle, this notion is similar to that encountered in atomistic simulations. With a fundamental set of force fields, one can simulate a wide spectrum of biological systems. Similarly, by creating components with fundamental properties, we can design a wide variety of lipid and proteins molecules needed to study cellular processes.

In terms of the toy problems presented in this thesis, we can pursue several directions to gain mechanistic insights. For example, in the context of nuclear envelope, we can perform Fourier analysis on the shape fluctuations and estimate the effective mechanical properties. We can also include the underlying protein layer and revisit the equilibrium geometries. In the context of lipid mixing, we can vary the lipid compositions and the vesicle geometry to investigate their role in domain formation. Since endocytosis typically involves a large set of membrane remodeling proteins, we can perform in silico experiments with multiple protein species. These simulations can potentially give new insights into the endocytosis puzzle by disentangling the roles of different proteins. For example, we can simulate both clathrin and banana-shaped proteins to predict their synergistic roles in vesicle formation [12].

The scope of the framework is not only limited to study membrane-protein systems and their shape evolution. We have developed a selective binding site structure and created reaction components that are capable of simulating protein-mediated reactions in cells. For example, one can build coarse-grained models for actin, DNA, ion channels, motor proteins etc to characterize their physical properties.

Bibliography

[1] H. T. McMahon and J. L. Gallop, "Membrane curvature and mechanisms of dynamic cell membrane remodelling," *Nature*, vol. 438, no. 7068, pp. 590–596, 2005.

[2] "nuclear envelope." `https://www.britannica.com/science/nuclear-envelope#/media/1/421602/114952`. Accessed: 7-05-2021.

[3] Y. Kanamaru, S. Sekine, H. Ichijo, and K. Takeda, "The phosphorylation-dependent regulation of mitochondrial proteins in stress responses," *Journal of signal transduction*, vol. 2012, p. 931215, 07 2012.

[4] A. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem (1936)*, pp. 51–60. 02 2021.

[5] J. F. Robeson, *Logistics handbook*. Simon and Schuster, 1994.

[6] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1, pp. 19–25, 2015.

[7] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. Berendsen, "Gromacs: fast, flexible, and free," *Journal of computational chemistry*, vol. 26, no. 16, pp. 1701–1718, 2005.

[8] H. J. Berendsen, D. van der Spoel, and R. van Drunen, "Gromacs: a message-passing parallel molecular dynamics implementation," *Computer physics communications*, vol. 91, no. 1-3, pp. 43–56, 1995.

[9] E. Lindahl, B. Hess, and D. Van Der Spoel, "Gromacs 3.0: a package for molecular simulation and trajectory analysis," *Molecular modeling annual*, vol. 7, no. 8, pp. 306–317, 2001.

[10] L.-s. Atomic and M. M. P. Simulator, "Lammps," *available at: http:/lammps. sandia. gov*, 2013.

[11] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of computational physics*, vol. 117, no. 1, pp. 1–19, 1995.

[12] G. Gompper and D. Kroll, "Triangulated-surface models of fluctuating membranes," *Statistical mechanics of membranes and surfaces*, pp. 359–426, 2004.

[13] D. Kroll and G. Gompper, "The conformation of fluid membranes: Monte carlo simulations," *Science*, vol. 255, no. 5047, pp. 968–971, 1992.

[14] G. Gompper and D. M. Kroll, "Network models of fluid, hexatic and polymerized membranes," *Journal of Physics: Condensed Matter*, vol. 9, no. 42, p. 8795, 1997.

[15] P. S. Kumar, G. Gompper, and R. Lipowsky, "Budding dynamics of multicomponent membranes," *Physical review letters*, vol. 86, no. 17, p. 3911, 2001.

[16] H. Noguchi, "Shape transitions of high-genus fluid vesicles," *EPL (Europhysics Letters)*, vol. 112, no. 5, p. 58004, 2015.

[17] H. Noguchi, "Construction of nuclear envelope shape by a high-genus vesicle with pore-size constraint," *Biophysical journal*, vol. 111, no. 4, pp. 824–831, 2016.

[18] H. Noguchi and G. Gompper, "Shape transitions of fluid vesicles and red blood cells in capillary flows," *Proceedings of the National Academy of Sciences*, vol. 102, no. 40, pp. 14159–14164, 2005.

[19] J. L. McWhirter, H. Noguchi, and G. Gompper, "Flow-induced clustering and alignment of vesicles and red blood cells in microcapillaries," *Proceedings of the National Academy of Sciences*, vol. 106, no. 15, pp. 6039–6043, 2009.

[20] H. Noguchi, "Membrane tubule formation by banana-shaped proteins with or without transient network structure," *Scientific reports*, vol. 6, no. 1, pp. 1–8, 2016.

[21] N. Cordella, T. J. Lampo, S. Mehraeen, and A. J. Spakowitz, "Membrane fluctuations destabilize clathrin protein lattice order," *Biophysical journal*, vol. 106, no. 7, pp. 1476–1488, 2014.

[22] N. Cordella, T. J. Lampo, N. Melosh, and A. J. Spakowitz, "Membrane indentation triggers clathrin lattice reorganization and fluidization," *Soft matter*, vol. 11, no. 3, pp. 439–448, 2015.

[23] S. Mehraeen, N. Cordella, J. S. Yoo, and A. J. Spakowitz, "Impact of defect creation and motion on the thermodynamics and large-scale reorganization of self-assembled clathrin lattices," *Soft Matter*, vol. 7, no. 19, pp. 8789–8799, 2011.

[24] M. Giani, *Modeling the self-assembly of clathrin coats*. University of Twente, 2017.

[25] M. Giani, W. K. den Otter, and W. J. Briels, "Clathrin assembly regulated by adaptor proteins in coarse-grained models," *Biophysical journal*, vol. 111, no. 1, pp. 222–235, 2016.

[26] M. Simunovic, G. A. Voth, A. Callan-Jones, and P. Bassereau, "When physics takes over: Bar proteins and membrane curvature," *Trends in cell biology*, vol. 25, no. 12, pp. 780–792, 2015.

[27] C.-L. Lai, C. C. Jao, E. Lyman, J. L. Gallop, B. J. Peter, H. T. McMahon, R. Langen, and G. A. Voth, "Membrane binding and self-association of the epsin n-terminal homology domain," *Journal of molecular biology*, vol. 423, no. 5, pp. 800–817, 2012.

[28] C. Mim, H. Cui, J. A. Gawronski-Salerno, A. Frost, E. Lyman, G. A. Voth, and V. M. Unger, "Structural basis of membrane bending by the n-bar protein endophilin," *Cell*, vol. 149, no. 1, pp. 137–145, 2012.

[29] N. Ramakrishnan, P. S. Kumar, and J. H. Ipsen, "Monte carlo simulations of fluid vesicles with in-plane orientational ordering," *Physical Review E*, vol. 81, no. 4, p. 041922, 2010.

[30] R. W. Tourdot, R. P. Bradley, N. Ramakrishnan, and R. Radhakrishnan, "Multiscale computational models in physical systems biology of intracellular trafficking," *IET systems biology*, vol. 8, no. 5, pp. 198–213, 2014.

[31] R. W. Tourdot, N. Ramakrishnan, and R. Radhakrishnan, "Defining the free-energy landscape of curvature-inducing proteins on membrane bilayers," *Physical Review E*, vol. 90, no. 2, p. 022717, 2014.

[32] "Complimentary action of structured and unstructured domains of epsin supports clathrin-mediated endocytosis at high tension," *Communications Biology*, vol. 3, 12 2020.

[33] M. Kaksonen and A. Roux, "Mechanisms of clathrin-mediated endocytosis," *Nature reviews Molecular cell biology*, vol. 19, no. 5, pp. 313–326, 2018.

[34] J. P. Ferguson, N. M. Willy, S. P. Heidotting, S. D. Huber, M. J. Webber, and C. Kural, "Deciphering dynamics of clathrin-mediated endocytosis in a living organism," *Journal of Cell Biology*, vol. 214, no. 3, pp. 347–358, 2016.

[35] H. T. McMahon and E. Boucrot, "Molecular mechanism and physiological functions of clathrin-mediated endocytosis," *Nature reviews Molecular cell biology*, vol. 12, no. 8, pp. 517–533, 2011.

[36] S. D. Conner and S. L. Schmid, "Regulated portals of entry into the cell," *Nature*, vol. 422, no. 6927, pp. 37–44, 2003.

[37] I. K. Jarsch, F. Daste, and J. L. Gallop, "Membrane curvature in cell biology: An integration of molecular mechanisms," *Journal of Cell Biology*, vol. 214, no. 4, pp. 375–387, 2016.

[38] E. Cocucci, F. Aguet, S. Boulant, and T. Kirchhausen, "The first five seconds in the life of a clathrin-coated pit," *Cell*, vol. 150, no. 3, pp. 495–507, 2012.

[39] E. M. Schmid and H. T. McMahon, "Integrating molecular and network biology to decode endocytosis," *Nature*, vol. 448, no. 7156, pp. 883–888, 2007.

[40] D. Bucher, F. Frey, K. A. Sochacki, S. Kummer, J.-P. Bergeest, W. J. Godinez, H.-G. Kräusslich, K. Rohr, J. W. Taraska, U. S. Schwarz, *et al.*, "Clathrin-adaptor ratio and membrane tension regulate the flat-to-curved transition of the clathrin coat during endocytosis," *Nature communications*, vol. 9, no. 1, pp. 1–13, 2018.

[41] B. L. Scott, K. A. Sochacki, S. T. Low-Nam, E. M. Bailey, Q. Luu, A. Hor, A. M. Dickey, S. Smith, J. G. Kerkvliet, J. W. Taraska, *et al.*, "Membrane bending occurs at all stages of clathrin-coat assembly and defines endocytic dynamics," *Nature communications*, vol. 9, no. 1, pp. 1–9, 2018.

[42] J. E. Hassinger, G. Oster, D. G. Drubin, and P. Rangamani, "Design principles for robust vesiculation in clathrin-mediated endocytosis," *Proceedings of the national academy of sciences*, vol. 114, no. 7, pp. E1118–E1127, 2017.

[43] S. Boulant, C. Kural, J.-C. Zeeh, F. Ubelmann, and T. Kirchhausen, "Actin dynamics counteract membrane tension during clathrin-mediated endocytosis," *Nature cell biology*, vol. 13, no. 9, pp. 1124–1131, 2011.

[44] N. Walani, J. Torres, and A. Agrawal, "Endocytic proteins drive vesicle growth via instability in high membrane tension environment," *Proceedings of the national academy of sciences*, vol. 112, no. 12, pp. E1423–E1432, 2015.

[45] J. P. Ferguson, S. D. Huber, N. M. Willy, E. Aygün, S. Goker, T. Atabey, and C. Kural, "Mechanoregulation of clathrin-mediated endocytosis," *Journal of cell science*, vol. 130, no. 21, pp. 3631–3636, 2017.

[46] J. G. Joseph and A. P. Liu, "Mechanical regulation of endocytosis: new insights and recent advances," *Advanced biosystems*, vol. 4, no. 5, p. 1900278, 2020.

[47] M. G. Ford, I. G. Mills, B. J. Peter, Y. Vallis, G. J. Praefcke, P. R. Evans, and H. T. McMahon, "Curvature of clathrin-coated pits driven by epsin," *Nature*, vol. 419, no. 6905, pp. 361–366, 2002.

[48] M. Messa, R. Fernández-Busnadiego, E. W. Sun, H. Chen, H. Czapla, K. Wrasman, Y. Wu, G. Ko, T. Ross, B. Wendland, *et al.*, "Epsin deficiency impairs endocytosis

by stalling the actin-dependent invagination of endocytic clathrin-coated pits," *Elife*, vol. 3, 2014.

[49] V. Legendre-Guillemin, S. Wasiak, N. K. Hussain, A. Angers, and P. S. McPherson, "Enth/anth proteins and clathrin-mediated membrane budding," *Journal of cell science*, vol. 117, no. 1, pp. 9–18, 2004.

[50] S. S. Holkar, S. C. Kamerkar, and T. J. Pucadyil, "Spatial control of epsin-induced clathrin assembly by membrane curvature," *Journal of Biological Chemistry*, vol. 290, no. 23, pp. 14267–14276, 2015.

[51] A. Sen, K. Madhivanan, D. Mukherjee, and R. C. Aguilar, "The epsin protein family: coordinators of endocytosis and signaling," *Biomolecular concepts*, vol. 3, no. 2, pp. 117–126, 2012.

[52] S. M. Smith, M. Baker, M. Halebian, and C. J. Smith, "Weak molecular interactions in clathrin-mediated endocytosis," *Frontiers in molecular biosciences*, vol. 4, p. 72, 2017.

[53] D. J. Busch, J. R. Houser, C. C. Hayden, M. B. Sherman, E. M. Lafer, and J. C. Stachowiak, "Intrinsically disordered proteins drive membrane curvature," *Nature communications*, vol. 6, no. 1, pp. 1–11, 2015.

[54] J. C. Stachowiak, E. M. Schmid, C. J. Ryan, H. S. Ann, D. Y. Sasaki, M. B. Sherman, P. L. Geissler, D. A. Fletcher, and C. C. Hayden, "Membrane bending by protein–protein crowding," *Nature cell biology*, vol. 14, no. 9, pp. 944–949, 2012.

[55] M. Gleisner, B. Kroppen, C. Fricke, N. Teske, T.-T. Kliesch, A. Janshoff, M. Meinecke, and C. Steinem, "Epsin n-terminal homology domain (enth) activity as a function of membrane tension," *Journal of Biological Chemistry*, vol. 291, no. 38, pp. 19953–19961, 2016.

[56] J. Hutchison, A. Karunanayake M, R. Weis, and A. Dinsmore, "Osmotically-induced tension and the binding of n-bar protein to lipid vesicles," *Soft Matter*, vol. 12, 01 2016.

[57] B. Capraro, Y. Yoon, W. Cho, and T. Baumgart, "Curvature sensing by the epsin n-terminal homology (enth) domain measured on cylindrical lipid membrane tethers," *Journal of the American Chemical Society*, vol. 132, pp. 1200–1, 02 2010.

[58] G. Drin and B. Antonny, "Amphipathic helices and membrane curvature," *FEBS letters*, vol. 584, pp. 1840–7, 10 2009.

[59] H. Cui, E. Lyman, and G. Voth, "Mechanism of membrane curvature sensing by amphipathic helix containing proteins," *Biophysical journal*, vol. 100, pp. 1271–9, 03 2011.

[60] Z. Shi and T. Baumgart, "Membrane tension and peripheral protein density mediate membrane shape transitions," *Nature communications*, vol. 6, p. 5974, 01 2015.

[61] J. Huang and A. MacKerell, "Charmm36 all-atom additive protein force field: Validation based on comparison to nmr data," *Journal of computational chemistry*, vol. 34, 09 2013.

[62] S. Jo, T. Kim, V. Iyer, and W. Im, "Charmm-gui: a web-based graphical user interface for charmm," *Journal of computational chemistry*, vol. 29, pp. 1859–65, 08 2008.

[63] W. Humphrey, A. Dalke, and K. Schulten, "Vmd: Visual molecular dynamics," *Journal of molecular graphics*, vol. 14, pp. 33–8, 27, 03 1996.

[64] D. Horton, G. Bourne, and M. Smythe, "J. comput. aided mol. des," 07 2021.

[65] P. Wiggins and R. Phillips, "Analytic models for mechanotransduction: gating a mechanosensitive channel," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 4071–4076, 2004.

[66] P. Wiggins and R. Phillips, "Membrane-protein interactions in mechanosensitive channels," *Biophysical journal*, vol. 88, pp. 880–902, 03 2005.

[67] H. T. Tien and A. Ottova-Leitmannova, *Membrane biophysics: as viewed from experimental bilayer lipid membranes*. Elsevier, 2000.

[68] W. Helfrich, "Elastic properties of lipid bilayers: theory and possible experiments," *Zeitschrift für Naturforschung C*, vol. 28, no. 11-12, pp. 693–703, 1973.

[69] M. Meyer, P. Oder, and A. Barr, "Discrete differential-geometry operators in nd," 09 2000.

[70] J. Vince, *Barycentric Coordinates*, pp. 307–336. 08 2017.

[71] F. Dabek, N. Zeldovich, F. Kaashoek, D. Eres, and R. Morris, "Event-driven programming for robust software," *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop, EW 10*, 08 2002.

[72] A. R. Braun and J. N. Sachs, "Determining structural and mechanical properties from molecular dynamics simulations of lipid vesicles," *Journal of chemical theory and computation*, vol. 10, no. 9, pp. 4160–4168, 2014.

[73] H. Beams and R. Kessel, "The golgi apparatus: structure and function," *International review of cytology*, vol. 23, pp. 209–276, 1968.

[74] M. W. Hetzer, "The nuclear envelope," *Cold Spring Harbor perspectives in biology*, vol. 2, no. 3, p. a000539, 2010.

[75] M. Torbati, T. P. Lele, and A. Agrawal, "Ultradonut topology of the nuclear envelope," *Proceedings of the National Academy of Sciences*, vol. 113, no. 40, pp. 11094–11099, 2016.

[76] A. Agrawal and T. P. Lele, "Geometry of the nuclear envelope determines its flexural stiffness," *Molecular biology of the cell*, vol. 31, no. 16, pp. 1815–1821, 2020.

[77] T. Ursell, W. Klug, and R. Phillips, "Morphology and interaction between lipid domains," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, pp. 13301–6, 08 2009.

[78] T. Baumgart, S. T. Hess, and W. W. Webb, "Imaging coexisting fluid domains in biomembrane models coupling curvature and line tension," *Nature*, vol. 425, no. 6960, pp. 821–824, 2003.

[79] J. Liu, Y. Sun, D. Drubin, and G. Oster, "The mechanochemistry of endocytosis," *PLoS biology*, vol. 7, p. e1000204, 09 2009.