Investigations of Mechanical Properties of Polymer Brushes and Hydrogels Through Molecular Dynamics Simulation

A dissertation presented

by

Yangpeng Ou

to

The Department of Physics

In partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in the field of

Physics

Northeastern University

Boston, MA

April 23, 2013
INVESTIGATIONS OF MECHANICAL PROPERTIES OF POLYMER BRUSHES AND HYDROGELS THROUGH MOLECULAR DYNAMICS SIMULATION

by

Yangpeng Ou

ABSTRACT OF DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Physics in the College of Science of Northeastern University April 23, 2013
Abstract

In this thesis, the forces between surfaces coated with polyelectrolye brushes and polymer gels were investigated via molecular dynamics simulation on a computer workstation that I built.

The first project investigated the reasons for why grafted polyelectrolye brushes have smaller friction coefficients than grafted neutral polymer brushes. The flexible neutral polymer brush is treated as a bead-spring model, and the polyelectrolyte brush is treated the same way except that each bead is charged and there are counterions present to neutralize the charge. We investigated the friction coefficient, monomer density, and brush penetration for the two kinds of brushes with both the same grafting density and the same normal force under good solvent conditions. We found that polyelectrolyte brushes have smaller friction coefficients in both simulations. We present evidence that the reason for this is that the extra normal force contribution provided by the counterion osmotic pressure that exists for polyelectrolyte brushes permits them to support the same load as identical neutral polymer brushes of higher grafting density. Because of the resulting lower monomer density for the charged brushes, fewer monomer collisions take place per unit time, resulting in a lower friction coefficient.

The second project investigated interactions between two grafted polymer gels. Unlike polyelectrolyte polymer brushes, polymer chains are linked to other neighboring polymer chains. We studied a defect-free network of
diamond-like topology containing 8 tetra-functional nodes linked by 16 non-crossing chains. We studied the shear force, friction coefficient, and density profiles of these polymer gels. We found that polymer gels have very small friction coefficients in our simulation, which is consistent with previous experimental measurements. In order to understand and explain this very low friction coefficient behavior, we computed the interpenetrations between two polymer gels, the pressure tensor at polymer gels’ interface, and the radial distribution function of the interacting monomers. From our analysis, we found that the linked network in the polymer gels restricts the motion of the monomers. As a result, there is little interpenetration between two apposing polymer gels, and hence little interaction between two polymer gels. The low interpenetration results smaller friction force, which leads to a small friction coefficients for polymer gels in our simulations.
Acknowledgments

First and foremost I would like to thank my advisor Prof. Sokoloff for his guidance and supervision through the entire course of my research years at Northeastern University. I appreciate his patience in many times of useful discussions. I also want to thank Pro. Sokoloff for funding me to build my own computer workstation. His support and guidance were invaluable during my PhD study.

I would also like to thank Dr. Mark Stevens at Sandia National Laboratory. Without his help, it would have been difficult to complete my research. Dr. Stevens taught me to apply molecular dynamics simulation to polymer brush problems, shared his private programs, and guided me through my research progress through hundreds of emails.

I would also like to thank the members of my thesis committee, Prof. Stepanyants and Prof. Markiewicz, for their teaching and other help over the years.

I would also like to thank my parents for their unlimited support for my PhD study. They gave me financial support and confidence to pursue the PhD degree.
## Contents

Abstract 2

Acknowledgments 4

1 Introduction 8

2 Theory 12

2.1 Polymer chains 12

2.1.1 Ideal chain 12

2.1.2 Polymer brushes 14

2.2 Simulation methods 15

2.3 Parallel simulation engines 15

2.4 Application of Nvidia CUDA to the molecular dynamics 28

2.5 Periodic boundary conditions 31

2.6 Dimensionless units 34

2.7 Integration algorithms 36

3 Polymer Brush 38
Chapter 1

Introduction

Polymers are long molecule chains of chemically bonded large molecules. They play very important roles in industrial engineering applications, and have attracted interest of academic researches. The polymerization of polymer, which is defined as the number of monomers per chain, is typically $10^2$ to $10^6$. The large magnitude of the polymerization makes it possible to use statistical mechanics to study them\textsuperscript{1}. Polymer brushes are formed when polymer chains are attached to a substrate surface through end anchor groups. The chains stretch from surfaces and repel each other. The balance between elasticity and repulsion of the chains generates conformations and properties completely different from those for isolated chains.\textsuperscript{2}.

A polymer brush consists of polymer chains densely grafted to a solid surface, which is immersed in a solvent. The chains stretch from surfaces and repel each other. The balance between elasticity and repulsion of the
chains generates completely different conformations and properties than for isolated chains. Polymer brushes have many technological applications, such as colloidal stabilization and lubrication. Electrostatic interaction involved in polyelectrolyte brushes results in a number of additional physical properties. Mutual repulsion between polymer segments and electrostatic forces between charged monomers and counterions strongly influences the conformation of the polyelectrolyte brush [3, 4]. A great deal of theoretical and experimental research has been conducted to investigate the frictional behavior when lateral sliding between two apposing polymer brush surfaces occurs [5, 6, 7, 8, 9, 10, 11]. The normal force and shear force have been well studied experimentally by the surface force apparatus (SFA) both on neutral polymer brush surfaces and polyelectrolyte brush coated surfaces. These experiments on neutral polymer brushes showed that the polymer brush surfaces have ultralow friction coefficients when the surfaces are brought into contact [12]. Molecular dynamics simulations for a bead-spring model provide excellent agreement with experimental results [5]. The experiment conducted by Raviv, et al., on two apposing polyelectrolyte brush walls using the surface force apparatus (SFA) also found very low friction coefficients when one polyelectrolyte brush slides relative to the other [13]. Liberelle used the SFA to study weakly charged polymer brushes, investigating the friction is a function of salt (NaCl) concentration [14]. Because of the small distance involved and limitation of the experimental apparatus, it is difficult to observe what is happening inside the polymer brushes when they slide. Molecular dynamics
simulations give us insights into what may be occurring inside the brushes. Earlier molecular simulations examined shear and compression of apposing neutral brush surfaces providing good agreement with experiment [5]. Using the same bead-spring model, we compare the behavior of neutral polymer brushes and polyelectrolyte brushes through molecular dynamics simulation.

Over the past decades, polymer gels have gained a lot of attention from both science and industry[15, 16]. Industry utilizes polymer gels’ ability to swell by absorbing huge amounts of solvent molecules, in numerous applications, such as drug delivery, agriculture, and environmental cleanup operations.

Polymer gels are three dimensional networks, composed of polymer molecules, connected to each other through functional groups. The large number of molecules inside networks of swollen gels gives the gels solidity and coherence, but gels can undergo a large deformation. Polymer gels are neither solid nor liquid, but have some features of solid and some features of liquid. [17, 18, 19, 20, 21]. Because of these specific properties of polymer gels, they have many unique properties, such as phase transition, chemomechanical behavior, shape memory effect, etc [22, 23, 24].

Some biological surfaces show ultra low friction coefficients. For example, cartilages of animal joints have a very low friction coefficients, about $0.001 - 0.03$[25, 26, 27]. The reason why the cartilage friction of the joints is so low when the pressure between the bone surfaces is as high as $3 - 18$ MPa is not clear now [28, 29, 30]. Animal cartilage consists of a three dimensional
collagen network filled with a synovial fluid. These facts inspired people to investigate the frictional behaviors of polymer gels, because they are similar to those of cartilage. A number of experiments have been performed to measure the friction coefficients between polymer networks [21, 31, 32].
Chapter 2

Theory

This chapter reviews fundamental theoretical methods, which are widely used in dealing with polymer related problems. The chapter also reviews several numerical methods used to solve the polymer brush problem. In addition to normal integration schemes, the chapter reviews parallel algorithms to solve systems of equations used in molecular dynamics simulations.

2.1 Polymer chains

2.1.1 Ideal chain

The simplest model of an ideal polymer chain is the freely jointed chain model. The configuration of a polymer chain is identified by the positions of $N+1$ monomers, $\{R_0, R_1, \cdots, R_N\}$, as shown in figure 2.1. The bond vectors connecting neighboring monomers on the polymer chains are $r_i = R_{i+1} - R_i$. 
and are considered to have identical length, equal to $b$ in the ideal free chain model. The monomers positions are assumed to result from a series of random walks with fixed length.

\[ R = R_N - R_0 = \sum_{i=0}^{N-1} r_i, \quad (2.1) \]

The average end-to-end distance $\langle R \rangle$ is 0, but the mean square end-to-end distance $\langle R^2 \rangle$ is not, and it is given by

\[ \langle R^2 \rangle = \sum_{i=0}^{N-1} \langle r_i^2 \rangle + 2 \sum_{i<j} \langle r_i r_j \rangle = N b^2, \quad (2.2) \]

The mean size of polymer chain $l = \sqrt{N} b$ is approximately correct for flexible polymer chains, since neighboring bonds in flexible chains have little correlation.
2.1.2 Polymer brushes

Consider a substrate with polymer chains grafted on it, so that the ends of the polymer chains, which are attached to the substrate, are not free to move. We define the grafting density $\sigma$ as the ratio of the number of such grafted polymer chains $N_{\text{chain}}$ to the area of substrate surface $S$,

$$\sigma = \frac{N_{\text{chain}}}{S}, \quad (2.3)$$

If the grafting density $\sigma$ is small, neighboring chains have little interaction, and behave like isolated chains on surface. This is the so called “mushroom regime”, since the polymers curl up and hence resemble mushrooms. When the grafting density increases, neighboring chains have much stronger interactions with each other, so that polymer chains are no longer so curled up, and they will stand up on surface like a brush. This is known as the “brush regime”. There are also other surface substrate configurations sometimes found in nature, such as a spherical substrates. However, this thesis will only focus on chains grafted on planar surfaces.

There are both neutral and charged brushes. Unlike neutral polymer brushes, monomers on charged brushes have positive or negative charges on them. The physical properties of a polyelectrolyte brush are normally different from those of a neutral brush. In addition to pairwise interactions between neighboring chains, there is also electrical interactions between the monomers on a polyelectrolyte brush. Coulomb interactions between charges
on molecules, and counter-ions in solvent, influence the behaviors of poly-
electrolyte brushes, and make them different from neutral brushes. Other
external conditions also influence behavior of polyelectrolyte brushes. For
example, They behave differently in monovalent counterion solvent and mul-
tivalent counterion solvent.

2.2 Simulation methods

Computational simulation methods have been widely used in investigate
behaviors and properties of polymer systems for many years. The thesis ap-
plies these simulation methods to planar polyelectrolyte brushes and polymer
gels.

2.3 Parallel simulation engines

I used the LAMMPS (Large-scale Atomic Molecular Massively Parallel
Simulation) molecular dynamics computation program to perform all my
computer simulations. LAMMPS is a classical molecular dynamics code
that models an ensemble of particles in a liquid, solid, or gaseous state.
It can model atomic, polymeric, biological, metallic, granular, and coarse-
grained systems using a variety of force fields and boundary conditions[33,
34]. LAMMPS is a parallel molecular dynamics computation program, devel-
oped using C++ language. It partitions the simulation domains into smaller

15
sub-domains using spatial decomposition techniques, and assigns each smaller divided simulation domain to one processor using MPI[35].

The molecular dynamics method is a widely used computational tool to investigate the properties of liquids, solids, and molecules. Each of the atoms or molecules in the simulation is treated as a point mass, and Newton’s equations are integrated to compute their motion. A good deal of useful microscopic and macroscopic information can be extracted from the motion of atoms or molecules in the system, such as structural or conformational properties, phase diagrams, and friction coefficients. The motion of these atoms or molecules in the simulation system and properties of the simulated systems are controlled by interaction energy functions between these atoms or molecules, and the force equation of each atom or molecule unit is derived from these total potential energies.

The molecular dynamics computation does not consume a large amount of memory, but it does use a huge amount of CPU. During the molecular dynamics computation, the program only needs to keep a record of position and velocity information of each atom or molecule in the theory. Suppose the simulated system contained $10^6$ atoms, and the computation program stored position points and velocities with a double precision floating point format. The memory storage needed to keep record of position points and velocities for the system would be $8 \times 2 \times 10^6$ bytes, or 16MB. Since even personal computers in the current market have 4GB memory, memory has never been a bottleneck for the molecular dynamics simulation. The bottleneck for the
molecular dynamics simulation is determined by CPU performance, that is how many floating point calculations per second the processor provides. The simulation systems always contain thousands or millions of particles in order to allow the simulated system to represent real materials in nature without distortions. And the number of all atoms or molecules $N$ ranges from $10^3$ to $10^6$. The simulation time step is always chosen to be less than $0.01\tau$ in Lennard Jones (LJ) units (LJ units defined in section 2.6), and the system is required to run millions of time steps to allow the system to relax and reach equilibrium. Consider the simplest molecular dynamics simulation algorithm on an Intel i7 CPU. For each atom or molecule in the system, one calculates the interaction forces between one atom and other $(N - 1)$ atoms, $F_{ij}(i \in (1, \cdots, N), j = 1, 2, \cdots, i-1, i+1, \cdots, N)$. To find the net force for each atom or molecule in the simulated system, it is necessary to read $N \times (N - 1)$ times positions of molecules, approximately $1000000 \times 1000000 = 10^{12}$ times for a simulated system containing 10000000 molecules. The Intel i7 CPU can provide $50 \times 10^9$ floating-point calculations per second\(^1\). Assuming that it takes 1 floating point calculation to find each $F_{ij}$, it could cost 20 seconds for each time step run during this simplest molecular dynamics simulator. In this simplest algorithm, it costs a huge of time to compute the net force on each atom in the system.

Because the computation requires us to simulate thousands or millions of molecules for millions of time step runs, and the limitation of computa-

---

\(^1\)http://www.intel.com
tion power provided by current CPU, researchers made efforts to optimize molecular dynamics computation[36, 37]. In most situations, each atom or monomer could potentially interact with any atom or monomer of the other remaining $N - 1$ particles. To determine the net force $F_i$ on each atom or molecule, it is necessary to examine $\frac{N(N-1)}{2}$ pairs of particles each time step, and hence it will take $O(N^2)$ time to finish. This simple method works for systems with a small number of particles $N$, but it takes too much time for systems with a large number of particles $N$. Therefore, it is necessary to look for an efficient method to find which particles interact with the particle $i$. For short-force interactions, the interaction forces decay very rapidly when two particles move far away from each other, and only the particles inside a cutoff distance $r_c$ of the particle $i$ contribute significantly to the net force $F_i$ exerted by the other particles. For this reason, the neighbor table was introduced. The neighbor table of a particle $i$ stores all particles near it, whose interaction force with the particle $i$ cannot be neglected. A significant reduction of computation time can be achieved by using a neighbor table to store particles inside the range $r_c + r_s$ for each particle $i$. $r_s$ is skin distance, and a suitable value can be chosen to optimize this method[38, 39]. If a larger skin distance $r_s$ is chosen, although the positions and velocities of particles near the atom $i$ changed after a time step in the simulation, the old neighbor table of the particle $i$ still could capture particles inside the cutoff $r_c$ distance of particle $i$, and it is not necessary to update the neighbor table again. Therefore, a larger skin distance $r_s$ decreases the frequency to update
neighbor table between time steps in the simulation, but it increases the cost to build the neighbor table. A good choice of skin distance $r_s$ could improve the computation performance.

Molecular dynamics simulations are great candidates for parallel programming. For each time step in the simulation, the computations of the net force on each atom or molecule $i$ are independent, because the net force computation only depends on the old positions of each particle in the simulation system. Several parallel computation algorithms have been developed to speed up the molecular dynamics computation, and make simulating systems with a large number of atoms or molecules possible[40, 35, 41, 42].

**Atom Decomposition Algorithm**

The atom decomposition algorithm, also called replicated-data algorithms, assigns $M = \frac{N}{P}$ atoms or molecules to each processor, when $N$ total particles are simulated parallel on $P$ processors. The assignment occurs in the beginning of simulation, and does not change during the whole simulation run. Each processor only computes the net forces of particles assigned to it, and updates their positions and velocities during the simulation. The calculations of net force, position, and velocities of each particle $i$ are performed on a specific processor $k$ during the whole simulation run.

The matrix 2.4 gives the force matrix of the simulation system. The atom decomposition method assigns particles 1, 2, · · · , and $M$ to the 1st processor, particle $M+1$, · · · and $2M$ to the 2nd processor, and particle $N-M+1$, · · · ,and
$N$ to the last processor $P$. The 1$^{st}$ processor is responsible for first $M$ rows of force matrix $F_{ij}$, the 2$^{nd}$ processor is responsible for rows $M+1, \cdots, 2M$ of force matrix $F_{ij}$, and the last processor is responsible for rows $N-M+1, \cdots, N$ for force matrix $F_{ij}$.

For the 1$^{st}$ particle in the 1$^{st}$ processor, it is necessary to compute $F_{1,2}, F_{1,3}, \cdots, F_{1,N}$ in order to compute the net force on the 1$^{st}$ exerted by other $N-1$ particles. Therefore, the 1$^{st}$ processor needs to know all positions of $N$ particles in the simulation system, and each processor stores a copy of all positions of $N$ particles. This means that processors have to communicate to exchange position information of their assigned particles every time step. Fox’s algorithm is a commonly used one to communicate position information in the
The atom decomposition algorithm divides the computation of positions and velocities equally across all processors. However, the algorithm requires a large number of communications between processors, in order to let each process achieve updated positions of all particles in the system. For a large simulation system with huge number of particles, this communication every time step would be a great burden for the computers. Therefore, the algorithm is not suitable for simulating large systems, but its simplicity makes it a good candidate for simulating small systems.

\[
\begin{bmatrix}
F_{1,1} & \cdots & F_{1,M} & \cdots & F_{1,N-M+1} & \cdots & F_{1,N} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
F_{M,1} & \cdots & F_{M,M} & \cdots & F_{M,N-M+1} & \cdots & F_{M,N} \\
F_{M+1,1} & \cdots & F_{M+1,M} & \cdots & F_{M+1,N-M+1} & \cdots & F_{M+1,N} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
F_{2M,1} & \cdots & F_{2M,M} & \cdots & F_{2M,N-M+1} & \cdots & F_{2M,N} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
F_{N-M+1,1} & \cdots & F_{N-M+1,M} & \cdots & F_{N-M+1,N-M+1} & \cdots & F_{N-M+1,N} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
F_{N,1} & \cdots & F_{N,M} & \cdots & F_{N,N-M+1} & \cdots & F_{N,N} \\
\end{bmatrix}
\]
Force Decomposition Algorithm

The atom decomposition algorithm divides the interaction force matrix $F_{ij}$ into $P$ force matrices $F'_{ij}$, which have the same dimension $M \times N$, ($M = \frac{N}{P}$). Unlike the atom decomposition algorithm, the force decomposition discussed in this section divides the interaction force matrix $F_{ij}$ blockwise into $P$ smaller sub matrices, which have the same dimension $M \times M$, ($M = \frac{N}{\sqrt{P}}$) [40]. Here the algorithm assumes that the number of particles could divide $\sqrt{P}$, the square root of the number of processors $P$.

The parallel computation program assigns each of the $P$ processors to compute each of $P$ smaller sub matrices. Each processor only stores positions and velocities contained within the sub force matrix $F'_{ij}$ it was assigned. The $1^{st}$ processor is assigned to compute the interaction force between particle $1, 2, \ldots, \frac{N}{\sqrt{P}}$. It stores positions of particle $1, 2, \ldots, \frac{N}{\sqrt{P}}$, but does not know the positions of other remaining points. It computes the interaction force of particle $i$, $(i = 1, \ldots, \frac{N}{\sqrt{P}})$ due to other particles $1, \ldots, i - 1, i + 1, \ldots$, and $\frac{N}{\sqrt{P}}$. From the $1^{st}$ processor computation, it can get the partial net total force on particle 1. The formula 2.5 gives

$$F_1(P_1) = \sum_{j=2}^{\frac{N}{\sqrt{P}}} F_{1,j}, \quad (2.5)$$

the partial net force on $1^{st}$ particle exerted by the other $\sqrt{P} - 1$ particles assigned to processor 1, $P_1$. Likewise, the $2^{nd}$ processor $P_2$ could give the
partial net force $F_1(P_2)$ exerted on particle 1 by particles $\sqrt{P} + 1, \cdots, 2\sqrt{P}$; the processor $P_{\sqrt{P}}$ compute the partial force $F_1(P_{\sqrt{P}})$ on particle 1 due to particles $N - \sqrt{P} + 1, \cdots, N$. From these computation, the total net force on 1st particle $F_1$ is shown in equation 2.6,

$$F_1 = F_1(P_1) + F_1(P_2) + \cdots + F_1(P_{\sqrt{P}}). \quad \text{(2.6)}$$

For example, the matrix 2.7 specifies the interaction force matrix for a system containing 9 particles.

\[
\begin{pmatrix}
F_{1,1} & F_{1,2} & F_{1,3} & F_{1,4} & F_{1,5} & F_{1,6} & F_{1,7} & F_{1,8} & F_{1,9} \\
F_{2,1} & F_{2,2} & F_{2,3} & F_{2,4} & F_{2,5} & F_{2,6} & F_{2,7} & F_{2,8} & F_{2,9} \\
F_{3,1} & F_{3,2} & F_{3,3} & F_{3,4} & F_{3,5} & F_{3,6} & F_{3,7} & F_{3,8} & F_{3,9} \\
F_{4,1} & F_{4,2} & F_{4,3} & F_{4,4} & F_{4,5} & F_{4,6} & F_{4,7} & F_{4,8} & F_{4,9} \\
F_{5,1} & F_{5,2} & F_{5,3} & F_{5,4} & F_{5,5} & F_{5,6} & F_{5,7} & F_{5,8} & F_{5,9} \\
F_{6,1} & F_{6,2} & F_{6,3} & F_{6,4} & F_{6,5} & F_{6,6} & F_{6,7} & F_{6,8} & F_{6,9} \\
F_{7,1} & F_{7,2} & F_{7,3} & F_{7,4} & F_{7,5} & F_{7,6} & F_{7,7} & F_{7,8} & F_{7,9} \\
F_{8,1} & F_{8,2} & F_{8,3} & F_{8,4} & F_{8,5} & F_{8,6} & F_{8,7} & F_{8,8} & F_{8,9} \\
F_{9,1} & F_{9,2} & F_{9,3} & F_{9,4} & F_{9,5} & F_{9,6} & F_{9,7} & F_{9,8} & F_{9,9} \\
F_{10,1} & F_{10,2} & F_{10,3} & F_{10,4} & F_{10,5} & F_{10,6} & F_{10,7} & F_{10,8} & F_{10,9} \\
F_{11,1} & F_{11,2} & F_{11,3} & F_{11,4} & F_{11,5} & F_{11,6} & F_{11,7} & F_{11,8} & F_{11,9} \\
F_{12,1} & F_{12,2} & F_{12,3} & F_{12,4} & F_{12,5} & F_{12,6} & F_{12,7} & F_{12,8} & F_{12,9}
\end{pmatrix}
\]
Assume the molecular dynamics computation was run on 9 CPU processors, and the algorithm made the assignment specified in the matrix 2.8.

\[
\begin{array}{ccc}
1 & 2 & 3 \\
\hline
1 & P_1 & P_2 & P_3 \\
2 & P_4 & P_5 & P_6 \\
3 & P_7 & P_8 & P_9 \\
\end{array}
\] (2.8)

From the equation 2.6, we have to exchange \(\sqrt{P}\) partial net forces of the 1st atom in order to compute the net force on it. In general, the algorithm must exchange \(\sqrt{P}\) partial net forces across \(\sqrt{P}\) processors along the same rows. Unlike the atom-force decomposition, the force decomposition algorithm only exchanges \(\frac{N}{\sqrt{P}}\) partial net forces between \(\sqrt{P}\) CPU processors, instead of exchanging \(N - 1\) positions across all processors. It saves a lot of communication between processors at each step, compared to the previous atom decomposition algorithm.

In summary, the force decomposition algorithm divides the interaction force matrix \(F_{ij}\) block-wise. In each processor, it only needs \(O\left(\frac{N}{\sqrt{P}}\right)\) floating point calculations to compute the partial net force, and exchange \(O\left(\frac{N}{\sqrt{P}}\right)\) partial net force information along \(\sqrt{P}\) processors. Besides the reduction of communication cost, the force composition algorithm still keeps the simplicity of parallel programing using the atom decomposition algorithm.
Spatial Decomposition Algorithm

The spatial decomposition algorithm is a dynamic algorithm. The atom decomposition algorithm and force decomposition algorithm only assign the computation of the sub matrices of interaction force $F_{ij}$ once. The computations of particles are fixed to assigned CPU processor and remain until the simulations end. The assignments are made before the parallel programs run. Unlike the atom decomposition algorithm and force decomposition algorithm, the computations of particles in the spatial decomposition algorithm are not assigned to specific processor. Perhaps the 1st particle was computed in the 1st processor at time $t_1$, but it is computed by a different processor at time $t_2$. Instead, the computations of atoms or molecules inside a position sub domain of the simulation box are fixed to a specific CPU processor.

The atom decomposition algorithm divides the interaction force matrix $F_{ij}$ by rows, and needs all CPU processors to exchange positions of all atoms or molecules in the system, in order to let each CPU processor know the positions of any other particles in the system. The force decomposition algorithm makes an improvement relative to the atom decomposition. It divides the interaction force matrix $F_{ij}$ by sub block matrices. Only the CPU in the same rows need to communicate, and they only exchange the partial forces $F_i(P_j)$, in order to compute the total net force $F_i$ on the $i$th particle exerted by other particles. The 2nd algorithm only needs $\sqrt{P}$ processors to exchange $\frac{N}{\sqrt{P}}$ partial forces; it saves considerably more computation time compared to the atom decomposition algorithm.
Figure 2.2: Divide the simulation box into $10 \times 8 \times 6$ sub boxes. Assign the computation of net force and integration for these sub spatial domains to $10 \times 8 \times 6$ processors. For example, the computation work of a sub domain colored by blue is assigned to 1st CPU processor.

The spatial decomposition algorithm divides the physical domain of a simulated system into smaller 3D cubic domains. Assume the physical domain of the simulated system has the dimensions $W \times W \times W$, where $W$ is length of the cube, and there are $P$ CPU processors. According to the idea of the spatial decomposition algorithm, we divide the original simulation box into $P$ smaller cubic boxes, each with dimension $\frac{W}{\sqrt[3]{P}}$. The computation for the net forces of atoms or molecules, belonging to each smaller cubic box, is assigned to a CPU processor\cite{44, 35}.

The smaller cubic box is divided in such as way that the length, width,
or height of these smaller cubic box is larger than the cut off distance \( r_c \) of short interaction force. Therefore, any atom or molecule inside a smaller sub domain could interact with other remaining particles in the same sub domains, and other particles in the sub domains, all of which neighbor with the sub domain. The atoms or molecules in the rest of sub domains, are at distances larger than the cut off distance \( r_c \) to any particle in the small domain, and therefore the interaction force contributed by these particles in the rest of smaller cubic box can be neglected. For the above reason, in order to compute the net forces and velocities of these atoms or molecules included in its assigned sub domain, each CPU processor only need communicate with its neighboring sub domains and processors to gain the positions of particles in these neighboring sub domains. At any time, each CPU processor communicates with its 8 neighboring sub domains, which have approximate \( \frac{N}{P^3} \) particles. Therefore, the communication cost is \( O(\frac{N}{P^3}) \). This communication is much smaller than \( O(N) \) in the force decomposition algorithm. The low communication cost advantage of spatial decomposition make it perform much better for systems with larger number of particles and systems with large simulation domains.

In the spatial decomposition algorithm, each processor keeps 2 tables. One table records position and velocity information for all particles in this processor, and the other table records position and velocity information for all particles in this processor’s neighbors. After one time step integration, positions and velocities of all atoms or molecules in the system are updated.
Whenever one particle moves out of the original sub domain, or another particle from other sub domains moves into the sub simulation box, each processor rebuilds these two tables again every time step.

In summary, the spatial decomposition algorithm divides the physical simulation box of systems into smaller sub domains, and assigns $P$ CPU processor to update the positions and velocities of every $P$ sub domains. Unlike the atom decomposition and force decomposition methods, the spatial decomposition algorithm only performs local communications. This advantage of the spatial decomposition algorithm makes it a good candidate to simulate larger systems. However, the programming of the spatial decomposition algorithm is much more complicated than the previous two algorithms.

2.4 Application of Nvidia CUDA to the molecular dynamics

GPU(graphics processing units) in a video card is mainly used in video games to render geometric shapes. For several years, GPU performance has doubled every 8 months, instead of doubling in every 12 months like CPU performance. An Intel i7-980X CPU, the most powerful desktop CPU with 6 physical cores in the current market, could provide 100GFLOPS of floating point computation. However, a Nvidia graphics card, GTX580, which is only half the price of Intel i7-980X, could provide 15 times as much floating point computation.  

\[^2\text{http://en.wikipedia.org/wiki/FLOPS}\]
point computation as an Intel i7-980X.  

Primarily, three factors make the GPU a great candidate for molecular dynamics simulations. First, molecular dynamics computations are very suitable for multi-thread parallel programming, and the GPUs in video cards have hundreds or even thousands of stream processors. Second, with the software development and hardware support by graphics card companies, the GPU supports advanced condition and loop programming language. Although the GPUs still don’t support recursive calls, this weakness does not prevent them from doing molecular dynamics simulations. Third, with the rapid development of video cards, a normal video card could provide 10 ~ 20 times more floating point computation than a CPU of the same price, and provide better computation performance per dollar investment.[45, 46, 47, 33, 48, 49, 50, 46, 51].

Here shows the design architecture of a GPU in the Nvidia GTX 570, which is the model I used for my molecular dynamics simulation. The GPUs in the GTX 570 have 15 stream processors, each having 32 cores for a total of 480 cores. The device supports very high speed data transfer, 150GB/s, and could reach 1500GLFOPS floating point computation at its peak.

Figure 2.3 presents a brief parallel communication model used in the CUDA computation. In the CUDA program, there are two different but

---

4. [http://www.theregister.co.uk/2012/05/18/inside_nvidia_kepler2_gk110_gpu_tesla/page2.html](http://www.theregister.co.uk/2012/05/18/inside_nvidia_kepler2_gk110_gpu_tesla/page2.html)
6. [http://www.pgroup.com/lit/articles/insider/v2nia5.htm](http://www.pgroup.com/lit/articles/insider/v2nia5.htm)
communicated computing environments, host CPU and device GPU. Host CPU reads from the hard drive, writes files to the hard drive, accesses computer memory, calls external libraries, and call programs, which should run on the device GPUs. After the host CPU transfers data from host memory to device memory, the device GPUs can execute parallel programs on each core. And the device chip copies computed result back to host memory after they finished the computation.

The CUDA programs are similar to C programs. Unlike C programs, however the CUDA programs need a CUDA nvcc compiler to compile. The code, which will be run on the host CPU, will be compiled like normal C
languages. The remaining parts of code, which will be executed on device GPUs, will be used to generate the parallel thread execution assembly. When the host CPU gives instructions to run parallel codes on GPUs, a parallel thread execution assembly will be read by a device driver, and converted to machine code. This strategy makes the compiled parallel thread execution assembly compatible for different GPU chips.

2.5 Periodic boundary conditions

Molecular dynamics simulation are usually performed on a small number of atoms or molecules, \( N \) ranging from 100 to 10000. Even though a parallel spatial decomposition algorithm can achieve \( O(N) \) time to calculate net forces on atoms or molecules in the simulated system, the simulation of a large size system will undoubtedly cost more computation resources and time than a smaller size system. In addition, the nature of molecular dynamics requires the computation to run millions of integration time steps before reaching equilibrium, and therefore simulating a small size system is less expensive and more efficient than simulating a large size system.

In order to simulate bulk systems, it is essential to apply the periodic boundary condition to mimic any atom or molecule in the simulated systems having interaction with infinite number of atoms or molecules. In fact, in a three dimensional simulation box, a non negligible number of atoms or molecules appears on cube surfaces. For example, 1000 molecules in a
Figure 2.4: Periodic boundary conditions on two dimensions. The primary cell is surrounded by image cells. The middle cell (0,0) labeled by the thick line means the real simulation domain, and atoms inside it are labeled by solid balls. The eight images ((-1,1) ... (1,-1)) labeled by dashed lines are eight neighboring images of the center simulation cell, and the ghost atoms in these image cells are labeled by circles.
simulation box with dimension $10 \times 10 \times 10$, 48% of molecules lie on cube faces\cite{38, 52}. For such simulated system, the interaction forces between molecules are not sufficient to represent interaction forces in a bulk system. Periodic boundary conditions are used to overcome this obstacle.

To use the periodic boundary conditions in the molecular dynamics simulations, we imagine that the volume of the simulated system contains two parts. One part is the original physical domain, which is labeled by $(0, 0)$ in figure 2.4, other cells of volumes are images of the original physical domain, such as $(-1, 1), \cdots, (1, -1)$. The image volumes not only contain the same number of atoms or molecules as the primitive domain, but also these imaged atoms have the same type and relative positions as atoms or molecules in the primitive cells. For example, the $i^{th}$ imaged particle in the $(i, j)$ images is

$$R_i^{(i,j)} = R_i + (i, j) \times L,$$   \hfil (2.9)

The $R_i$ in the equation 2.9 is the position of $i^{th}$ particle in the primitive cell.

After applying the periodic boundary condition in the molecular dynamics simulation, an atom or molecule now interact with all other atoms or molecules in the infinite dimension systems. An atom or molecule not only interacts with other atoms or molecules in the primitive cell, but also interacts with all particles in other image cells. The net force on the $k^{th}$ particle
in the primitive cell is

\[ F_k = \sum_{i=-\infty, i \neq 0}^{\infty} \sum_{j=-\infty, j \neq 0}^{\infty} \sum_{t=1}^{N} F(P_k, P_t + (i, j)L), \] (2.10)

Where \( F(P_k, P_t + (i, j)L) \) is the interaction force between \( k^{th} \) particle in the primitive cell and the \( t^{th} \) particle in the image cell labeled by \((i, j)\).

In summary, the use of the periodic boundary condition in the molecular dynamics simulation removes surface effects, and also let us investigate properties of bulk systems through simulating a finite size system. The use of the periodic boundary condition may give rise to correlations not present in the real bulk systems. However, for equilibrium properties, such as the conformation structure, the effects of the artificial periodic boundary conditions are very small, and even less than the systemic error[53, 54, 55, 56].

2.6 Dimensionless units

The international system of units (SI) is not suitable in the molecular dynamics simulations. The scale of atom or molecules in the molecular dynamics simulations is about 1\( nm \). The stored quantities and calculations done with floating point operations have more truncation errors for such small numbers, and there are more chances to underflow and overflow[57]. Therefore, it is beneficial to apply dimensionless units in the molecular dynamics simulations so that physical quantities in these units are of order one.
The LJ units is the dimensionless units commonly used in molecular dynamics simulations. The fundamental units in the LJ units are energy, distance, and mass, which are represented in the units of $\epsilon, \sigma,$ and $m$. Under LJ units, the bead mass $m$ is 1, and the parameters in the Lennard-Jones potential $\sigma$ and $\epsilon$ are both 1.0[54].

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Dimensionless Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>$r^* = \frac{r}{\sigma}$</td>
</tr>
<tr>
<td>Energy</td>
<td>$E^* = \frac{E}{\epsilon}$</td>
</tr>
<tr>
<td>Mass</td>
<td>$m^* = \frac{mass}{m}$</td>
</tr>
<tr>
<td>Time</td>
<td>$t^* = \frac{t}{\sqrt{\frac{m}{\sigma^2 \epsilon}}}$</td>
</tr>
<tr>
<td>Velocity</td>
<td>$v^* = \frac{v}{\sqrt{\frac{m}{\epsilon}}}$</td>
</tr>
<tr>
<td>Force</td>
<td>$F^* = \frac{F}{\sigma \epsilon}$</td>
</tr>
<tr>
<td>Density</td>
<td>$\rho^* = \frac{N \sigma^3}{V}$</td>
</tr>
<tr>
<td>Temperature</td>
<td>$T^* = \frac{k_B T}{\epsilon}$</td>
</tr>
<tr>
<td>Pressure</td>
<td>$P^* = \frac{P \sigma^3}{\epsilon}$</td>
</tr>
</tbody>
</table>

Table 2.1: Some physical quantities in molecular dynamics simulations expressed through the LJ units.

The dimensionless units are indicated by an asterisk (*), the Lennard-Jones potential in LJ units can be written as

$$U^*(r^*) = 4 \left[ \left( \frac{1}{r^*} \right)^{12} - \left( \frac{1}{r^*} \right)^6 \right]$$

(2.11)

The dimensionless units $r^*$ and $U^*$ in the equation 2.11 are defined in the table 2.1.
2.7 Integration algorithms

To solve the ordinary differential equations, the finite difference method is a commonly used method. In the molecular dynamics programs, we attempt to find velocities, and positions at a later time $t + \delta t$, if we have already known the velocities and positions of atoms or molecules in the system before time $t$. There are two integration algorithms widely used in molecular dynamics softwares\[58, 59, 60, 39\].

The Verlet algorithm is a simple but moderate precision integration method. The method is directly derived from Taylor expansion.

\[
r(t + \delta t) = r(t) + v(t)\delta t + \frac{1}{2}a(t)\delta t^2 \quad (2.12)
\]

\[
r(t - \delta t) = r(t) - v(t)\delta t + \frac{1}{2}a(t)\delta t^2 \quad (2.13)
\]

Summing these two equations in 2.12, we have

\[
r(t + \delta t) = 2r(t) - r(t - \delta t) + a(t)\delta t^2 \quad (2.14)
\]

\[
v(t) = \frac{r(t + \delta t) - r(t - \delta t)}{2\delta t} \quad (2.15)
\]

The Verlet algorithm uses positions and accelerations at current time $t$ to compute positions at time $t + \delta t$. The advantages of the Verlet algorithm are that it is simple, and it requires less storage than other algorithms. However, the computation in the equation 2.14 has errors $O(\delta t^2)$ in the computation of $r(t + \delta t)$.
The other commonly used integration algorithm is Verlet velocity algorithm, which is a variation from the Verlet algorithm above.

\[ r(t + \delta t) = r(t) + v(t)\delta t + \frac{1}{2}a(t)\delta t^2 \quad (2.16) \]
\[ v(t + \delta t) = v(t) + \frac{1}{2}[a(t) + a(t + \delta t)]\delta t \quad (2.17) \]

The Verlet velocity algorithm in the equation 2.16 first computes the velocities at time \( t + \frac{\delta t}{2} \) using

\[ v\left(t + \frac{\delta t}{2}\right) = v(t) + \frac{1}{2}a(t)\delta t \quad (2.18) \]

And then uses equation 2.16 to compute the positions at time \( t + \delta t \), and therefore we can compute the accelerations at time \( t + \delta t \) through new positions at time \( t + \delta t \). Velocities at time \( t + \delta \) are computed by

\[ v(t + \delta t) = v(t + \frac{\delta t}{2}) + \frac{1}{2}a(t + \delta t)\delta t \quad (2.19) \]

The advantages of the Verlet velocity algorithm are simplicity, and numerical stability, which make it widely used in molecular dynamics computations.
Chapter 3

Polymer Brush

We have simulated the relative shear motion of both neutral and polyelectrolyte end-grafted polymer brushes using molecular dynamics. The flexible neutral polymer brush is treated as a bead-spring model, and the polyelectrolyte brush is treated the same way except that each bead is charged and there are counterions present to neutralize the charge. We investigated the friction coefficient, monomer density, and brush penetration for the two kinds of brushes with both the same grafting density and the same normal force under good solvent conditions. For the case of equal grafting density, we find that polyelectrolyte brushes have a smaller friction coefficient and monomer penetration than neutral polymer brushes, although the polyelectrolyte brushes supported a much higher normal load than the neutral
brushes for the same degree of compression. Charged and neutral brushes with their grafting densities chosen so that they support the same load exhibited approximately the same degree of interpenetration, but the polyelectrolyte brush exhibited a significantly lower friction coefficient. We present evidence that the reason for this is that the extra normal force contribution provided by the counterion osmotic pressure that exists for polyelectrolyte brushes permits polyelectrolyte brushes to support the same load as an identical neutral polymer brush of higher grafting density. Because of the resulting lower monomer density for the charged brushes, fewer monomer collisions take place per unit time, resulting in a lower friction coefficient.

3.1 Overview

A polymer brush consists of polymer chains densely grafted to a solid surface, which is immersed in a solvent. The chains stretch from surfaces and repel each other. The balance between elasticity and repulsion of the chains generates completely different conformations and properties than for isolated chains. Polymer brushes have many technological applications, such as colloidal stabilization and lubrication. Electrostatic interaction involved in polyelectrolyte brushes results in a number of additional physical properties. Mutual repulsion between polymer segments and electrostatic forces between charged monomers and counterions strongly influence the conformation of the polyelectrolyte brush [3, 4]. A great deal of theoretical and
experimental research has been conducted to investigate the frictional behavior when lateral sliding between two apposing polymer brush surfaces occurs [5, 6, 7, 8, 9, 10, 11]. The normal force and shear force have been well studied experimentally by the surface force apparatus (SFA) both on neutral polymer brush surfaces and polyelectrolyte brush surfaces. These experiments on neutral polymer brushes showed that the polymer brush surfaces have ultra low friction coefficients when the surfaces are brought into contact [12]. Molecular dynamics simulations for a bead-spring model provide excellent agreement with experimental results [5]. The experiment conducted by Raviv, et al., on two opposing polyelectrolyte brush walls using the SFA also found very low friction coefficient when one polyelectrolyte brush slides relatively to the other [13]. Liberelle used the SFA to study the salt (NaCl) concentration dependency of the friction for weakly charged polymer brush friction changes due to different [14]. Because of the small distance involved and limitation of the experimental apparatus, it is difficult to observe what is happening inside the polymer brushes when they slide. Molecular dynamics simulations give us insights into what may be occurring inside the brushes. Earlier molecular simulations examined shear and compression of opposing neutral brush surfaces providing good agreement with experiment [5]. Using the same bead-spring model, we compare the behavior of neutral polymer brushes and polyelectrolyte brushes through molecular dynamics simulation.
3.2 Introduction and Model

To simulate the motion inside of polymer brushes which are sliding relative to each other, we applied the standard coarse bead-spring polymer chain model, which has been used previously in molecular dynamics simulations of polymer brushes [5, 9, 5, 61]. We investigate two opposing walls with grafted polymer brushes at wall separation distance $D$. $N_p = 16$ polymers chains with length $N = 32$ monomers are grafted on each wall surface in a square lattice with lattice constant $a = 3.5\sigma$ for the charged brushes, or $2.4\sigma$ for the neutral brush, where $\sigma$ is the length parameter in the Lennard-Jones potential. The corresponding grafting densities are $\rho_g = 0.082/\sigma^2$ or $\rho_g = 0.17/\sigma^2$. This set of parameters corresponds to the strongly stretched brush regime, because the grafting density is many times larger than $1/(\pi R_g^2)$, where $R_g$ is the radius of gyration of an isolated polymer. For polyelectrolyte brushes, the monomers on the chains, except for the end-grafted beads, each carry a single negative charge, and $N \times N_p$ explicit monovalent positive counterions are added to neutralize the system [9, 62]. Periodic boundary conditions are applied in $x$ and $y$ directions. Figure 3.1 shows the configuration of the two opposing polymer brushes.

All particles interact via the shifted Lennard-Jones (LJ) potential,

$$V_{LJ}(r_{ij}) = \begin{cases} 4\epsilon_{ij}(\sigma_{ij}/r_{ij})^{12} - (\sigma_{ij}/r_{ij})^6 + \frac{1}{4} & r_{ij} < r_c \\ 0 & r_{ij} > r_c \end{cases} \quad (3.1)$$
Figure 3.1: The schematics of two opposing polymer brushes for (a) a neutral brush and (b) a charged brush. Each polymer chain is grafted to its surface by its end monomer. Each monomer on charged chains carries one negative charge except for the end-grafted monomer. In addition, explicit monovalent counterions are added to the polyelectrolyte brushes to neutralize the system.
where $r_{ij}$ is the distance between two particles, which could be two monomers, two counterions or a monomer and a counterion for polyelectrolyte brushes. The monomers and counterions have the same bead size and mass, and interact through the same LJ potential parameters, i.e., $\sigma = \sigma_{ij}$, and $\epsilon = \epsilon_{ij}$ for all particle pairs. Instead of including explicit solvent, we mimic a good solvent by setting the cutoff distance to $r_c = 2^{1/6}\sigma$ in order to keep only repulsive interaction. The bonds of polymer chains are modeled using a finitely extensive nonlinear elastic (FENE) potential,

\[
V_{FENE}(r) = \begin{cases} 
-0.5kR_0^2 \ln(1 - r^2/R_0^2) & r < R_0 \\
\infty & r > R_0
\end{cases}
\]

(3.2)

where the spring constant $k = 30\epsilon/\sigma^2$ and the maximum monomer separation allowed by the potential $R_0 = 1.5\sigma$. The choice of such LJ potential and FENE bond potential parameters prevents bond crossings and yields realistic dynamics of polymer chains [63, 64]. The average bond length, found by minimizing $V_{FENE}(r) + V_{LJ}(r)$, is $b = 0.98\sigma$.

For polyelectrolyte brushes, the Coulomb interaction between two charged particles is included. Charged particles interact with each other through a solvent medium, which is presumed to be water in the simulation. Then the Coulomb potential between two charged particles $i$ and $j$ in our simulation system is given by

\[
V_{coul} = k_BTb \frac{q_i q_j}{r_{ij}},
\]

(3.3)
where $q_i$ and $q_j$ are charges on two particles in units of an electron charge, and $l_B$ is the Bjerrum length, $e^2/4\pi\varepsilon\varepsilon_0 k_B T$. $\varepsilon_0$ and $\varepsilon$ are the vacuum permittivity and the dielectric constant of the solvent respectively. The Bjerrum length of bulk water is 7.1Å at room temperature. We choose $l_B = 0.92\sigma$ in our simulation, and therefore $\sigma$ is 7.75Å and the Manning ratio $\varepsilon = l_B/b = 0.94$, where $b$ is the bond length of a polymer chain. The particle-particle particle-mesh (PPPM) method implemented in LAMMPS was used to calculate electrostatic potential between charged particles [65]. All particles except anchored segments interact repulsively with the walls at $z = 0$ and $z = L_z$ with the Lennard-Jones (LJ) potential,

$$V_{\text{wall}}(z) = \begin{cases} 
4\varepsilon(\sigma/z)^{12} - (\sigma/z)^6 + \frac{1}{4} & z < z_c \\
0 & z > z_c 
\end{cases}$$

(3.4)

where $z$ is the distance between particles and the wall to which the polymer under consideration is attached. Cutoff distance $z_c$ is set to $2^{1/6}\sigma$ to only keep the repulsive interaction. Therefore the total potential energy of the polyelectrolyte brush system is given by

$$U_{\text{total}} = V_{\text{LJ}} + V_{\text{FENE}} + V_{\text{wall}} + V_{\text{coul}}$$

(3.5)

The simulations are carried out in the constant number of particles, constant volume and constant temperature (NVT) ensemble. Constant temperature is achieved by the Langevin thermostat [5]. In this case, the equation
of the $i$th particle is

$$m_i \frac{dv_i(t)}{dt} = -\nabla_i V - \gamma m_i v_i(t) + \varepsilon_i(t) \quad (3.6)$$

where $-\nabla_i V$ gives net force on particle $i$ due to other particles, and $\gamma$ is the damping constant used to control the relaxation rate in order to maintain the desired temperature. $\varepsilon_i(t)$ is a random force that is uncorrelated in time and between particles, which means that it is given by

$$<\varepsilon_i(t)\varepsilon_j(t')> = 6m_i\gamma k_B T \delta_{ij} \delta(t-t') \quad (3.7)$$

The damping constant $\gamma$ and random forces $\varepsilon(t)$ are believed to mimic collisions of monomers with solvent [7]. For initial equilibrium runs, all three velocity components are coupled to the thermal bath. In the simulations, $\gamma$ and $T$ are assigned the values to $0.5 \tau^{-1}$ and $1.2\varepsilon/k_B$, where $\tau$ is defined as $\sigma(m/\epsilon)^{1/2}$. In the shear simulations only the $v_y$ (vorticity) component is coupled to the reservoir in order to not to bias the shear flow [5]. The Verlet algorithm is used to integrate equation 4.1 using timestep $\delta t = 0.005 \tau$.

The simulation was performed by the following procedure. The initial configuration of each polymer brush was prepared by building individual polymer chains randomly. Counterions were added in the neighborhood of the monomers in the simulation box for the polyelectrolyte brush. The initial system was equilibrated for $1 \times 10^6$ time steps, almost $5000\tau$ in LJ units. Then the final configuration was followed for another $1.5 \times 10^6$ time steps.
while sliding the top polymer brush relative to the bottom polymer brush. To investigate the shear behavior of the polymer brushes, we average over the normal forces, shear forces, monomer densities and brush interpenetrations computed in the simulations.

3.3 Simulation Results and discussions

In this section, we present results for lateral sliding of one planar polymer brush relative to the other. We shear the system by moving the upper wall at constant velocity \((v, 0, 0)\) while keeping the distance between the two surfaces constant. We calculate the normal force \(F_n\) and shear force \(F_s\) directly by adding together the vertical \(z\) components and the horizontal \(x\) components, respectively, of all forces acting between the substrates and the monomers. The corresponding stresses so obtained are labeled \(\sigma_n\) and \(\sigma_s\) respectively. The friction coefficient is defined as \(\mu = \sigma_s/\sigma_n\).

3.3.1 Neutral brush and polyelectrolyte brush having the same grafting density

In this section, we make a comparison of the behavior under shear of charged and uncharged brushes with the same grafting density. For both neutral and charged brushes, the polymer chain length \(N\) is 32 monomers, the number of polymer chains on each wall \(N_p\) is 16, and the spacing between two neighboring chains \(a = 3.5\sigma\). The corresponding grafting density is
Table 3.1: The table gives the normal force pressures on wall surfaces with separation distance $D$. Both neutral and polyelectrolyte brushes have the same polymer grafting density $\rho_g = 0.082/\sigma^2$.

<table>
<thead>
<tr>
<th>D($\sigma$)</th>
<th>Neutral brushes $\sigma_n(\epsilon/\sigma^3)$</th>
<th>Polyelectrolyte brushes $\sigma_n(\epsilon/\sigma^3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.0</td>
<td>0.41</td>
<td>6.29</td>
</tr>
<tr>
<td>16.0</td>
<td>0.25</td>
<td>3.70</td>
</tr>
<tr>
<td>18.0</td>
<td>0.17</td>
<td>2.45</td>
</tr>
</tbody>
</table>

In figure 3.2, we compare friction coefficients between neutral brushes and polyelectrolyte brushes with parameters specified in the figure caption. From our molecular dynamics simulation results, the friction coefficient between neutral polymer brushes is much larger than the value between two charged brushes under same grafting density and surface separation. The normal force was almost constant for both neutral and charged systems as a function of shear velocity. The increase in $\mu$ with velocity is due to the increase in the shear force. When the wall separation distance $D$ was decreased, the normal force and shear force on the wall increased for both neutral and charged brushes. This result comes from stronger interactions between beads in the system because decreasing the surface separation distance $D$ results in an increase in the global monomer density $\rho_m(r)$ of the polymer brushes, which can be seen in figure 3.3, and the counterion density $\rho_c(r)$. Also, the values of normal forces on the walls in the polyelectrolyte brush systems are about an order of magnitude greater than the values in the neutral brushes, even though both of them have the same grafting density and chain length.
Figure 3.2: Variation of friction coefficient as a function of velocity for (a) neutral brushes and (b) charged brushes. The polymer grafting density $\rho_g$ is $0.082/\sigma^2$ for all runs. Points $\square$, $\triangle$ and $\circ$ represent runs for charged brushes with $D = 14.0\sigma$, $D = 16.0\sigma$ and $D = 18.0\sigma$, respectively. Points $\blacksquare$, $\blacktriangle$ and $\bullet$ represent runs for neutral brushes with $D = 14.0\sigma$, $D = 16.0\sigma$ and $D = 18.0\sigma$, respectively. The normal force pressures on the walls for these runs are provided in the table 3.1.
Figure 3.3: The monomer density profile of one brush vs. distance from the wall surface for (a) neutral brushes and (b) charged brushes at $v = 0$. Points □, △ and ○ represent runs for charged brushes with $D = 14.0\sigma$, $D = 16.0\sigma$ and $D = 18.0\sigma$, respectively; points ■, ▲ and ● represent runs for neutral brushes with $D = 14.0\sigma$, $D = 16.0\sigma$ and $D = 18.0\sigma$, respectively. For all simulation runs, $N_p = 16$ polymer chains of length $N = 32$ monomers anchored to the wall surface in a square lattice at the grafting density $\rho_g\sigma^2 = 0.082$. 
Figure 3.4: The monomer density profile of one brush vs distance from the grafting surface wall under different sliding velocity at a separation distance $D = 14\sigma$ for (a) neutral brushes and (b) charged brushes. Points $\Box$, $\Delta$ and $\circ$ represent runs for charged brushes with sliding velocity $v = 0.00\sigma/\tau, 0.04\sigma/\tau$, and $0.10\sigma/\tau$, respectively; points $\blacksquare$, $\blacktriangle$ and $\bullet$ represent runs for neutral brushes with sliding velocity $v = 0.00\sigma/\tau, 0.04\sigma/\tau$, and $0.10\sigma/\tau$, respectively. For all simulation runs, $N_p = 16$ polymer chains of length $N = 32$ monomers are anchored to the wall surface in a square lattice at the grafting density $\rho_g\sigma^2 = 0.082$. 

50
The difference results from the existence of counterions in the polyelectrolyte brushes, which results in osmotic pressure which adds to the normal force. However, the friction coefficient actually decreases slightly as the wall separation distance decreases in the simulations. This is opposite to findings in experiments. This is because the normal force increased much more rapidly than the shear force when the separation distance $D$ decreased. This behavior was seen in previous simulations of neutral brushes and it was pointed out that the difference in the geometry of the crossed cylinders in the SFA experiments and the parallel plates of the simulations does not allow direct comparison [5].

![Figure 3.5: Variation of interpenetration as a function of velocity for neutral and charged brushes. The labeling system here is the same as in Fig 3.2. For both neutral and charged polymer brushes, $N_p = 16$ polymer chains of length $N = 32$ monomers are anchored to the wall surface in a square lattice at the grafting density $\rho_g \sigma^2 = 0.082$.](image-url)
Figure 3.4 shows how the monomer density profiles of bottom brushes evolve according to different sliding velocities of top brushes for both neutral and polyelectrolyte brushes. If bottom and top brushes were to completely repel each other, the bottom and top brushes would occupy space $z < D/2$ and $z > D/2$ separately. Therefore, the monomer densities for $z/D > 0.5$ in the figure 3.4 represent the part of the monomer density profile which penetrates into the top brush. The region defined by $z/D > 0.5$ of the monomer density profile quantifies the amount of penetration between top and bottom brushes. For both neutral and polyelectrolyte brushes, the monomer density of the lower brush for $z > D/2$ decreased as the relative sliding velocity between top and bottom brushes increased. The reason is that the increase of relative shear motion between top and bottom brushes increases the interaction frequency between monomers in the brushes. In figures 3.4(a) and 3.4(b), comparing monomer density profiles of both neutral and polyelectrolyte brushes with the top brush sliding at $v = 0.00\sigma/\tau$ and $0.10\sigma/\tau$, we can see that the monomer density of the lower charged brush for $z > D/2$ decreased from its static value more than for neutral brushes. Also, the monomer density profile for $z > D/2$ with sliding velocity $v = 0.10\sigma/\tau$ is smaller for the charged brush. Therefore, when the top brush slides relative to bottom brush, it take less work to pull these interacting monomers apart for charged brushes.

The amount of interpenetration between top and bottom brushes when
the two wall surfaces are separated by a distance $D$ is defined by

$$I(D) = \frac{\int_{D/2}^{D} \rho(z) dz}{\int_{0}^{D} \rho(z) dz},$$

(3.8)

where the quantity $\rho(z)$ is the monomer density profile of the lower brush found from figure 3.4. This quantity defines the thickness of the region containing interacting monomers between top and bottom brushes. The larger the interpenetration between top and bottom brushes, the more interaction between them.

The first result from figure 3.5 is that charged brushes have smaller interpenetration of top and bottom brushes than neutral brushes when they have the same grafting density and move at the same velocity. Therefore, it takes less work to pull apart these entangled monomers for charged brushes than neutral brushes. The second result from figure 3.5 is that the interpenetration between top and bottom brushes decreases as the relative shear velocity of the top brush increases. The decrease of interpenetration comes from increasing interaction between top and bottom brushes as monomers interact more frequently for larger velocities.

Matsen[66] has performed mean field theory calculations on two polyelectrolyte brushes in contact. Figure 10 of Ref. [66] gives the degree of interpenetration of the brushes as a function of plate separation. For a plate separation corresponding to $D = 18b$ in our notation, Matsen finds a degree of interpenetration of 0.05 for the highest grafting density that he considers,
which is smaller than our grafting density of 0.85 for this value of \( D \). Since Matsen’s model is for a \( \Theta \) solvent (which has no monomer-monomer hard core interaction), however, and our model includes hard core interaction, we do not expect precise agreement, but it is encouraging that the results of the two approaches are not so significantly different.

To determine directly the amount of contact at the interfaces between top and bottom polymer brushes, we present radial distribution function results in the figure 3.6. The radial distribution function \( g_2(\mathbf{r}_i, \mathbf{r}_j) \) or simply \( g(r) \) gives the probability of finding a pair of monomers a distance \( r \) apart, relative to the probability expected for a completely random distribution at the same density [67, 38]. Here \( \mathbf{r}_i \) and \( \mathbf{r}_j \) are the position vectors of monomers in the top and bottom polymer brush respectively. In the canonical ensemble

\[
g(\mathbf{r}_1, \mathbf{r}_2) = \frac{N(N-1)}{\rho^2 Z_{NVT}} \int d\mathbf{r}_3 \ldots d\mathbf{r}_N \exp \left(-\beta V(\mathbf{r}_1, \ldots, \mathbf{r}_N)\right),
\]

where \( V(\mathbf{r}_1, \ldots, \mathbf{r}_N) \) is the potential energy of the monomers. The choice \( i = 1, j = 2 \) is arbitrary in the simulation system of identical monomers. An equivalent definition takes an ensemble average over pairs

\[
g(r) = \rho^{-2} \left( \sum_i \sum_{j \neq i} \delta(\mathbf{r}_i) \delta(\mathbf{r}_j - \mathbf{r}) \right) = \frac{V}{N^2} \left( \sum_i \sum_{j \neq i} \delta(\mathbf{r} - \mathbf{r}_{ij}) \right)
\]

It can be thought of as the number of monomer pairs between top and bot-
tom polymer brushes with a given separation $r$ compared to the number of monomer pairs with the same density and separation $r$ in an ideal gas.

From figure 3.6, we could see that the radial distribution function of neutral brush systems is larger than that for the charged brushes close to $r = 1\sigma$, which is about the hard core radius of Lennard-Jones potential in our simulation. The radial distribution close to $r = 1\sigma$ represents the amount of contact between top and bottom polymer brushes. This result is consistent with figure 3.5, both of them show that monomers of neutral brushes have more interactions than monomers of polyelectrolyte brushes when they are pressed together.

In summary, the charged polymer brushes have smaller friction coefficients than identical neutral brushes. From the above analysis, the monomer density profile (Fig. 3.4), interpenetration between top and bottom brushes (Fig. 3.5), and radial distribution function of two opposing brushes (Fig. 3.6) show that charged brushes have normal forces an order of magnitude larger and a smaller amount of monomer contacts per unit time than neutral brushes, which yields a smaller friction coefficient for charged brushes.

### 3.3.2 Neutral brushes and polyelectrolyte brushes having same normal force pressure on the walls

In this section, we present a comparison of charged and uncharged brushes with almost the same normal force pressures on their walls, in order to show
Figure 3.6: The radial distribution function between top brush and bottom brush for (a) neutral brushes and (b) charged brushes with sliding top brushes at constant velocity $v = 0.04\sigma/\tau$. The labeling method here is the same as in Fig 3.2. For both neutral and charged polymer brushes, $N_p = 16$ polymer chains of length $N = 32$ monomers are anchored to the wall surface in a square lattice at the grafting density $\rho\sigma^2 = 0.082$. 

56
that a polyelectrolyte brush is a better lubricant than a neutral one under the same external normal force pressure.

Because of the existence of counterions in the polyelectrolyte brushes, the normal force pressures exerted on the wall surfaces of the polyelectrolyte brushes is much larger than that for identical neutral brushes under the same surface separation distance \( D \). For example, this difference in behavior of the two kinds of polymer brushes can be seen in table 3.1. In order to demonstrate that polyelectrolyte brushes exhibit smaller friction than neutral brushes under same external force pressures on wall surfaces, we select different polymer grafting densities for the neutral and charged brushes and keep other parameters, such as the chain length, \( l \) the same in order to exert the same normal force pressures on the grafted wall surfaces under the same separation distance \( D \). For example, in table 3.2, neutral brushes with grafting density \( \rho_g = 0.174/\sigma^2 \) and charged brushes with grafting density \( \rho_g = 0.082/\sigma^2 \) have similar values of normal force pressures on their wall surfaces under different surface separation distances if we keep the chain length of their polymers the same.

Figure 3.7 presents the kinetic friction coefficients of neutral brushes with grafting density \( \rho_g = 0.174/\sigma^2 \). Compared to the friction coefficients of charged brushes with grafting density \( \rho_g = 0.082/\sigma^2 \) in figure 3.2, the polyelectrolyte brushes still exhibit lower friction by a factor of about 2. In the previous section, we found that the charged brushes have a much lower friction coefficient, in part because the normal force was much larger for charged
Table 3.2: The table gives the normal force pressures on wall surfaces for neutral and charged brushes with different grafting density but the same polymer chain length. Polyelectrolyte brushes are anchored to the surface with square lattice constant 3.5σ, which is the same as grafting density \( \rho_g = 0.082/\sigma^2 \). Neutral brushes are anchored to the wall surface with square lattice constant 2.4σ, which is the same as grafting density \( \rho_g = 0.174/\sigma^2 \).

<table>
<thead>
<tr>
<th>D(σ)</th>
<th>Neutral brushes ( \sigma_n(\epsilon/\sigma^3) )</th>
<th>Polyelectrolyte brushes ( \sigma_n(\epsilon/\sigma^3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.0</td>
<td>5.76</td>
<td>6.29</td>
</tr>
<tr>
<td>16.0</td>
<td>3.38</td>
<td>3.70</td>
</tr>
<tr>
<td>18.0</td>
<td>2.03</td>
<td>2.45</td>
</tr>
</tbody>
</table>

Figure 3.7: Variation of friction coefficient as a function of velocity for neutral brushes with grafting density \( \rho_g = 0.174/\sigma^2 \). Points ■, ▲ and • represent runs for neutral brushes with \( D = 14.0\sigma \), \( D = 16.0\sigma \) and \( D = 18.0\sigma \), respectively. The normal force pressures on walls for these runs are provided in table 3.2.
Figure 3.8: Variation of the interpenetration as a function of velocity for neutral and charged brushes with almost the same normal force on wall surfaces. Points □ and ○ represent runs for charged brushes with $D = 14.0\sigma$ and $D = 18.0\sigma$ at grafting density $\rho_g = 0.082/\sigma^2$; points ■ and ● represent runs for neutral brushes with $D = 14.0\sigma$ and $D = 18.0\sigma$ at grafting density $\rho_g = 0.174/\sigma^2$. For both neutral and charged polymer brushes, $N_p = 16$ polymer chains of length $N = 32$ monomers are anchored to the wall surface in a square lattice.
Figure 3.9: The radial distribution function $g(r)$ between the top brush and bottom brush for neutral brushes with top brush sliding constant velocity $v = 0.04\sigma/\tau$. Points ■, ▲, and ○ represent runs for neutral brushes with $D = 14.0\sigma$, $D = 16.0\sigma$ and $D = 18.0\sigma$, respectively. For all runs, $N_p = 16$ polymer chains of polarization $N = 32$ anchored to the wall surface with the grafting density $\rho_g = 0.174/\sigma^2$.

brushes. In these data, where the systems have almost the same normal forces, the difference must come from the contact interactions. Figure 3.8 shows that neutral brushes and charged brushes, with grafting densities chosen so as to exhibit equal normal pressures, have almost the same monomer interpenetration even though the charged brush is a much better lubricant than neutral brush under the same external normal force pressures on walls. This is explained through the comparison between radial distribution function of these two kinds of polymer brushes. The radial distribution function $g(r)$ of neutral brushes with grafting density $\rho_g = 0.174/\sigma^2$ in figure 3.9 has
higher values close to \( r = 1\sigma \) than charged brushes with grafting density \( \rho_g = 0.082/\sigma^2 \) in figure 3.6(b). Here, \( r = 1\sigma \) is the hard core radius of Lennard-Jones potential in our simulation. Therefore, the larger values of \( g(r) \) close to \( r = 1\sigma \) imply more contacts per unit time between top and bottom brushes. The larger number of contacts per unit time between top and bottom brushes for neutral brushes with grafting density \( \rho_g = 0.174/\sigma^2 \) than charged brushes with grafting density \( \rho_g = 0.082/\sigma^2 \) results from larger monomer density in the neutral brushes, which results in higher friction than for charged brushes with same normal force pressure on walls.

In summary, for neutral and charged brushes with different grafting densities but the same normal external force pressures on their wall surfaces, the charged brushes exhibit lower friction than corresponding neutral brushes. The larger friction of the neutral brush results from the larger number of contacts between top and bottom brushes per unit time.

### 3.4 Conclusion

From the comparison between charged brushes and neutral brushes with the same grafting density, we demonstrated the existence of a lower friction coefficient (order of magnitude) and lower monomer density penetration in polyelectrolyte than neutral brushes. For these systems, the normal force is much larger in the charged system and the number of contacts is smaller in the charged system. Our results suggest that a polyelectrolyte brush ex-
hibits lower friction than an otherwise identical but neutral polymer brush, because for the polyelectrolyte brushes the counterions make a significant contribution to the normal force. Secondly, from a comparison between neutral and charged brushes under the similar external normal force pressures on the grafting wall surfaces, we also find that the charged brush has a lower friction coefficient than the neutral brush by about a factor of two. The main factor likely to explain the difference in the friction coefficients in this case is the number of contacts per unit time between the monomers of the two brushes. Surprisingly, the interpenetrations between top and bottom brushes of both neutral brushes and charged brushes are almost same when we exert the same normal force pressure on the grafting wall surfaces of these two kinds of brushes.
Chapter 4

Polyelectrolyte Gel

As discussed in the introduction, polymer gels are three dimensional networks, composed of polymers, that are connected to each other through functional groups. The large number of molecules inside networks of swollen gels gives the gels solidity and coherence, but gels can undergo a large deformation. Polymer gels are neither solid nor liquid, but have some features of solid and some features of liquid. [17, 18, 19, 20, 21]. Because of these specific properties of polymer gels, they have many unique properties, such as phase transition, chemomechanical behavior, shape memory effect, etc [22, 23, 24].

We have simulated the dynamic structures, the change of structures while apposing gels are pressed together, and the relative shear motion of two polyelectrolyte gels relative to each other using molecular dynamics method. The polymer gels are treated as a linked network of polymer chains. The polymer chains between linking nodes are similar to polymer chains in the
polyelectrolyte brush except that their end beads are linked to other chains, and thus they are no longer completely free to move. We investigated the friction, monomer density, and monomer penetration.

4.1 Overview

Over the past decades, polymer gels have gained a lot of attention from both science and industry[15, 16]. Industry utilizes polymer gels’ ability to swell by absorbing huge amounts of solvent molecules in numerous applications, such as drug delivery, agriculture, and environmental cleanup operations.

Polymer gels are three dimensional networks, composed of molecules, polymers, particles, counter-ions, etc., connected to each other through functional groups. Gels always contain many solvent molecules inside their networks, which gives polymer gels solidity and coherence, but gels can undergo large deformations. Polymer gels are neither solid nor liquid, but have some features of solid and some features of liquid [17, 18, 19, 20, 21]. Because of these specific properties of polymer gels, gels have unique behaviors, such as phase transition, chemomechanical behavior, shape memory effect, etc [22, 23, 24].

Some biological surfaces show ultra low friction. For example, cartilages of animal joints have a friction coefficients of about 0.001 – 0.03[25, 26, 27]. The reason why the cartilage friction of the joints is so low when the pressure
between the bone surfaces reaches as high as $3 - 18 \text{ MPa}$ is not clear now [28, 29, 30]. Animal cartilage consists of a three dimensional collagen networks filled with a synovial fluid. These facts inspired people to investigate the frictional behaviors of polymer gels. A number of experiments have been performed to measure the friction coefficients between polymer networks [21, 31, 32].

### 4.2 Models for Simulating Polymer networks

There have been many computer simulations of polymer networks in the past decades.

G. Grest and K. Kremer investigated the static and dynamic properties of random-built polymer networks [68, 69]. In their model, the system contains a number of free polymer chains and some tetra-functional ($f = 4$) groups in the begin of simulation. In order to build random polymer networks, they randomly attached tetra-functional cross-linkers to the chain ends of available free polymer chains. When a free end came within a short reaction distance of an unsaturated cross-linker node, the chain end was attached to this functional group, and the available linker space for that node was decreased by one. Their simulation results found that the number of unsaturated cross-linkers decays very slowly when the remaining unsaturated cross-linkers and free ends of polymer chains are far from the mean end-to-end distance of the original free polymer chains. It is possible that there exists unsaturated
cross-linker nodes in the end.

Different from Kremer’s randomly built polymer network model, S. Schneider and P. Linse developed a static polymer network model\[70, 71, 72, 73, 74, 75\]. Schneider and Mann used this model in their research on the swelling of polyelectrolyte networks.

Figure 4.1: The schematic illustration of one unit cell of a defect-free network of diamond topology polymer networks\[75\]. Each unit cell contains 8 linker nodes and 16 non-crossing polymer chains.

Figure 4.1 is a unit cell of polymer networks in this model. The linker nodes are configured with the topology of the diamond lattice, and free chains are linked between pairs of linker nodes. All chains between two cross-linked nodes have the same length. The counter-ions are explicitly added into the system to neutralize the system. The polymer chain length between two linker monomers was $N_c = 13$ in our simulations.
In our simulation, we used this diamond topology polymer network model. This geometry topology was conserved during the simulation. We considered two gels in contact with each other. In our simulation, both top polymer gel and bottom polymer gel consisted of $2 \times 2 \times 2$ diamond cubic cells in the figure 4.1. Our polymer gels were periodic in horizontal $X$ and $Y$ direction. The ends of polymer chains attached to the surfaces could freely move along the grafted surface in the $X-Y$ plane, but they could not move up or down in $Z$ direction.

Similarly to polymer brushes, the polymer chains in our gel problem were modeled as bead spring chains, with all beads interacting with each other with a Lennard-Jone (LJ) potential and beads with a chain interacting with a FENE (Finite Extension Nonlinear Elastic) bond potential, $U_{FENE}(r) = 1\frac{1}{2}k_F r_F^2 \ln[1 - (r/r_F)^2]$ for $r \leq r_F$ and 0 otherwise. The parameters were $k_F = 30\frac{\epsilon}{\sigma^2}$, and $r_F = 1.5\sigma^2$, where $\sigma$ is the monomer diameter in LJ units. The standard Lennard-Jones potential used here was $U_{LJ}(r) = 4\epsilon[(\sigma/r)^{12} - (\sigma/r)^6 + \frac{1}{4}]$ if $r \leq r_c$ and 0 otherwise. We modeled a good solvent case by choosing $r_c = 2^{1/6}\sigma$, only keeping the repulsive part of the interactions. The choice of such LJ potential and FENE bond potential parameters prevents bond crossings and yields realistic dynamics of polymer chains. The average bond length, found by minimizing $U_{FENE}(r) + U_{LJ}(r)$, is $b = 0.98$.

Because we simulated polyelectrolyte gels, the Coulomb interaction between two charged particles was also included in the simulation. Charged particles interact with each other through a solvent medium, which is presumed
to be water in our simulations. Then the Coulomb potential between two charged particles \( i \) and \( j \) in our simulation system is given 

\[ U_{\text{coul}} = k_B T l_B \frac{q_i q_j}{r_{ij}}, \]

where \( q_i \) and \( q_j \) are charges on two particles in units of an electron charge, and \( l_B \) is Bjerrum length, \( e^2/4\pi\epsilon\epsilon_0 k_B T \). \( \epsilon_0 \) and \( \epsilon \) are the vacuum permittivity and the dielectric constant of the solvent respectively. The Bjerrum length of bulk water is 7.1 Å at room temperature. We choose \( l_B = 0.92\sigma \) in our simulation, and therefore \( \sigma = 7.75\sigma \) and the manning ration \( \epsilon = l_B/b = 0.94 \), where \( b \) is the bond length of a polymer chain[63, 64].

The particle-particle particle-mesh (PPPM) method implemented in LAMMPS was used to calculate the electrostatic potential between charged particles[65]. All particles except those anchored to the walls interact repulsively with the walls at \( z = 0 \) and \( z = D \) with the LJ potential

\[ U_{\text{wall}} = 4\epsilon[(\sigma/z)^{12}-(\sigma/z)^6]+\frac{1}{4} \]

if \( z < z_C \), and 0 otherwise. Here \( z \) is the distance between particles and the wall to which polymer gel is attached. Cutoff distance \( z_C \) is set to \( 2^{1/6}\sigma \) to keep the interaction purely repulsive.

The total potential energy of polymer gel in our simulation is given by

\[ U_{\text{total}} = U_{\text{LJ}} + U_{\text{FENE}} + U_{\text{wall}} + U_{\text{coul}}. \]

The simulations are carried out for a constant number of particles, constant volume and constant temperature (NVT) ensemble. Constant temperature is achieved by the Langevin thermostat[5]. In this case, the equation motion of the \( i \)th particle is

\[ m_i \frac{dv_i(t)}{dt} = -\nabla_i V - \gamma m_i v_i(t) + \xi(t) \quad (4.1) \]
where $-\nabla_i V$ gives net force on particle $i$ by other particles, and $\gamma$ is the damping constant used to control relaxation rate in order to maintain the desired temperature. The quantity $\varepsilon_i(t)$ is a random force that is uncorrelated in time and between particles, which means that it is given by

$$<\varepsilon_i(t)\varepsilon_j(t')> = 6m_i \gamma k_B T \delta_{ij} \delta(t-t')$$

(4.2)

The damping constant $\gamma$ and random forces $\varepsilon(t)$ are believed to mimic collisions of monomers with solvent [7]. For initial equilibrium runs, all three velocity components are coupled to the thermal bath. In the simulations, $\gamma$ and $T$ are assigned the values to $0.5 \tau^{-1}$ and $1.2 \epsilon/k_B$, where $\tau$ is defined as $\sigma(m/\epsilon)^{1/2}$. In the shear simulations only the $v_y$ (vorticity) component is coupled to the reservoir in order to not bias the shear flow [5]. The Verlet algorithm is used to integrate equation 4.1 using a timestep $\delta t = 0.005 \tau$.

### 4.3 Simulation results

The simulation was performed by the following procedure. The initial configuration of each polymer gel was prepared by linking straight polymer chains in a diamond topology network. Counter-ions were added in the neighborhood of the monomers in the simulation box for the polyelectrolyte gels. The initial system was equilibrated under constant number of particles, constant pressure, and constant temperature (NPT), where the simulation
box could expand or shrink in the horizontal $X - Y$ plane. This process was not stopped until the system reached equilibrium and once equilibrium was reached the simulation box did not change size any longer. The next stage was to fix the simulation box $X - Y$ dimensions, and compress two polymer gels together but without allowing them to expand in the $X - Y$ plane. Then the final configuration was followed for another $1.5 \times 10^6$ time steps while sliding the top polymer gel relative to the bottom polymer gel.

To investigate the shear behavior of the polymer gels, we averaged over the normal forces, shear forces, monomer densities and brush inter penetrations computed in the simulations.

4.3.1 Normal forces between two apposing polyelectrolyte networks

In order to investigate the normal forces between two apposing polyelectrolyte gels, we moved the top polymer gel slowly towards the bottom polymer gel, but prohibited polymer chains from expanding in the horizontal $xy$ direction. We calculate the normal force $F_n$ directly by adding together the vertical components of all forces acting between the substrates and the monomers.

When two polymer gels are brought into contact, they will repel each other. The normal forces between polymer gels were experimentally measured by N.Kampf [31]. N.Kampf, Uri Raviv, and Jacob Klein used a mica
(a) Normal force $F_n$ vs separation distance $D(\sigma)$ of compressed polymer gels in the simulation runs. In all simulation runs with different separation distance $D$, each polymer gel has $2 \times 2 \times 2$ units of diamond lattice cubes with $L_x = L_y = 42\sigma$.

Surface balance to measure interactions between mica surfaces bearing chitosan. To measure the interaction forces between polymer networks, they added a polynionic cross-linking agent to cause a gelation of the absorbed layers. Their experimental measurements, that showed cross-linked layers had longer ranged and more repulsive forces. Kampf, et.al believed that it was probably due to the network formation of polymer gels resulting a higher modulus of the layers.

When polymer gel surfaces are compressed together, there is inter-penetration and compression of the gels. Similar to the discussions in the polymer brush section, we computed the interpretation between two polymer gels from

$$ I(D) = \frac{\int_D^0 \rho(z)dz}{\int_0^D \rho(z)dz} $$  \hspace{1cm} (4.3)
Figure 4.2: Interpenetration $I(D)$ vs Separation distance $D$ of two apposing polymer gels. In all simulation runs with different separation distance $D$, each polymer gel has $2 \times 2 \times 2$ units of diamond lattice cubes with $L_x = L_y = 42\sigma$.

Figure 4.2 displays how the interpenetration of two apposing polymer gels changed as we compressed them together. The two polymer gels first showed visible interpenetration at $D = 76.9364\sigma$, which is about two times the static height of a single polymer gel. As the two polymer gels are compressed closer, they have a higher interpenetration, and higher repulsive interaction force with each other. The interpenetration between two polymer gels results, when two polymer gels repel each other, and the larger normal force $F_n$ exerted on surfaces the more interpenetration there is between two polymer gels.
4.3.2 Local Pressure $P_{zz}$

Pressure is one of the fundamental thermodynamics variables. Local pressure is an important method to investigate the mechanical responses of materials to external strain[76]. Gerhard Hummer and Bjørn Hafskjold developed numerical methods to compute pressures of charged systems, where they treated Coulomb interactions with Ewald lattice summation or similar particle-mesh Ewald method[77, 78]. There are also several applications of numerical local pressure calculations in the analysis of polymers and thin films[79, 80, 81, 82]. Cao applied the local pressure tensor analysis in their research work on shear force between polyelectrolyte brushes[10].

J. Irving and J. Kirkwood’s pioneering works described how to numerically compute local stress tensors[83]. They derived the equations of hydrodynamics from classical statistical mechanics, and obtained expressions for the stress tensor in terms of molecular variables, including intermolecular forces in their calculations as well.

The definition of local stress tensor can be more clearly explained using figure 4.3. The calculation of local pressure tensor at point $(a, b, c)$ use an infinitesimally small cube containing the point $(a, b, c)$ [84]. The forces $F_{\beta}, (\beta = x, y, z)$ acting on surfaces $A_{\alpha}, (\alpha = x, y, z)$ give the local stress tensor $p_{\beta\alpha}$ as

$$p_{\beta\alpha} = \frac{F_{\beta}}{A_{\alpha}}$$

(4.4)

Assume our system has $N$ atoms or molecules, and there are only pairwise
Figure 4.3: A simple graph to illustrate the definition of local stress pressures. \( \vec{x} \) is orthonormal to surface \( A_x \). The similar schemes are for \( \vec{y}, A_y \) and \( \vec{z}, A_z \).

two-body interactions between them in our systems, such as Lennard-Jones interactions and Coulomb interactions. Irving and Kirkwood developed a method to calculate local stress tensor in their works\cite{83}. They gave the formula as the

\[
p_{\beta\alpha} = \frac{1}{2A_a\Delta\alpha} \left[ \sum_{i \in (\Delta V=2A_a\Delta\alpha)} m_i v_{i\beta} v_{i\alpha} \right] + \frac{1}{A_a} \left[ \sum_{r_{ij} \cap A_{\alpha}} F_{ij\beta} \right] \tag{4.5}
\]

In order to find the local pressure tensor \( p_{\beta\alpha} \) for a point \( (a, b, c) \), the equation 4.5 tells us how to compute it. First, construct a small cube containing the point \( (a, b, c) \) with dimensions \( A_a \times 2\Delta\alpha \). Second, consider only molecules inside the small cube and compute the kinetic contributions to the local tensor \( p_{\beta\alpha} \). Third, compute the component of \( p_{\beta\alpha} \) contributed by internal forces. This is the tricky part in the equation 4.5. The Irving and Kirkwood formula only considers two-body forces \( F_{ij} \) between particles, whose position vectors \( r_{ij} \) dissect the small cube surface \( A_{\alpha} \) passing through the specific
point \((a, b, c)\).

The need to determine whether a vector \(r_{ij}\) dissects the small cube surface \(A_\alpha\) makes the calculation of local pressure tensor a little complicated. Most of the time we are interested in the properties of \(p_{\beta \alpha}(\alpha)\), such as \(P_{zz}(z)\), and want to know how the local pressure tensor changes along a specific direction. Therefore, we actually only need to compute the average cross-sectional pressure tensor at \(Z = z\). Todd and Evans simplified computation method of Irving and Kirkwood, and introduced an average cross-sectional pressure calculation method \[85\].

\[
\bar{p}_{\beta \alpha}(\alpha) = \frac{1}{2A_\alpha \Delta \alpha} \left\{ \sum_{\alpha - \Delta \alpha \leq \alpha_i \leq \alpha + \Delta \alpha} m_i v_{i\beta} v_{i\alpha} \right\} + \frac{1}{A_\alpha} \left[ \sum_{i=1}^{N} F_{i\beta} \text{sgn}(\alpha_i - \alpha) \right] \tag{4.6}
\]

Similar to the Irving-Kirkwood method, in order to compute \(\bar{p}_{\beta \alpha}(\alpha)\), we need consider a small slice with dimensions \( [\alpha - \Delta \alpha, \alpha + \Delta \alpha] \times [0, L_\beta] \times [0, L_\gamma] \). \(F_{i\beta}\) specifies the \(\beta\) component of total force on molecule \(i\) by all other \(N - 1\) molecules. Because of Newton’s’ Third law, we have \(\sum_{i=1}^{N} F_i = 0\) in our system.

We applied Todd and Evans’ local tensor method in our data analysis, to determine how \(P_{zz}\) change for different separation distances \(D\) and \(z\). Figure 4.4 shows our calculation of the average cross-sectional stress pressure \(P_{zz}\) by two methods, the Irving-Kirkwood method and the direction Newton force method.

The computation method of Irving-Kirkwood is based on formula 4.6,
(a) The stress $P_{zz}$ at two polymer gels surfaces $D = 40\sigma$ apart.

(b) The stress $P_{zz}$ at two polymer gels surfaces $D = 60\sigma$ apart.

Figure 4.4: Average cross-sectional pressure $P_{zz}$ computed by both Irving-Kirkwood method and direct stress pressure method $\frac{F_z}{A_z}$, where $F_z$ is computed by Newton’s third law in normal force section.
the newton force direct computation method is \( \frac{F_n}{x} \), where \( F_n \) has already been found in the normal force section 4.3.1. From figure 4.4, we could see that calculation of \( P_{zz} \) by these two different methods gave consistent results. Even though there are oscillations of \( P_{zz}(z) \) in Irving-Kirkwood method, if we take an average of \( P_{zz}(z) \) across the \( z \) direction, we obtain a value very close \( P_{zz} \) found by the direct force computation method.

### 4.3.3 Density

In figure 4.5, we present the monomer density profiles for the top polymer gels, the bottom polymer gels, and counter-ions at several different surface separation distances. When polymer gels were very far apart, the equilibrium height for each gel was \( 40\sigma \) for our models.

When we moved two polymer gels close, they reached the first contact at \( D = 80\sigma \). From figure 4.2, we found that the interpenetration between top and bottom polymer gels was 0.0000 at \( D = 80\sigma \), and 0.0025% at \( D = 75\sigma \) in our simulation results. And the figure 4.5 also showed that monomers of top and bottom polymer gels at \( D = 75\sigma \) had a very tiny overlap region.

When two polymer gels moved closer together and the separation distance of two polymer gels reached \( D = 55\sigma \), more monomers of two gels appeared at the interface region between two gels. Therefore there was greater monomer density around the interface, as is shown in figure 4.5, where the monomer density around \( 0.5z/D \) increased as surface separation distance \( D \) decreased. However, unlike polymer brushes, polymer chains inside polymer gels are
Figure 4.5: The density profiles of top gel monomers, bottom gel monomers and counter-ions at $D = 75\sigma, 55\sigma, 35\sigma$. In all simulation runs with different separation distance $D$, each polymer gel has $2 \times 2 \times 2$ units of diamond lattice cubes with $L_x = L_y = 42\sigma$. 

78
linked together, they have much more constraint on their motion, making it
difficult to penetrate into the opposite gels. As is showed in figure 4.5, al-
though the monomer density close to the interface \((0.5z/D)\) increased when
we pushed two polymer gels together, they increased slowly. We see from fig-
ure 4.2, that the interpenetration between two polymer gels with separation
distance \(D = 55\sigma\) is only 0.3400%.

The top graph in figure 4.5 shows what monomer density profiles of poly-
mer gels look like when the walls are as close as \(D = 35\sigma\). Because the
static height of single polymer gel in our model was \(40\sigma\), it is reasonable to
believe that we compressed the two polymer gels to less than 50% of their
uncompressed equilibrium height when \(D = 35\sigma\). The remarkable result is
that the interpenetration was only about 1% in this case. This result showed
that even though these two polymer gels suffered a great compression, there
was little interpenetration of monomers between top and bottom polymer
gels. This behavior was strongly related to the network geometry of polymer
gels, which restricted free motions of monomers on polymer gels, and thus
resulted in a small interpenetration between monomers in polymer networks.

In figure 4.5, we could see the wave-like oscillations in the density profiles.
There were small oscillations in the case \(D = 75\sigma\), where two gels were far
apart and only very small interactions existed between two polymer gels.
The wave-like oscillations appeared more clearly in the case \(D = 35\sigma\), where
two polymer gels were compressed greatly and moved together.

In order to explain this strange behavior, it is necessary to consider
Figure 4.6: The density profiles of linkers in the top gel and bottom at \( D = 75\sigma, 55\sigma, 35\sigma \). In all simulation runs with different separation distance \( D \), each polymer gel has \( 2 \times 2 \times 2 \) units of diamond lattice cubes with \( L_x = L_y = 42\sigma \).
the network geometry in our model. We used diamond-lattice topology in our model, where linker monomers occupied points in a diamond lattice, and straight polymer chains were connected between these linker monomers. Also, there was $2 \times 2 \times 2$ diamond cubic in our simulations.

In figure 4.6, we plotted the distribution of linker monomers in the top and bottom polymer gels. From the figure, we can see that the distribution of linker monomers was about uniform at $D = 75\sigma$, when two polymer gels started to contact. When we moved two walls close to $D = 55\sigma$, there were more interactions between them, and at the same time the distribution of linker monomers had some peaks, showing a tendency to form two layers. When we compressed two polymer networks to $D = 35\sigma$, the peak in the distribution of linker monomers became much clearer. The great similarity between the distribution of linker monomers and the density profiles of polymer monomers gives us hints to explain why there exists wave-like oscillations in figure 4.5. When we compressed diamond geometry polymer networks, there exist layers of polymer linkers, and therefore these polymer linkers hold their connected polymer chains around them, and will generate wave-like oscillations in figure 4.5.

### 4.3.4 Friction coefficients

Figure 4.7b shows profiles of friction coefficients and friction force between two sliding polymer gels grafted on wall surfaces with parameters specified in the figure caption. In our molecular dynamics simulation, similar to polymer
Figure 4.7: In all simulation runs with different separation distance $D$, each polymer gel has $2 \times 2 \times 2$ units of diamond lattice cubes with $L_x = L_y = 42\sigma$. 

(a) Friction coefficient $\mu$ as a function of sliding velocity $v(\sigma/\tau)$.

(b) Friction force $f$ as a function of sliding velocity $v(\sigma/\tau)$. 


brush’s simulation, the normal force was almost constant as a function of shear velocity. At a specific surface separation distance $D$, the increase in friction coefficient $\mu$ with velocity is due to the increase in the shear force. When the wall separation distance $D$ was decreased, the normal force and shear force increased in our computer simulations. The result comes from stronger interactions between monomers in the top and bottom polymer gels, because decreasing the surface separation distance $D$ results an increase in the monomer density at their interface ($z = \frac{D}{2}$).

When two walls are moved together, the normal load and friction forces at the interface of two gels during sliding becomes larger, but the friction coefficient is smaller. When two gels are moved closer, the repulsion force between them increases, a thicker counter-ions layer forms between two gels, and decreases the sliding friction coefficient. Therefore, under assumptions of our model, although there is larger sliding friction force between two poly-electrolyte gels when we push two gels surfaces closer, the friction coefficient is decreased[86].

J.Gong made measurements of surface friction of gels[86]. In their experiments, they used polyelectrolyte gels in NaCl solutions. Because there is no explicit solvent molecule, our simulation only captured frictions coming from interactions between monomers on the chains. They found that, the increase of charge density would give rise to a non-flow network density, and an increase in the friction force.
4.3.5 Radial distribution function of polymer gel

Figure 4.8: The radial distribution function $g(r)$ between top and bottom polymer gels with separation distance $D = 35\sigma, 50\sigma, 75\sigma$. In all simulation runs with different separation distance $D$, each polymer gel has $2 \times 2 \times 2$ units of diamond lattice cubes with $L_x = L_y = 42\sigma$.

To determine directly the amount of contact at the interfaces between top and bottom polymer brushes, we present radial distribution function results in figure 4.8. The radial distribution function $g(r)$ gives the probability of finding a pair of monomers a distance $r$ apart, relative to the probability expected for a completely random distribution at the same density [67, 38].
We computed the radial distribution function $g(r)$ by the following formula

$$
g(r) = \rho^{-2} \left( \sum_i \sum_{j \neq i} \delta(r_i) \delta(r_j - r) \right) = \frac{V}{N^2} \left( \sum_i \sum_{j \neq i} \delta(r - r_{ij}) \right) \quad (4.7)
$$

where $r_i$ is position of a monomer in the bottom polymer gel, and $r_j$ is the position of a monomer in the top polymer gel. It can be thought of as the number of monomer pairs between top and bottom polymer gels with a given separation $r$ compared to the number of monomer pairs with the same density and separation $r$ in an ideal gas.

From figure 4.8, we could see that the radial distribution function of polymer gels is greatly different from the radial distribution function of polymer brushes in figure 3.6 close to $r = 1\sigma$, which is about the hard core radius of Lennard-Jones potential in our simulation. The radial distribution close to $r = 1\sigma$ represents the amount of contact between top and bottom polymers. Figures 4.8 and 3.6 show that two apposing polymer brushes have much stronger interaction at their interface than two compressed polymer gels. This result is consistent with the interpenetration figures 3.5 and 4.2; both of them show that monomers of polymer brushes have more interactions than monomers of polyelectrolyte gels when they are pressed together, and therefore polymer gels have much smaller friction coefficients than polymer brushes.
4.4 Conclusions

We simulated the compression and shear behavior of two apposing polymer gels using static diamond topology linked polymer networks. From our molecular dynamics simulations, we demonstrated the existence of a small friction coefficient and small monomer density penetration in polyelectrolyte gels. Our analysis suggests that a polyelectrolyte gel exhibits a small friction coefficient, because its networks hold polymer chains around its linker monomers, and monomers of polymer gels are no longer free to move like monomers in polymer brushes. Hence they have difficulty in penetrating into other polymer gels.
Appendix A

Dissipative particle dynamics (DPD) method

The dissipative particle dynamics (DPD) method is a popular simulation techniques for mesoscale polymer systems, and received lots of attention since it was introduced by Hoogerbrugge in 1992 and improved by Espanol in 1995[87, 88, 89, 90, 91].

The basic idea behind the DPD method is to simulate a system by clusters of molecules instead of individual molecules. The advantages of the DPD method over the conventional molecular dynamics method is that the DPD method could simulate a smaller number of particles with a larger time steps than normal molecular dynamics method to statistically represent a materials’ properties.

The dissipative particle dynamics method solves the Newtonian equations
for particles subject to conservative, dissipative, and random forces.

\[ \ddot{\vec{r}}_i = \frac{\vec{p}_i}{m_i} \quad (A.1) \]

\[ \ddot{\vec{p}}_i = \sum_{j \neq i} \vec{F}_{ij}^C + \vec{F}_{ij}^D + \vec{F}_{ij}^R \quad (A.2) \]

where \( F_{ij}^C \) is a slowly varying conservative force acting between particle \( i \) and particle \( j \), \( F_{ij}^D \) is the dissipative force, and \( F_{ij}^R \) is the random force on particle \( i \).

The inter-particle, conservative forces are linear,

\[ F_{ij}^C = a_{ij} \omega^R(\vec{r}_{ij})\hat{r}_{ij} \quad (A.3) \]

with

\[ \omega^R(\vec{r}_{ij}) = \begin{cases} a_{ij} (1 - r_{ij}/r_c)\hat{r}_{ij} & r_{ij} \leq r_c \\ 0 & r_{ij} > r_c \end{cases} \quad (A.4) \]

where \( a_{ij} \) is the reduced repulsion parameter and \( r_{ij} = |\vec{r}_i - \vec{r}_j| \), \( \hat{r}_{ij} = \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|} \), and \( r_c \) is the cut-off distance. The dissipative force \( F_{ij}^D \) is expressed by

\[ F_{ij}^D = -\gamma \omega^D(\vec{r}_{ij})(\dot{\vec{r}}_i \dot{\vec{r}}_j \hat{r}_{ij})\hat{r}_{ij} \quad (A.5) \]

with

\[ \omega^D(\vec{r}_{ij}) = [\omega^R(\vec{r}_{ij})]^2 \quad (A.6) \]
where $\gamma$ is a constant. The random force $F_{ij}^R$ is given by

$$F_{ij}^R = \sigma \omega^R(\vec{r}_{ij})\theta_{ij}(\delta t)^{-\frac{1}{2}}\dot{\vec{r}}_{ij}$$  \hspace{1cm} (A.7)

$$\sigma^2 = 2\gamma k_B T$$  \hspace{1cm} (A.8)

where $\theta_{ij}$ is a random number with zero mean and unit variance.

Several novel dissipative particle dynamics integration schemes were discussed by Nikunen in the paper [92]. They found the integration algorithm below is best most of the time,

$$\vec{r}_i(t + \delta t) = \vec{r}_i + \delta t\vec{v}_i(t) + \frac{\vec{F}_i(t)}{2m_i}(\delta t)^2$$  \hspace{1cm} (A.9)

$$\vec{v}_i(t + \frac{\delta t}{2}) = \vec{v}_i(t) + \frac{\vec{F}_i}{2m_i}\delta t$$  \hspace{1cm} (A.10)

$$\vec{F}_i(t + \delta t) = \vec{F}_i(\vec{r}_i(t + \delta t), \vec{v}_i(t + \frac{\delta t}{2}))$$  \hspace{1cm} (A.11)

$$\vec{v}_i(t + \delta t) = \vec{v}_i(t) + \frac{\vec{F}_i(t) + \vec{F}_i(t + \delta t)}{2m_i}\delta t$$  \hspace{1cm} (A.12)
Appendix B

Radial distribution function

The structure of polymers is characterized by a distribution function of particle positions. A common distribution is the radial distribution function $g_2(r_i, r_j)$ or simply $g(r)$. The function gives the probability of finding a pair of monomers a distance $r$ apart, relative to the probability expected for a completely random distribution at the same density [67, 38]. Here $r_i$ and $r_j$ are the position of monomers in the polymer systems. In the canonical ensemble

$$g(r_1, r_2) = \frac{N(N-1)}{\rho^2 Z_{NVT}} \int dr_3 \ldots r_N \exp \left( -\beta V(r_1, \ldots, r_N) \right), \quad (B.1)$$

where $V(r_1, \ldots, r_N)$ is the potential energy of the monomers. The choice $i = 1, j = 2$ is arbitrary in the simulation system of identical monomers. An
equivalent definition takes an ensemble average over pairs

\[
g(r) = \rho^{-2} \left( \sum_i \sum_{j \neq i} \delta(r_i) \delta(r_j - r) \right) = \frac{V}{N^2} \left( \sum_i \sum_{j \neq i} \delta(r - r_{ij}) \right) \quad (B.2)
\]

The last formula could be used to compute the radial distribution function \( g(r) \) numerically by computer simulation. The term \( \delta(r - r_{ij}) \) can be approximated by a histogram algorithm. If \( r \) falls into the bin range \([r_{ij}, r_{ij} + \delta r]\), the term \( \delta(r - r_{ij}) \) is assumed to be 1, otherwise 0.

The algorithm to compute radial distribution \( g(r) \) is [38]

\[
\text{for } i = 1 \rightarrow N \text{ do} \\
\quad \text{for } j = i + 1 \rightarrow N \text{ do} \\
\quad \quad r_{ij} \leftarrow \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \\
\quad \quad \text{bin} \leftarrow 1 + \text{INT}(r_{ij}/\text{binSize}) \\
\quad \quad \text{Hist}[\text{bin}] \leftarrow 1 + \text{Hist}[\text{bin}] \\
\quad \text{end for} \\
\text{end for} \\
\text{end for}
\]

The next step is to normalize the above histogram.

\[
\text{Hist}[\text{bin}] = \frac{\text{Hist}[\text{bin}]}{N} \quad (B.3)
\]
Then the radial distribution \( g(r) \) could be found by

\[
g(r + \frac{1}{2} \delta r) = \frac{\text{Hist}[\text{bin}]}{\text{Hist}[\text{bin}]^G} \tag{B.4}
\]

where \( \text{Hist}[\text{bin}]^G \) is the histogram for ideal gas, and is defined by

\[
\text{Hist}[\text{bin}]^G = \frac{4\pi \rho}{3} [(r + \delta r)^3 - r^3] \tag{B.5}
\]
Appendix C

Figure Permission
This is a License Agreement between Yangpeng Ou ("You") and Elsevier ("Elsevier") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Elsevier, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Elsevier Limited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registered Company Name</td>
<td>1982084</td>
</tr>
<tr>
<td>Customer name</td>
<td>Yangpeng Ou</td>
</tr>
<tr>
<td>Customer address</td>
<td>26 Newhall st, Malden, MA 02148</td>
</tr>
<tr>
<td>License number</td>
<td>3134960358358</td>
</tr>
<tr>
<td>License date</td>
<td>Apr 23, 2013</td>
</tr>
<tr>
<td>Licensed content publisher</td>
<td>Elsevier</td>
</tr>
<tr>
<td>Licensed content publication</td>
<td>Polymer</td>
</tr>
<tr>
<td>Licensed content title</td>
<td>Monte Carlo simulation of two interpenetrating polymer networks: Structure, swelling, and mechanical properties</td>
</tr>
<tr>
<td>Licensed content author</td>
<td>Samuel Edgecombe, Per Linse</td>
</tr>
<tr>
<td>Licensed content date</td>
<td>1 April 2008</td>
</tr>
<tr>
<td>Licensed content volume number</td>
<td>49</td>
</tr>
<tr>
<td>Type of Use</td>
<td>reuse in a thesis/dissertation</td>
</tr>
<tr>
<td>Portion</td>
<td>figures/tables/illustrations</td>
</tr>
<tr>
<td>Number of pages</td>
<td>12</td>
</tr>
<tr>
<td>Start Page</td>
<td>1981</td>
</tr>
<tr>
<td>End Page</td>
<td>1992</td>
</tr>
<tr>
<td>Format</td>
<td>electronic</td>
</tr>
<tr>
<td>Are you the author of this Elsevier article?</td>
<td>No</td>
</tr>
<tr>
<td>Will you be translating?</td>
<td>No</td>
</tr>
<tr>
<td>Order reference number</td>
<td></td>
</tr>
<tr>
<td>Title of your thesis/dissertation</td>
<td>Investigations of Mechanical Properties of Polymer Brushes and Hydrogels Through Molecular Dynamics Simulation</td>
</tr>
</tbody>
</table>
INTRODUCTION

1. The publisher for this copyrighted material is Elsevier. By clicking “accept” in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. (“CCC”), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

GENERAL TERMS

2. Elsevier hereby grants you permission to reproduce the aforementioned material subject to the terms and conditions indicated.

3. Acknowledgement: If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source, permission must also be sought from that source. If such permission is not obtained then that material may not be included in your publication/copies. Suitable acknowledgement to the source must be made, either as a footnote or in a reference list at the end of your publication, as follows:

“Reprinted from Publication title, Vol / edition number, Author(s), Title of article / title of chapter, Pages No., Copyright (Year), with permission from Elsevier [OR APPLICABLE SOCIETY COPYRIGHT OWNER].” Also Lancet special credit - “Reprinted from The Lancet, Vol. number, Author(s), Title of article, Pages No., Copyright (Year), with permission from Elsevier.”

4. Reproduction of this material is confined to the purpose and/or media for which permission is hereby given.

5. Altering/Modifying Material: Not Permitted. However figures and illustrations may be altered/adapted minimally to serve your work. Any other abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of Elsevier Ltd. (Please contact Elsevier at permissions@elsevier.com)

6. If the permission fee for the requested use of our material is waived in this instance, please be advised that your future requests for Elsevier materials may attract a fee.

7. Reservation of Rights: Publisher reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC’s Billing and Payment terms and conditions.

8. License Contingent Upon Payment: While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless
and until full payment is received from you (either by publisher or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received on a timely basis, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC’s Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

9. Warranties: Publisher makes no representations or warranties with respect to the licensed material.

10. Indemnity: You hereby indemnify and agree to hold harmless publisher and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

11. No Transfer of License: This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

12. No Amendment Except in Writing: This license may not be amended except in a writing signed by both parties (or, in the case of publisher, by CCC on publisher's behalf).

13. Objection to Contrary Terms: Publisher hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC’s Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and publisher (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC’s Billing and Payment terms and conditions, these terms and conditions shall control.

14. Revocation: Elsevier or Copyright Clearance Center may deny the permissions described in this License at their sole discretion, for any reason or no reason, with a full refund payable to you. Notice of such denial will be made using the contact information provided by you. Failure to receive such notice will not alter or invalidate the denial. In no event will Elsevier or Copyright Clearance Center be responsible or liable for any costs, expenses or damage incurred by you as a result of a denial of your permission request, other than a refund of the amount(s) paid by you to Elsevier and/or Copyright Clearance Center for denied permissions.

LIMITED LICENSE

The following terms and conditions apply only to specific license types:

15. Translation: This permission is granted for non-exclusive world English rights only unless your license was granted for translation rights. If you licensed translation rights you may only translate this content into the languages you requested. A professional translator must perform all translations and reproduce the content word for word preserving the integrity of the article. If this license is to re-use 1 or 2 figures then permission is granted for non-exclusive world rights in all languages.

16. Website: The following terms and conditions apply to electronic reserve and author websites: Electronic reserve: If licensed material is to be posted to website, the web site is to be
password-protected and made available only to bona fide students registered on a relevant course if:
This license was made in connection with a course,
This permission is granted for 1 year only. You may obtain a license for future website posting,
All content posted to the web site must maintain the copyright information line on the bottom of each image,
A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx or the Elsevier homepage for books at http://www.elsevier.com, and
Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.

17. **Author website** for journals with the following additional clauses:

All content posted to the web site must maintain the copyright information line on the bottom of each image, and the permission granted is limited to the personal version of your paper. You are not allowed to download and post the published electronic version of your article (whether PDF or HTML, proof or final version), nor may you scan the printed edition to create an electronic version. A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx. As part of our normal production process, you will receive an e-mail notice when your article appears on Elsevier’s online service ScienceDirect (www.sciencedirect.com). That e-mail will include the article’s Digital Object Identifier (DOI). This number provides the electronic link to the published article and should be included in the posting of your personal version. We ask that you wait until you receive this e-mail and have the DOI to do any posting.

Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.

18. **Author website** for books with the following additional clauses:
Authors are permitted to place a brief summary of their work online only. A hyper-text must be included to the Elsevier homepage at http://www.elsevier.com. All content posted to the web site must maintain the copyright information line on the bottom of each image. You are not allowed to download and post the published electronic version of your chapter, nor may you scan the printed edition to create an electronic version.

Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.

19. **Website** (regular and for author): A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx or for books to the Elsevier homepage at http://www.elsevier.com

20. **Thesis/Dissertation**: If your license is for use in a thesis/dissertation your thesis may be submitted to your institution in either print or electronic form. Should your thesis be published commercially, please reapply for permission. These requirements include permission for the Library and Archives of Canada to supply single copies, on demand, of the complete thesis and include permission for UMI to supply single copies, on demand, of the complete thesis. Should your thesis be published commercially, please reapply for permission.

21. **Other Conditions**:
If you would like to pay for this license now, please remit this license along with your payment made payable to "COPYRIGHT CLEARANCE CENTER" otherwise you will be invoiced within 48 hours of the license date. Payment should be in the form of a check or money order referencing your account number and this invoice number RLNK501006331. Once you receive your invoice for this order, you may pay your invoice by credit card. Please follow instructions provided at that time.

Make Payment To:
Copyright Clearance Center
Dept 001
P.O. Box 843006
Boston, MA 02284-3006

For suggestions or comments regarding this order, contact RightsLink Customer Support: customercare@copyright.com or +1-877-622-5543 (toll free in the US) or +1-978-646-2777.

Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.
Bibliography


[47] Changjun Hu, Yali Liu, and Jianjiang Li. Efficient Parallel Implementation of Molecular Dynamics with Embedded Atom Method on Multi-


