TOWARDS RATIONAL DESIGN OF NANOSCALE LUBRICANTS AND ELUCIDATION OF THE HYDRATION LUBRICATION MECHANISM USING MOLECULAR SIMULATION

By

Christoph Klein

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Chemical and Biomolecular Engineering

December 16, 2017

Nashville, Tennessee

Approved:

Peter T. Cummings, Ph.D.

Clare McCabe, Ph.D.

Paul E. Laibinis, Ph.D.

John T. Wilson, Ph.D.

Paul D. Sheldon, Ph.D

To my family, thank you for always supporting me.

ACKNOWLEDGMENTS

Clare McCabe and Peter T. Cummings, I cannot thank you enough for accepting me into your research groups. Your cogent advising, the funding you provided and the opportunities that you created to speak at conferences around the world have shaped me into a better scientist and a better person. Christopher R. Iacovella , thank you for your invaluable mentorship from getting me on my feet in the lab to helping me navigate the academic landscape to the honest feedback you have provided throughout the years.

Much the software work described in this dissertation was performed under the mentorship of Janos Sallai. Our weekly pair-programming sessions and candid discussions throughout the years kept me focused and excited about my work. I owe Janos much of my programming knowledge and a newly gleaned fascination with many aspects of computer science. I would not be in the position I am today without your guidance.

I am also grateful for the friends and colleagues in the Cummings and McCabe labs who contributed to the daily research experience, many excellent conference trips and other outings in and around Nashville, especially Timothy Moore, Andrew Summers, Matthew Thompson and Remco Hartkamp.

Lastly, a significant portion of my work took the form of open-source software development. Through these projects, I have had the pleasure to collaborate with others at Vanderbilt and beyond. I am especially grateful to Michael Shirts (CU Boulder), Jason Swails (Rutgers), Robert McGibbon (Stanford), Kyle Beauchamp (MSKCC), and Matthew Harrigan (Stanford) all of whom welcomed my contributions to their projects and provided excellent feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Harnessing the often unusual phenomena that occur at the nanoscale has led to a vast number of applications in areas ranging from manufacturing,[1] to medicine[2] and even household products.[3] Most notably perhaps, many bulk properties of materials do not persist down to the nanoscale. For example, while a conventional combustion engine may be well lubricated by oil, these lubricants are unable to dissipate frictional forces in many micro- and nano-electromechanical systems (MEMS and NEMS). This failure occurs at separation distances on the order of several nanometers due to a phase change from liquid to solid accompanied by a several order magnitude increase in viscosity. Nanoconfined dodecane, for example, has been observed to undergo this transition when separation distances are between six and seven molecular layers.[4] Similarly, a macroscopically smooth material may be rough on the nanoscale. In such devices, sliding surfaces with nanoscale roughness lead to stiction due to atomic-level interactions that quickly lead to wear, decreased device performance, and ultimately device failure. Designing better lubricants for ever decreasing device sizes requires fundamental understanding of the relationships between surface structure, film properties and solvent interactions among many other factors.

The U.S. Materials Genome Initiative (MGI) "aims to double the speed at which we discover, develop and manufacture new materials".[5] Computational tools comprise one of the four pillars that form the materials innovation infrastructure as described in the initiative. Specifically, the goal is to use computational tools to design and explore new material candidates. Several projects in line with this initiative are already successfully producing promising results such as the MIT Materials Project,[6] Northwestern University's metal organic framework screening[7] and the Harvard Clean Energy Project.[8] However, these successful projects are not tackling soft-matter design problems such as designing lubricants.

Soft-matter's predominant energy scales and dynamic nature present a more complex sampling problem for simulation studies. Many of the performance critical phenomena in soft materials occur at energy levels comparable to the system thermal energies. Therefore, soft-matter systems will exhibit far more, often unpredictable, energy minima. Such problems requires dynamical measurements and therefore long simulation times of systems with far more complex initial simulation setups than are required when e.g. finding the minimum energy of a metal organic framework.[7]

The work described herein falls under two main categories: 1) the development of tools for the screening and design of nanolubricating systems as well as more generally enhancing reproducibility of simulation studies (Chapters 3 & 4) and 2) the simulation of nanolubricating systems (Chapters 5-7). Both tasks have been undertaken concurrently to provide a continuous feedback loop between the scientific requirements and features provided by the tools.

First, in Chapter 3, I describe `mBuild`, a tool developed to aid in the construction of complex chemical systems with a particular emphasis on simplifying construction of surface functionalized systems such as self-assembled monolayers and tethered nanoparticles. The design of the data structure, underlying algorithms and several use cases are presented.

Once a chemical system is constructed in `mBuild`, the next step towards running a molecular dynamics simulation is to apply a force field to the system that describes the potential energy function governing its dynamics. Chapter 4 describes `Foyer`, a tool for atom-typing as well as applying and disseminating such force fields. Details are provided on the atom-typing process and the efforts made to promote development and dissemination of force fields in a more reproducible and transparent way than is current practice.

Chapter 5 explores a portion of the chemical parameter space of self-assembled monolayers by leveraging `mBuild` to construct the systems while exposing variables to tune the chemistry and `Foyer` to automate the application of forcefield models to the systems. The tribological properties of a family of self-assembled monolayers are screened by varying

1) chain length, 2) chain backbones, 3) chain head groups and 4) surface structure. This study provides trend-level insight into the relative factors that impact self-assembled monolayer performance as lubricants, identifies several pitfalls associated with using idealized surfaces and showcases the ability of `mBuild` and `Foyer` to facilitate screening of across chemical parameter space of soft matter systems.

In Chapters 6 & 7, the same tools are leveraged to provide insight into the hydration lubrication mechanism and its role in recently synthesized brush structures, zwitterionic monolayers and polymer brushes. Hydration lubrication is the mechanism by which such materials are thought to provide ultra-low friction coefficients as discussed in greater detail in the following chapter. The results presented in Chapters 6&7 support the experimentally posited hypothesis that hydration lubrication manifests itself in these zwitterionic systems and identifies the specific chemical moieties most likely to be responsible for its actuation.

Lastly, Chapter 8 provides an overview of the dissertation's primary findings and recommendations for future avenues of research.

1.1 Bibliography

[1] Zhao, X.-M.; Xia, Y.; Whitesides, G. M. *Journal of Materials Chemistry* **1997**, *7*, 1069–1074.

[2] Moghimi, S. M.; Hunter, A. C.; Murray, J. C. *FASEB journal : official publication of the Federation of American Societies for Experimental Biology* **2005**, *19*, 311–30.

[3] Sadrieh, N.; Wokovich, A. M.; Gopee, N. V.; Zheng, J.; Haines, D.; Parmiter, D.; Siitonen, P. H.; Cozart, C. R.; Patri, A. K.; McNeil, S. E.; Howard, P. C.; Doub, W. H.; Buhse, L. F. *Toxicological sciences : an official journal of the Society of Toxicology* **2010**, *115*, 156–66.

[4] Cui, S. T.; Cummings, P. T.; Cochran, H. D. *The Journal of Chemical Physics* **2001**, *114*, 7189.

[5] Http://www.whitehouse.gov/mgi,

[6] Jain, A.; Ong, S. P.; Hautier, G.; Chen, W.; Richards, W. D.; Dacek, S.; Cholia, S.; Gunter, D.; Skinner, D.; Ceder, G.; Persson, K. A. *APL Materials* **2013**, *1*, 011002.

[7] Wilmer, C. E.; Leaf, M.; Lee, C. Y.; Farha, O. K.; Hauser, B. G.; Hupp, J. T.; Snurr, R. Q. *Nature chemistry* **2012**, *4*, 83–9.

[8] Hachmann, J.; Olivares-Amaya, R.; Atahan-Evrenk, S.; Amador-Bedolla, C.; Sanchez-Carrera, R. S.; Gold-Parker, A.; Vogt, L.; Brockway, A. M.; Aspuru-Guzik, A. *The Journal of Physical Chemistry Letters* **2011**, *2*, 2241–2251.

CHAPTER 2

BACKGROUND

2.1 Lubricating Micro- and Nano Electromechanical Systems

Hard disk drives (HDD) are an example of a system where separation distances have decreased to the nanoscale. In modern HDDs, the read/write head, through which information is transfered to and from the physical storage platter, floats on a cushion of air while navigating the platter's surface at up to 15,000 rotations per minute. This cushion of air has decreased from ~10 nm to ~5 nm in thickness over the past decade.[1,2] As manufacturers aim to increase data density in HDDs, this separation distance will have to decrease to ~3 nm in order to hit their target goal of 10 terabytes/in^2 in the next generation of hard disk drives.[2] In order to achieve separation distances of 3 nm, read/write heads will no longer be able to rely on air to provide a similar cushion.[3] The heads will instead have to be designed for constant contact with the surface below.

Surface functionalization presents a promising avenue through which to lubricate various MEMS and NEMS devices, including hard disk drives. In particularly, self-assembled monolayers (SAMs) exhibit many of the desired characteristics and present a tunable platform to achieve specific performance metrics. They can for example be tuned to withstand high shear rates.[4] This is partially accomplished by adjusting the chain length of monomers[5–7] which also controls the minimum separation distance in a given device. Additionally, the disorder of the chains[6] as well as the species comprising the monolayer[8–11] are adjustable parameters that influence monolayers' stability, compressibility and ability to lubricate. By tuning such parameters, phenomena unique to the nanoscale can be exploited to enhance performance. For example, frictional forces can be more efficiently dissipated under certain conditions by designing monolayers that can form liquid-like layers.[10,12]

Even though self-assembled monolayers provide a highly tunable platform for designing nanoscale lubricants, monolayer systems synthesized to date exhibit poor stability which has hindered their application in lubricating systems.[4,12] Booth et al., for example, reported lifetimes on the order of hours to days. The precise mechanism of degradation is not fully understood[13] but ultimately hinges on the preferential scission of a bond type which then induces localized disorder in the monolayer. Under shear, this local defect destabalizes its immediate environment which cascades into utimate failure of the material.[14] Designing monolayers that are stable enough to find application in NEMS/MEMS will require a thorough understanding of this degradation mechanism in order to directly counteract it.

2.2 Hydration Lubrication

The mammalian synovial joint presents another example of a system where nanoscale effects appear to cause unanticipated behavior based on the bulk properties of the system's constituents. Within the synovium, long chain hyaluronic acid chains interact with both water and various surface proteins and lipids which enable the joint to sustain friction coefficients in the range of 0.001[15] to 0.1[16] while subject to physiological pressures. However, the precise mechanism of lubrication in these systems remains challenging to resolve.

More recently it was discovered that surfaces functionalized with zwitterionic polymer brushes mimic aspects of a joint's composition[17] and can yield friction coefficients as low or lower than those found in natural synovial joints.[18–23] Such materials resist compression up to normal loads of several MPa while maintaining friction coefficients orders of magnitude lower than comparable neutral polymers in non-polar[24] and aqueous solvents[25] as well as non-grafted charged polymers in aqueous solution.[26] Kyomoto *et al.* compared polymer brushes comprised of different neutral, charged and zwitterionic monomers and found that those grown from 2-methacryloyloxyethyl phosphorylcholine (MPC) exhibited the lowest coefficients of friction while enduring negligible wear over millions of cycles[22]

and proposed that these brushes could improve the performance of artificial hips.[27] The mechanism by which such materials are thought to provide ultra-low friction coefficients has been termed "hydration lubrication".[28–31]

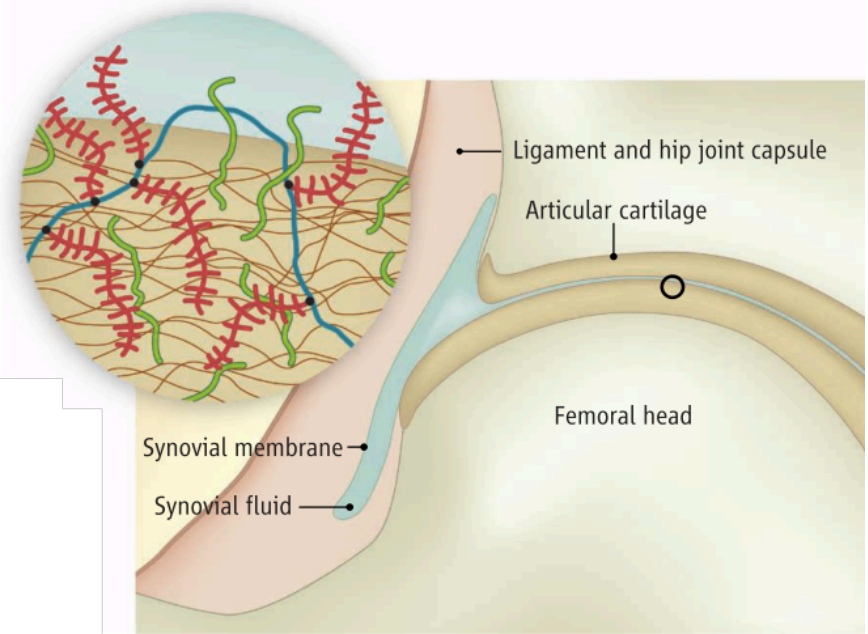


Figure 2.1: A close-up view of the articular cartilage surface. Schematic section through part of a hip joint. The friction coefficients between the articular cartilage layers, compressed to 50 atmospheres or more in a hip joint, can be as low as 0.001. (Inset) The detailed structure at the outer cartilage surface is thought to include charged macro-molecules - mainly hyaluronic acid (blue), to which are attached aggrecans (red) and lubricins (green) - that extend from the surface to form a brushlike layer. Synthetic charged brushes lead to low friction similar to that in articular cartilage, although, to date, only up to much lower pressures than in human joints. From Klein, J. (2009). Repair or replacement: a joint perspective. Science, 323(5910), 47-48. Reprinted with permission from AAAS.

Qualitatively, water is hypothesized to facilitate this mechanism by rapidly relaxing between hydration shells surrounding charged groups while simultaneously supporting high normal loads due to the tightly bound nature of these hydration shells. The rapid relaxation of water molecules allows systems to exhibit a fluid shear response. However, the precise nature of this mechanism and hence the validity of the hypothesized mechanism has yet to be determined.[31] Additionally, the hydration lubrication mechanism itself presents but one pathway for dissipation of frictional forces in complex systems like zwitterionic polymer

brushes.[19,25,32] Molecular dynamics simulation, as used herein, provides a promising avenue for studying this mechanism and its role in more complex systems as it grants direct insight into the dynamic motion of charged groups and water molecules during shearing.

2.3 Rational Design of Nanoscale Lubricants

Biolubrication as well as hard disk drive lubrication, and in a broader sense NEMS/MEMS lubrication, present an engineering design problem. We understand many, but not all, of the phenomena that can be exploited to design effective materials to lubricate these systems yet we lack the ability to truly explore this design landscape and automate the discovery of novel lubricants. For example, the hydration lubrication mechanism is somewhat speculative, where it is "assumed qualitatively that [the mechanism] operates via a fluid response to shear of the hydration shells".[31] Furthermore, this mechanism is difficult to resolve via experimental techniques alone. In a recent review of the subject, Jacob Klein - a pioneer and leading researcher in the field - highlighted the need for, amongst other methods, simulation to define the precise nature of the mechanism and to probe its limits. Simulation has been used extensively to both understand tribological phenomena at the nanoscale[33–36] but also to design better self-assembled monolayer based lubricants.[10]

Efficient utilization molecular dynamics simulation to explore this design space is currently hampered by a lack of a coherent tool set to simplify and automate system construction for surface functionalized materials. By contrast, the biophysics simulation community has invested considerable effort into creating tools and databases for building and parameterizing biological molecules with minimal effort, e.g. the Protein Data Bank,[37] VMD,[38] AmberTools,[39] the Omnia suite.[40] Such toolchains allow researchers to generate input files for complex structures, such as proteins and DNA, that can run on most molecular dynamics simulation engines with little to no manual intervention and have lead to successful studies which harness the largest computing resources available to conduct studies such as exploring the human kinome.[41,42] However, while the biophysics community's tools

provide excellent functionality for biological system setup, they do not allow one to easily generate arbitrary structures found outside this biochemical niche. For example, surface bound brushes or tethered nanoparticles, which often feature semi-infinite substrates and/or irregular surface bonding sites, require a less specialized approach. These systems may not be regular and thus defining a small unit cell and replicating it is not always possible. Additionally, many tools are tied to a specific simulation environment[39] or are operated via a custom language that complicates integration with a broader scientific ecosystem of tools for performing tasks not specific to the domain of molecular simulation, such as statistical analysis and visualization.

2.4 Bibliography

[1] Ambekar, R.; Gupta, V.; Bogy, D. B. *Journal of Tribology* **2005**, *127*, 530.

[2] Sorkin, V.; Sha, Z. D.; Branicio, P. S.; Pei, Q. X.; Zhang, Y. W. *IEEE Transactions on Magnetics* **2013**, *49*, 5227–5235.

[3] Chen, H.; Guo, Q.; Jhon, M. S. *IEEE Transactions on Magnetics* **2007**, *43*, 2247–2249.

[4] Booth, B. D.; Vilt, S. G.; Lewis, J. B.; Rivera, J. L.; Buehler, E. a.; McCabe, C.; Jennings, G. K. *Langmuir* **2011**, *27*, 5909–17.

[5] Chandross, M.; Grest, G. S.; Stevens, M. J. *Langmuir* **2002**, *18*, 8392–8399.

[6] Chandross, M.; Webb, E.; Stevens, M.; Grest, G.; Garofalini, S. *Physical Review Letters* **2004**, *93*, 166103.

[7] Chandross, M.; Lorenz, C. D.; Grest, G. S.; Stevens, M. J.; Iii, E. B. W. *JOM* **2005**, *57*, 55–61.

[8] Lorenz, C.; Webb, E.; Stevens, M.; Chandross, M.; Grest, G. *Tribology Letters* **2005**, *19*, 93–98.

[9] Lorenz, C. D.; Chandross, M.; Grest, G. S.; Stevens, M. J.; Webb, E. B. *Langmuir* **2005**, *21*, 11744–8.

[10] Lewis, J. B.; Vilt, S. G.; Rivera, J. L.; Jennings, G. K.; McCabe, C. *Langmuir* **2012**, *28*, 14218–26.

[11] Rivera, J. L.; Jennings, G. K.; McCabe, C. *The Journal of Chemical Physics* **2012**, *136*, 244701.

[12] Vilt, S. G.; Leng, Z.; Booth, B. D.; McCabe, C.; Jennings, G. K. *The Journal of Physical Chemistry C* **2009**, *113*, 14972–14977.

[13] Summers, A. Z.; Iacovella, C. R.; Billingsley, M. R.; Arnold, S. T.; Cummings, P. T.; McCabe, C. *Langmuir* **2016**, *32*, 2348–2359.

[14] Bhushan, B. *Microelectronic Engineering* **2007**, *84*, 387–412.

[15] Clarke, I. C.; Contini, R.; Kenedi, R. M. *Journal of Lubrication Technology* **1975**, *97*, 358.

[16] Roberts, B. J.; Unsworth, A.; Mian, N. *Annals of the Rheumatic Diseases* **1982**, *41*, 217–224.

[17] Klein, J. *Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology* **2006**, *220*, 691–710.

[18] Kobayashi, M.; Terayama, Y.; Hosaka, N.; Kaido, M.; Suzuki, A.; Yamada, N.; Torikai, N.; Ishihara, K.; Takahara, A. *Soft Matter* **2007**, *3*, 740.

[19] Chen, M.; Briscoe, W. H.; Armes, S. P.; Klein, J. *Science (New York, N.Y.)* **2009**, *323*, 1698–701.

[20] Chen, M.; Briscoe, W. H.; Armes, S. P.; Cohen, H.; Klein, J. *European Polymer Journal* **2011**, *47*, 511–523.

[21] Zhang, Z.; Morse, A. J.; Armes, S. P.; Lewis, A. L.; Geoghegan, M.; Leggett, G. J. *Langmuir : the ACS journal of surfaces and colloids* **2011**, *27*, 2514–21.

[22] Kyomoto, M.; Moro, T.; Saiga, K.; Hashimoto, M.; Ito, H.; Kawaguchi, H.; Takatori, Y.; Ishihara, K. *Biomaterials* **2012**, *33*, 4451–9.

[23] Banquy, X.; Burdyńska, J.; Lee, D. W.; Matyjaszewski, K.; Israelachvili, J. *Journal of the American Chemical Society* **2014**, *136*, 6199–202.

[24] Klein, J.; Kumacheva, E.; Mahalu, D.; Perahia, D.; Fetters, L. J. *Nature* **1994**, *370*, 634–636.

[25] Raviv, U.; Giasson, S.; Kampf, N.; Gohy, J.-F.; Jérôme, R.; Klein, J. *Nature* **2003**, *425*, 163–5.

[26] Kampf, N.; Raviv, U.; Klein, J. *Macromolecules* **2004**, *37*, 1134–1142.

[27] Kyomoto, M.; Moro, T.; Takatori, Y.; Kawaguchi, H.; Ishihara, K. *Clinical orthopaedics and related research* **2011**, *469*, 2327–36.

[28] Raviv, U.; Klein, J. *Science (New York, N.Y.)* **2002**, *297*, 1540–3.

[29] Briscoe, W. H.; Titmuss, S.; Tiberg, F.; Thomas, R. K.; McGillivray, D. J.; Klein, J. *Nature* **2006**, *444*, 191–4.

[30] Gaisinskaya, A.; Ma, L.; Silbert, G.; Sorkin, R.; Tairy, O.; Goldberg, R.; Kampf, N.; Klein, J. *Faraday Discussions* **2012**, *156*, 217.

[31] Klein, J. *Friction* **2013**, *1*, 1–23.

[32] Ma, L.; Gaisinskaya-kipnis, A.; Kampf, N.; Klein, J. *Nature Communications* **2015**, *6*, 1–6.

[33] Cui, S. T.; Cummings, P. T.; Cochran, H. D. *The Journal of Chemical Physics* **2001**, *114*, 7189.

[34] Onodera, T.; Morita, Y.; Suzuki, A.; Koyama, M.; Tsuboi, H.; Hatakeyama, N.; Endou, A.; Takaba, H.; Kubo, M.; Dassenoy, F.; Minfray, C.; Joly-Pottuz, L.; Martin, J.-M.; Miyamoto, A. *The Journal of Physical Chemistry B* **2009**, *113*, 16526–36.

[35] Onodera, T.; Morita, Y.; Nagumo, R.; Miura, R.; Suzuki, A.; Tsuboi, H.; Hatakeyama, N.; Endou, A.; Takaba, H.; Dassenoy, F.; Minfray, C.; Joly-Pottuz, L.; Kubo, M.; Martin, J.-M.; Miyamoto, A. *The Journal of Physical Chemistry B* **2010**, *114*, 15832–8.

[36] Yue, D.-C.; Ma, T.-B.; Hu, Y.-Z.; Yeon, J.; van Duin, A. C. T.; Wang, H.; Luo, J. *The Journal of Physical Chemistry C* **2013**, *117*, 25604–25614.

[37] Bernstein, F. C.; Koetzle, T. F.; Williams, G. J.; Meyer, E. F.; Brice, M. D.; Rodgers, J. R.; Kennard, O.; Shimanouchi, T.; Tasumi, M. *Archives of Biochemistry and Biophysics* **1978**, *185*, 584–591.

[38] Humphrey, W.; Dalke, A.; Schulten, K. *Journal of Molecular Graphics* **1996**, *14*, 33–38.

[39] Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2013**, *3*, 198–210.

[40] Available at http://www.omnia.md.

[41] Parton, D. L.; Hanson, S. M.; Rodríguez-Laureano, L.; Albanese, S. K.; Gradia, S.; Jeans, C.; Seeliger, M. A.; Chodera, J. D. *bioRxiv* **2016**, 038711.

[42] Daniel L. Parton, S. M. H. K. A. B. J. D. C., Patrick B. Grinaway *PLoS Computational Biology* **2016**, *12*, e1004728.

CHAPTER 3

A HIERARCHICAL, COMPONENT BASED APPROACH TO SCREENING PROPERTIES OF SOFT MATTER

This chapter describes the tool `mBuild` which we developed for constructing arbitrarily complex input configurations for molecular simulation in a programmatic fashion. Basic molecular components are connected using an equivalence operator which reduces and often removes the need for users to explicitly rotate and translate components as they assemble systems. Additionally, the programmatic nature of this approach and integration with the scientific Python ecosystem seamlessly exposes high-level variables that users can tune to alter the chemical composition of their systems, such as mixtures of polymers of different chain lengths and surface patterning. Leveraging these features, I demonstrate how `mBuild` serves as a stepping stone towards screening and performing optimizations in chemical parameter space of complex materials by performing automated screening studies of monolayer systems as a function of graft type, degree of polymerization, and surface density. The work detailed here is based on references,[1,2] and.[3]

3.1 Introduction

The preliminary concepts underpinning `mBuild`'s functionality were introduced in Sallai et al.[1] Since then, `mBuild` has evolved into a Python package designed to simplify the construction of complex, regular and irregular structures and topologies as well as integrate seamlessly with the Python scientific stack and more recently developed Python tools in the area of molecular simulation.[4–8] `mBuild` adopts a hierarchical approach to system construction that relies on equivalence relations to connect chemical building blocks (components). Every component can recursively contain particles and other components to

generate arbitrary, hierarchical structures where every particle represents a leaf in the hierarchy. Low-level components, such as an alkyl group or a monomer, can be hand-drawn using software like Avogadro[9] and then connected using an equivalence operator which matches defined attachment sites between two components - the operator forces two sets of points in space to overlap thus translating and rotating components into the desired positions. This approach minimizes and often even eliminates the need for users to explicitly translate or rotate components while constructing initial configurations - users simply specify which components should be connected. Additionally, the hierarchical nature of this approach allows for complex families of chemical structures to be encapsulated in a single component class which exposes user defined, tunable parameters that adjust the structural properties of the system (e.g. chain length, surface coverage). By providing a more natural avenue to express such structures, where the requirement for mental visualization of spatial arrangements is minimized, `mBuild` provides a stepping stone towards the goals outline by the Materials Genome Initiative,[10] by enabling screening of and optimizations in chemical parameter space of complex, soft-materials.

Here, I provide an overview of the algorithms associated with `mBuild` including several recent improvements, and demonstrate its use as a means for automating screening of soft matter systems. I illustrate the construction of basic components, how they can be connected programmatically into complex chemical systems, and finally showcase this functionality by generating and performing parameter sweeping simulations of an ensemble of monolayers constructed of alkanes and polyethylene glycol (PEG) where, through the functionality of `mBuild`, we trivially vary surface density, patterning and chain length in an automated, programmatic way.

3.2 Software Concept

While the basic concepts and algorithms underlying `mBuild` were outlined in Ref.,[1] additional refinement and development has been undertaken, as reported here, in partic-

ular to simplify and increase the generality of the data structure and provide enhancements with regards to connecting individual components via equivalence transforms. The primary building blocks of an `mBuild` hierarchy are `Compounds`; every user-created component inherits from this class. Each `Compound` can contain an arbitrary amount of other `Compounds`, allowing for systems to be flexibly built in a hierarchical manner. The programmatic connection of `Compounds` in three dimensional space is facilitated by an equivalence transform. This concept is formalized and implemented via the `Port` class which defines connection sites and orientation. These are each discussed below.

3.2.1 Data Structure

The hierarchical data structure of `mBuild` is composed of `Compounds`. `Compounds` maintain an ordered set of children which are other `Compounds`. `Compounds` at the bottom of an `mBuild` hierarchy, i.e., the leafs of the tree, are referred to as `Particles` and can be instantiated as, for example, `lj = mb.Particle(name='lennard-jonesium')`. Note however, that this merely serves to illustrate that this `Compound` is at the bottom of the hierarchy; `Particle` is an alias for `Compound` which can be used to clarify the intended role of an object you are creating.

Every `mBuild` hierarchy also maintains a network of bonds between its `Particles` in the form of a graph as provided by the `NetworkX` package.[11] This graph is maintained by the root (top level component) of the given hierarchy. When two `Compounds` with bonds are added together, their bond graphs are composed.

Additionally, `Compounds` have built-in support for copying and deepcopying `Compound` hierarchies, enumerating particles or bonds in the hierarchy, proximity based searches, visualization, I/O operations, and a number of other convenience methods that enable complex topologies to be constructed with little user effort.

3.2.2 Equivalence Transforms

When connecting components in 3D space, their relative orientations must be specified. In `mBuild`, this is accomplished via an equivalence transform. The equivalence operator described here declares points in a component's local coordinate system to be equivalent to points in another component's coordinate system. Using these point pairs, it is possible to compute a rigid transformation, specifically an affine coordinate transformation conserving scaling and orientation (chirality), that, when applied to one component, will transform its designated points to the other component's respective points. Specifying four or more pairs of non-coplanar points is sufficient to compute an unambiguous transformation matrix in 3D space.

Using a rigid transformation F, one can map a point vector v to its image $F(v)$ in a different coordinate system. This operation can be expressed as a multiplication by a rotation matrix $R \in \mathbb{R}^{3\times3}$ and a translation with vector $t \in \mathbb{R}^{3\times1}$.

$$F(v) = Rv + t \tag{3.1}$$

R and t can be solved for using the singular value decomposition to get the pseudoinverse given four or more points $P_i(x_i, y_i, z_i)$ and their images $P_i^{'}(x_i^{'}, y_i^{'}, z_i^{'})$ in the target 3-dimensional coordinate system:

$$\begin{bmatrix} x_1^{'} & x_2^{'} & \dots & x_n^{'} \\ y_1^{'} & y_2^{'} & \dots & y_n^{'} \\ z_1^{'} & z_2^{'} & \dots & z_n^{'} \\ 1 & 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{3.2}$$

where the lower elements in the transformation matrix (0 and 1) are of dimensions 1×3 and 1×1 respectively.

In `mBuild`, this equivalence transform is used to force four points of one compound

to overlap with four points of another. Achieving this generally requires that the same arrangement of four non-coplanar points must be added to any compound intended to make use of the equivalence transform.

3.2.3 Ports

To formalize, simplify, and enable this behavior to function with any compound, `mBuild` provides the `Port` class, which is a simple `Compound` containing four untyped `Particles` in a compact, non-coplanar arrangement (see Fig. 3.1). Note that for most use cases, it is not desirable to print these untyped, extra `Particles` when outputting the final structure to a file, which is the default behavior of the `Compound.save()` method, but they can be saved if desired.

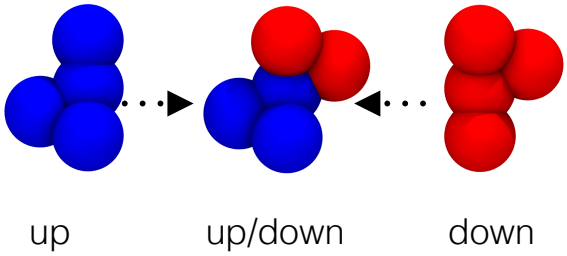


Figure 3.1: The spatial arrangement of the particles within a `Port`. Both `up` and `down` contain the same arrangement of four non-coplanar particles except that they face opposite directions.

Instead of having to explicitly define an equivalence relation between four pairs of points, `mBuild` allows for declaring two `Ports`, one in each compound, to be equivalent. When performing an equivalence transform on two `Ports`, one of the `Compounds` that the two `Ports` are a part of is rotated and translated, such that the untyped particles inside their respective ports overlap (see Fig. 3.2). Since it is common that `Ports` represent bonding sites where molecule fragments need to be attached, `mBuild` allows for defining an *anchor* `Compound` associated with a `Port`. After the affine transformation is applied, `mBuild` will automatically create a bond between the two respective anchors, relieving the user from this often tedious task.

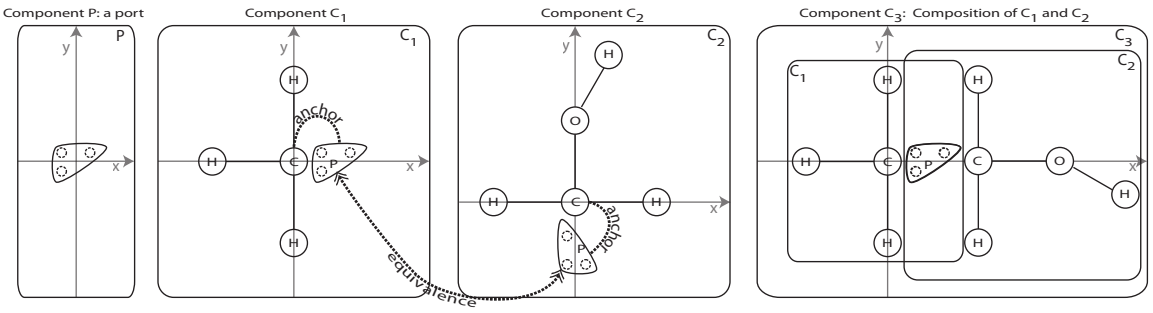


Figure 3.2: A `Port` is a compound with two pairs of four `Particles`. Here, one pair of three points is shown to illustrate this 2D example. `Ports` are attached to any other `Compound`, most commonly anchored to a `Particle` where a chemical bond should exist. Compound C1 is a methyl group with a `Port` anchored to the carbon atom. C2 is a methylene bridge already connected to a hydroxyl group. C1 and C2 are then attached using the equivalence relation described in Eq. 3.2 to create C3, an ethanol molecule. By default, a `Bond` is created between the two anchoring carbons. Adapted with permission from Fig 2. in Sallai, J. et al. (2013) *Web- and Cloud-based Software Infrastructure for Materials Design*. Procedia Computer Science: Elsevier.

Notice that ports have directionality, as well. Consider *Component* C_1 in Fig. 3.2, representing a methyl group. It is not possible to create an ethane molecule from two such components, because the equivalence transform would render not just the untyped atoms in the ports, but also the carbon and hydrogen atoms to overlap. While one way of solving this problem would be to have two flavors of each such `Compound` class, one with an "outward pointing" Port, and another one with an "inward pointing" one, `mBuild` takes an alternate approach. The actual implementation of the `Port` class contains not four, but eight untyped atoms: four of them forming an "inward pointing", while the other four comprising an "outward pointing" collection of points. When performing an equivalence transform, `mBuild`, in fact, computes two affine transformation matrices, and choses the one that avoids the overlap of the compounds' typed atoms. This is achieved by checking which of the two transformations forces the anchor atoms as far away from one another as possible (see Fig. 3.1 for an illustration of how these quartets of `Particles` are arranged). Fig. 3.2 highlights this procedure via the construction of an ethanol molecule. Additional documentation is included at the development website (http://imodels.github.io/mbuild/)

via an interactive IPython notebook.[12]

3.3 Applications

Below, I highlight the basics of assembling low level components into successively more complex structures in `mBuild` and how to programmatically control these workflows to perform automated screening for monolayer systems. All the examples discussed below are also available as tutorials in IPython notebook format where users can seamlessly visualize components as they are constructed from Python code via a widget provided by the `imolecule` package.[6] Static versions of these notebooks are also hosted on our documentation page at http://imodels.github.io/mbuild/. Many additional example systems of varying complexity are also provided together with the `mBuild` source code on GitHub.

3.3.1 Defining and connecting basic components

The simplest way to define a basic component in `mBuild` is to draw the component using software such as Avogadro,[9] output it as a `.mol2` or `.pdb` file with defined bonds and then use the `load` function in `mBuild`. Adding a `Port` to a compound that a user wants to be able to connect to other compounds requires placing the `Port` where a bond could be formed and specifying an anchor particle with which the `Port` is associated. Just as with any other `Compound`, `Ports` can not only be translated but also rotated thus allowing non-linear arrangements to be constructed. This procedure is highlighted in Listing 3.1; basic components can be stored and reused for future system construction thus minimizing the need for users to place `Ports`, as will be demonstrated as part of the construction of alkane monolayers in the screening application below.

Listing 3.1: Example code to generate a CH_2 group and attach two ports.

```
1
2  ch2 = mb.load('ch2.pdb')
3  mb.translate(ch2, -ch2[0].pos) # Move carbon to origin.
4
5  port1 = mb.Port(anchor=ch2[0]) # Anchor the port on the carbon.
6  mb.translate(port1, [0, 0.07, 0]) # Approx. half a C-C bond
       length in nm.
7
8  port2 = mb.Port(anchor=ch2[0])
9  mb.translate(port1, [0, -0.07, 0]) # Placed on opposite side of
       carbon.
10
11 ch2.add(port1, label='up')
12 ch2.add(port2, label='down')
```

Any two `Ports` can be forced to overlap using the equivalence transform. Listing 3.2 demonstrates how this functionality can be leveraged via the simple yet common use case of creating an alkane polymer chain which will be used for screening - in this example, a CH_2 group with the ports "up" and "down" defined.

Listing 3.2: Example code for polymerizing CH_2 groups.

```
1  import mbuild as mb
2  from mbuild.lib.moieties import CH2
3
4  polymer = mb.Compound()
5  last_monomer = CH2()
6
7  polymer.add(last_monomer)
8  for _ in range(10):
9      this_monomer = mb.clone(last_monomer)
10     mb.equivalence_transform(this_monomer, this_monomer['up'],
           last_monomer['down'])
11     polymer.add(this_monomer)
12     last_monomer = this_monomer
```

To further simplify the composition of basic components into more complex structures, several classes and functions have been developed to more naturally express many commonly performed tasks. For example, the functionality of the example in Listing 3.2 is encapsulated within the `Polymer` class which reduces the above for loop to one line for end users. For example, the PEG chains referenced in the following examples, are created with the code in Listing 3.3.

Listing 3.3: Using the `Polymer` class to create PEG chains

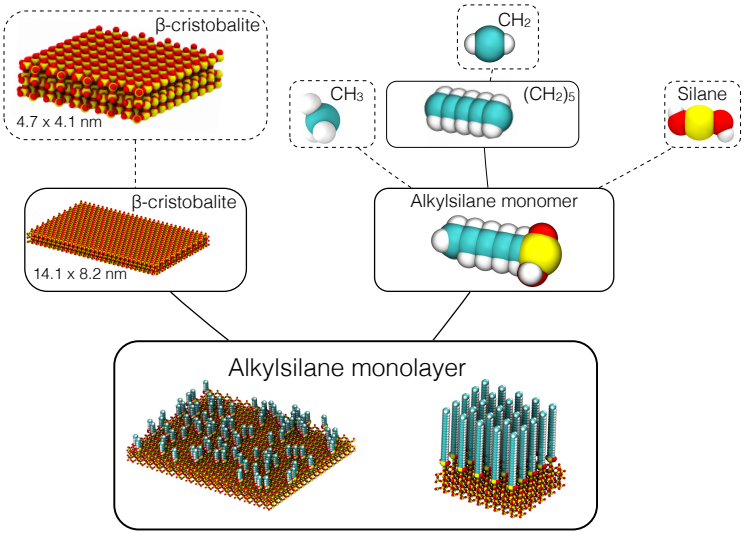
```
import mbuild as mb
from mbuild.lib.moieties import CCO

peg = mb.Polymer(CCO(), n=10)
```

3.3.2 Patterning Surfaces

`mBuild` provides functionality for patterning of surfaces in arbitrary ways. Below, I highlight this feature via the patterning of scientifically relevant 2D and 3D systems.

In the example shown in Fig. 3.3, the `TiledCompound` class is used to replicate a periodic substrate in the x- and y-dimensions. This class also internally adjusts periodic bonds. In the final tier of the hierarchy, the patterning functionality, which can be used to create patterns on, for example, substrates or spherical particles, is used to randomly disperse polymer brushes on the substrate. Functionality is provided in `mBuild` for a variety of 2D and 3D patterns including random, grid-like, disks and spherical patterns. Ultimately, a multi-tiered hierarchy of components is assembled, from simple "hand-drawn" monomers, through polymerization and replication of periodic substrates. This functionality is expressed with minimal code via creating a new Python class (shown at the bottom of Fig. 3.3) to expose the desirable tunable parameters. Here, the number of monomers in the chain, the number of chains on the surface, the pattern on the surface, and the size of the surface can all be trivially modified during screening.

The surface patterning illustrated in Fig. 3.3 was limited to a two-dimensional surface; however, the underlying functionality in `mBuild` naturally generalizes to three dimensions as well with essentially no changes to the user-level code. Fig. 3.4 and Listing 3.4 show how this could be used to functionalize a spherical nanoparticle with various polymer chains. The code utilized to attach chemical groups to two-dimensional systems can be reused for three dimensional structures without significant modification or further effort by the end user.




```
pattern = Random2DPattern(100)
chain_length = 5
n_tiles = [2, 3, 1]
```

```
pattern = Grid2DPattern(5, 5)
chain_length = 20
n_tiles = [1, 1, 1]
```

```
monolayer = AlkaneMonolayer(chain_length, pattern, n_tiles)
```

Figure 3.3: Hierarchy of compounds used to generate an alkylsilane monolayer on a β-cristobalite substrate. Dashed boxes indicate base components for which .mol2 or .pdb files exist, e.g. drawn using software such as Avogadro.[9] The code snippet used to generate the structures with all of the tunable parameters exposed is shown at the bottom for two different parameter combinations.

Listing 3.4: Example code to tether PEG chains to a silica nanoparticle.

```
import mbuild as mb
from mbuild.lib.surfaces import SilicaNP
from mbuild.examples import PEGSilane

peg_silane = PEGSilane(peo_units=5) # Length based on number of
    CCO moieties.
silica_np = SilicaNP()

pattern = mb.SpherePattern(25)
peg_chains, _ = pattern.apply_to_compound(guest=peg_silane,
    host=silica_np)

tnp = mb.Compound([silica_np, peg_chains])
```

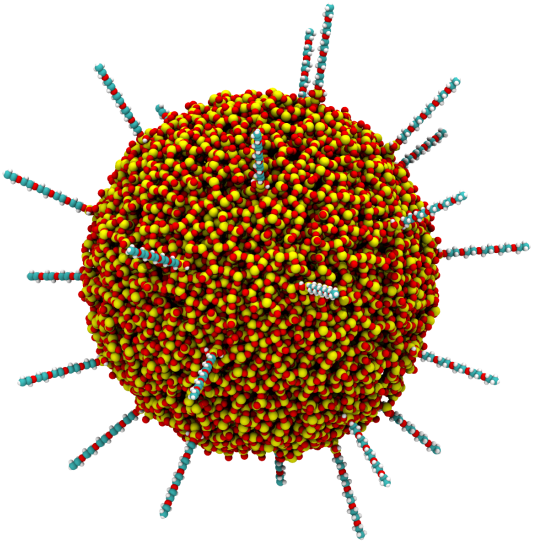


Figure 3.4: An 8 nm diameter silica nanoparticle sparsely functionalized with PEG chains bound to the surface by siloxane attachments.

3.4 Screening Soft Matter Systems: Self-assembled Monolayers

Building upon the prior examples, monolayers are constructed in a programmatic way to demonstrate the use of `mBuild` for screening applications. Monolayers encompass a vast chemical parameter space that can be tuned for applications such as lubrication[13] and anti-fouling,[14] and their behavior and properties often strongly depend on the substrate, binding moiety, chain type, composition of multiple chain types, surface patterning, etc. Sampling more than one or two dimensions of this parameter space using experimental techniques, while technically possible, quickly becomes limited by practical considerations. Molecular dynamics can be used as a screening step to inform subsequent experimental studies and dramatically cut down the relevant search space. Here, using `mBuild` substrate density, chain length, and chain type of monolayer systems are programmatically varied in order to perform a basic screening.

The first step to performing a screening procedure across chemical space involves building the input topologies. Ideally, a user should have seamless access to any variables of interest thus enabling them to adjust these to mimic a statistical distribution. As discussed

previously, the hierarchical nature of `mBuild` provides an avenue to expose an arbitrary set of variables to the end user and thus enables users to leverage the scientific Python ecosystem to apply standard optimization techniques and analysis to explore chemical parameter space. As highlighted above, in `mBuild`, the only explicit rotation and translation occurs in the lowest level of the hierarchy when placing ports. Once these simple components have been fitted with ports, they can be stored in the database for future use thus completely eliminating the need for explicit rotation and translation when building many systems; here, we reuse many of the components previously defined in the prior examples. Each higher tier in the hierarchy contains only a few lines of code to express which ports to connect to one another.

Listing 3.5 shows the `mBuild` code that generates configurations for a simple screening procedure of alkane and PEG monolayers on silica substrates. This code varies the chain length and the number of chains on the surface for both molecules types. It is important to note that the code to generate both monolayer types are nearly identical due to the hierarchical nature of `mBuild`; the `Monolayer` function is generic, as it simply expects a `Compound` with a `Port` defined for attachment. Thus it can readily accept either the `Alkane` or `PEG Compounds` (or mixture thereof) that have previously been define, where each of these `Compounds` accepts an argument to define the length of the desired polymer chain. As such, this example can be trivially extended by creating a different molecule `Compound`, and substituting this in place of either the `Alkane` or `PEG Compound`.

Fig. 3.5 illustrates two of the systems created using this procedure post-equilibration. In this example, the monolayers were patterned in a 2D grid but the patterning of the surface is also tunable if desired, as shown previously. Each monolayer that was created was sampled for 10 ns using GROMACS[15] and the OPLS-aa forcefield[16] with modifications as described by Lorentz et al.[17]

Listing 3.5: Example code to generate alkane and PEG monolayers differing in both chain length and number of surface grafted chains. Note that most of the code can be reused to create both the PEG and alkane monolayer; the only difference is the chain class that is instantiated.

```
import mbuild as mb
from mbuild.examples import Alkane, PEG
from mbuild.lib.atoms import H
from mbuild.lib.surfaces import Betacristobalite

hydrogen = H()
surface = Betacristobalite()

for sqrt_n_chains in range(4, 11): # Amount of chains on surface.
    pattern = mb.Grid2DPattern(sqrt_n_chains, sqrt_n_chains)

    for chain_length in range(6, 22, 3): # Length of chains on
        surface.
        chain = Alkane(chain_length) # Use alkane chains.
        monolayer = mb.Monolayer(surface, chain, pattern,
            backfill=hydrogen)
        monolayer.save('{}_{}_alkane.mol2'.format(sqrt_n_chains**2,
            cl))

        chain = PEG(peo_units=chain_length / 3) # Use PEG chains.
        monolayer = mb.Monolayer(surface, chain, pattern,
            backfill=hydrogen)
        monolayer.save('{}_{}_peg.mol2'.format(sqrt_n_chains**2,
            cl))
```

Fig. 3.6 shows the average nematic order parameter, $S2$, of the chains on monolayer.[18,19] $S2$ measures the orientational ordering of the chains, where for monolayers, values below 0.7 indicate a fluid-like state (i.e., low order) whereas values that approach unity indicate a high degree of crystalline orientational ordering. It has been shown that $S2$ influences the frictional properties of monolayers, where lower values of $S2$ for monolayers tend to be correlated with higher frictional forces when the monolayers are brought together in sliding contact.[20] Thus $S2$ serves as a useful surrogate for rapidly screening monolayers to determine which regimes are likely to produce high/low coefficients of frictions. While additional simulations and sampling are required to draw more robust conclusions, several regimes are readily apparent. A clear transition from disordered, fluid-like monolayer

states to ordered states occurs for both systems. This transition occurs at lower surface coverages for alkane chains as compared to PEG chains. That is, PEG systems appear to have a smaller regime of well order states, which can be accounted for by the increased flexibility in PEG. In both cases, systems with the highest values of $S2$ tend to occur for higher surface densities and longer chains, and thus one would expect materials in these regimes to demonstrate the most favorable frictional properties. Interestingly, these screening simulations also reveal a second regime for PEG occurring for low surface coverage and short chain length; in this regime moderate values of $S2$ are observed, which, upon visual inspection, appears associated with chains lying flat along the surface. The ability to rapidly screen, evaluate and cross-correlate metrics like the nematic order parameter will accelerate our ability to rationally design soft materials in complex parameter landscapes.

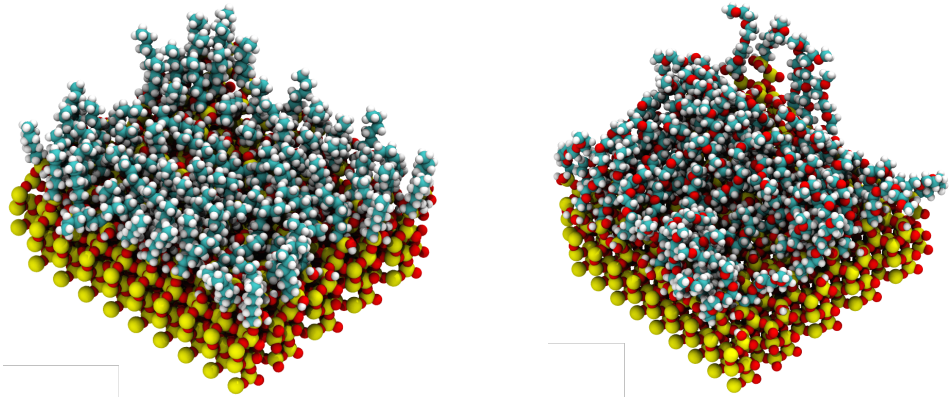


Figure 3.5: An alkane system with 81 chains with 7 carbons each (left) and a PEG system with 64 chains and 13 carbons/oxygens (right). Both shown post-equilibration.

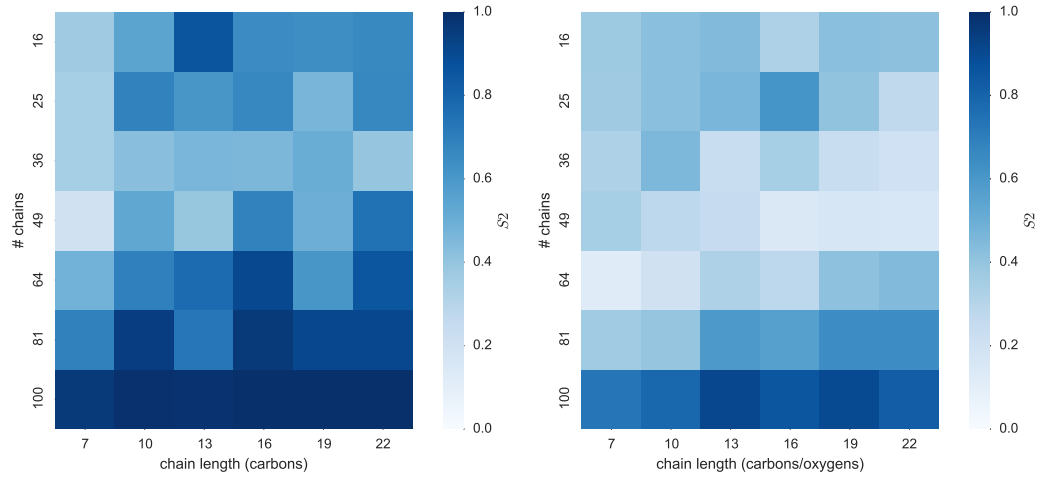


Figure 3.6: Average nematic order parameter of every system after 10 ns of sampling. The total process of constructing all 84 systems with `mBuild` takes a few minutes on a modern laptop and the simulations each take approximately 0.5-3 hours depending on system size using a GTX980 and 8 CPU cores of an Intel Xeon E5 2600v3.

3.5 Bibliography

[1] Sallai, J.; Varga, G.; Toth, S.; Iacovella, C.; Klein, C.; McCabe, C.; Ledeczi, A.; Cummings, P. T. *Procedia Computer Science* **2014**, *29*, 2034–2044.

[2] Klein, C.; Sallai, J.; Jones, T. J.; Iacovella, C. R.; McCabe, C.; Cummings, P. T. *Foundations of Molecular Modeling and Simulation*; Springer Singapore, 2016; pp 79–92.

[3] Iacovella, C. R.; Klein, C.; Sallai, J.; Ismail, A. E. *Introduction to Scientific and Technical Computing* **2016**, 261.

[4] Abbott, L. J.; Hart, K. E.; Colina, C. M. *Theoretical Chemistry Accounts* **2013**, *132*, 1–19.

[5] McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. *Biophysical Journal* **2015**, *109*, 1528–1532.

[6] Fuller, P. Available at https://github.com/patrickfuller/imolecule.

[7] Eastman, P. et al. *Journal of Chemical Theory and Computation* **2013**, *9*, 461–469.

[8] Anderson, J. A.; Lorenz, C. D.; Travesset, A. *Journal of Computational Physics* **2008**, *227*, 5342–5359.

[9] Hanwell, M. D.; Curtis, D. E.; Lonie, D. C.; Vandermeersch, T.; Zurek, E.; Hutchison, G. R. *Journal of cheminformatics* **2012**, *4*, 17.

[10] Http://www.whitehouse.gov/mgi,

[11] Aric Hagberg, P. S., Dan Schult Available at https://networkx.github.io/.

[12] Pérez, F.; Granger, B. E. *Computing in Science and Engineering* **2007**, *9*, 21–29.

[13] Bhushan, B.; Israelachvili, J. N.; Landman, U. *Nature* **1995**, *374*, 607–616.

[14] Brzoska, J. B.; Shahidzadeh, N.; Rondelez, F. *Nature* **1992**, *360*, 719–721.

[15] Pronk, S.; Páll, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; Van Der Spoel, D.; Hess, B.; Lindahl, E. *Bioinformatics* **2013**, *29*, 845–854.

[16] Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *Journal of the American Chemical Society* **1996**, *118*, 11225–11236.

[17] Lorenz, C.; Webb, E.; Stevens, M.; Chandross, M.; Grest, G. *Tribology Letters* **2005**, *19*, 93–98.

[18] Lagomarsino, M. C.; Dogterom, M.; Dijkstra, M. *J. Phys. Chem.* **2003**, *119*, 719–721.

[19] Wilson, M. R. *Journal of Molecular Liquids* **1996**, *68*, 23 – 31.

[20] Black, J. E.; Iacovella, C. R.; Cummings, P. T.; McCabe, C. *Langmuir* **2015**, *31*, 3086–3093.

CHAPTER 4

FORMALIZING FORCEFIELD ATOMTYPING AND APPLICATION FOR MOLECULAR SIMULATION

This chapter describes our efforts to simplify the process of atom-typing as well as applying and disseminating force fields in a more reproducible and transparent manner. These efforts are implemented in the package `Foyer` which enables users to encode atom-type definitions that are both human and machine readable directly into the file that encodes the force field parameters. Details of the force field format and atom-typing procedure are described as well as efforts to expose this functionality in a community friendly way to encourage other researchers to leverage `Foyer`'s capabilities when disseminating their force field development efforts.

4.1 Introduction

The availability of forcefields for molecular simulation has reduced the effort researchers must devote to the difficult and costly task of determining the interactions between species, allowing them to instead focus on the motivating scientific questions. However, determining which parameters in a forcefield to use is still often a tedious and error prone task. This difficulty is related to the strong dependence of the parameters on the chemical context of the atoms; the chemical context may depend, among other factors, on the local bonded environment of an atom in a molecule, the local environment of neighboring atoms and the type of molecule(s) being considered. Forcefields can contain tens or hundreds of different types for the same element, where each type represents the element in a different chemical context. Atom typing can be challenging, often requiring the user to consult textual comments scattered in parameter files or the scientific literature where the parameters were

published. Unfortunately, as of today, the documentation of many forcefields tends to be scarce and unstructured, commonly expressed in plain English or in an ad-hoc shorthand notation, leading to ambiguities and increasing the likelihood of incorrect usage. While there are freely available tools to aid in atom-typing, these are typically specific to a particular forcefield or simulator and capture the atom-typing and parameterization rules in ways that are hard to maintain, debug, and evolve. Here we propose a forcefield agnostic formalism to express atom-typing and parameterization rules in a way that is expressive enough for human consumption, while being machine readable to enable automation in complex scientific workflows.

Classical molecular simulations rely on forcefields to describe the various interactions that exist between atoms and/or groups of atoms, including non-bonded interactions (van der Waals and electrostatics) and bonded interactions (bonds, angles, and torsions). These forcefields are typically expressed as a set of analytical function with adjustable fitting parameters for different atomic/molecular species. Considerable efforts have been undertaken by many research groups to develop accurate forcefields, both in determining the mathematical functions and associated fitting parameters, for a large variety of molecular species under different conditions. Numerous forcefields (and associated parameters), have been devised including: AMBER,[1] CHARMM,[2] OPLS,[3] SKS,[4] TraPPE,[5] COMPASS[6] and GROMOS.[7] The availability of these forcefields can significantly reduce or completely eliminate the difficult and costly task of determining the interactions between species, allowing researchers to instead focus their efforts on the motivating scientific questions.

However, while researchers do not necessarily need to spend time developing the forcefield parameters, determining *which* parameters to use (*i.e.*, atom-typing) is still often a tedious and error prone task. In many forcefields, the appropriate interaction parameters will depend on the local topological environment of atoms. For example, the non-bonded interactions of carbon atoms in *n*-alkanes are typically different for terminal carbons verses "middle" carbons.[8] Similarly, when identifying the correct parameters for a torsional term,

one must typically consider not just the backbone, (*e.g.*, C-C-C-C), but also the bonds of each atom in the backbone (*e.g.*, CH_3-CH_2-CH_2-CH_3, *vs.* CF_3-CF_2-CF_2-CF_3).[3,9] Consequently, a given forcefield may include multiple unique parameters for a given atom, whose usage depends on the *chemical context* of the atom. That is, the appropriate parameters will typically depend on a number of factors, including the specific molecule in which the atom appears (*e.g.*, alkanes *vs.* perfluoroalkanes), where in that molecule the atom appears (*e.g.*, terminal *vs.* middle), the system type being studied (*e.g.*, hydrated lipid bilayer *vs.* bulk fluid), and the statepoint/phase of the system (*e.g.*, parameters may differ for liquid *vs.* vapor). As a clear illustration of this, we note that the OPLS all-atom forcefield parameter database (as provided in the TINKER molecular modeling software[10]) contains 427 different "types" of carbon atoms, which are all differentiated based on their chemical context. Furthermore, determining which of the multitude of forcefield parameters is most appropriate, based on the chemical context of an atom, is often accomplished through brief, unstructured – and sometimes ambiguous – annotations located in the forcefield parameter files. Even journal articles associated with forcefield parameters can be unclear, where parameters are typically listed in table form, often with limited annotations or examples of their usage. Also, given their static nature, parameters provided in journal articles may not be the most up-to-date, whether resulting from typographic errors or modifications in later work. As a result, for many forcefields, the formal logic required to distinguish atom types may be difficult to find, difficult to interpret, out-of-date, or simply ambiguous. This ambiguity is confounded by the fact that many published journal articles that rely on forcefields often provide only vague citations for the source of the parameters. A few forcefields, such as GAFF, CGenFF and more recently OPLS, do provide parameterization tools that serve as the reference implementation and thus implicitly define atom types. However, to our knowledge there does not exist a generic tool that simultaneously provides a human and machine readable mechanism of force field distribution that is agnostic to the forcefield that it defines.

Many research groups use some combination of hand parameterization and logic based codes to facilitate the process of atom-typing. Identifying atom types by hand, while easy for small/simple molecules and important for validation, becomes impractical for large molecules or systems with significant heterogeneity. For molecules with only a few atomic species and little variability, *ad hoc*, logic-based codes are relatively straightforward to write and test. However, as the number of unique atom types increases, properly defining the appropriate if/else statements required in most logic codes becomes onerous, even for people who are well versed in programming and have domain expertise in molecular modeling. Also, nested logic statements are often difficult to adequately test and debug and thus introduce a potential source of error. Since *ad hoc* codes are not typically released to the general community, any errors in the code may go undiscovered and thus data based on flawed atom-typing may appear in published scientific literature. To address this, several community tools and approaches have been developed to aid in atom-typing,[11–16] some of which are more generally applicable than others. Forcefields developed in the biophysics community tend to have exceptionally well vetted parameterization codes, such as AMBER's antechamber,[17] but such tools are typically designed to only work with their associated forcefield and may also produce output specific to a given simulation package, rather than a general form. In general, these tools all rely on a hierarchy where rules that identify more specialized atom types must be called in precise order, such that more general atom types are only chosen when more specialized matches do not exist (*i.e.*, they include rule precedence). Maintaining, let alone constructing, these hierarchies is extremely error prone and, just as in *ad hoc* codes, typically results in source code with deeply nested if/else statements. In these hierarchical schemes, in order to add a new atom type or correct an error, a developer must have a complete picture of the hierarchy and know exactly where the relevant rule should be placed, such that it does not inadvertently override other rules. This may impose practical limits on functionality, where, for example, a user is not able to easily extend the rules to include newer parameters, or that such attempts to extend rules

result in incorrect atom-typing for other molecular species.

To address these issues, we developed a framework for atom-typing, based upon first order logic over graph structures. This logic is implemented via SMARTS strings which allow us to embed the atomtype definition in the force field parameter file as opposed to the source code of an atomtyping tool. Additionally, our method of resolving atomtypes differs slightly from most tools in that it is iterative and does not rely on rigid rule hierarchies. In combination, these features simplify the force field development and dissemination processes which is illustrated via a force field recently developed in our group for perfluoropolyethers.

4.2 File Format

Force fields are stored in the OpenMM XML format which is extensively documented (http://docs.openmm.org/7.0.0/userguide/application.html#creating-force-fields). To briefly summarize, atomtypes and forces are encoded as XML tags with various attributes defining the types of atoms (by name only) that they apply to as well as the associated parameters for that interaction (e.g. the equilibrium bond length and spring constant for a harmonic bond). The flexible nature and ubiquitous adoption of the XML format makes it both easily extensible and easy to work with given the plethora of high quality XML parsers. It also enables us to perform additional validation through the use of XML schemas which allow us to perform sanity checks on any force field file used by `Foyer`.

4.3 SMARTS Based Atom-Typing

`Foyer` allows users to describe atomtypes using a modified version of SMARTS (). SMARTS is a straightforward extension of the more commonly used SMILES () notation which provides additional tokens that enable users to express further chemical detail.

4.3.1 Basic Usage

Consider the following example defining the OPLS-AA atomtypes for a methyl group carbon and its hydrogen atoms:

Listing 4.1: Atomtype definitions for a methyl carbon and its hydrogen atoms

```
<ForceField>
 <AtomTypes>
  <Type name="opls_135" class="CT" element="C" mass="12.01100"
     def="[C;X4](C)(H)(H)H" desc="alkane CH3"/>
  <Type name="opls_140" class="HC" element="H" mass="1.00800"
     def="H[C;X4]" desc="alkane H"/>
 </AtomTypes>
</ForceField>
```

The above example utilizes two additional XML attributes supported by `Foyer`: `def` and `desc`. The atomtype that we are attempting to match is always the first token in the SMARTS string, in the above example, `[C;X4]` and `H`. The `opls_135` (methyl group carbon) is defined by a SMARTS string indicated by a carbon with 4 bonds, a carbon neighbor and 3 hydrogen neighbors. The `opls_140` (alkane hydrogen) is defined simply as a hydrogen atom bonded to a carbon with 4 bonds.

4.3.2 Overriding Atomtypes

When multiple atomtype definitions can apply to a given atom, we must establish precedence between those definitions. Many other atomtypers determine rule precedence by placing more specific rules first and evaluate those in sequence, breaking out of the loop as soon as a match is found.

While this approach works, it becomes more challenging to maintain the correct ordering of rules as the number of atomtypes grows in particular if one is making a small addition to a larger force field. Instead, `Foyer` iteratively runs all rules on all atoms and each atom maintains a whitelist (rules that apply) and a blacklist (rules that have been superceded by another rule). The set difference between the white- and blacklists yields the correct atomtype if the force field is implemented correctly.

We can add a rule to a blacklist using the `overrides` attribute in the XML file. To illustrate an example where overriding can be used consider the following types describing alkenes and benzene:

Listing 4.2: Atomtype definitions for alkenes and benzene highlighting the overrides syntax and mechanism for referencing other atomtypes

```
<ForceField>
 <AtomTypes>
  <Type name="opls_141" class="CM" element="C" mass="12.01100"
     def="[C;X3](C)(C)C" desc="alkene C (R2-C=)"/>
  <Type name="opls_142" class="CM" element="C" mass="12.01100"
     def="[C;X3](C)(C)H" desc="alkene C (RH-C=)"/>
  <Type name="opls_144" class="HC" element="H" mass="1.00800"
     def="[H][C;X3]" desc="alkene H"/>
  <Type name="opls_145" class="CA" element="C" mass="12.01100"
     def="[C;X3;r6]1[C;X3;r6][C;X3;r6][C;X3;r6][C;X3;r6][C;X3;r6]1"
     overrides="opls_141,opls_142"/>
  <Type name="opls_146" class="HA" element="H" mass="1.00800"
     def="[H][C;%opls_145]" overrides="opls_144" desc="benzene
     H"/>
 </AtomTypes>
</ForceField>
```

When atomtyping a benzene molecule, the carbon atoms will match the SMARTS patterns for both `opls_142` and `opls_145`. Without the `overrides` attribute, `Foyer` will notify you that multiple atomtypes were found for each carbon. Providing the `overrides` indicates that if the `opls_145` pattern matches, it should supercede the specified rules.

4.3.3 Implementation of SMARTS Matching

The actual implementation of the SMARTS based atomtyping scheme is broken down into several steps and internally relies on a subgraph isomorphism to detect matches as highlighted in Figure 4.1. First, a SMARTS string is parsed into an abstract syntax tree (AST) from which we populate a `SMARTSGraph` object. This class inherits from the `Graph` class in the `NetworkX` package[18] - the de facto standard network analysis library in Python. Atoms in this `SMARTSGraph` are represented as nodes and chemical bonds as edges. Inheriting from `NetworkX` is convenient in that it allows us to leverage most of the

algorithms and visualization methods already implemented there. The primary distinguishing feature of the `SMARTSGraph` is the set of methods that encode the logic for matching the more complex SMARTS tokens. These methods can be directly used by `NetworkX`'s implementation of the VF2 subgraph isomorphism algorithm.[19]

A thin wrapper provided by the `find_matches` method allows a `SMARTSGraph` instance to search for all subgraph isomorphisms within a bare chemical topology (an unatomtyped graph of just elements and bonds). This method returns the indices of all atoms that match the first token in the SMARTS string - the atomtype that we are looking for. Successfully matching atoms have the atomtype definition added to their whitelist and any overridden types added to their blacklist.

4.4 Validation of Force Field Files

`Foyer` performs several validation steps to help prevent malformed force field files and SMARTS strings from making it into production code. The goal is to provide human readable error messages that users who may not be intimately familiar with XML or our SMARTS parsing grammar can easily act upon.

4.4.1 XML Schema

As a first line of defense, any force field file loaded by `Foyer` is validated by an XML schema definition. Here we enforce which elements (e.g. `HarmonicBondForce`) are valid and how their attributes should be formatted. Additionally, the schema ensures that atomtypes are not 1) defined more than once and that 2) atomtypes referenced in other sections are actually defined in the `<AtomTypes>` element.

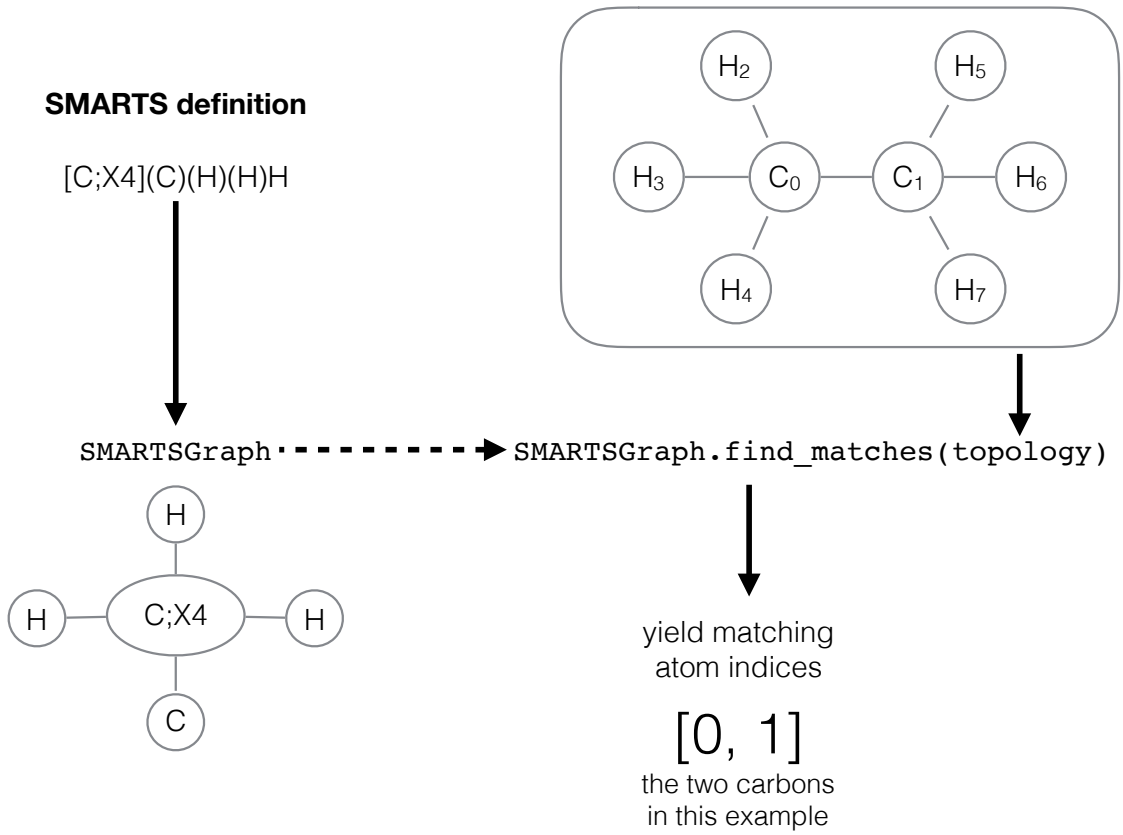


Figure 4.1: Schematic of the workflow to apply SMARTS patterns to chemical topologies. The SMARTS strings used to define atomtypes are read into a `SMARTSGraph` class which inherits from `NetworkX`'s core data structure. Using the `find_matches` method, a `SMARTSGraph` instance can search for subgraph isomorphisms of itself within a provided chemical topology and will yield all atoms that match the first token in the original SMARTS string - the atomtype that we are looking for.

4.4.2 SMARTS Validation

All SMARTS strings used to define atomtypes are parsed. Parsing errors are captured and re-raised with error messages that allow you to pin point the location of the problem in the XML file and within the SMARTS string. Wherever possible, we attempt to provide helpful hints and we welcome any contributions that help improve the clarity of our error messages.

Additionally, we ensure that any atomtypes referenced using the `%type` or `overrides` syntax are actually defined in the `<AtomTypes>` element.

4.5 Usage Examples

`Foyer` supports atomtyping of both all-atom and coarse-grained molecular systems, and also allows for separate force fields to be used to atomtype separate portions of a molecular system.

4.5.1 Parameterizing a Simple System

In Listing 4.4, we use mBuild to construct a box filled with ethane molecules and use `Foyer` to atomtype the system, applying the OPLS force field, and save to run-able GROMACS files.

Listing 4.3: Script to fill a box with ethane and apply the OPLS-AA force field to it

```
import mbuild as mb
from mbuild.examples import Ethane

ethane_box = mb.fill_box(compound=Ethane(), n_compounds=100,
    box=[2, 2, 2])
ethane_box.save('ethane-box.gro')
ethane_box.save('ethane-box.top', forcefield_name='oplsaa')
```

4.5.2 Support for Coarse-Grained Force Fields

`Foyer` also supports atomtyping for coarse-grained force fields. Representing non-atomic species is not immediately possible using standard SMARTS due to the its reliance on the presence of an element specification for each atom. To circumvent this limitation, `Foyer` allows users to define custom ”elements” by prefixing their string representation with an underscore, e.g. `_CCC` could represent a coarse-grained bead intended to represent three carbon atoms. In its current implementation, `Foyer` makes a first pass through force field files to detect any custom element definitions. Those that are detected are injected into the grammar that parsers SMARTS strings and are given priority over standard elements. An additional quirk of this methodology is that care must be given to properly arrange all custom elements in a way such that element names that are substrings of other element

names must appear later in the grammar. For example, the definition for _CH2 would need to be injected into the grammar before _CH since the greedy nature of the parser would simply stop after the H character and identify _CH. This nuance is handled automatically and should not concern any end-users.

In practice, coarse-grained force fields can be used almost identically as all-atom force fields. Listing 4.4 shows how one can define a united-atom description of ethane and apply the TraPPE force field during atom-typing. Note how particle names are prefixed with an underscore so that Foyer knows these particles are non-atomistic.

Listing 4.4: Script to fill a box with coarse-grained ethane and apply the TraPPE force field to it

```
import mbuild as mb

ethane_UA = mb.Compound()
ch3_1 = mb.Particle(name='_CH3', pos=[0, 0, 0])
ch3_2 = mb.Particle(name='_CH3', pos=[0.15, 0, 0])
ethane_UA.add([ch3_1, ch3_2])
ethane_UA.add_bond((ch3_1, ch3_2))

ethane_UA_box = mb.fill_box(ethane_UA, 100, box=[2, 2, 2])
ethane_UA_box.save('ethane-UA-box.gro')
ethane_UA_box.save('ethane-UA-box.top',
   forcefield_name='trappe-ua')
```

4.5.3 Combining Force Fields

In some instances, the multiple force fields may be required to describe a molecular system. For example, the user may want to apply one force field for a surface and another for a fluid in the same system. Foyer supports this functionality by allowing the user to separately atomtype parts of a system. In this example, we take a system featuring bulk united atom ethane above a silica surface and apply the OPLS force field to the surface and the TraPPE force field to the ethane. The two atomtyped Parmed structures are then combined using a simple + operator and saved to any format supported by ParmEd.

Listing 4.5: Script to build a system of ethane adsorbed onto an amorphous silica substrate and apply a different force field to the substrate and fluid respectively

```
from foyer import Forcefield
from foyer.test.utils import get_fn
import mbuild as mb
from mbuild.examples import Ethane
from mbuild.lib.atoms import H
from mbuild.lib.bulk_materials import AmorphousSilica

silica = mb.SilicaInterface(bulk_silica=AmorphousSilica())
silica = mb.Monolayer(surface=silica, chains=H(),
   guest_port_name='up')

box = mb.Box(mins=[0, 0, max(silica.xyz[:,2])],
          maxs=silica.periodicity + [0, 0, 4])

ethane_box = mb.fill_box(compound=Ethane(), n_compounds=200,
   box=box)

opls = Forcefield(name='oplsaa')
opls_silica =
   Forcefield(forcefield_files=get_fn('opls-silica.xml'))
ethane_box = opls.apply(ethane_box)
silica = opls_silica.apply(silica)

system = silica + ethane_box

system.save('ethane-silica.gro')
system.save('ethane-silica.top')
```

4.6 Promoting Reproducible Force Field Dissemination

The fact that the force field parameters and its usage semantics are encoded within the same, text based document allows it to be easily version controlled as one would any other piece of software. The now ubiquitous git+github/bitbucket based distribution process for software development, allows force field developers to disseminate their force field files to a public, version controlled repository that can be referenced from relevant publications. Additionally, a variety of other features of this standard software development process translate nicely to the force field development process. Revisions can easily be made to the force field as it evolves or is corrected in a transparent number. Whenever the developers

wish to, they can create a new release of the force field that generates a new, citable DOI. Verification and validation is also simplified by the use of automated testing tools that can perform checks on every new iteration of the force field content.

To promote these practices, we have created a template repository which contains all the basics required to create, test and publish a new force field as well as a guided tutorial that introduces users to the SMARTS based atomtyping scheme (https://github.com/mosdef-hub/forcefield_template). This process was successfully used in recent work that derived force field parameters for perfluoropolyethers,[20] a novel lubricant class. The force field was published in conjunction with the manuscript and made freely available on github (https://github.com/mosdef-hub/forcefield_perfluoroethers). The specific version of the force field at time of publication is citable via a separate DOI.[21] Any adjustments or improvements to the force field could now be released under a new DOI while the old one would still exist and point to the originally published force field in order to maintain provenance. Automated testing assert upon each modification that the force field file is correctly formatted and that the SMARTS definitions continue to accurately resolve a range of chemical topologies pertinent to this force field.

4.7 Bibliography

[1] Weiner, S. J.; Kollman, P. A.; Case, D. A.; Singh, U. C.; Ghio, C.; Alagona, G.; Profeta, S.; Weinerl, P. *Journal of the American Chemical Society* **1984**, *106*, 765–784.

[2] MacKerell, A. D.; Banavali, N.; Foloppe, N. *Biopolymers* **2000**, *56*, 257–65.

[3] Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *Journal of the American Chemical Society* **1996**, *118*, 11225–11236.

[4] Siepmann, J. I.; Karaborni, S.; Smit, B. Simulating the critical behaviour of complex fluids. 1993.

[5] Potoff, J. J.; Siepmann, J. I. *AIChE Journal* **2001**, *47*, 1676–1682.

[6] Sun, H. *Journal of Physical Chemistry* **1998**, *5647*, 7338–7364.

[7] Oostenbrink, C.; Villa, A.; Mark, A. E.; Van Gunsteren, W. F. *Journal of Computational Chemistry* **2004**, *25*, 1656–1676.

[8] Martin, M. G.; Siepmann, J. I. **1998**, *5647*, 2569–2577.

[9] Watkins, E. K.; Jorgensen, W. L. *The Journal of Physical Chemistry A* **2001**, *105*, 4118–4125.

[10] Ponder, J. W. TINKER Molecular Modeling Software. http://dasher.wustl.edu/tinker/.

[11] Bush, B. L.; Sheridan, R. P. *Journal of Chemical Information and Computer Sciences* **1993**, *33*, 756–762.

[12] Schüttelkopf, A. W.; Van Aalten, D. M. F. *Acta Crystallographica Section D: Biological Crystallography* **2004**, *60*, 1355–1363.

[13] Ribeiro, A. A. S. T.; Horta, B. A. C.; De Alencastro, R. B. *Journal of the Brazilian Chemical Society* **2008**, *19*, 1433–1435.

[14] Malde, A. K.; Zuo, L.; Breeze, M.; Stroet, M.; Poger, D.; Nair, P. C.; Oostenbrink, C.; Mark, A. E. *Journal of Chemical Theory and Computation* **2011**, *7*, 4026–4037.

[15] Vanommeslaeghe, K.; MacKerell, a. D. *Journal of chemical information and modeling* **2012**, *52*, 3144–54.

[16] Yesselman, J. D.; Price, D. J.; Knight, J. L.; Brooks, C. L. *Journal of computational chemistry* **2012**, *33*, 189–202.

[17] Wang, J.; Wang, W.; Kollman, P. a.; Case, D. a. *Journal of molecular graphics & modelling* **2006**, *25*, 247–60.

[18] Schult, D. A.; Swart, P. Exploring network structure, dynamics, and function using NetworkX. Proceedings of the 7th Python in Science Conferences (SciPy 2008). 2008; pp 11–16.

[19] Cordella, L. P.; Foggia, P.; Sansone, C.; Vento, M. *IEEE transactions on pattern analysis and machine intelligence* **2004**, *26*, 1367–1372.

[20] Black, J. E.; Silva, G. M.; Klein, C.; Iacovella, C. R.; Morgado, P.; Martins, L. F.; Filipe, E. J.; McCabe, C. *The Journal of Physical Chemistry B* **2017**,

[21] Black, J.; Silva, G.; Klein, C.; Iacovella, C.; Morgado, P.; Martins, L.; Filipe, E.; McCabe, C. OPLS-AA compatible parameters for perfluoroethers. 2017; https://doi.org/10.5281/zenodo.583310.

CHAPTER 5

SCREENING THE CHEMICAL PARAMETER SPACE OF SELF-ASSEMBLED MONOLAYERS

This chapter describes a screening study across the chemical parameter space of self-assembled monolayers. Using the toolchain described in Chapters 3&4, a pipeline is developed to rapidly screen self-assembled monolayers across alkane and PEG monomers with varying chain lengths and head groups on both crystalline and amorphous silica substrates. Our results suggest that the added flexibility of PEG chains slightly decreases coefficient of friction on more realistic, amorphous substrates and even though it decreases monolayer ordering. On rougher substrates, the added flexibility provides more fluid pathways for frictional dissipation when surfaces slide past one another. Additionally, the automated procedure used to screen these monolayers revealed several quirks in the simulation setup that further strengthen the case for using realistic, amorphous silica substrates to model self-assembled monolayers. While crystalline silica substrates are frequently used in simulation studies for convenience, our results demonstrate that these idealized systems can result in anomalous behavior related to lateral vibrations during shearing.

5.1 Introduction

Self-assembled monolayers present an appealing platform for creating fine tuned lubrication schemes for micro- and nanoelectromechanical devices (MEMS and NEMS). In addition to the choice of substrate and its roughness, monomers can vary in length as well as functional group and mixtures of monomers further expand the chemical parameter space of possible materials. This vast chemical space, while appealing from a design perspective, is not practical to significantly explore experimentally. Non-equilibrium molecular dynam-

ics simulations provide a convenient and cheap pathway for exploring these chemistries by providing trend level analyses of friction coefficients and additionally providing molecular level insights into the structure and dynamics of the monolayers as they are subjected to shearing.

Using the tools described in Chapters 3&4 in conjunction with the workflow manager Signac, a pipeline is developed that allows users to tune the chemistry of the monolayers in mBuild, use Foyer to apply the forcefield and then execute a series of equilibration and compression steps before finally shearing the monolayers and extracting the desires data from the results. This pipeline is freely available on Github and allows for full reproducibility of the simulations discussed below.

5.2 Simulation Details

Initial configurations of all monolayers are created using `mBuild` and the OPLS-AA force field is applied using `Foyer`. The procedure to construct the chemical topology, apply the force field and save it to GROMACS input files was wrapped in a simple template function which can be passed the desired parameters of interest. Here, the tuned parameters were 1) chain length (7, 10, 13, 16 and 19 heavy atoms), 2) chain type (alkane, methyl terminated PEG or hydroxyl terminated PEG) and 3) surface type (amorphous silica or β-cristobalite).

Simulations were run in GROMACS 5.0.[1] Initial configurations were minimized using a steepest decent procedure and then equilibrated for 1 ns in the NVT ensemble at 298.15 K using a Berendsen thermostat. Next, a constant normal (z-directed) force was applied to the bottom surface while the lowest layer of atoms was fixed in the x- and y-dimensions. This compression was run for 1 ns at which point the two surfaces had made contact. Lastly, the top surface was moved at a constant velocity of 10 m/s in the x-direction for 50 ns using a 1 fs timestep and velocity rescaling to maintain a temperature of 298.15 K. Lennard-Jones interactions were modeled with a cutoff of 0.9 nm and a switching function between 0.9

and 1.0 nm. Electrostatic interactions were modeled with a real-space cutoff of 1 nm and the particle-mesh Ewald algorithm. Subsequent trajectory analysis was performed using the GROMACS analysis tools and MDTraj.[2]

5.3 Friction Measurements

Figures 5.1-5.6 show the effects of chain length and normal load on the friction force experienced by the different monolayer chemistries. The alkane systems on crystalline substrates exhibit anomalously high shear forces which are further discussed in Section 5.5. With the exception of the crystalline alkane system, the differences across system chemistries are not dramatic. However, PEG systems on the more realistic amorphous substrates, Figures 5.4 & 5.6, have slightly lower friction coefficients than their alkane counterpart. On amorphous substrates, the more flexible PEG backbone allows for a more fluid shear response thus lowering the energy barriers required of the chains to circumnavigate the roughness of the opposing monolayer surface.

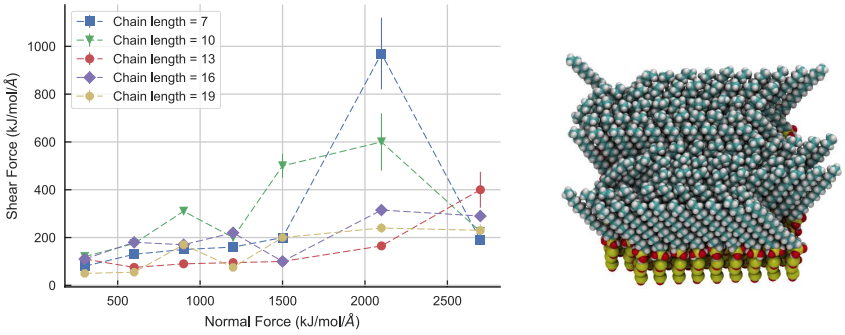


Figure 5.1: Shear force as a function of normal force for alkanes on a crystalline substrate.

5.4 Impact of Chemistry on Monolayer Order

The ordering of self-assembled monolayers impacts both the frictional forces experienced by the SAMs[3] and the stability of the monolayer.[4] Figure 5.7 shows the nematic order parameter of all the systems considered here averaged over all normal forces applied

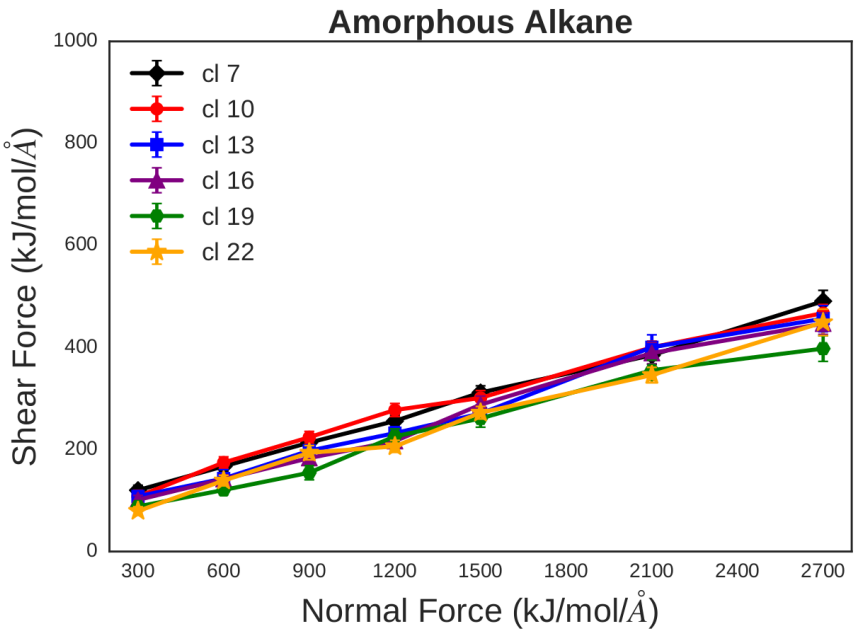


Figure 5.2: Shear force as a function of normal force for alkanes on an amorphous substrate.

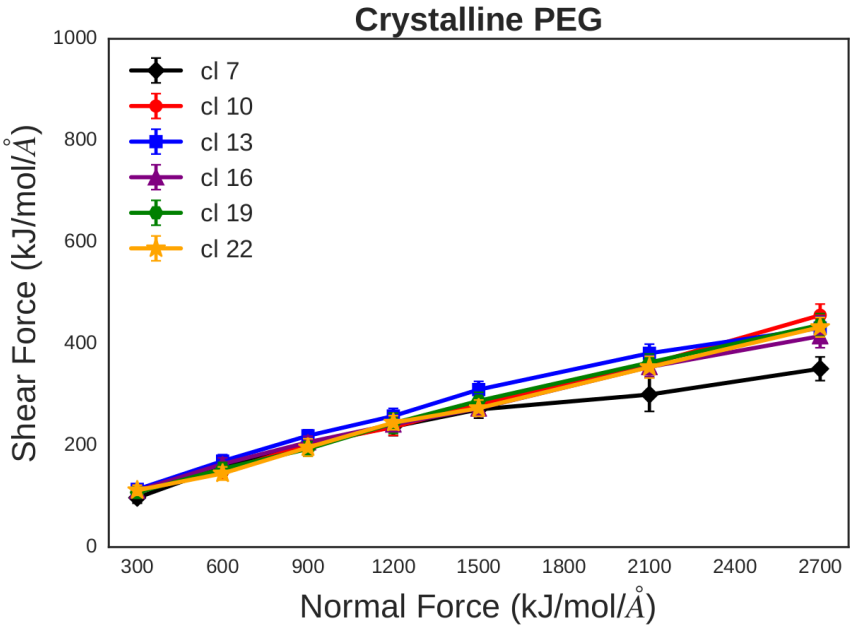


Figure 5.3: Shear force as a function of normal force for PEG chains terminated with a methyl group on a crystalline substrate.

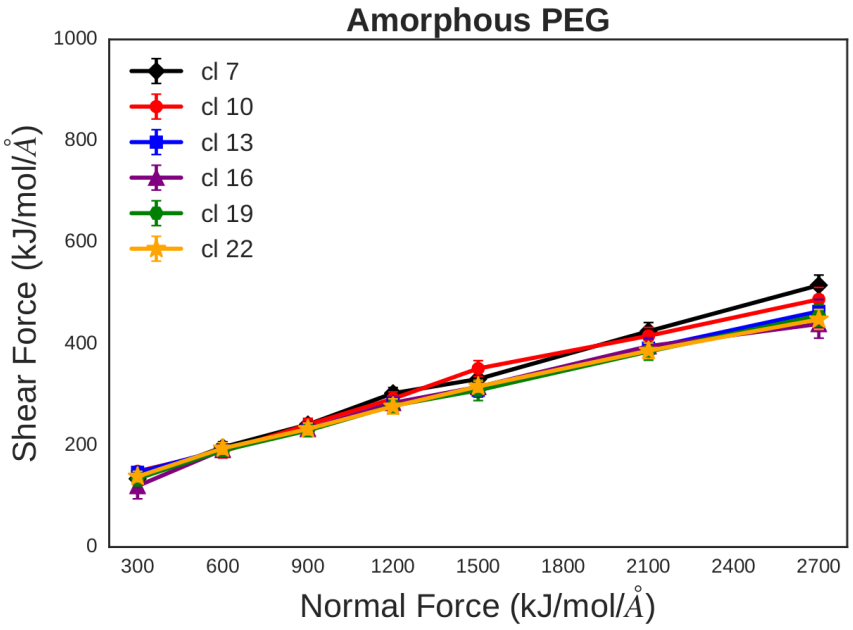


Figure 5.4: Shear force as a function of normal force for PEG chains terminated with a methyl group on an amorphous substrate.

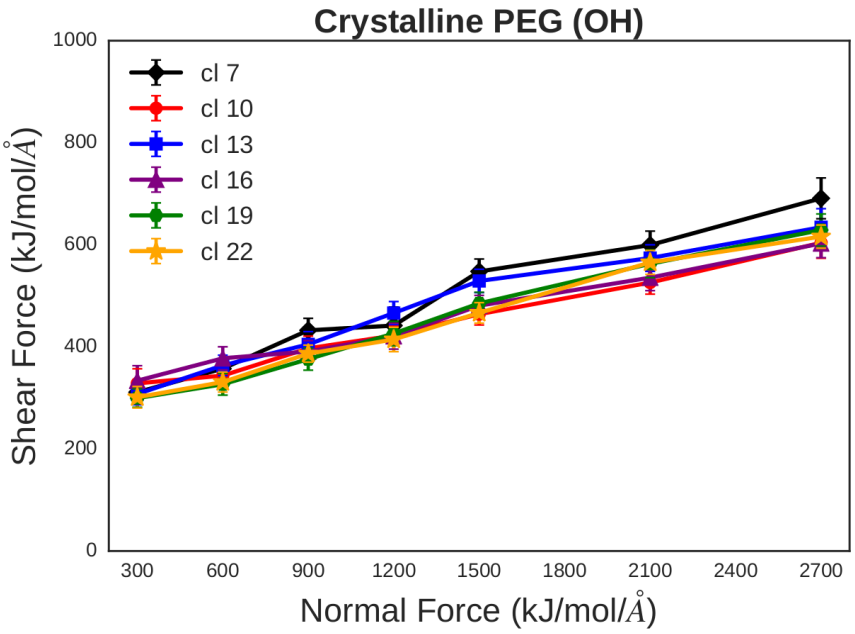


Figure 5.5: Shear force as a function of normal force for PEG chains terminated with a hydroxyl group on a crystalline substrate.

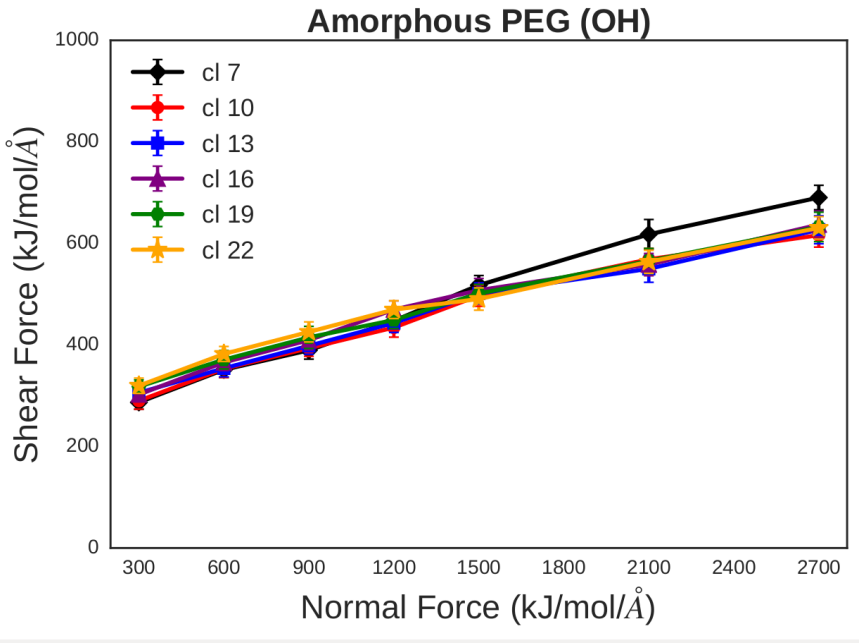


Figure 5.6: Shear force as a function of normal force for PEG chains terminated with a hydroxyl group on an amorphous substrate.

to each system. The nematic order parameter, S, is defined as the largest eigenvalue of the matrix Q

$$Q_{\alpha\beta} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{3}{2}u_{\alpha}^{i}u_{\beta}^{i} - \frac{\delta_{\alpha\beta}}{2}\right) \tag{5.1}$$

where α and β are the Cartesian coordinates x, y, z; u_{α}^{i} is the α component of the direction vector for a chain as weighted by its moment of inertia; and $\delta_{\alpha\beta}$ is the Kronecker delta function. For S, a value of 0 indicates a completely isotropic system while a value of 1 indicates that all chains are pointing in exactly the same direction.

Crystalline substrates clearly induce a higher degree of order into the monolayers as compared to more realistic amorphous substrates. Considering chain chemistry, alkane monomers exhibit the highest ordering. The more flexible PEG monomers exhibit less ordering but the hydroxyl capped versions are slightly more ordered than the methyl capped. Hydrogen bonding networks between headgroups are likely responsible for this behavior.

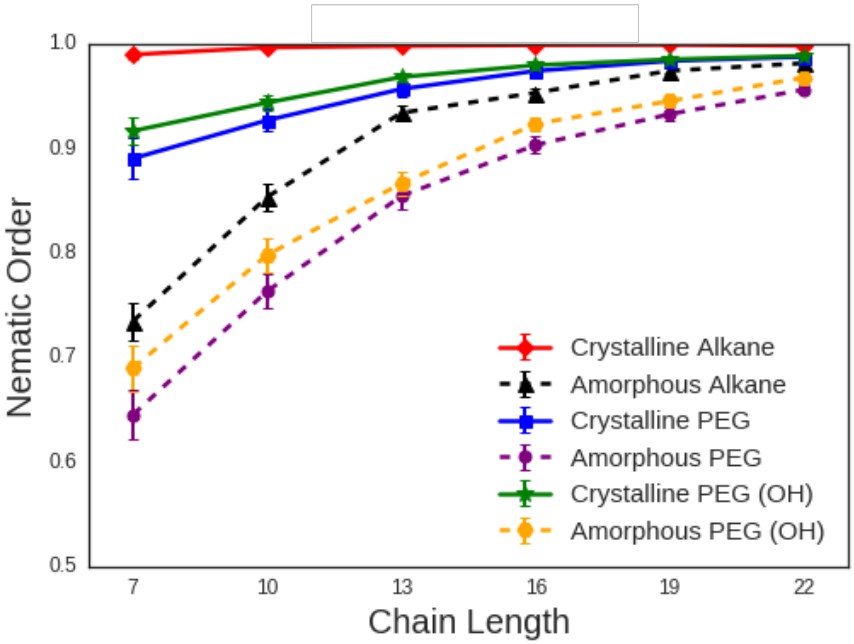


Figure 5.7: Nematic order parameters of the monolayers during shearing. A value of 0 indicates a completely isotropic system while a value of 1 indicates that all chains are pointing in exactly the same direction. Datapoints and errorbars are averages and standard deviations across all normal forces applied.

5.5 Pitfalls of using Idealized Crystalline Surfaces

Many modelling studies on the tribology of self-assembled monolayers used idealized crystalline surfaces to represent the substrate from which the monolayer is grown.[5–7] While some trend level analyses in these studies may hold true, more recent work[3] suggests that estimates of metrics including friction coefficients and monolayer order may be dramatically underestimated when crystalline surfaces are used in place of more realistic amorphous surfaces.

As discussed previously, using the relatively straightforward screening procedure described above, several anomalous data points were recorded in the alkane systems on crystalline substrates. In general, past studies and the data presented here suggest that coefficients of friction on crystalline surfaces are lower than for similar systems on amorphous substrates. As discussed in Black et al.,[3] this is largely due to artificially induced ordering

on the crystalline substrates that cannot occur on amorphous substrates. The anomalous spikes in Figure 5.1 turn out to correspond to systems that did not fully shear align as highlighted in the visualization of the same figure. During compression portions of the monolayer become trapped facing opposing directions. At high enough compressive forces, the systems are not able to fully shear align themselves within the 50 ns sampling period used for this study - a much longer time than all previous studies of similar systems to the best of our knowledge. This suggests that, without visual inspection or several other data points to establish a trend, this misalignment would not be readily apparent.

There does, however, appear to be a tell tale sign for this lack of ordering in this particular system: anomalously high perpendicular forces. Figure 5.8 shows the absolute values of forces experienced by monolayers in the direction perpendicular to the shearing direction. These data clearly illustrate that only the crystalline alkane systems experience significant lateral forces. However, these forces are not only experienced by systems that fail to shear align but all alkane systems on crystalline substrates. This effect may additionally be compounded by a combination of the alkane chains higher rigidity as compared to PEG as well as the fact that the shear direction does not perfectly align with a straight track through the hexagonal lattice of the β-cristobalite thus jostling the chains laterally as they pass over each lattice site. For a large automated screening as was performed here, perpendicular force data can be used to automatically identify systems that may not have equilibrated properly or are stuck in an unrealistic energy minimum. Such automated detection is desired for large parameter screening as it becomes extremely impractical to manually visually inspect each system.

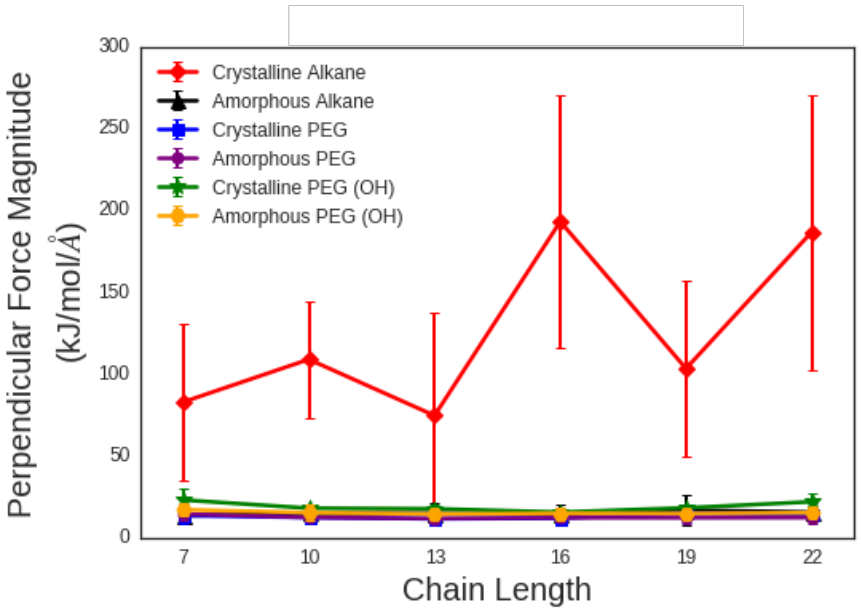


Figure 5.8: Absolute values of forces experienced by the monolayers perpendicular to the shear direction. The shear force is directed in the x-direction, compression is applied in the z-direction and this data represents the absolute values of the y-directed forces.

5.6 Bibliography

[1] Pronk, S.; Páll, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; Van Der Spoel, D.; Hess, B.; Lindahl, E. *Bioinformatics* **2013**, *29*, 845–854.

[2] McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. *Biophysical Journal* **2015**, *109*, 1528–1532.

[3] Black, J. M.; Baris Okatan, M.; Feng, G.; Cummings, P. T.; Kalinin, S. V.; Balke, N. *Nano Energy* **2015**, *15*, 737–745.

[4] Summers, A. Z.; Iacovella, C. R.; Billingsley, M. R.; Arnold, S. T.; Cummings, P. T.; McCabe, C. *Langmuir* **2016**, *32*, 2348–2359.

[5] Booth, B. D.; Vilt, S. G.; Lewis, J. B.; Rivera, J. L.; Buehler, E. a.; McCabe, C.; Jennings, G. K. *Langmuir* **2011**, *27*, 5909–17.

[6] Lewis, J. B.; Vilt, S. G.; Rivera, J. L.; Jennings, G. K.; McCabe, C. *Langmuir* **2012**, *28*, 14218–26.

[7] Rivera, J. L.; Jennings, G. K.; McCabe, C. *The Journal of Chemical Physics* **2012**, *136*, 244701.

CHAPTER 6

TUNABLE TRANSITION FROM HYDRATION TO MONOMER-SUPPORTED LUBRICATION IN ZWITTERIONIC MONOLAYERS

This chapter describes the tribology of 2-methacryloyloxyethyl phosphorylcholine monolayers in water molecular dynamics simulations. Our results show two distinct shear regimes where the first is dominated by hydration lubrication, exhibiting near zero friction coefficients, and the second by chain-chain interactions, resembling monomer-supported lubrication. We discuss the implications of these results as they relate to the much larger pMPC brush systems grown experimentally. The work detailed here is based on reference.[1]

6.1 Introduction

Previous studies have demonstrated the utility of molecular simulation in elucidating tribological mechanisms[2–6] but have not specifically focused on hydration lubrication. Other simulation studies have investigated both neutral[7] and zwitterionic[8–10] monolayer systems in aqueous environments. However, these simulation studies considered densely populated monolayers that do not compare to the environment existing within the experimentally studied brush systems thought to exhibit hydration lubrication. Moreover the foci of the studies considering zwitterionic monolayers were not centered directly on the behavior within zwitterionic brush films but rather on dense phospholipid and carboxybetaine coated surfaces and their interactions with solvated ions.

In this work, atomistic molecular dynamics simulations are used to study the frictional properties of sparse, silane-functionalized MPC (Figure 6.2A) monolayers in an aqueous environment. More specifically, the aim is not to directly model the behavior of pMPC brush films but rather gain a deeper understanding of the molecular level environment sur-

rounding individual monomers found in the brushes. Additionally, the monolayers studied herein differ from previously studied monolayers in two primary ways. First, the zwitterionic nature of the monomers clearly distinguishes them from neutral monolayers such as aklysilanes. Second, in contrast to previously studied densely packed monolayers (both neutral and zwitterionic), our sparse monolayers are designed to mimic the aqueous environment found in experimentally studied pMPC brushes where water can interpenetrate the polymer film. Clearly the experimental brush systems are more complex due to the network of chain-chain interactions but the goal of this study is to clearly capture the MPC-water interactions, largely eliminating the chain-chain interactions to establish a baseline behavior.

6.2 Simulation Details

Monolayers were created on 4.1x4.7 nm (19.7 nm^2) β-cristobalite silica substrates which provide an atomically smooth surface with well defined binding sites to enable examination of tribological phenomena under idealized conditions. Previous work has shown that this system size is sufficiently large to avoid any size effects related to periodic boundary conditions during tribological simulations.[11] Silane-functionalized MPC monomers were randomly attached to oxygen atoms with open valencies on the silica surfaces via the monomers' terminal silane groups.

Surface oxygen atoms not bound to a monomer were terminated as hydroxyl groups. Two systems were prepared with average chain surface densities of 1.02 and 2.03 chains per nm^2 which corresponds to monomers attached to 20 and 40% of the under-coordinated surface oxygens, respectively. The chain arrangement for both systems is shown in Figure 6.1 where opposing substrates were mirrored. The sparser chain density (1.02 chains per nm^2) was chosen because monomers in the experimentally studied polymer brush systems are likely to possess higher mobility than would be granted by a densely packed monolayer on a flat substrate. The denser system (2.03 chains per nm^2) compares to chain densities found

in previous simulation studies of zwitterionic monolayers (1.98 chains per nm^2)[8] and thus provides a reference point.

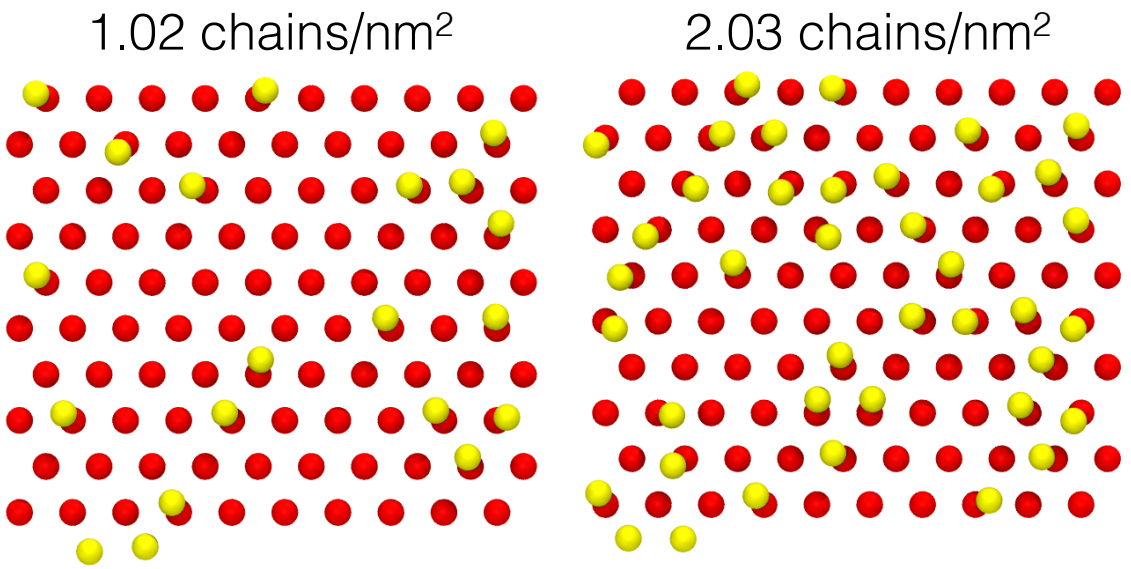


Figure 6.1: Binding site locations for both chain densities. Red shows available surface oxygens. Yellow shows silicon atoms at the base of bound monomers.

The previously studied monolayers, however, consisted of longer, more linear monomers than MPC and were formed in regular lattice arrangements that did not allow for water to penetrate into the monolayer. Additionally, observing similar results across two chain densities allows us to rule out anomalous behavior arising from one particular system. Both systems were then solvated in TIP3P water to create a density of $\sim$1.0 g/cm^3 in the interstitial water layer post-compression, at the largest separation distances where friction simulations were performed. The separation distance, D, between opposing monolayers is defined as the distance between the outermost layers of silicon atoms in each substrate. A schematic overview of the system setup is given in Figure 6.2B. The number of water molecules remains constant for all separation distances thus representing the edge case where compression happens fast enough to prevent significant squeeze out of water molecules during the elapsed simulation time. This model makes sense for a closed system where squeeze out occurs on a much longer timescale than the simulations performed here; e.g., we are considering a nanoscale segment of what would typically be a milli- to centimeter scale

substrate where water may not be able to rapidly escape from the highly absorbent zwitterionic films. The other edge case, not considered in this work, would consider complete equilibration with bulk water outside of the pore at each separation distance.

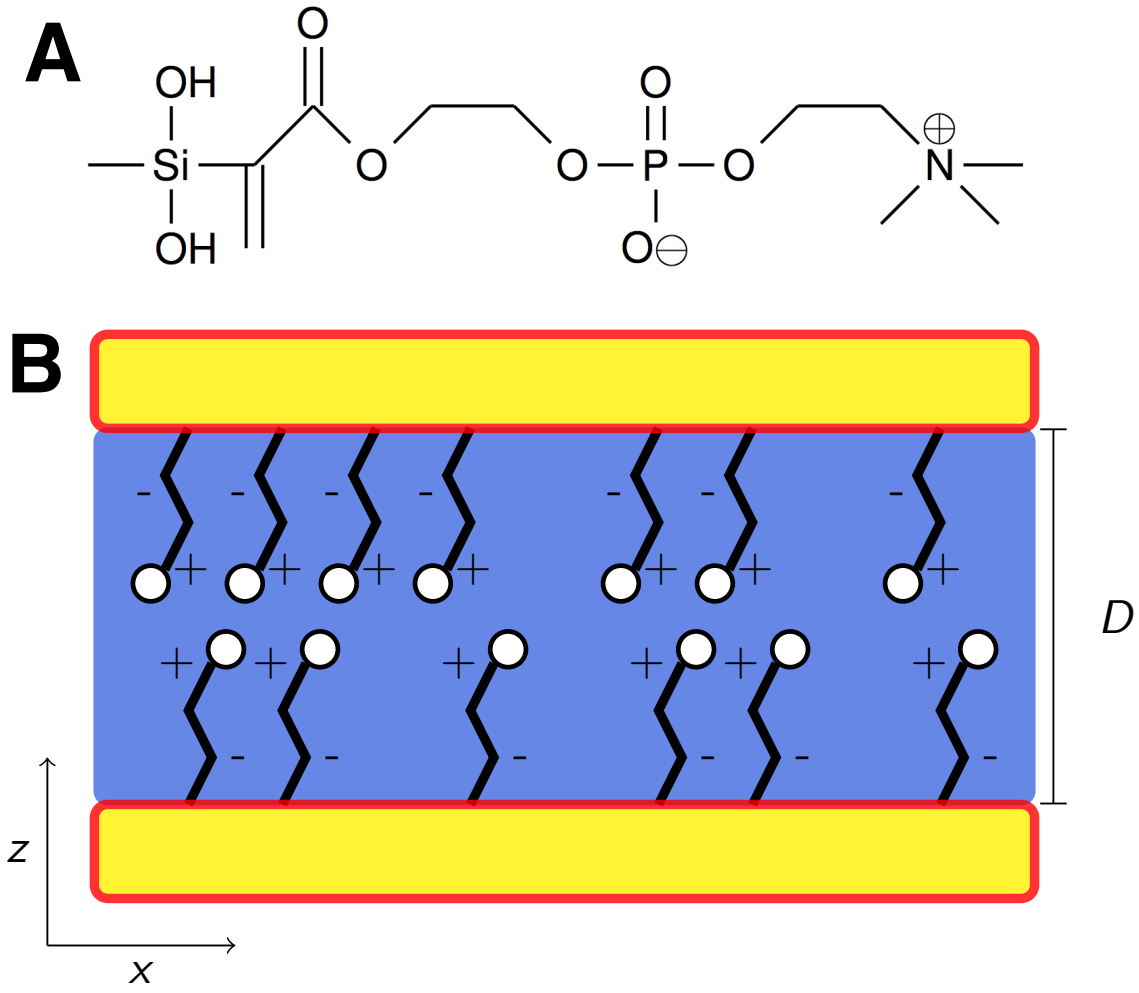


Figure 6.2: Overview of MPC monomer system. **A)** Silane functionalized MPC monomer. **B)** Schematic slice of solvated, unequilibrated systems. Silica substrates shown in red/yellow; chains in black/white; water in blue. Plus/minus symbols denote approximate locations of charged regions. The non-periodic z-dimension is defined as normal to the substrates; the x- and y-dimensions are both periodic and shearing is performed in the x-dimension.

Simulations were performed in LAMMPS[12] using OPLS-aa parameters for both the MPC monomers[13] and the substrate.[14] To ensure charge neutrality of the system, partial charges were adjusted from their values for free chains, which by construction are es-

sentially charge neutral; these adjusted charges were localized to the region around the attachment site to avoid significantly altering more relevant components of the original model. The charges used are detailed in Table 6.1 and Listing 6.1 The partial charges used in this work are summarized in Table S1. The rRESPA multi-timestep integrator[15] was applied to the MPC chains and water molecules with timesteps of 0.25 fs for bonds, 0.5 fs for angles/dihedrals and 1 fs for non-bonded interactions. Periodic boundary conditions were applied in the *x*- and *y*-directions to simulate an infinite surface. Electrostatic interactions were modeled using the 2D particle-particle particle-mesh algorithm[16] where the real space component is calculated within a 1 nm radius. Lennard-Jones interactions were calculated with a 1 nm cutoff radius. The temperature of the water and chain molecules was maintained at 300 K using the Nosé-Hoover thermostat.[17,18]

Table 6.1: Summary of partial charges in MPC monomer. Corresponding `.xyz` file for a monomer given below.

Name	q (e)
HC3	0.15
N3	0.30
CH3	-0.31
CT	-0.13
HC	0.06
CTD	0.41
HCD	-0.01
OSD	-0.61
P	1.42
O2D	-1.03
CTE	0.27
OS	-0.61
C	0.99
O	-0.69
CM3	-0.13
CM1	-0.30
HM	0.16
CH2	-0.26
OH	-0.88
HO	0.44
Si2	1.29

Listing 6.1: `.xyz` file for a single monomer

```
45
Silane MPC
Si2     -0.04172   -2.90982    0.22037
OH      -1.24454   -2.87736    0.00477
OH       1.10628   -3.12501   -0.13071
HO      -2.28453   -2.69327    0.34699
HO       2.19471   -3.31217   -0.36034
CH2      0.11586   -2.56520    1.31676
CM3      0.95504   -2.16260    2.51336
HC      -0.26119   -3.53452    1.71497
HC      -0.57509   -1.71410    1.52665
CM1      2.14119   -1.54573    2.37393
HM       2.55320   -1.34557    1.38814
HM       2.71388   -1.19372    3.22606
C        0.29712   -2.37195    3.84703
O       -0.83193   -2.82291    3.96183
OS       1.04002   -1.99788    4.91054
CTE      0.52719   -2.24033    6.24686
CTD      1.60798   -1.78750    7.24244
HC       0.33306   -3.31155    6.35865
HC      -0.40096   -1.68051    6.39816
OSD      1.42343   -2.42122    8.53518
HCD      1.60400   -0.69923    7.34603
HCD      2.58140   -2.10625    6.86123
P        0.69269   -1.67034    9.75920
OSD      2.24817   -1.93833   10.15396
O2D     -0.56520   -2.36912    9.40468
OSD      0.71235   -2.38080   11.20677
CTD     -0.04528   -1.85267   12.32707
CT       0.90614   -1.16063   13.33217
HCD     -0.54102   -2.69471   12.81585
HCD     -0.80235   -1.15413   11.96164
N3       0.29673   -0.57328   14.60028
HC       1.42836   -0.34232   12.82827
HC       1.64632   -1.88560   13.68450
CH3      1.39046    0.07911   15.41680
HC3      1.87522    0.86910   14.83652
HC3      2.14672   -0.65816   15.70080
HC3      0.98398    0.52350   16.33042
CH3     -0.35135   -1.65156   15.44106
HC3     -1.19803   -2.14701   14.95900
HC3     -0.71483   -1.17491   16.35686
HC3      0.38362   -2.41235   15.71859
CH3     -0.73911    0.47297   14.25681
HC3     -0.28687    1.24851   13.63252
HC3     -1.09334    0.93523   15.18306
HC3     -1.60419    0.06107   13.73049
```

Two distinct sets of simulations were performed. First, compression runs were car-

ried out by fixing the bottom substrate in space and displacing the top substrate down at 0.001 nm/ps. Second, using configurations obtained from the compression runs as input, systems with fixed separation were examined under shear. Shearing simulations were performed at separation distances of 4.1-2.8 nm and 3.8-2.5 nm in increments of 0.05 nm for chain densities of 2.03 and 1.02 chains per nm^2, respectively, for a total of 54 shearing simulations. Shear stresses were induced by displacing the top substrate at 10 m/s in the *x*-direction. From each independent shearing simulations, 15-30 ns of post-equilibration data were collected; systems with higher separation distances were sampled for longer periods of time due to larger fluctuations in shear forces. Normal and shear forces were calculated by summing the forces experienced by the bottom substrate plus chains in the *z*- and *x*-directions, respectively. These forces were sampled every 1 ps and block averaged across each simulation to yield the data points and standard errors shown in Figure 6.6.

Film thickness is calculated by binning atoms in each monolayer over time according to their z-position in the system (shown in 6.8). The film thickness is then defined as the difference between the z-value where the cumulative distribution function reaches 0.9 and the z-value closest to the surface.

Although shearing speeds of 10 m/s are greater than those typically studied experimentally, they are frequently encountered in practical applications. For example, micro- and nanoelectromechanical devices routinely operate at speeds on the order of 10-20 m/s and natural joints can experience shearing on the order of 1 m/s, *e.g.*, when a kicking motion is made where angular velocities can reach over 20 rad/s.[19] Considering artificial joints or cartilage scaffolds as real world applications for materials lubricated via the mechanisms studied here, speeds experienced in knee joints are of obvious interest. Additionally, both experimental[20] and simulation studies[11] of shearing, functionalized surfaces show little dependence on shearing velocities across several orders of magnitude. This is not to suggest that there is *no* dependence but that the effects are minimal within the, respectively, accessible ranges of shearing speeds for experimental and simulation studies.

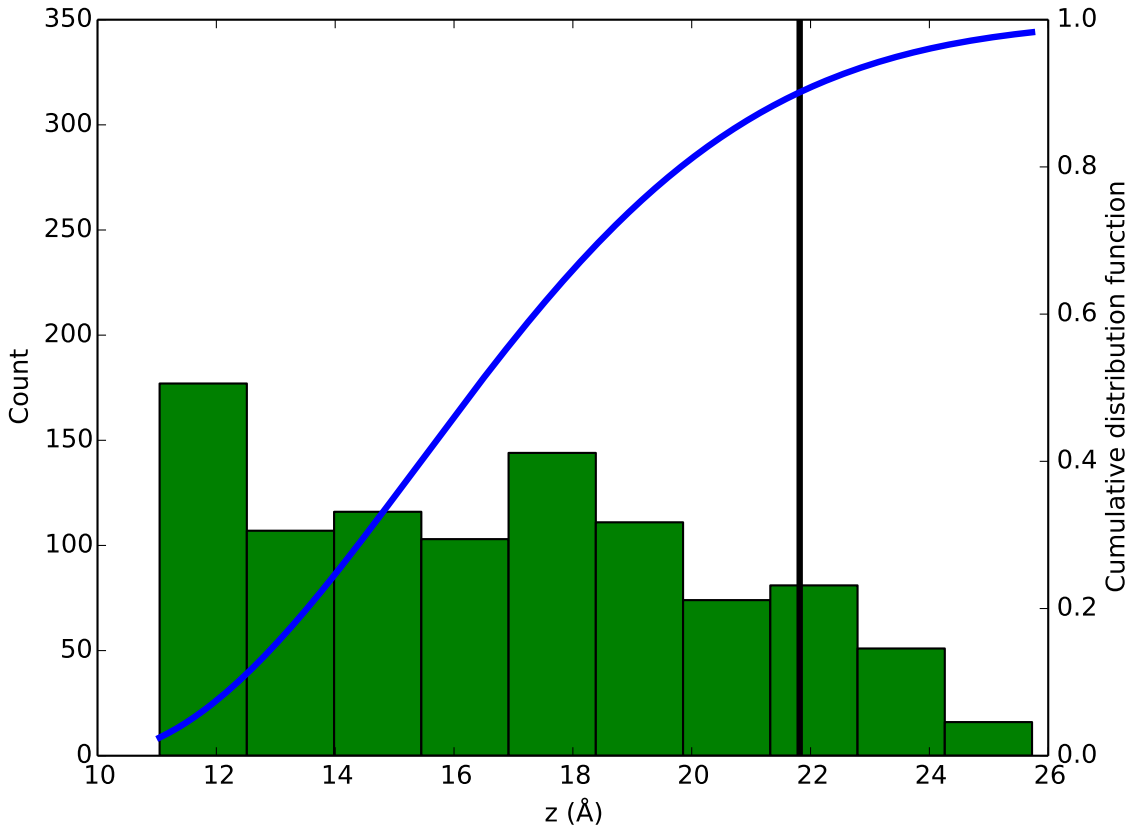


Figure 6.3: Example of film thickness calculation for a single frame. Green histogram shows vertical position of atoms in bottom film. Blue line denotes cumulative distribution function (CDF) of atoms in bottom film. Black vertical line denotes cutoff where CDF reaches 0.9.

6.3 Effects of Compression

In Figure 6.4 we present the normal force response for the aqueous MPC system as it is compressed for both surface densities of 1.02 and 2.03 chains per nm^2. Both systems show nearly identical *trends*, although the less dense system (1.02 chains per nm^2) appears to have its force curve shifted to smaller separations ($\sim$0.22 nm lower) and a lower normal force, roughly proportional to surface coverage. This phenomenon is highlighted in the inset of Figure 6.4 where the base line normal forces are shifted to zero and the separation distance of the denser system is shifted by 0.22 nm to show the overlap. The general trend shows that as separation distances between the substrates decrease, normal forces increase.

However, there are noticeable deviations from the overall trend.

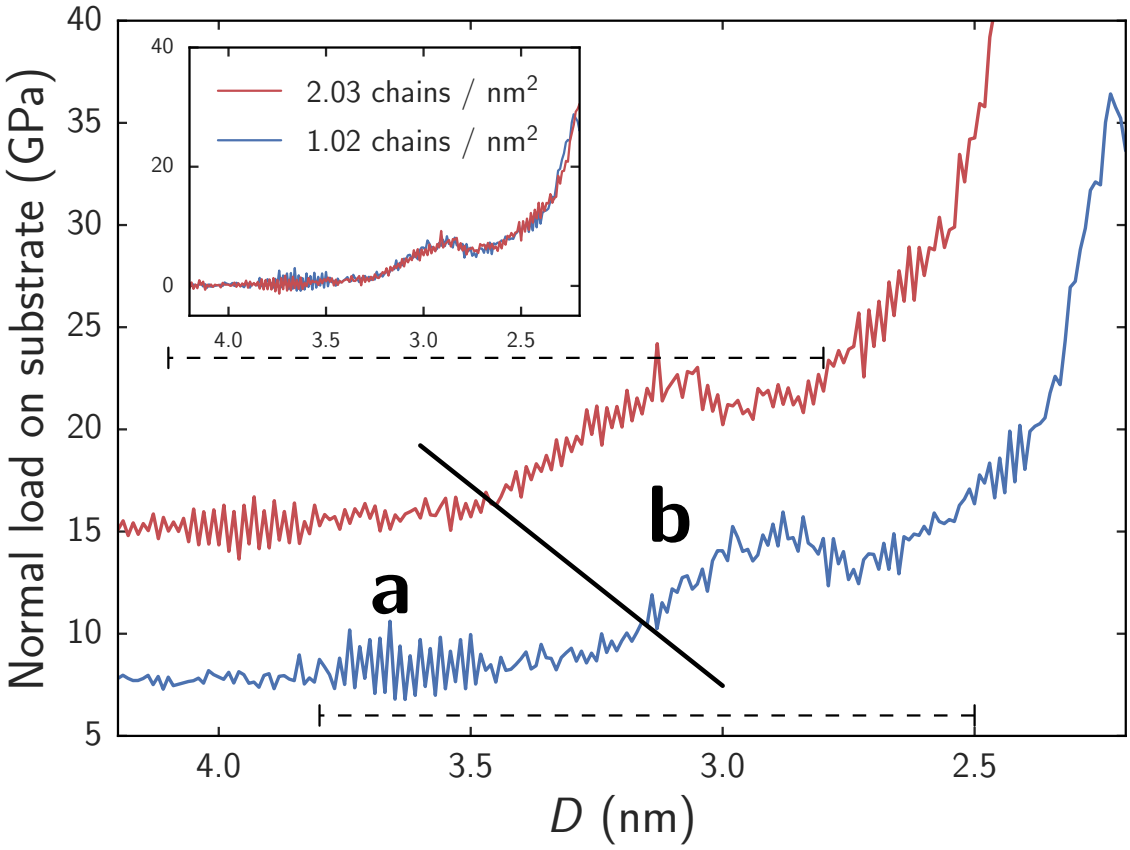


Figure 6.4: Normal force responses during compression of MPC systems. In main plot, shown forces are acting on substrate only in order to highlight the effects of chain density. Summing the forces acting on both the substrate and chains shifts the curves down the y-axis, yielding a base line normal force (at high separation) of approximately zero; additionally, shifting the separation distance by 0.22 nm for the denser system shows almost complete overlap in trend behavior for the two systems (shown in inset). The diagonal solid line approximately denotes the two regimes (**a** and **b**) discussed in text and highlighted in Figure 6.5. The upper and lower horizontal dashed lines correspond to ranges of separations across which we performed shearing simulations for the respective chain densities.

Two clear regimes are observed with respect to compression (hereinafter referred to as **a** and **b**). In systems of 2.03 and 1.02 chains per nm^2, the transitions between these two regimes respectively occur at approximately 3.4 nm & 3.1 nm. This transition is consistent between the two surface densities, although, as noted, shifted to lower separations for the less dense monolayer. In regime **a**, as highlighted in Figure 6.5, an interstitial water layer separates the opposing monolayers. The transition, from **a** to **b**, approximately corresponds

to the first contact of terminal groups of chains attached to opposing substrates, rather than just a pure interstitial water layer (compare Figure 6.5a to Figure 6.5b). Throughout the transition and beyond, a large spike in normal force occurs. Examination of the total amount of water within the monolayer region as a function of separation reveals a similar trend to Figure 6.4; relatively constant values within regime **a** and an increase as we cross over into regime **b** (see Figure S2). These phenomena suggests a high resistance to compression marked by swelling of the monolayer with water, which is qualitatively consistent with experimental studies of polymeric brushes built from the same monomers.[20–24]

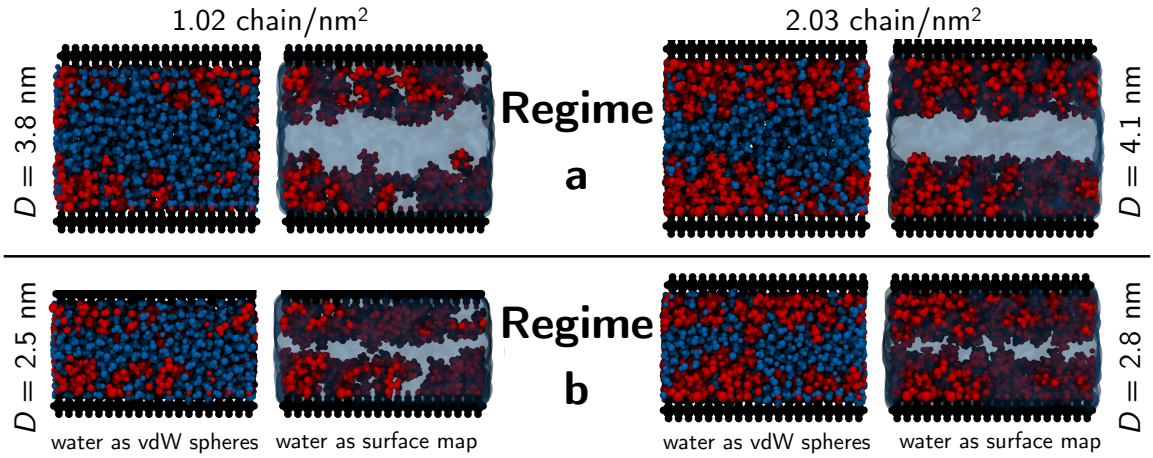


Figure 6.5: Snapshots of systems with 1.02 (left) and 2.03 (right) chains per nm^2 at various separation distances. **Regime a**) Interstitial water layer separates surfaces. **Regime b**) Interstitial water layer no longer exists and chains from opposing substrates contact one another. Substrates are shown as black van der Waals spheres, chains as red van der Waals spheres and water is depicted as blue van der Waals spheres in the left image for every configuration and a surface map in the right image for the same configuration. All visualizations were performed using VMD.[25]

6.4 Effects of Shearing

Figure 6.6 shows shear stresses experienced by substrates at different normal forces that fall within regimes **a** and **b**. The normal force values on the x-axis correspond to the aforementioned separation distances and do not fluctuate significantly. At normal forces below $\sim$35 nN, which corresponds to regime **a** and approximately 1.8 GPa, shear stresses

increase with a nearly linear trend. By performing linear least squares fitting of the data, the slope of F_s versus F_n was estimated which provides an estimate of the friction coefficient, μ. In regime **a**, we find friction coefficients of 0.014 for the sparser system and 0.016 for the denser system which are comparable to those exhibited by neutral, densely packed alkylsilane monolayers separated by water.[7] However, the MPC monolayers sustain this coefficient of friction at normal loads nearly an order of magnitude higher. Unlike in dense hydrophobic monolayers, the sparse, flexible, zwitterionic monolayers enable water to penetrate into the monolayer. Instead of being restricted to a narrow regime bounded by the chains' terminal groups, water is able to penetrate into the monolayers and provide similarly low friction coefficients at higher normal loads. Similarly, our sparse monolayers provide much lower friction coefficients when compared to densely packed phospholipid[9] and carboxybetaine[10] monolayers which range from ˜0.2-0.5 under the studied conditions. The friction coefficients found in regime **a** are, in fact, comparable to values reported for experimentally synthesized pMPC brush structures[21,22,24] yet slightly above the lowest reported values.[20,23] Since our system is a monolayer analogue to the experimentally studied one that features much larger polymer brushes, we cannot offer a direct comparison to these experimental results. However, the similarly low coefficients of friction at high normal loads observed here suggest that regime **a**, may be a good model for the local environment between brushes in the experimentally synthesized systems which are able to leverage the hydration lubrication mechanism to provide an efficient dissipation pathway for frictional forces. Here, chains attached to opposing surfaces or different polymer backbones, such as in the looped bottle brushes synthesized by Banquy *et al.*,[26] come into close proximity but are still separated by a few water molecules that form the hydration shells around monomers and can rapidly exchange between shells.

As normal loads trend towards zero, shown in the top left inset of Figure 6.6, the average shear stresses fluctuate about zero. Portions of this regime correspond to the normal force fluctuations observed during compression beginning around 3.8 nm for the sparser system

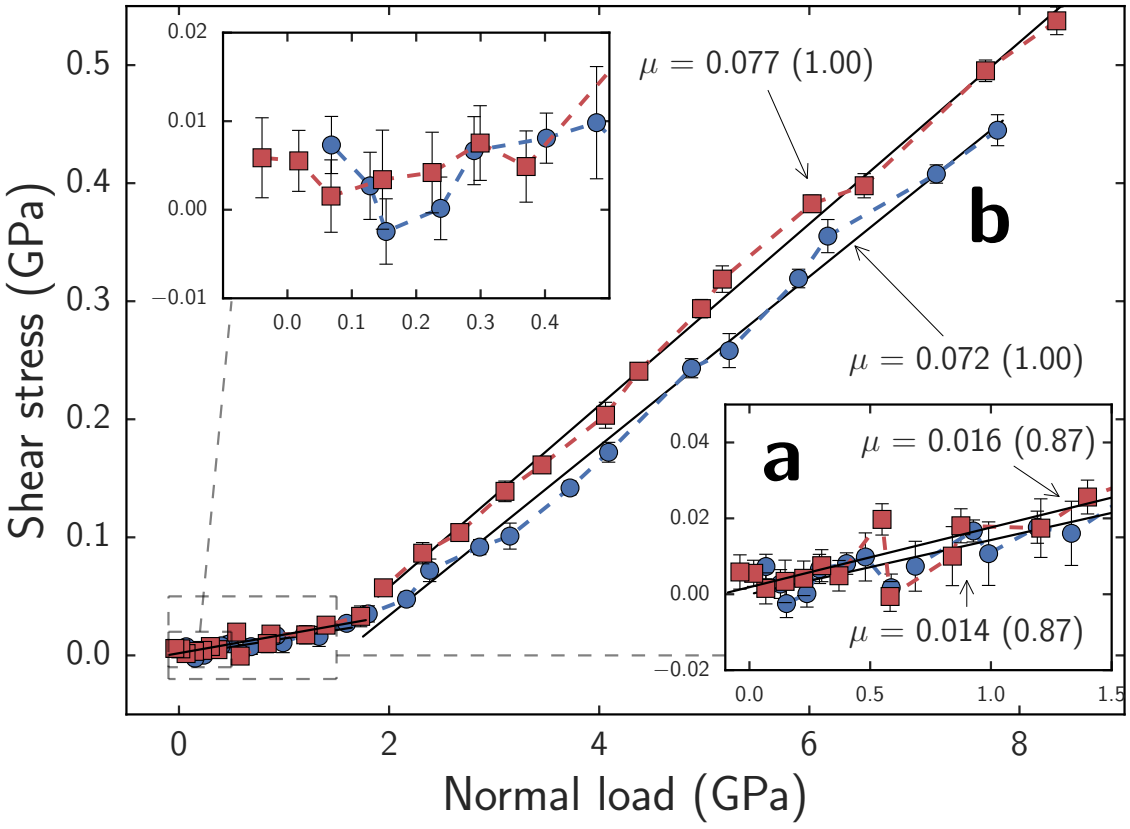


Figure 6.6: Block averaged shear stresses as a function of normal force per area. Colors correspond to Figure 6.4. Error bars represent standard errors of the means; normal load error bars are plotted but smaller than data markers for all points. Friction coefficients, μ, correspond to the slope of F_s versus F_n as determined by linear least squares fitting. R-values are shown in parentheses. The friction coefficients and fitted slopes near **a** and **b** respectively correspond to the regimes discussed in text and highlighted in Figure 6.5.

and 4.1 nm for the denser system (Figure 6.4). We suspect that some of this behavior is caused by the transition from the attractive regime, where the two substrates are not yet in compressive contact, to the repulsive regime.

At normal forces exceeding ∼35 nN (1.8 GPa), which corresponds to regime **b**, shear stresses monotonically increase in a linear trend. In regime **b**, we find friction coefficients of 0.072 for the sparser system and 0.077 for the denser system. This magnitude of friction coefficients is comparable to those reported in previous experimental and simulation studies of neutral, densely packed monolayers without water[27–31] but again at higher normal loads. Due to consistent chain-chain interactions between opposing monolayers, the shear

response in this regime is likely more similar to that of systems without water which gives rise to the comparable friction coefficients. Just as in other studies without water, chains from opposing monolayers provide the primary normal load and shear stress response as opposed to the water molecules exchanging via the hydration lubrication mechanism. The chains are far less mobile than water molecules and are thus unable to provide a comparably fluid shear response yet they are able to provide a strong resistance to compression as shown in Figure 6.4.

We also note that additional shearing simulations conducted at 1 m/s demonstrate the same trends; these results are summarized in Figure S3.

6.5 Water Distribution During Shearing

In the low shear stress regime, interactions between chains from opposing substrates are rare as is reflected in the velocity profiles shown in the top panels of Figure 6.7. Here, all parts of the chains are moving at approximately the speed of the substrate as is a layer of water approximately as tall as the chains. There is also a distinct layer of water separating the two monolayers. Above $\sim$35 nN (1.8 GPa), opposing chains begin to consistently interact as shown in the bottom panels of Figure 6.7. Opposing chains' terminal choline groups interpenetrate, drag each other and displace the interstitial water layer. The water velocity profiles in our systems differ from profiles of nanoconfined water[32,33] and water confined between alkylsilane monolayers;[7] where these systems exhibit a near linear slope from wall-to-wall or chain-to-chain with approximate boundary layers of less than 0.5 nm. The profiles in our systems assume an S-like shape with water layers over 1 nm thick within the sparse, zwitterionic monolayers. While monolayer thickness does decrease with decreasing separation distances (Figure 6.8), the change in thickness is relatively minimal. A decrease in separation distance of 1.3 nm for both systems, yields only a decrease in film thickness of about 0.13 and 0.12 nm for 1.02 and 2.03 chains per nm^2 respectively. This almost negligible change suggests that the monolayers swell with water as normal loads

increase - a phenomenon also observed experimentally.[24,34]

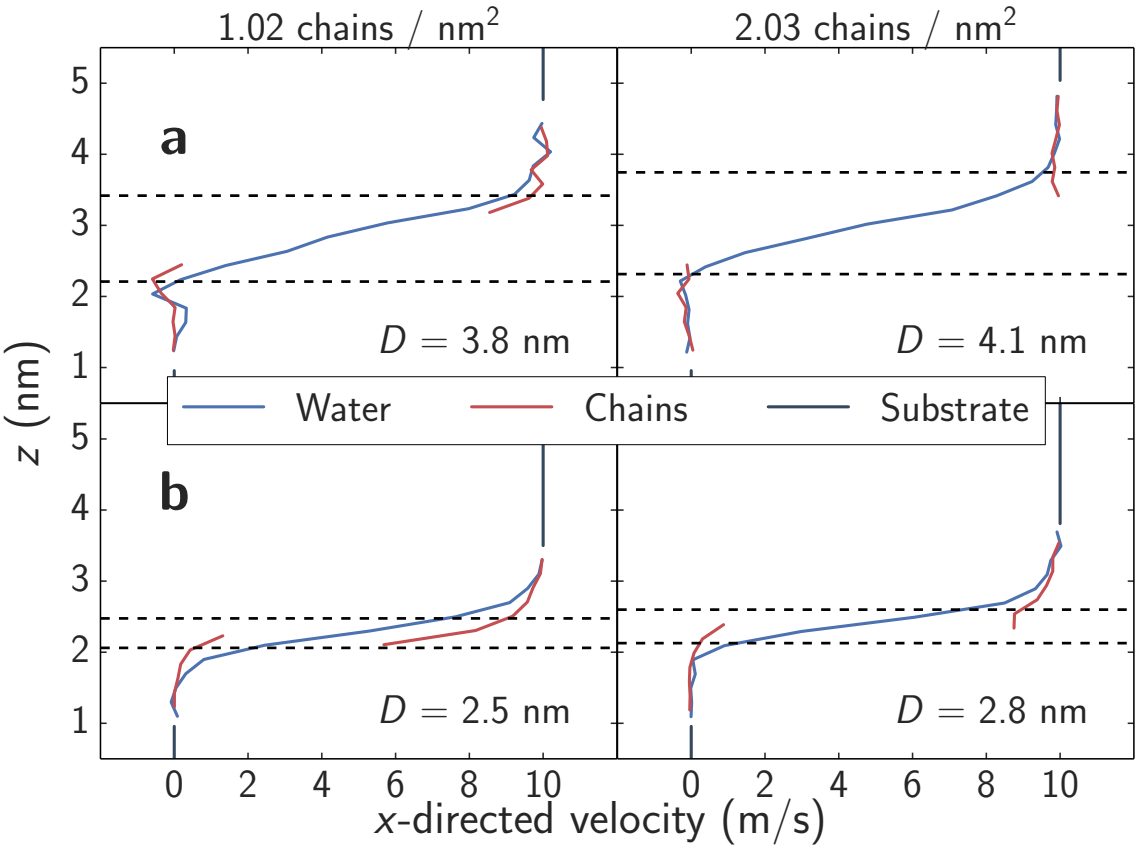


Figure 6.7: Velocity profiles of system components in the shear direction. The dashed lines denote the monolayer film thickness, wherein 90% of the monolayer atoms remain over the course of a simulation. Film thickness calculation is detailed in Figure S4 and trends as a function of separation are shown in Figure 6.8. Note that in the bottom two panels, interpenetration of chains from opposing monolayers occurs for portions outside the defined film thickness cutoff thus displacing the interstitial water layer.

Figure 6.9 further details the distribution of water across the pores. The bulk of the water remains in the interstitial area but spreads into the outer parts of the monolayers (approximately denoted by the dashed lines) where it appears to bind near where the choline headgroups are located. As the systems are compressed and they transition from regime **a** to **b** the interstitial water does densify but water is also pushed into the monolayers as is particularly evident in the sparser system.

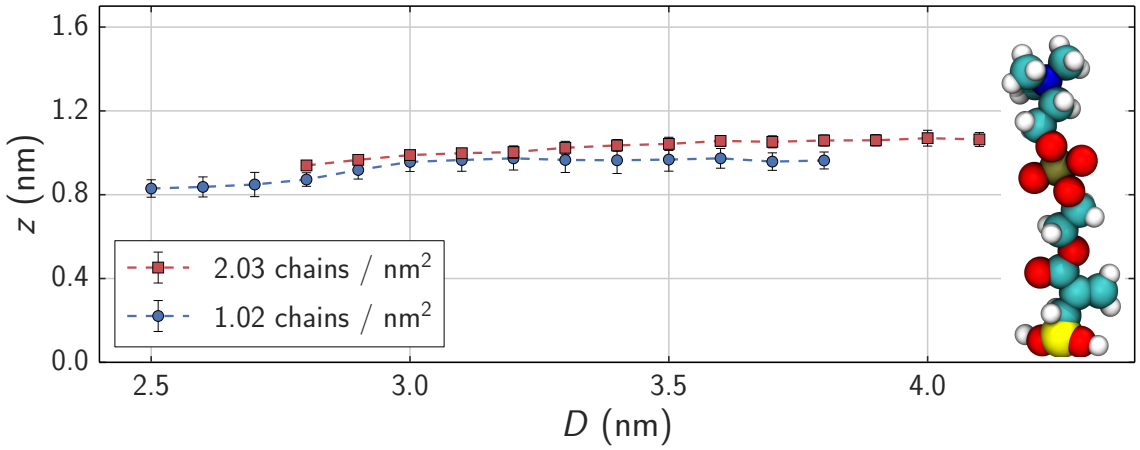


Figure 6.8: Dependence of monolayer thickness on separation distance during shearing. Equilibrated MPC chain *in vacuo* shown for reference (to scale of y-axis). Error bars correspond to standard deviations through time.

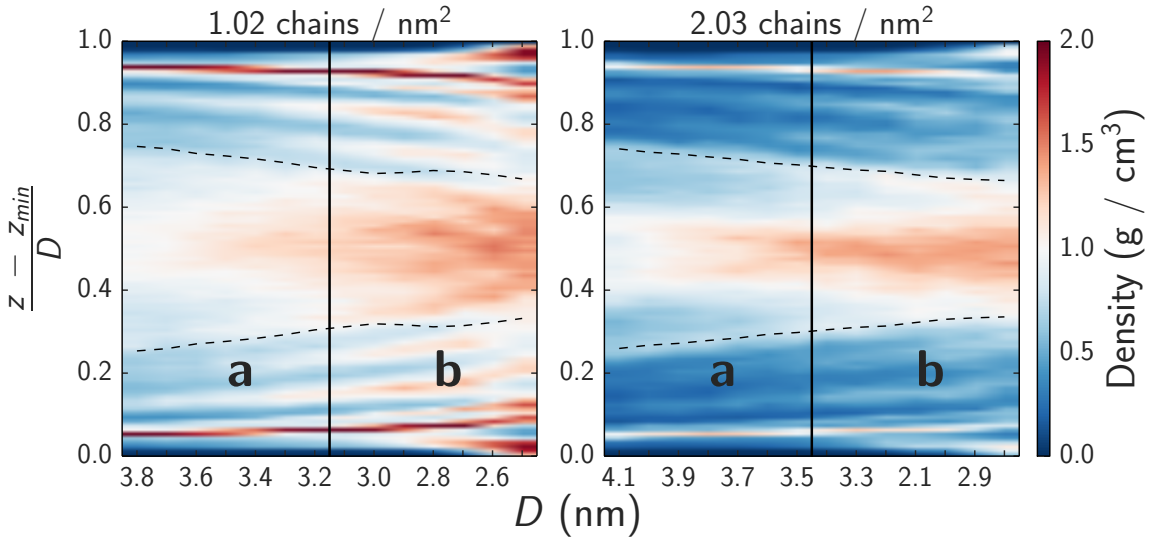


Figure 6.9: Water density across *xy*-planes as a function of normal force. The *y*-axis denotes the normalized position in the pore from the bottom substrate at 0 to the top substrate at 1. Solid vertical lines denote approximate transitions between regimes **a** and **b** as discussed in text and highlighted in Figure 6.5. The dashed lines denote the monolayer film thickness, wherein 90% of the monolayer atoms remain over the course of a simulation and which the choline headgroups frequently protrude from. Note that these are local *water* densities which do not include the monomers.

6.6 Water Mobility During Shearing

Experimental studies suggest that ultra-low shear stresses in hydration lubrication schemes arise due to rapid water exchange rates between hydration shells or hydration shells and a

bulk-like fluid state.[20,23,26,35–38] In Figure 6.10, we quantify water mobility, by considering flux through *xy*-planes as a function of normal force. Low normal forces allow for rapid water exchange in the interstitial region as well as with water bound to the choline headgroups. Comparing to Figure 6.6, at both chain densities, systems with the highest interstitial water flux correspond to those operating in regime **a** where fluxes are up to three times higher than in regime **b** and friction coefficients are up to an order of magnitude lower. Upon transitioning to regime **b**, where the hydration lubrication mechanism does not seem to be the dominant mechanism due to a lack of water mobility, the chains are still able to provide some form of lubrication while simultaneously resisting higher normal loads.

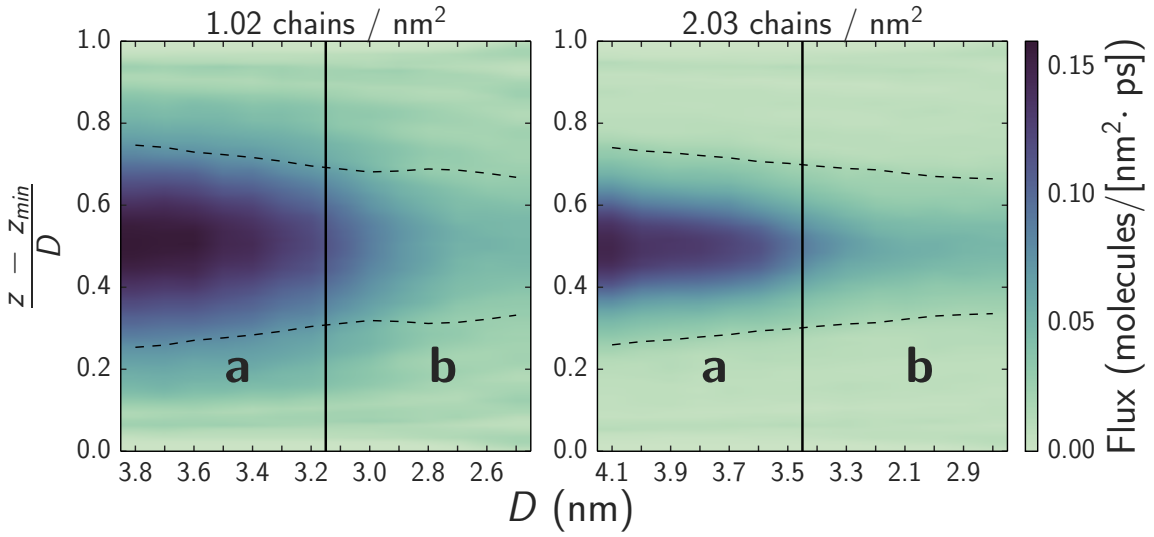


Figure 6.10: Water flux across *xy*-planes as a function of normal force. The *y*-axis denotes the normalized position in the pore from the bottom substrate at 0 to the top substrate at 1. Solid vertical lines denote approximate transitions between regimes **a** and **b** as discussed in text and highlighted in Figure 6.5. The dashed lines denote the monolayer film thickness, wherein 90% of the monolayer atoms remain over the course of a simulation and which the choline headgroups frequently protrude from.

6.7 Bibliography

[1] Klein, C.; Iacovella, C. R.; McCabe, C.; Cummings, P. T. *Soft matter* **2015**, *11*, 3340–3346.

[2] Cui, S. T.; Cummings, P. T.; Cochran, H. D. *The Journal of Chemical Physics* **2001**, *114*, 7189.

[3] Onodera, T.; Morita, Y.; Suzuki, A.; Koyama, M.; Tsuboi, H.; Hatakeyama, N.; Endou, A.; Takaba, H.; Kubo, M.; Dassenoy, F.; Minfray, C.; Joly-Pottuz, L.; Martin, J.-M.; Miyamoto, A. *The Journal of Physical Chemistry B* **2009**, *113*, 16526–36.

[4] Onodera, T.; Morita, Y.; Nagumo, R.; Miura, R.; Suzuki, A.; Tsuboi, H.; Hatakeyama, N.; Endou, A.; Takaba, H.; Dassenoy, F.; Minfray, C.; Joly-Pottuz, L.; Kubo, M.; Martin, J.-M.; Miyamoto, A. *The Journal of Physical Chemistry B* **2010**, *114*, 15832–8.

[5] Carrillo, J.-M. Y.; Russano, D.; Dobrynin, A. V. *Langmuir : the ACS journal of surfaces and colloids* **2011**, *27*, 14599–608.

[6] Yue, D.-C.; Ma, T.-B.; Hu, Y.-Z.; Yeon, J.; van Duin, A. C. T.; Wang, H.; Luo, J. *The Journal of Physical Chemistry C* **2013**, *117*, 25604–25614.

[7] Lorenz, C. D.; Chandross, M.; Lane, J. M. D.; Grest, G. S. Nanotribology of water confined between hydrophilic alkylsilane self-assembled monolayers. 2010; http://stacks.iop.org/0965-0393/18/i=3/a=034005?key=crossref.67a1e0d9a066677967ecb01caba49a69.

[8] Zheng, J.; He, Y.; Chen, S.; Li, L.; Bernards, M. T.; Jiang, S. *The Journal of Chemical Physics* **2006**, *125*, 174714.

[9] He, Y.; Chen, S.; Hower, J. C.; Bernards, M. T.; Jiang, S. *The Journal of chemical physics* **2007**, *127*, 084708.

[10] He, Y.; Shao, Q.; Chen, S.; Jiang, S. *The Journal of Physical Chemistry C* **2011**, *115*, 15525–15531.

[11] Chandross, M.; Grest, G. S.; Stevens, M. J. *Langmuir* **2002**, *18*, 8392–8399.

[12] Plimpton, S. *Journal of Computational Physics* **1995**, *117*, 1–19, http://lammps.sandia.gov.

[13] Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *Journal of the American Chemical Society* **1996**, *118*, 11225–11236.

[14] Lorenz, C.; Webb, E.; Stevens, M.; Chandross, M.; Grest, G. *Tribology Letters* **2005**, *19*, 93–98.

[15] Tuckerman, M.; Berne, B. J.; Martyna, G. J. *The Journal of Chemical Physics* **1992**, *97*, 1990.

[16] Yeh, I.-C.; Berkowitz, M. L. *The Journal of Chemical Physics* **1999**, *111*, 3155.

[17] Hoover, W. *Physical Review A* **1985**, *31*, 1695–1697.

[18] Nose, S. *The Journal of Chemical Physics* **1984**, *81*, 511.

[19] Reilly, T.; Williams, A. M. *Science and Soccer*; Psychology Press, 2003; p 111.

[20] Chen, M.; Briscoe, W. H.; Armes, S. P.; Klein, J. *Science (New York, N.Y.)* **2009**, *323*, 1698–701.

[21] Kobayashi, M.; Terayama, Y.; Hosaka, N.; Kaido, M.; Suzuki, A.; Yamada, N.; Torikai, N.; Ishihara, K.; Takahara, A. *Soft Matter* **2007**, *3*, 740.

[22] Zhang, Z.; Morse, A. J.; Armes, S. P.; Lewis, A. L.; Geoghegan, M.; Leggett, G. J. *Langmuir : the ACS journal of surfaces and colloids* **2011**, *27*, 2514–21.

[23] Chen, M.; Briscoe, W. H.; Armes, S. P.; Cohen, H.; Klein, J. *European Polymer Journal* **2011**, *47*, 511–523.

[24] Kyomoto, M.; Moro, T.; Saiga, K.; Hashimoto, M.; Ito, H.; Kawaguchi, H.; Takatori, Y.; Ishihara, K. *Biomaterials* **2012**, *33*, 4451–9.

[25] Humphrey, W.; Dalke, A.; Schulten, K. *Journal of Molecular Graphics* **1996**, *14*, 33–38.

[26] Banquy, X.; Burdyńska, J.; Lee, D. W.; Matyjaszewski, K.; Israelachvili, J. *Journal of the American Chemical Society* **2014**, *136*, 6199–202.

[27] Lewis, J. B.; Vilt, S. G.; Rivera, J. L.; Jennings, G. K.; McCabe, C. *Langmuir* **2012**, *28*, 14218–26.

[28] Booth, B. D.; Vilt, S. G.; Lewis, J. B.; Rivera, J. L.; Buehler, E. a.; McCabe, C.; Jennings, G. K. *Langmuir* **2011**, *27*, 5909–17.

[29] Rivera, J. L.; Jennings, G. K.; McCabe, C. *The Journal of Chemical Physics* **2012**, *136*, 244701.

[30] Chandross, M.; Webb, E.; Stevens, M.; Grest, G.; Garofalini, S. *Physical Review Letters* **2004**, *93*, 166103.

[31] Chandross, M.; Lorenz, C. D.; Grest, G. S.; Stevens, M. J.; Iii, E. B. W. *JOM* **2005**, *57*, 55–61.

[32] Qiao, R.; Aluru, N. R. *The Journal of Chemical Physics* **2003**, *118*, 4692.

[33] Leng, Y.; Cummings, P. T. *The Journal of chemical physics* **2006**, *125*, 104701.

[34] Kyomoto, M.; Moro, T.; Takatori, Y.; Kawaguchi, H.; Ishihara, K. *Clinical orthopaedics and related research* **2011**, *469*, 2327–36.

[35] Raviv, U.; Giasson, S.; Kampf, N.; Gohy, J.-F.; Jérôme, R.; Klein, J. *Nature* **2003**, *425*, 163–5.

[36] Briscoe, W. H.; Titmuss, S.; Tiberg, F.; Thomas, R. K.; McGillivray, D. J.; Klein, J. *Nature* **2006**, *444*, 191–4.

[37] Gaisinskaya, A.; Ma, L.; Silbert, G.; Sorkin, R.; Tairy, O.; Goldberg, R.; Kampf, N.; Klein, J. *Faraday Discussions* **2012**, *156*, 217.

[38] Klein, J. *Friction* **2013**, *1*, 1–23.

CHAPTER 7

HYDRATION STRUCTURE AND DYNAMICS OF POLY(2-METHACRYLOYLOXYETHYL PHOSPHORYLCHOLINE)

In this chapter, we explore the hydration structure and dynamics of poly(2-methacryloyloxyethyl phosphorylcholine) in solution. The conformations assumed by individual monomers preferentially orient the phosphoryl oxygens away from the polymer, enabling the formation of hydration shells and thus suggesting that the phosphoryl group enables hydration lubrication in these materials. By contrast, poly(2-methacryloyloxyethyl phosphate), or pMP_{50}, an analog without the choline group that exhibits inferior lubrication properties, promotes hydrogen bond-formation between phosphate groups which hinders the formation of hydration shells. The work detailed here is based on a manuscript in preparation.

7.1 Introduction

Previous work has explored the conditions under which water favorably hydrates surface bound MPC monomers and enables such materials to utilize the hydration lubrication mechanism.[1] Here, we further characterize the molecular level structure and dynamics of this hydration water around the much larger pMPC molecules that are experimentally grafted onto substrates for biomimetic lubrication.

7.2 Simulation Details

Initial configurations of a pMPC and a pMP 50-mer were created using mBuild[?] and the OPLS-AA force field[2] was applied using foyer.[?] Each molecule was then solvated in TIP4P-FB water[3] using the "gmx solvate" tool packaged with GROMACS. Using GROMACS 5.1.0,[4] the initial configurations were minimized using a steepest decent procedure

and then equilibrated for 2 ns in the NVT ensemble at 298.15 K using a Berendsen thermostat and then for 10 ns in the NPT ensemble at 298.15 K using velocity rescaling and 1 bar using a Berendsen barostat. Production simulations of XXX ns, for pMPC, and XXX ns, for pMP, were conducted using a 2 fs timestep in the NPT ensemble at 298.15 K using velocity rescaling and 1 bar using a Parrinello-Rahman barostat. All hydrogen bonds were constrained using the LINCS algorithm. Lennard-Jones interactions were modeled with a cutoff of 0.9 nm and a switching function between 0.9 and 1.0 nm. Electrostatic interactions were modeled with a real-space cutoff of 1 nm and the particle-mesh Ewald algorithm. Subsequent trajectory analysis was performed using the GROMACS analysis tools and MDTraj.[5]

7.3 Backbone Dynamics

The polymer backbones of $pMP_{50}$and $pMPC_{50}$exhibit similar degrees of flexibility as shown in Figure 7.1. While the total radius of gyration fluctuates across an approximately 1 nm range, the end-to-end distance of the polymers ranges from approximately 3-10 nm. Visually, one can see the polymers rapidly alternating between a collapsed globular state and a fully extended rod-like state. The lack of the choline group in pMP_{50}does not appear to significantly alter the polymer-scale behavior.

7.4 Intra-Polymer Associations

The radial distribution functions (RDF) presented in Figure 7.2 highlight how individual monomers of the full $pMPC_{50}$molecule associate with one another. Most importantly, the nitrogen-phosphorus RDF exhibits a plateau between ˜0.7 and 1.5 nm. This range corresponds to a choline group, at the end of a monomer, associating with a neighboring monomer's phosphoryl group, at the center of a monomer as illustrated in the visualization of Figure 7.2.

Frequent associations like these between neighboring monomers have multiple effects.

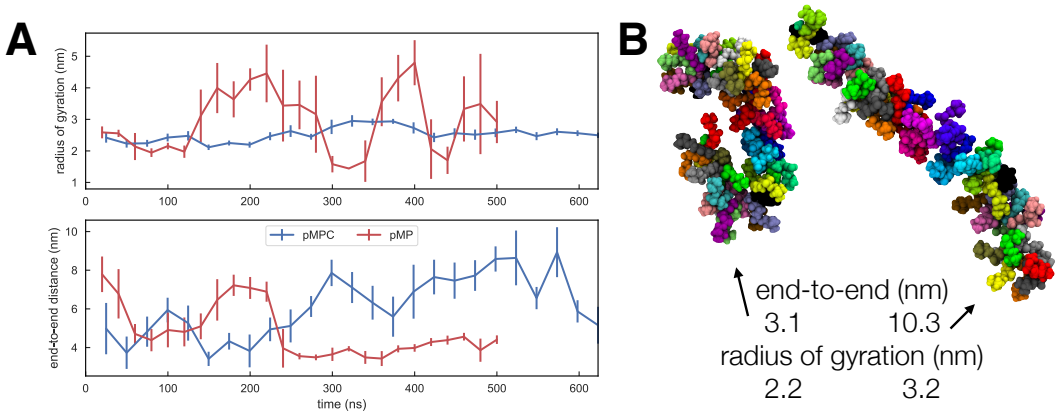


Figure 7.1: Backbone conformations of pMPC over time. A) Radius of gyration (top) and end-to-end distance of pMPC and pMP over time. B) Visualizations of pMPC in a compact (left) and extended (right) state. Colors represent individual monomers.

Firstly, with the choline groups primarily self-associating, they preclude further hydration of the polymer as compared to a fully extended confirmation that would allow for additional hydration shells to form around the choline group. Secondly, folding onto neighboring monomers creates a compact core around the backbone of the polymer which could contribute to pMPC's exceptional durability under shear.[6] Thirdly, to form these associations, the monomers form a kink at the phosphoryl group which exposes the two most negatively charged oxygens to the surrounding aqueous environment as described in the following section.

7.5 Polymer-Water Interactions

The conformation adopted by most monomers orients the two most negatively charged phosphoryl oxygens away from the polymer backbone. The RDFs in Figure 7.3 detail the hydration shell structure around those oxygens. At the shortest distances (˜0.17-0.35 nm),

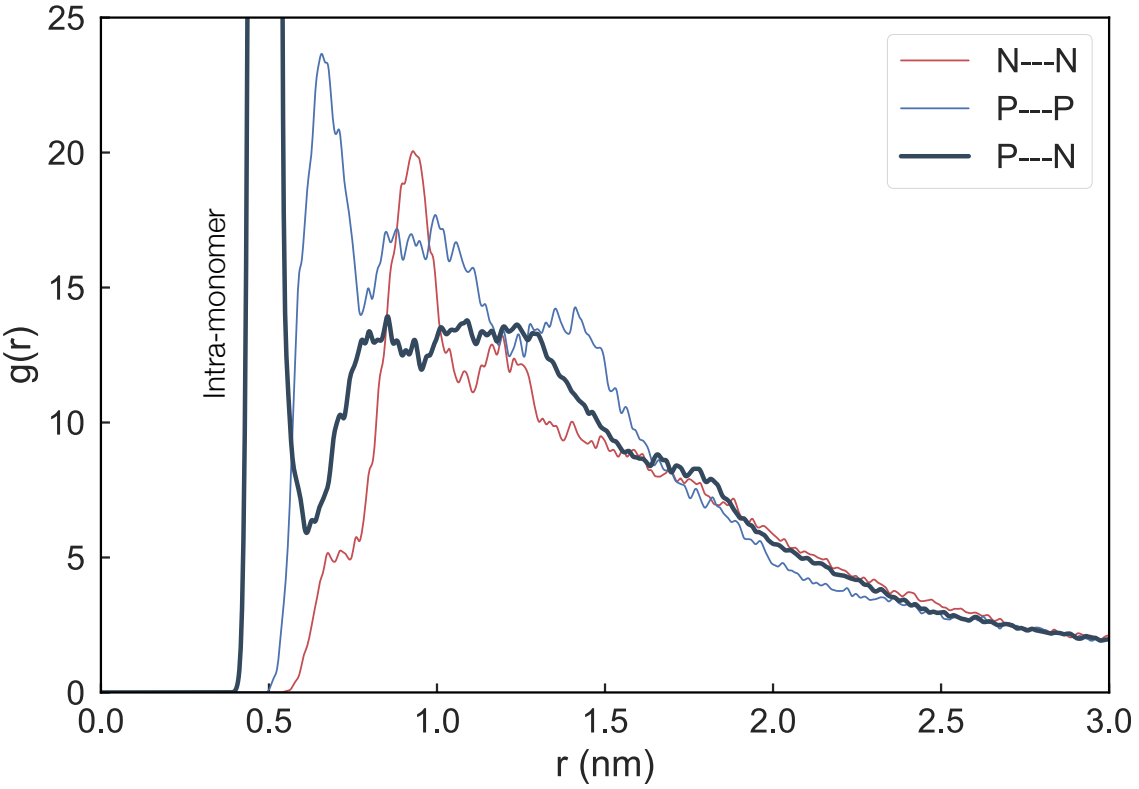


Figure 7.2: Relative intra-molecular spatial arrangement of nitrogen and phosphorus atoms. All three radial distribution functions between phosphorus and nitrogen atoms within $pMPC_{50}$molecule.

we observe a strong hydrogen bonding peak as evidenced both by the range of distances and the distinct separations of the oxygen and hydrogen peaks. Beyond these peaks, long range structure also exists in the water with distinct hydration shells forming out to at least 0.8 nm. This result suggests that if hydration lubrication does indeed play a significant factor in lubricating MPC-based materials, the phosphoryl group is likely the primary actuator of this lubrication mechanism.

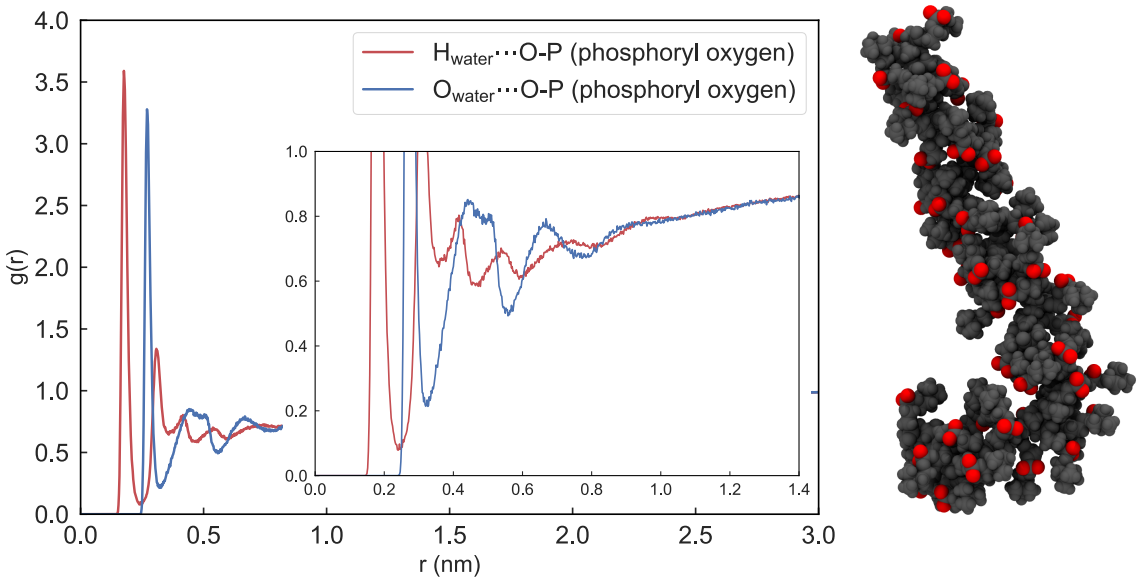


Figure 7.3: Radial distribution functions between water atoms and exposed oxygen atoms bound to phosphorus. The visualization highlights these exposed oxygens in red.

In comparison to the phosphoryl group, Figure 7.3 shows that water does not exhibit a long range structure around the positively charged choline group. This is attributable to 1) the positive charge being localized on the nitrogen atom which is shielded by hydrophobic methyl groups 2) the lack of exposed hydration bond acceptors or donors within the choline group and 3) the confirmation adopted by most monomers where the choline group folds onto its neighbors as discussed in section 7.4.

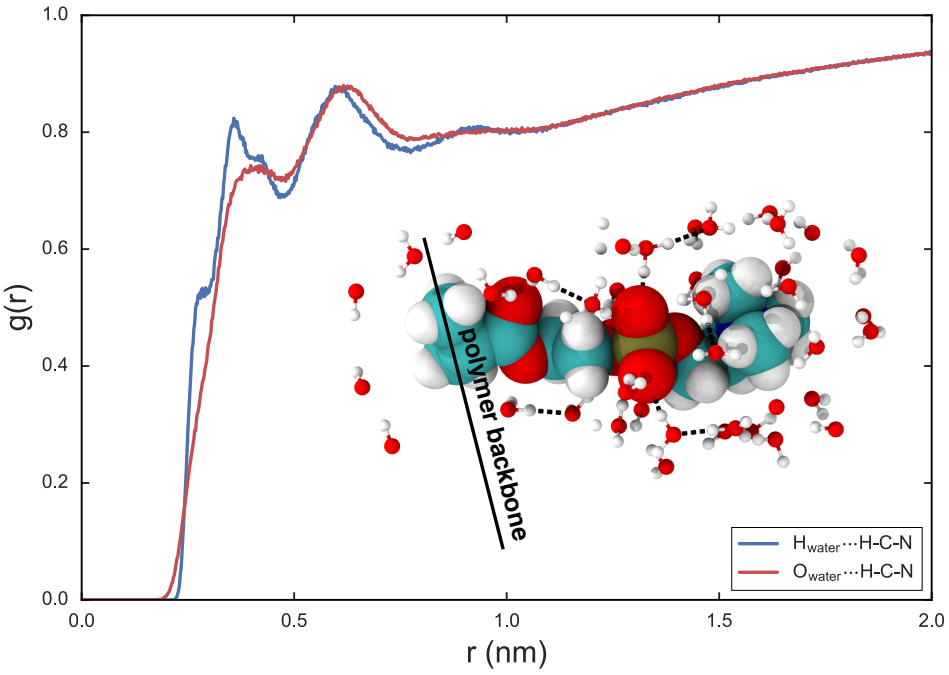


Figure 7.4: Radial distribution functions between water atoms and hydrogens in choline group. Inset depicts an example of a choline group not currently folded onto its neighbors but exposed to the aqueous environment with hydrogen bonds depicted as black dashed lines. Even so, the water is not significantly structured around the choline group.

7.6 Effect of Removing Choline Group

One of the key results of the Kyomoto et al.[6] study of materials for artificial joint lubrication is that removing the choline group from $pMPC_{50}$results in both an increase in friction coefficient and wear thus impacting both material performance and life-time. The results in Figure 7.5 show that the phosphorus containing moieties associate far more closely in pMP_{50}than in $pMPC_{50}$due to their ability to form hydrogen bonds with one another.

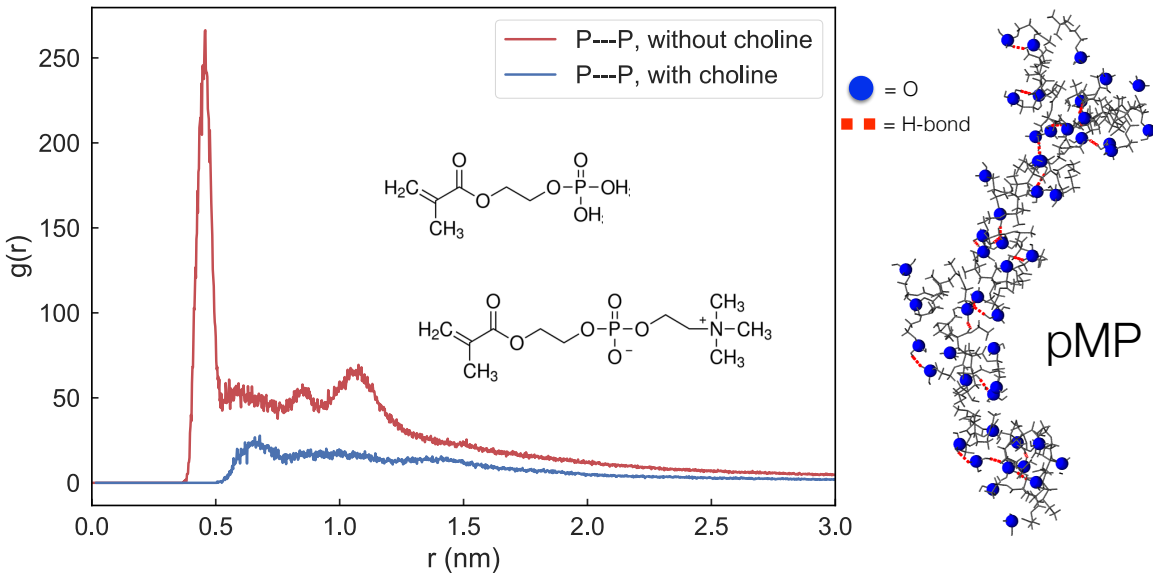


Figure 7.5: Radial distribution function between phosphorus atoms in pMPC and pMP. Inset chemical structures depict single monomers of the respective polymers. Visualization to the right of plot highlights phosphorus bound oxygens exhibiting hydrogen bonds with neighboring phosphoryl groups.

While a tight association between neighboring phosphate groups might suggest a more stable overall polymer structure, experimental results suggest otherwise.[6] Figure 7.6 shows that the phosphorus containing moieties are hydrated in pMP_{50}as compared to $pMPC_{50}$. This also diminishes the long range structured of hydration shells exhibited by $pMPC_{50}$as detailed in Figure 7.3. These results suggest that hydration shells play a crucial role in providing stability to the overall polymeric material.

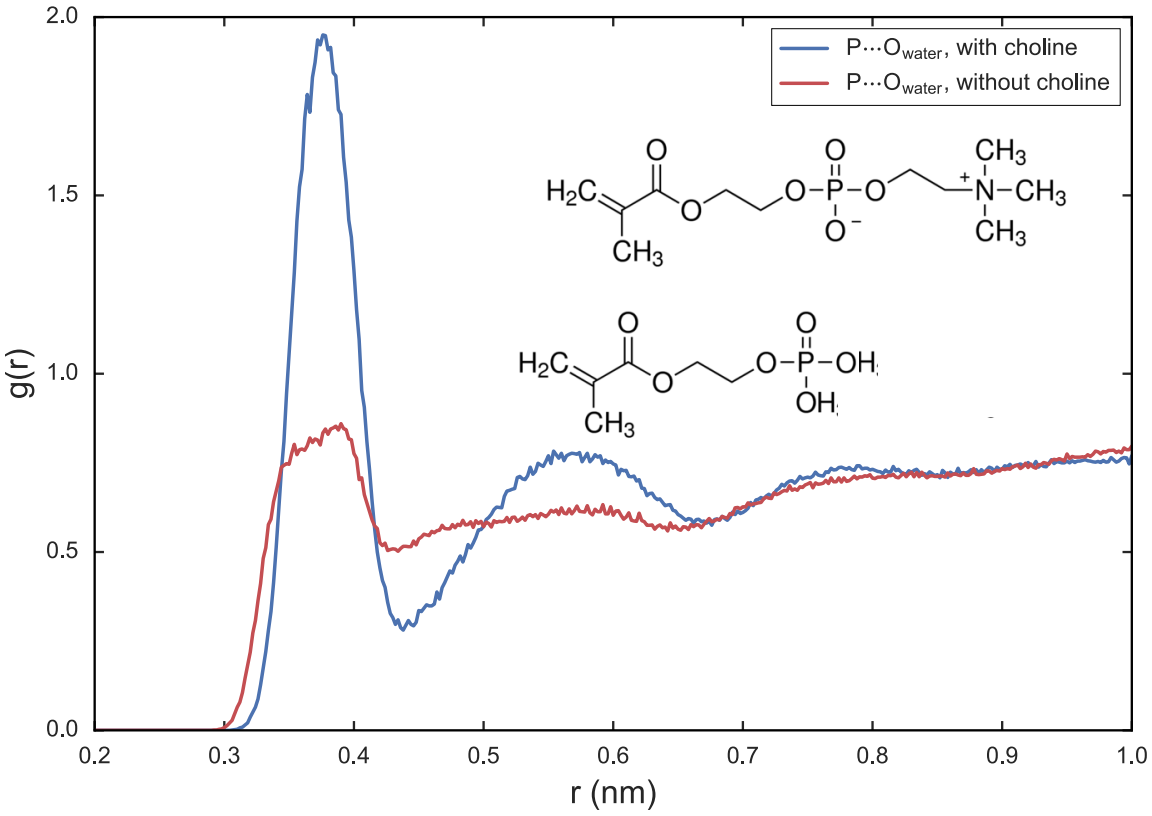


Figure 7.6: Radial distribution function between water oxygens and phosphorus atoms in pMPC and pMP. Inset chemical structures depict single monomers of the respective polymers.

7.7 Bibliography

[1] Klein, J. *Friction* **2013**, *1*, 1–23.

[2] Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *Journal of the American Chemical Society* **1996**, *118*, 11225–11236.

[3] Wang, L. P.; Martinez, T. J.; Pande, V. S. *Journal of Physical Chemistry Letters* **2014**, *5*, 1885–1891.

[4] Pronk, S.; Páll, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; Van Der Spoel, D.; Hess, B.; Lindahl, E. *Bioinformatics* **2013**, *29*, 845–854.

[5] McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. *Biophysical Journal* **2015**, *109*, 1528–1532.

[6] Kyomoto, M.; Moro, T.; Saiga, K.; Hashimoto, M.; Ito, H.; Kawaguchi, H.; Takatori, Y.; Ishihara, K. *Biomaterials* **2012**, *33*, 4451–9.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

Through the development of new tools that enable computationally exploring the chemical parameter spaces of soft matter systems, this dissertation provides insight into two classes of materials used for nanoscale lubrication and specifically into the hydration lubrication mechanism as it pertains to surface bound zwitterionic brush based materials. The tools developed towards this end were intentionally designed to be broadly applicable to constructing and executing molecular dynamics simulations and in particular to promote reproducibility and transparency in future simulation studies. They are already finding application in the study of non-surface functionalized nanoscale lubrications, tethered nanoparticles and ionic liquid based super-capacitors.

Chapter 3 describes `mBuild` - the first step in our workflow for simulating soft matter systems. `mBuild` provides a programmatic pathway to constructing arbitrary, complex input topologies for molecular simulations. The use of an equivalence operator typically eliminates the need for users to explicitly rotate or translate components while assembling chemical structures. The core data structures of `mBuild` and how the equivalence operator is implemented and used in practice are described and the pathway from basic component creation all the way through constructing several complex example hierarchies illustrated. The format-agnostic nature of `mBuild` allows for flexible interoperability with other tools in the scientific Python and molecular modelling communities, such as packmol,[1] polymatic,[2] MDTraj,[3] imolecule,[4] OpenMM[5] and HOOMD-blue.[6] Using monolayers as an example, the power of this approach is highlighted by performing a small parameter sweeping simulation study, demonstrating clear regimes of highly ordered monolayers which are likely correlated with favorable friction coefficients. This example demonstrates how this approach can be leveraged to more broadly study, design and optimize complex materials.

Source code and interactive tutorials in the Jupyter notebook format, which reinforce the basics of component construction and how to re-use components to assemble more complex systems, are also provided on the `mBuild` website (http://imodels.github.io/mbuild/).

Upon constructing a molecular system using `mBuild`, a force field is applied using the tool `Foyer` as described in Chapter 4. Foyer supports an extension of the OpenMM XML file format for force field specification that enables atomtypes to be defined based on the local chemical environment as encoded by SMARTS strings. Coupled with an iterative atomtyping scheme that removes the need for rigid rule hierarchies where more specific rules must appear first in the file, `Foyer` simplifies the force field development and dissemination process. In particular, `Foyer` promotes the dissemination of force fields via github and a template repository has been developed to assist new users in publishing their force fields in this fashion with a citable DOI, automated verification and validation testing, and a full version history for force fields that evolve over time.

Chapter 5 showcases a screening study across self-assembled monolayer parameter space by leveraging the capabilities of `mBuild` and `Foyer`. The chemistry of monolayer systems is easily tuned using `mBuild` which enables us to do a parameter sweep across chain length, chain type and substrate type to compare the relative impacts of these variables. Of particular note are the results obtained for alkane monolayers on β-cristobalite, an idealized representation of the amorphous silica often used to grow self-assembled monolayers. The parameter screens broad scope and the individual simulations' long simulation times as compared to previous studies revealed several pitfalls associated with using these idealized surfaces.

The results of Chapter 6 qualitatively support the hypothesized hydration lubrication mechanism where tight binding of water molecules to charged groups allows materials to support high normal loads while simultaneously allowing water molecules to rapidly relax between bound states thus providing a fluid shear response.[7] The results show that substrates sparsely functionalized with zwitterionic MPC monolayers in water provide strong

resistance to normal loads by allowing water molecules to swell into the monolayers and allowing strongly repulsive monomer-monomer interactions to resist compression. Simultaneously, when shearing at normal loads where water mobility remains high, shear stresses fluctuate about zero and yield friction coefficients as low as 0.014. Our results show two regimes with distinct shear responses. In the high shear stress regime, monomer-monomer interactions dominate the shear response which thus resembles the response of previously studied dense alkyl silane monolayers without water.[8–12] The low shear regime, where friction coefficients compare to those observed in experimental brush systems,[13–16] is characterized by a thin layer of water separating monomers in opposing substrates. Estimates of friction coefficients in these regimes approximately agree with other simulation and experimental studies of systems with respectively comparable environments. In the low shear regime, hydration shells surrounding opposing headgroups efficiently slide past one another by exchanging water molecules via the hydration lubrication mechanism. While experimental studies have thus far examined large polymer brushes composed of the monomers studied here, our simulations present a picture of the local environment surrounding individual monomers and how phenomena occurring on this scale result in favorable frictional properties. The examination of zwitterionic monolayers further demonstrates the importance of water mobility and how sparse zwitterionic monolayers behave differently than dense monolayer systems, both in their shearing mechanisms and their ability to sustain lower coefficients of friction to higher normal loads.

Chapter 7 provides further molecular level insight into the hydration lubrication mechanism and *how* it is likely to play a role in the tribology of zwitterionic brush layers comprised of poly(2-methacryloyloxyethyl phosphorylcholine), or pMPC. The charged choline groups of individual monomers tend to fold onto neighboring monomers. This has the threefold effect of (1) compacting the polymer, (2) excluding significant volume where hydration shells could otherwise form around the choline groups and (3) bending the monomers at the phosphate group thus preferentially exposing the two strongly nega-

tively charged oxygen atoms in the phosphate group to the aqueous environment. These results suggest that the predominant contribution to hydration lubrication in pMPC based materials is derived from water hydrating the phosphate group and that the choline group plays a lesser or perhaps negligible role. However, the strong association of the choline groups with neighboring monomers may serve to stabilize the overall polymer structure which would contribute to the high resistance to compression exhibited by hydrated pMPC brush layers.

8.1 Recommendations for Future Work

8.1.1 Screening Workflow Development

Both `mBuild` and `Foyer`, while functional and being actively used to conduct research, are still under active development. Aside from generally improving the reliability of the code base, several features are being actively worked on or have been identified as near term goals.

Prominent features in `mBuild`'s development pipeline include 1) automatically detecting and orienting binding sites for arbitrary surfaces, 2) carving arbitrary shapes from bulk solids, 3) expanding its coarse-graining functionality to include SMARTS based coarse-graining, 4) generating compounds from SMILES and/or SMARTS strings and 5) improving the bilayer and lattice builders.

`Foyer`'s verification and validation procedures are rapidly expanding in scope and improving in user friendliness. Several features of the SMARTS language are still being implemented and features to automatically generate bibliographies for manuscripts based on the force field parameters used are also being improved upon. Significant work is also being invested into improving the force field template repository and tutorial which serve to promote dissemination of force fields in more a reproducible manner.

8.1.2 Large Scale Self-Assembled Monolayer Screening

The full workflow for screening self-assembled monolayers is now robust enough to be applied to larger problems. A study currently being performed on Titan is exploring the impact of chain length and head group type over a space of 20 headgroups. Additionally, it is now easily possible to adjust 1) surface roughness, 2) surface types, 3) monomer backbones (e.g. to introduce intra-chain hydrogen bonding), 4) monolayers of mixed chain compositions and 5) explore the effects of shearing opposing monolayers with entirely different chemical compositions, thus greatly increasing the parameter space. Particular effort in this area should be placed into quantifying and discovering monolayer compositions that resist degradation as this is the primary hurdle towards wider spread adoption of self-assembled monolayers as nanoscale lubricants.

8.1.3 Hydration Structure of pMPC brush layers

The work presented here lends strong support to the hypothesized hydration lubrication mechanism, provides insight into the molecular level arrangements of the key moieties and hypothesizes how these arrangements could give rise to the observed macroscopic properties. However, it is still unclear how confining pMPC strands to a brush layer would alter the hydration structure of the material and how this larger complex responds to shear stresses. Using significantly larger molecular dynamics simulations (2-10 million atoms per system) than presented in this work, it should be possible to provide direct mechanistic insight into the dynamic behavior of water and interactions between neighboring pMPC strands that form the brush layers. The largest of these systems overlap with the smallest brush layer thicknesses that can be grown experimentally and should allow direct comparison between experimental and simulation studies.

8.1.4 Application of Screening Workflow to other Soft Matter Systems

The workflow described herein to screen self-assembled monolayers is readily adaptable to other soft-matter systems. In general, the workflow can be adapted to, and is suited to performing, any parameter screening or optimization study that requires adjusting the chemical composition of a system. Current work being pursued using `mBuild` and `Foyer` includes tethered nanoparticle design based on the spherical patterning methods in `mBuild`, studying the impact of lipid composition in bilayers found in the skin and ionic liquid based supercapacitor design. The application areas best served by `mBuild`'s capabilities, as opposed to other system initialization tools, involve working with substrates and in particular systems where those substrates are functionalized by another chemical species. An area of significant importance is adsorption of biological material such as proteins or DNA onto substrates for applications ranging from anti-fouling coatings to drug delivery. Our screening workflow could be adapted to, for example, run adsorption simulations of chemically functionalized surfaces interacting with proteins starting from 1) one or more PDB protein structures, 2) a substrate unit cell, 3) a SMILES representation of solvent(s), 4) a SMILES representation of the moieties functionalizing the surface and 5) parameters describing the surface film height and density.

8.2 Bibliography

[1] Martínez, L.; Andrade, R.; Birgin, E. G.; Martínez, J. M. *Journal of computational chemistry* **2009**, *30*, 2157–2164.

[2] Abbott, L. J.; Hart, K. E.; Colina, C. M. *Theoretical Chemistry Accounts* **2013**, *132*, 1–19.

[3] McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. *Biophysical Journal* **2015**, *109*, 1528–1532.

[4] Fuller, P. Available at https://github.com/patrickfuller/imolecule.

[5] Eastman, P. et al. *Journal of Chemical Theory and Computation* **2013**, *9*, 461–469.

[6] Anderson, J. A.; Lorenz, C. D.; Travesset, A. *Journal of Computational Physics* **2008**, *227*, 5342–5359.

[7] Klein, J. *Friction* **2013**, *1*, 1–23.

[8] Lewis, J. B.; Vilt, S. G.; Rivera, J. L.; Jennings, G. K.; McCabe, C. *Langmuir* **2012**, *28*, 14218–26.

[9] Booth, B. D.; Vilt, S. G.; Lewis, J. B.; Rivera, J. L.; Buehler, E. a.; McCabe, C.; Jennings, G. K. *Langmuir* **2011**, *27*, 5909–17.

[10] Rivera, J. L.; Jennings, G. K.; McCabe, C. *The Journal of Chemical Physics* **2012**, *136*, 244701.

[11] Chandross, M.; Webb, E.; Stevens, M.; Grest, G.; Garofalini, S. *Physical Review Letters* **2004**, *93*, 166103.

[12] Chandross, M.; Lorenz, C. D.; Grest, G. S.; Stevens, M. J.; Iii, E. B. W. *JOM* **2005**, *57*, 55–61.

[13] Kobayashi, M.; Terayama, Y.; Hosaka, N.; Kaido, M.; Suzuki, A.; Yamada, N.; Torikai, N.; Ishihara, K.; Takahara, A. *Soft Matter* **2007**, *3*, 740.

[14] Zhang, Z.; Morse, A. J.; Armes, S. P.; Lewis, A. L.; Geoghegan, M.; Leggett, G. J. *Langmuir : the ACS journal of surfaces and colloids* **2011**, *27*, 2514–21.

[15] Kyomoto, M.; Moro, T.; Saiga, K.; Hashimoto, M.; Ito, H.; Kawaguchi, H.; Takatori, Y.; Ishihara, K. *Biomaterials* **2012**, *33*, 4451–9.

[16] Banquy, X.; Burdyńska, J.; Lee, D. W.; Matyjaszewski, K.; Israelachvili, J. *Journal of the American Chemical Society* **2014**, *136*, 6199–202.