UC Merced
UC Merced Electronic Theses and Dissertations

Title
Viscosity of Nanoconfined Fluids and the Effect on Elastohydrodynamic Lubricated Interfaces

Permalink
https://escholarship.org/uc/item/2qw846j7

Author
Len, Michelle

Publication Date
2016

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, MERCED

Viscosity of Nanoconfined Fluids and the Effect on Elastohydrodynamic Lubricated Interfaces

A Thesis submitted in partial satisfaction of the requirements
for the degree of Master of Science

in

Individual Graduate Program with emphasis in
Mechanical Engineering

by

Michelle Len

Committee in charge:

Professor Ashlie Martini, Chair
Professor Venkattraman Ayyaswamy
Professor Yanbao Ma

2016

© Michelle Len, 2016
All rights reserved.

This Thesis of Michelle Len is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Ashlie Martini, Chair	Date
Venkattraman Ayyaswamy	Date
Yanbao Ma	Date

University of California, Merced

2016

To my parents.

ABSTRACT

The elastohydrodynamic lubrication (EHL) regime is characterized by the elastic deformation of the bodies in contact, hydrodynamics of viscous friction caused by shearing the lubricant, and dramatic viscosity increase due to high pressure. As technology for mechanical components advances towards higher energy efficiency and less energy loss from viscous friction, this advancement implies that lubricant films are decreasing towards thicknesses on the nanometer scale. However, these lubricant films may behave differently from what is predicted by classical EHL theory. While viscosity is affected by temperature, pressure, and shear rate, researchers have discovered that there is a new dependence of film thickness on viscosity, indicating that extreme confinement in EHL can affect the viscosity of the lubricant. In this work, we attempt to characterize the relationship between film thickness and viscosity for squalane using molecular dynamics (MD) simulations so that we may observe how the lubricant behaves with film thicknesses on the nanometer scale. However, we were unable to successfully obtain this relationship due to issues with the MD simulations. In addition, we attempt to improve the ability of EHL simulations to accurately capture the behavior of ultra-thin film lubricated interfaces by incorporating empirical models that describe the effect of confinement on fluid viscosity. The enhanced model reproduces trends seen in some EHL experiments where the film thickness of very thin film lubricants is larger than expected compared to classical EHL theory. Our enhanced EHL simulation successfully produces film thickness results with improved ability to reproduce to experimental data.

TABLE OF CONTENTS

ABSTRACT . . . . . **ii**
LIST OF FIGURES . . . . . **vii**
LIST OF TABLES . . . . . **x**
LIST OF SYMBOLS . . . . . **xi**

1 INTRODUCTION **1**

1.1 Background 1

1.1.1 Motivation 1
1.1.2 Lubrication Regimes 1

1.1.2.1 Boundary Lubrication 2
1.1.2.2 Mixed Lubrication 3
1.1.2.3 Hydrodynamic Lubrication 3
1.1.2.4 Elastohydrodynamic Lubrication 4

1.2 EHL 4

1.2.1 Analytical Model 5
1.2.2 Numerical Simulation 6
1.2.3 Experimental Measurement 6
1.2.4 Deviation from Classical Theory 6

1.3 Viscosity 8

1.3.1 Parameters that Affect Viscosity 9
1.3.2 Temperature 9
1.3.3 Pressure 10

1.3.4 Shear Rate 10
1.3.4.1 Shear Thickening 11
1.3.4.2 Shear Thinning 11
1.4 Film Thickness Effects on Viscosity 12
1.4.1 Experimental Work 12
1.4.2 Simulation Work 13
1.4.3 Summary of Observations 15
1.5 Research Objective & Thesis Outline 15
2 METHODS 17
2.1 Molecular Dynamics 17
2.1.1 Simulation Platform 18
2.1.2 Molecular Construction Platform 18
2.1.3 Simulations of Confined Fluids 18
2.1.3.1 Simple Model 18
2.1.3.2 Molecular Model 19
2.2 Numerical EHL Simulation 21
2.2.1 Input Parameters 21
2.2.2 Initial Estimates 22
2.2.3 Reynolds Equation 23
2.2.4 Pressure–Viscosity Model 24
2.2.5 Pressure–Viscosity Coefficient 25
2.2.6 Numerical Procedure 26
2.2.7 Equation for Viscosity Dependence on Film Thickness 26
2.2.8 Enhanced EHL Simulation for Thin Films 27
3 RESULTS 28
3.1 MD Predictions of Viscosity vs. Film Thickness 28
3.1.1 Simple Model Viscosity vs. Film Thickness Predictions 28

3.1.2 Molecular Model Viscosity vs. Film Thickness Predictions . . 29

3.2 Numerical EHL Simulation 31

3.2.1 Validation of the Traditional Model 31
3.2.2 Enhanced EHL Simulation for Thin Films 34

4 CONCLUSIONS AND RECOMMENDATIONS 38

4.1 Research Summary 38
4.2 Areas for Further Study 38

4.2.1 MD Simulation Issues & Areas of Improvement 39

4.2.1.1 Ultra–Thin Films & Computational Limitations . . . 39
4.2.1.2 Signal–to–Noise Ratio & RNEMD 40
4.2.1.3 Alternative Lubricant Selection 41

4.2.2 EHL Simulation Issues & Areas of Improvements 41

4.2.2.1 Convergence & Error 42
4.2.2.2 Optical Interferometry Accuracy 43
4.2.2.3 Optical Interferometry Phase Change 44

4.3 Conclusion 44

BIBLIOGRAPHY 46

APPENDIX

A LAMMPS INPUT FILES 52

A.1 run.nemdljwall 52
A.2 run.sqaequilibrate 53
A.3 run.sqashearwalls 54

B MOLTEMPLATE FILES 56

B.1 ch1group.lt 56
B.2 ch2group.lt 56
B.3 ch3group.lt 57

B.4 squalane.lt 57
B.5 oplsaa.lt 59
B.6 system.in.charges 63
B.7 system.in.init 63
B.8 system.in.settings 63

C POST–PROCESSING SCRIPTS 66

C.1 ljviscshear.py 66
C.2 sqaviscshear.py 72
C.3 plotEHLph.py 79

LIST OF FIGURES

1.1 Illustration of the energy consumption of a passenger car 2

1.2 Stribeck curve showing the relationship of the Hersey number on the friction coefficient μ 3

1.3 Image of film thickness obtained from a ball–on–disc EHL experiment produced by using optical interferometry 5

1.4 Cross section schematic of EHL film thickness and pressure with respect to the fluid entrainment direction 5

1.5 Schematic of an EHL ball–on–disc test rig 7

1.6 Examples of experiments where the measured film thicknesses deviated from the classical theory. The symbols in the left plot represent film thicknesses of cyclohexane obtained from experiments and the dotted line represents the Hamrock–Dowson theory. The open symbols in the right plot indicate film thicknesses of squalane obtained from experiments by Hartl et al. and the solid symbols represent the film thicknesses obtained from numerical simulations with the lines used as a guide 9

1.7 A modified surface forces apparatus was used to measure viscosity of polybutadiene. Viscosity increases with decreasing film thickness for very thin films of polybutadiene 13

1.8 Summary of the behavior of the viscosity of a bulk and confined fluid 14

2.1 Model of Lennard–Jones fluid at film thicknesses of $h' = 5$ (Left) and $h' = 9$ (Right) generated by a free visualization software called OVITO. The walls are represented by gray spheres and the fluid is represented by blue spheres 19

2.2 The squalane molecule illustrated using OVITO. The carbon atoms are represented in blue and the hydrogen atoms are represented in purple. 20

2.3 The process for creating a confined squalane simulation is shown. The carbon atoms are represented in blue and the hydrogen atoms are represented in purple. The first step is to create the squalane molecule in Moltemplate. Next, the simulation is set up in LAMMPS. Then, system is equilibrated and compressed. Lastly, the system is sheared by applied velocities to the walls. All images were generated using OVITO 21

2.4 Diagram of algorithm for classical EHL simulation solver developed by Hu and Zhu in 2000 where the prime symbols ($\prime$) indicate normalized values 22

2.5 Diagram of algorithm for enhanced EHL simulation incorporating film thickness dependence on viscosity (Equation 2.16). 27

3.1 Viscosity as a function of shear rate for two film thicknesses. The data points were obtained from MD simulations and the dashed lines indicate fits to the Carreau viscosity model. All values are in reduced Lennard–Jones units. See Appendix C.1 for the Python script used to post–process the data. 29

3.2 Effective viscosity with respect to effective shear rate. See Appendix C.2 for the Python script used to post–process the data. 31

3.3 Actual viscosity with respect to actual shear rate. See Appendix C.2 for the Python script used to post–process the data. 32

3.4 Contour plot of the relative errors of the central film thickness obtained from our enhanced EHL simulation compared to the expected experimental film thickness value. 35

3.5 Comparison of the film thicknesses produced by our EHL solver, the values obtained by Hartl et al. and the original experimental values. The open symbols represent the experimental data obtained by Liu et al. The black symbols represent the film thickness values obtained from our original traditional EHL simulation. Lastly, the red symbols represent the film thickness values obtained from our enhanced EHL simulation that accounts for film thickness dependence on calculating viscosity 36

4.1 Plot of the reduced viscosity versus reduced shear rate for experimental and simulation data of squalane 39

4.2 Velocity profile vs. x for the 200 molecule system (1.29 nm) with an imposed wall speed of 0.8 m/s. 41

4.3 Velocity profile vs. x for the 200 molecule system (1.29 nm) with an imposed wall speed of 200 m/s. 41

4.4 Schematic of the RNEMD method 42

4.5 For values of $A = 205$ and $t = 1.5$, $\Delta p'$ converges to 9.96×10^{-7}. . . 43

4.6 For values of $A = 205$ and $t = 1.5$, h_{cen} converges to 6.224 nm. . . . 43

LIST OF TABLES

3.1 Reduced parameters obtained from fitting our simulation data of two different film thicknesses to the Carreau viscosity model. 30

3.2 Validation of our EHL simulation to the literature data provided by Liu et al. using the Roelands viscosity model and a pressure–viscosity coefficient $\alpha_{film} = 20.15$ GPa^{-1} 33

3.3 Validation of our EHL simulation to the literature data provided by Liu et al. using the Barus viscosity model and a pressure–viscosity coefficient $\alpha_{film} = 20.15$ GPa^{-1} 33

3.4 Validation of our EHL simulation to the literature data provided by Liu et al. using a 256×256 mesh, the Barus viscosity model and a pressure–viscosity coefficient $\alpha_{film} = 20.15$ GPa^{-1} 34

3.5 Fitted parameters to the simplified Hamrock–Dowson equation. . . 34

3.6 Calculated percent relative errors between our simulation film thicknesses and the experimental data. h_{cen} refers to the central film thickness and h_{min} refers to the minimum film thickness 37

LIST OF SYMBOLS

μ	Friction coefficient	2
η	Viscosity $[Pa \cdot s]$	2
ω	Rotational speed $[1/s]$	2
p	Pressure $[Pa]$	2
h_{cen}	Central film thickness [m]	5
h_{min}	Minimum film thickness [m]	5
U	Entrainment speed otherwise known as entraining surface velocity that is an average between the velocities of the contact surfaces [m/s]	5
η_0	Ambient viscosity or viscosity at atmospheric pressure [Pa·s]	5
E'	Reduced Young's modulus	5
R'	Reduced radius of curvature	5
α	Pressure–viscosity coefficient [m^2/N]	5
W	Load [N]	5
k	Non–dimensional ellipticity parameter consisting of the ratio between the semi–axes in the transverse direction and direction of motion, a and b, respectively	5
τ	Shear stress $[Pa]$	8
$\dot{\gamma}$	Shear rate $[s^{-1}]$	8
ν	Kinematic viscosity $[m^2/s]$	10
T	Temperature $[K]$	10
η_p	Viscosity at pressure 'p' [Pa·s]	10
η_∞	Viscosity at infinite shear rate $[Pa \cdot s]$	11
λ	Relaxation (characteristic) time constant $[s]$	11
n	Power-law behavior constant for the Carreau model	11
σ	Finite distance at which the inter-particle potential is zero, often used to indicated reduced Lennard–Jones units	13
r	The distance separating a pair of atoms, used in the Lennard–Jones potential equation	17
K	Harmonic spring force	17
l	Bond length	17
θ	Bond angle	17
ϕ_i	i^{th} dihedral angle	17
V_j^i	j^{th} coefficient in the Fourier series	17

p_h	Hertzian pressure [Pa]	22
a	Hertzian contact radius [m]	22
V	Dimensionless elastic deformation	22
Ω	Solution domain used in Boussinesq integration	23
$\overline{\xi}$	Corresponding x value in the computational domain for calculating elastic deformation V	23
$\overline{\zeta}$	Corresponding y value in the computational domain for calculating elastic deformation V	23
ρ	Density [kg/m^3]	23
u	Velocity component in the x–direction	23
v	Velocity component in the y–direction	23
w	Velocity component in the z–direction	23
X_a	Body force needed to accelerate a fluid element in the x–direction	23
Y_a	Body force needed to accelerate a fluid element in the y–direction	23
Z_a	Body force needed to accelerate a fluid element in the z–direction	23
ξ_a	Dilatation otherwise known as the measure of fluid expansion	23
ρ'	Dimensionless density normalized from the density at ambient conditions $\rho' = \rho/\rho_0$	24
ρ_0	Ambient density or density at atmospheric pressure [kg/m^3]	24
η'	Dimensionless viscosity normalized from the density at ambient conditions $\eta' = \eta/\eta_0$	24
η_g	Glass transition viscosity [Pa·s]	25
A_1	Yasutomi constant used in the Yasutomi free–volume viscosity model	25
A_2	Yasutomi constant used in the Yasutomi free–volume viscosity model	25
B_1	Yasutomi constant used in the Yasutomi free–volume viscosity model	25
B_2	Yasutomi constant used in the Yasutomi free–volume viscosity model	25
C_1	Yasutomi constant used in the Yasutomi free–volume viscosity model	25
C_2	Yasutomi constant used in the Yasutomi free–volume viscosity model	25
α_{film}	New definition of the pressure–viscosity coefficient defined by Liu et al. [GPa^{-1}]	25
α^*	Blok's reciprocal asymptotic isoviscous pressure–viscosity coefficient [GPa^{-1}]	25
ϕ	The net phase change that occurs upon reflection in an EHL experiment that uses optical interferometry	44

Chapter 1

INTRODUCTION

1.1 Background

In this section, we will discuss the motivation and background associated with this project. At the end of the section, our research objectives will be described and contents of this thesis will be outlined.

1.1.1 Motivation

An important challenge in mechanical engineering and tribology is to increase the efficiency of machine components. Friction is a significant cause of material wear and energy loss. Minimizing friction can reduce material wastage and increase mechanical performance [1]. One industry where friction is prevalent is the automotive industry. For example, in a passenger car, fuel energy is divided into the various functions needed to operate a car, as shown in Figure 1.1. 33% of fuel energy is due to friction losses that come from mechanical systems that make up the components of the car, such as the engine, transmission, and brakes. While the 33% of friction loss comes from a variety of sources, such as tire-road contact and braking contact, 23.3% of the loss comes from viscous friction [2]. In another study, the energy loss in a paper production mill was analyzed. It was found that 32% of friction loss comes from sources such as bearings and gears. However, 13.9% of that friction loss comes from viscous friction as well [3]. Technology is advancing towards higher energy efficiency in mechanical components and less energy loss from viscous friction. To reduce viscous friction, components are using less viscous lubricants. This advancement implies that lubricant films are becoming thinner. However, these thin lubricant films may behave differently from what is predicted by classical theory, in part because extreme confinement can affect the viscosity of the lubricant. Therefore, it is important to understand the behavior of thin film lubricated interfaces.

1.1.2 Lubrication Regimes

When two surfaces come in contact and slide relative to each other, both surfaces experience friction, which can cause wear. Lubricants were developed to help mitigate friction and wear [4]. A majority of lubricants are liquids composed

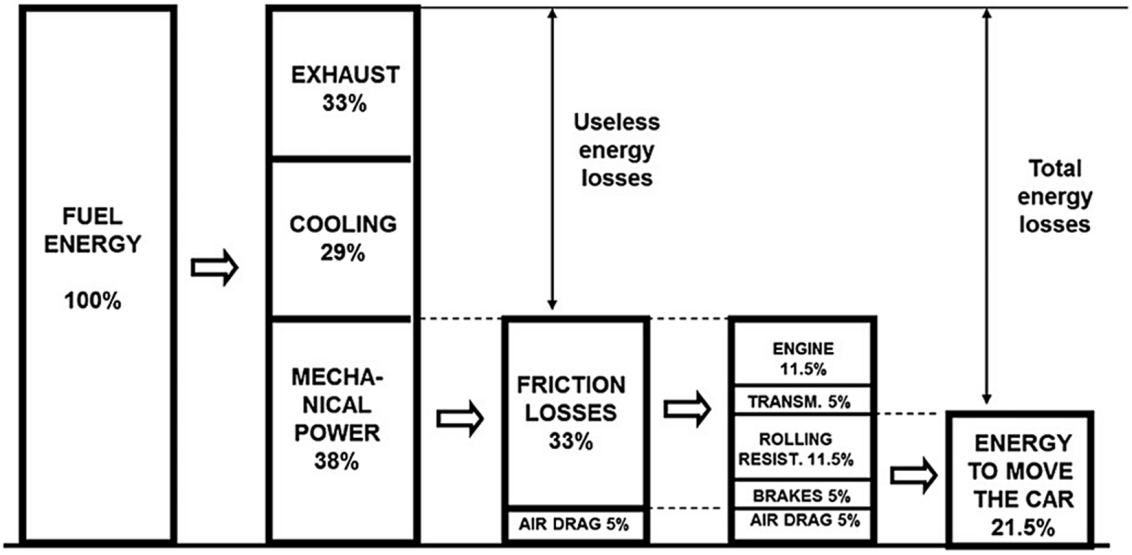

Figure 1.1: Illustration of the energy consumption of a passenger car from Reference [2].

of either refined mineral oils or synthetic oils such as polyalphaolefins, polyolesters, and polyglycols [5]. Lubricants act as a layer between two sliding surfaces to enable smooth motion and reduce shear stress [6]. In a lubricated system, there are four possible lubrication regimes that a lubricated system can function in: boundary, mixed, hydrodynamic, and elastohydrodynamic. These lubrication regimes vary by component velocity, pressure and lubricant viscosity. The Stribeck curve as shown in figure 1.2 depicts an overall view of the friction variation in the entire range of possible lubrication regimes. The Stribeck curve shows that friction due to viscous shear, denoted by μ, varies with the value of the dimensionless Hersey number, which is a product of the viscosity (η) and rotational speed (ω) divided by the average pressure (p) [7].

1.1.2.1 Boundary Lubrication

In the boundary lubrication regime, the layer of lubricant that was initially separating the two surfaces is "squeezed" out, meaning that the surfaces are not separated by lubricant, and there is a large amount of asperity contact and a high amount of friction. However, boundary lubrication does not mean that the system is unlubricated. Even though the friction coefficient is high, it is not as high as it would be if the two surfaces were completely unlubricated. The film thickness varies between 1 nm and 10 nm, depending on the molecular size of the lubricant [4]. The

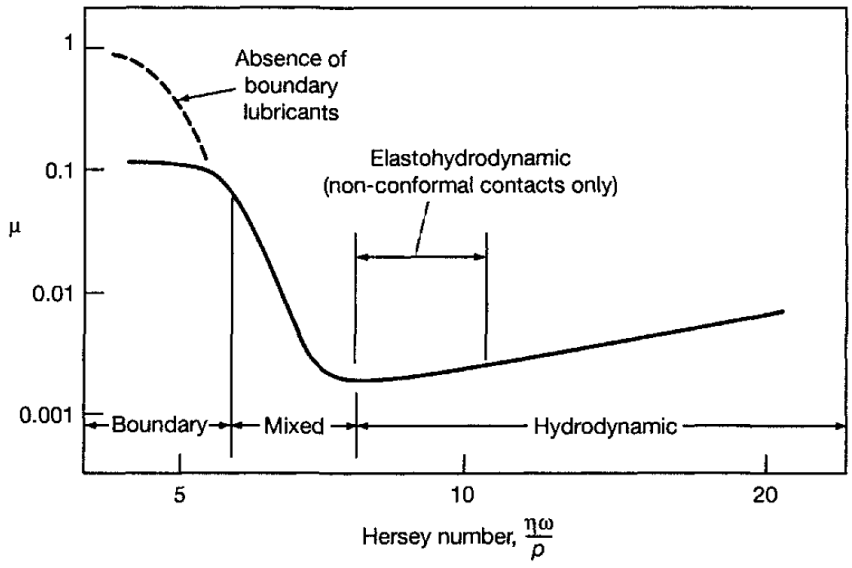

Figure 1.2: Stribeck curve showing the relationship of the Hersey number on the friction coefficient μ from Reference [4].

possible causes for a lubricated system to enter the boundary lubrication regime are high pressure, low shearing speed, and low lubricant viscosity [6].

1.1.2.2 Mixed Lubrication

The mixed lubrication regime, also known as the partial lubrication regime, is characterized by a mixture of asperity contact and lubrication. This means that there are some regions where there is no lubricant with surface-surface contact and other regions where a lubricant film is present. Similar to boundary lubrication, high pressure and low operating speeds are the causes for the asperity contact in mixed lubrication. However, the mentioned conditions are not as extreme as what would be needed to create boundary lubrication. In the lubricant layer where there is no asperity contact, the bulk fluid is still intact. The average film thickness in a mixed lubrication system is between 10 nm and 1000 nm (0.01 µm and 1 µm) [4].

1.1.2.3 Hydrodynamic Lubrication

The hydrodynamic lubrication regime is usually found on conformal surfaces with a large lubrication area. The pressure applied to the system is typically no larger than 5 MPa, and the minimum film normally exceeds 1000 nm (1 µm). In oil bearings, the film thickness ranges from 1 µm to 1000 µm (1 mm) [6]. The film thickness is generally thick enough to where the two surfaces never come into contact,

which means that friction is low and there is wear. The friction that is present in this lubrication regime comes from the viscous friction caused from shearing the lubricant [4].

1.1.2.4 Elastohydrodynamic Lubrication

The elastohydrodynamic lubrication (EHL) regime lies between the mixed lubrication and hydrodynamic lubrication regimes. EHL is similar to hydrodynamic lubrication, but contains three specific characteristics: hydrodynamics, elastic deformation of the two surfaces, and a viscosity increase under extreme operating pressure [1]. While the pressure in the hydrodynamic lubrication regime is no larger than 5 MPa, the pressure at which an EHL system operates is between 0.5 GPa and 3 GPa [4]. Similar to the mixed lubrication regime, the film thickness in EHL is also between 10 nm and 1000 nm (0.01 µm and 1 µm). Figure 1.3 shows a film thickness image obtained from an EHL ball–on–disc experiment [8]. The "ears" seen in the figure are unique to EHL experiments. Each color in the figure represents film thicknesses of different values. Unlike other lubrication regimes, the pressure in an EHL regime causes a steep increase in lubricant viscosity, causing changes in film thickness as well. Examples of engineering applications that operate under EHL are rolling ball bearings, gears, and engine cams. In rolling contact bearings, the balls in a ball bearing roll against their race. In gears, the gear teeth rub against each other. In engine cams, the cams rub against a follower [6]. The common characteristic among these applications is that the surfaces are nonconformal, resulting in small lubrication contact areas and high pressures [1]. This lubrication regime is the focus of this thesis and will be discussed in more detail.

1.2 EHL

As mentioned previously, an EHL film is created by three effects: hydrodynamic film formation, elastic deformation of contacting bodies, and the increase of the lubricant's viscosity under pressure. An EHL interface is characterized by the distributions of pressure and film thickness in the contact. Pressure is defined as the force per area that is exerted from the fluid onto the surface, and film thickness is defined as the height of the layer of lubricant that separates two moving bodies in a mechanical system, respectively. Figure 1.4 shows the cross section of the film thickness and pressure along the x axis which is the direction of the lubricant entrainment. In the figure, there is a pressure spike, p_s, at the exit region of the contact that is a characteristic of EHL [10]. In the figure, the minimum film thickness occurs at the pressure spike and is represented by h_m. The central film thickness is the film thickness found at the center of contact, represented by h_c. In the context of this work, we will be using h_{min} and h_{cen} to represent the minimum and central film thicknesses, respectively. The pressure can be calculated from numerical approximations to the Navier–Stokes equations. However, we will be primarily focusing on

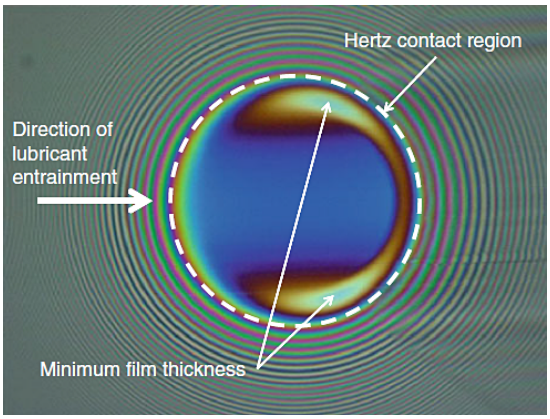

Figure 1.3: Image of film thickness obtained from a ball–on–disc EHL experiment produced by using optical interferometry [8].

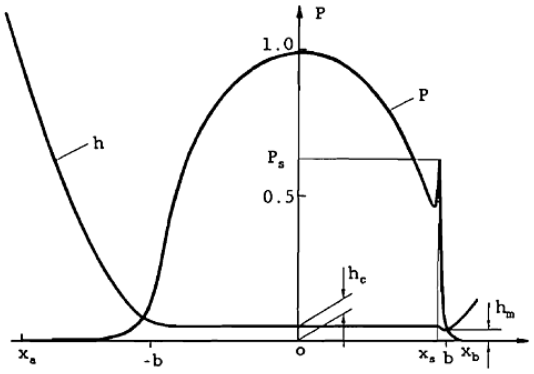

Figure 1.4: Cross section schematic of EHL film thickness and pressure with respect to the fluid entrainment direction [9].

film thickness. Two methods for determining film thickness are analytical models and numerical simulations. In addition, film thickness can also be measured directly in experiments [1].

1.2.1 Analytical Model

The most widely used theoretical EHL film thickness predictions came from Hamrock and Dowson, who created formulas for determining the central and minimum film thicknesses of any contact, such as point, linear or elliptical [11]. They can also be used to make reasonable estimates for many different material combinations, such as steel on steel surfaces, for pressures up to 4 GPa. The equations Hamrock and Dowson derived for the central and minimum film thicknesses of an EHL interface are the following:

$$\begin{aligned} \frac{h_{cen}}{R'} &= 2.69 \left(\frac{U\eta_0}{E'R'}\right)^{0.67} (\alpha E')^{0.53} \left(\frac{W}{E'R'^2}\right)^{-0.067} \left(1 - 0.61\exp^{-0.73k}\right) \\ \frac{h_{min}}{R'} &= 3.63 \left(\frac{U\eta_0}{E'R'}\right)^{0.68} (\alpha E')^{0.49} \left(\frac{W}{E'R'^2}\right)^{-0.073} \left(1 - \exp^{-0.68k}\right) \end{aligned} \tag{1.1}$$

where h_{cen} is the central film thickness, h_{min} is the minimum film thickness, U is the entrainment speed, η_0 is the ambient viscosity, E' is the reduced Young's modulus, R' is the reduced radius of curvature, α is the pressure–viscosity coefficient, W is the contact load, and k is the non–dimensional ellipticity parameter consisting of the ratio between the semi–axes in the transverse direction and direction of motion [1].

1.2.2 Numerical Simulation

Numerical simulations are also used to predict the film thickness of a lubricant subjected to EHL conditions. The simulations take an input of physical, material, and lubricant parameters. The simulation is run over a user–defined computational domain, where equations such as elastic deformation, film thickness, viscosity as a function of pressure, density, are solved analytically and the Reynolds equation is solved using numerical approximations. Numerical simulations not only give the central and minimum film thicknesses, but they also give a distribution of the film thickness, viscosity, and pressure throughout the lubricant domain. The details of our EHL simulation will be discussed further in the Methods chapter.

1.2.3 Experimental Measurement

Apart from analytical models and numerical simulations, there are experimental methods for predicting the film thickness of an EHL system. A majority of EHL research emerged in the second half of the 20th century, beginning in the 1950s [12]. While different kinds of methods were developed, the most effective method was optical interferometry. In 1961, Archard and Kirk discovered the optical interferometry method when they found that colored interference fringes appeared when shining a white light on the contact zone in their initial experimental setup [13]. In 1966, Cameron and Gohar, conducted their experiment using an apparatus consisting of a steel ball and glass disc with the optical interferometry. They found that the different colors in the fringes produced by the light gave film thicknesses in the contact region [14]. In 1996, the optical interferometry method was refined to yield better precision in measuring film thicknesses by using a chromium coated glass disc [15]. The steel ball and chromium glass disc apparatus coupled with optical interferometry became the standard procedure for modern EHL experiments [12].

1.2.4 Deviation from Classical Theory

The classical method of using the Hamrock–Dowson equations (Equation 1.1) or numerical EHL simulations, works well for thick lubricant films [11]. In 2009, Ma and Zhang measured film thicknesses of 80 cSt dimethylsilicone oil using relative optical interference intensity from a traditional ball–on–disc test rig [16]. An example of the ball–on–disc test rig is shown in Figure 1.5. Lubricant floods the space between the steel ball and glass disc. As the ball and disc rotate, the contact is illuminated by white light and filtered with an interference filter to obtain film thickness values [17]. Ma and Zhang saw good agreement to the Hamrock–Dowson prediction using speeds ranging from roughly 0.006 m/s to 1.75 m/s. Their resulting film thicknesses ranging from roughly 70 nm to 600 nm. However, thin films do not agree well with the Hamrock–Dowson prediction.

One of the first EHL experiments where researchers noticed a distinct deviation from the classical Hamrock–Dowson theory was in 2000, by Spikes and Rotoi.

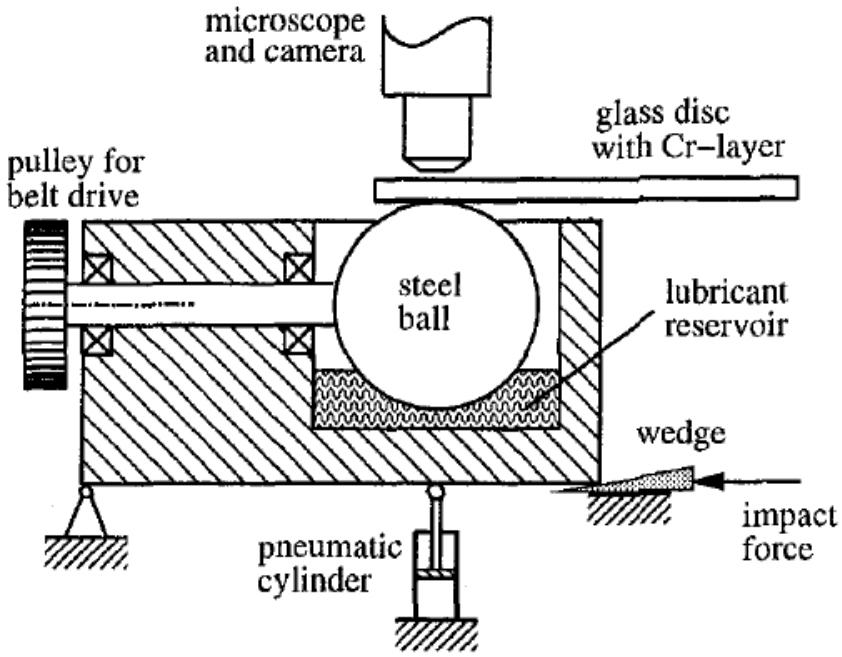

Figure 1.5: Schematic of an EHL ball–on–disc test rig from Reference [17].

They measured a speed range from approximately 0.0001 m/s to 1.25 m/s for a lubricants cyclohexane and octamethylcyclotetrasiloxane (OCMTS) using optical interferometry on a surface forces apparatus (SFA) where the lubricant was sheared in a ball–on–disc test rig with chromium and silica coated glass [18]. They found that, in their range of experimental film thicknesses from 1 nm to 20 nm, the film thicknesses began to deviate from the continuum theory at about 7 nm, where the film thicknesses were larger than the theory, as shown on the left of Figure 1.6. In 2001, Shen et al. conducted an experiment using relative optical interference intensity on a ball–on–disc test rig. They tested an unspecified liquid paraffin with four different coatings covering the glass disc. They measured speeds ranging from 0.3 m/s to 40 m/s with resulting film thicknesses ranging from 7 nm to 45 nm [19]. They found that the film thickness at which their experimental data began to deviate from the Hamrock–Dowson theory occurred at roughly 25 nm. Also in 2001, Hartl et al. measured the effect of entrainment speed on film thickness for hexadecane, di-(2-ethylhexyl)phthalate, squalane, a paraffinic base oil and a paraffinic base oil with triolein and stearic acid using thin film colorimetric interferometry. The speeds tested ranged from 0.001 m/s to 10 m/s with resulting film thicknesses ranging from 1 nm to 500 nm depending on the fluid [20]. Hartl et al. saw variations from the

classical theory in both the central and minimum film thicknesses for all fluids. In 2007, Pierre et al. studied a range of Newtonian and non–Newtonian fluids: an unspecified mineral paraffinic base oil, squalane, pennzane, Z25, PAO650, 5P4E and Santotrac 50. They used an EHL ball–on–disc test rig with a steel ball and a transparent glass disc. Hartl et al. found that, for both their unspecified mineral oil and squalane, the Hamrock–Dowson theory gave acceptable central film thickness predictions. The experimental data deviated from the classical theory at roughly 10 nm for the central film thickness for both lubricants. The deviation from the classical theory and the minimum film thickness occurred at roughly and roughly 60 nm for the mineral oil and 50 nm for squalane [21]. More recently, in 2015, Guegan et al. also found good film thickness agreement with the classical theory measuring film thicknesses for Shell Turbo 68, a Group II base oil with antioxidant additives placed in a traditional EHL ball–on–disc apparatus containing a smooth steel ball and glass disc rig [22]. Guegan et al. tested a larger range of speeds between 0.004 m/s and 3 m/s with resulting film thicknesses ranging from 10 nm to 750 nm. However, the theoretical prediction appears to not agree very well with film thicknesses under roughly 30 nm in this case.

The focus of this thesis will be on squalane, since it is a known reference fluid in EHL. There have been several EHL studies conducted using squalane. One notable study was conducted by Liu et al. in 2006. Liu et al. referenced the experimental squalane data from Hartl et al to use as validation for their newly developed numerical EHL simulation. Since their code was based on using continuum theory, the results for the central and minimum film thicknesses deviate from the experimental data as shown on the right of Figure 1.6. The deviation from the experiment occurs at roughly 15 nm for the central film thickness and 30 nm for the minimum film thickness. The data presented on the right of Figure 1.6 by Liu et al. will be analyzed and reproduced in this work.

1.3 Viscosity

Viscosity is a property used to characterize the "fluidity" of a fluid. In a lubricated system, the viscosity describes the relationship between the shear stress and rate of shear strain which has the following form,

$$\eta = -\frac{\tau}{\dot{\gamma}} \tag{1.2}$$

where τ is the shear stress, and $\dot{\gamma}$ is the velocity gradient which also known as the shear rate.

The viscosity equation can be used under standard temperature and pressure conditions of fluids where the shearing stress is linearly related to the rate of shearing strain. However, since lubricated mechanical systems are more complicated and do

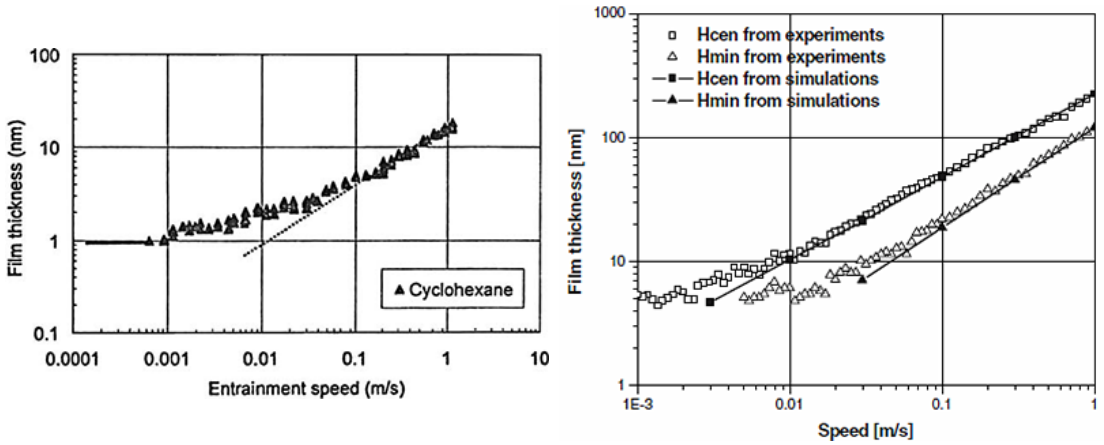

Figure 1.6: Examples (left [18] and right [23]) of experiments where the measured film thicknesses deviated from the classical theory. The symbols in the left plot represent film thicknesses of cyclohexane obtained from experiments and the dotted line represents the Hamrock–Dowson theory [18]. The open symbols in the right plot indicate film thicknesses of squalane obtained from experiments by Hartl et al. [20] and the solid symbols represent the film thicknesses obtained from numerical simulations with the lines used as a guide [23].

not operate under standard conditions, we need to understand what factors affect viscosity [24].

1.3.1 Parameters that Affect Viscosity

Viscosity is an important lubricant property. As previously shown in Figure 1.2, viscosity is a factor that contributes to friction. As the Stribeck curve from Figure 1.2 initially shifts from the boundary lubrication regime to the hydrodynamic regime with increasing viscosity, friction decreases, yielding better lubricant performance. However, the Stribeck curve also shows that friction increases with viscosity in the EHL and hydrodynamic lubrication regimes, indicating that high viscosity leads to high friction which creates power loss. It is important to understand the viscous characteristics of a lubricant in order to accurately predict the behavior of a lubricated system [1]. There are a few factors that are well–known to effect the viscosity of a lubricant: shear rate, pressure, and temperature.

1.3.2 Temperature

The viscosity of a lubricant can be greatly affected by the mechanical system's operating temperature. As a lubricant's temperature increases, their lubricant's viscosity decreases [6]. An operating temperature increase of 25°C can decrease the

lubricant viscosity by 80%. While there are a number of equations that characterize viscosity at varying temperatures, a common method is by referring to the viscosity-temperature charts for various lubricant oils developed by the American Society for Testing Materials (ASTM). The ATSM chart was developed based on Walther's viscosity-temperature equation,

$$(\nu + a) = bd^{1/T^c} \tag{1.3}$$

where a, b, c, d are constants, ν is the kinematic viscosity, and T is the absolute temperature. Under normal operating conditions, the ATSM chart works well for calculating viscosity with varying temperature [1] However, since our EHL simulations assume a constant temperature, a temperature dependent viscosity model is not applicable in our work.

1.3.3 Pressure

Pressure is another factor that can affect lubricant viscosity. Viscosity is sensitive to extreme pressure changes. For some lubricants, the rapid rate of pressure increase can increase the viscosity to where the lubricant behaves like a solid instead of a liquid. While many equations have been developed to describe the relationship between viscosity and pressure, one well-known formula is the Barus equation,

$$\eta_p = \eta_0 \exp^{\alpha p} \tag{1.4}$$

where p is the given pressure, η_p is the viscosity at pressure p, η_0 is the viscosity at atmospheric pressure, and α is the pressure–viscosity coefficient. α is a function of the lubricant's molecular features, such as structure, interlocking, packing, and rigidity. However, the pressure–viscosity coefficient is only applicable to low shear rates [1]. Even though the Barus equation is widely used to calculate viscosity at a given pressure, it is only accurate for moderate pressures [4]. Examples of models that have been shown to give better predictions of viscosity are the Roelands viscosity model [25] and the Yasutomi free-volume viscosity model [23].

1.3.4 Shear Rate

Fluids are typically characterized as Newtonian or non-Newtonian. If a fluid's viscosity is independent of shear rate, then it is designated as Newtonian. Lubricants are typically non-Newtonian, where they may appear Newtonian with constant viscosity at low shear rates, but increase or decrease in viscosity at moderate to high shear rates [4].

Mechanical engineering applications tend to operate at moderate to high shear rates, meaning shear rates larger than 10^6 1/s. The non-Newtonian behavior is a function of the molecular structural complexity of the lubricant. For dilatant

fluids, otherwise known as shear thickening fluids, viscosity increases with increasing shear rate. Pseudoplastic behavior, also known as a shear thinning, can also occur where the fluid's viscosity decreases with increasing shear rate [1].

1.3.4.1 Shear Thickening

Shear thickening, also known as dilatancy, is defined as the increase in viscosity with increasing shear rate. While shear thickening has been observed in some polymeric and detergent fluids, it is prevalent in suspensions. In fact, under the right circumstances, all suspensions containing solid particles exhibit shear thickening behavior. In dilatant fluids, the viscosity is constant at low shear rates. However, at some critical shear rate, the viscosity decreases in a power–law type behavior briefly before increasing, exhibiting shear thickening [26]. The viscosity increase is believed to be caused by the rearranging of the suspended solid particles, resulting in an increased resistance to fluid flow [1]. Shear thickening is counter–productive for industrial applications because the high solid particle content can lead to problems such as component wear and failure. Therefore, shear thickening fluids are generally avoided as lubricants for machine components. Hence, shear thinning lubricants are widely used instead [26].

1.3.4.2 Shear Thinning

Some lubricants are made up of polymer fluids, where the molecular structure is characterized by long chains of molecules with repeating units and branches. Under fast shear rates, polymer fluids align and form layers that reduce the viscosity, a process known as shear thinning. Shear thinning is a characteristic is common to multigrade oils since they are often formulated with polymer fluids [1].

There are a number of equations that are used to characterize the viscosity with respect to shear rate. One equation that is widely used is the Carreau model,

$$\frac{\eta - \eta_\infty}{\eta_0 - \eta_\infty} = \left[1 + (\lambda\dot{\gamma})^2\right]^{(n-1)/2} \tag{1.5}$$

where η is the viscosity, η_∞ is the viscosity at infinite shear rate, η_0 is the viscosity at zero shear rate (ambient viscosity), λ is the relaxation time, $\dot{\gamma}$ is the shear rate, and n is the power-law behavior constant [27]. n describes how quickly a given fluid shear thins. λ is also known as the characteristic time constant which indicates the inverse of the shear rate at which the viscosity of the given fluid begins to experience shear thinning [28]. The viscosity η_∞ at infinite shear rate is zero because, when a given fluid undergoes shear thinning, the viscosity decreases until it reaches zero. Therefore, the Carreau model can be further simplified to the following form [28]:

$$\eta\left(\dot{\gamma}\right) = \frac{\eta_0}{\left[1 + (\lambda\dot{\gamma})^2\right]^n} \tag{1.6}$$

1.4 Film Thickness Effects on Viscosity

While it is well–known that shear rate, pressure, and temperature affect viscosity, a new factor has been found to affect viscosity as well: film thickness. Both experimental and simulation studies have been conducted to observe how film thickness affects viscosity. In the following sections, we will review the experimental and simulation work that has been done, but there is no universal equation that can characterize the film thickness dependence on viscosity.

1.4.1 Experimental Work

One of the earliest experiments was performed in 1990 by Gee et al. Gee et al. conducted experiments with a branched chain, 2-methyloctadecane using a macroscopic droplet of fluid with a lateral sliding mechanism. They observed that, as film thickness decreases from 2.2 nm to 1.0 nm, friction increases exponentially [29]. The friction in this case is not due to surface-surface contact, but rather due to an increase in the viscous friction that is inherent within lubricants. Since friction is the result of the combination of viscosity, velocity and pressure, this experiment marked one of the earliest investigations to see how film thickness affects a given parameter.

In 1991, Hu et al. conducted experiments using a mixture of dodecane and a silicone oil, octamethylcyclotetrasiloxane (OMCTS), confined between circular parallel plates of single crystals of muscovite mica. They found that the effective viscosity increases as film thickness decreases from 3.5 nm to 2.5 nm [30]. In 2003, Zhu and Granick conducted similar experiments also using OMCTS confined between muscovite mica. They ran surface force experiments and measured viscosities of OMCTS slowly compressed from 14 nm to 1 nm. Zhu and Granick observed a peak viscosity at 3 nm that was 3 orders of magnitude higher than the bulk viscosity [31].

In 1997, Luengo et al. conducted experimental work with polybutadiene using a sphere on a flat surface of a modified surface forces apparatus [32]. Their experiments were conducted on film thicknesses decreasing from 180 nm to 12 nm. They found that if the polybutadiene film thickness was greater than roughly 40 nm, the viscosity behaved linearly. However, if the viscosity was less than 40 nm, the viscosity began to behave nonlinearly as film thickness decreased, which is shown in Figure 1.7.

In 2002, Yamada compiled a number of sources reporting shear rate, viscosity, and film thickness data of simple liquids and polymer melts. He proposed a viscosity map that described the effect of confinement on the relationship between shear rate and viscosity of a fluid film as seen in Figure 1.8 [33]. Yamada proposed that a confined fluid has a larger ambient viscosity than a bulk fluid. In addition, Yamada also proposed that a confined fluid begins shear thinning at a lower shear rate than its bulk counterpart [33].

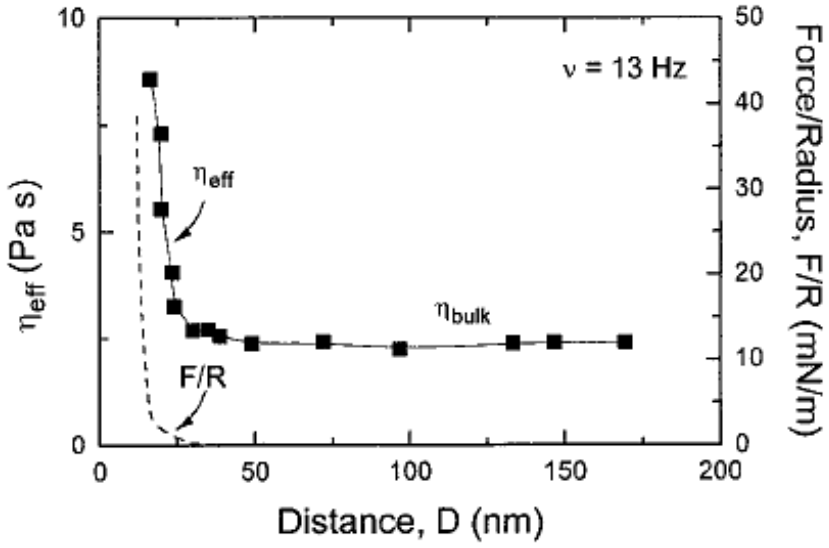

Figure 1.7: A modified surface forces apparatus was used to measure viscosity of polybutadiene. Viscosity increases with decreasing film thickness for very thin films of polybutadiene [32].

1.4.2 Simulation Work

As molecular dynamics (MD) simulations became more developed, simulations were used to analyze systems on the molecular level. Lubricants were one of the topics modeled in simulations to observe changes in viscosity under various conditions. In 1994, Padilla and Toxvaerd used MD simulations to report viscosity results for three confined fluid alkanes: n-butane, n-decane, and 5-butyl-nonane. In their simulations, the fluid alkanes were confined in a slit pore with a film thickness of 2.45 nm. Even though previous experimental work indicated a viscosity increase for confined fluids with respect to the bulk, Padilla and Toxvaerd were unable to reproduce that viscosity increase. Their reported viscosity values were within the uncertainty of the bulk fluid viscosity for all three alkanes [34].

In 1996, Manias et al. used non–equilibrium molecular dynamics (NEMD) simulations to observe the shear rate and viscosity relationships of three different kinds of fluids, linear hexamers, branched hexamers, and star heptamers. They compared film thicknesses of 6σ and 7σ, where σ indicates that the film thickness is in reduced Lennard-Jones units since the simulations were run using a Lennard-Jones potential. Manias et al. found that the simulations with the smaller film thickness produced a shear rate vs. viscosity curve that was much higher than the shear rate vs. viscosity curve for all three bulk fluid films [35]. Similarly, in

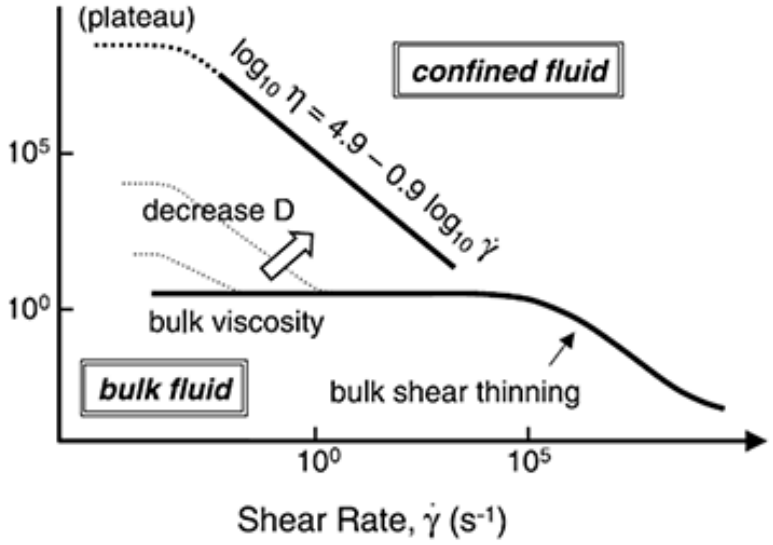

Figure 1.8: Summary of the behavior of the viscosity of a bulk and confined fluid from Reference [33].

1997, Stevens et al. also reported viscosities at three different film thicknesses that were considerably larger than their bulk counterparts. They ran NEMD simulations of hexadecane to obtain the viscosities for the bulk fluid, and used the constraint dynamics of the RATTLE algorithm and rRESPA multi-timestep dynamics for the confined fluid [36].

In 2003, Cui et al. used MD simulations of n-dodecane to observe how the viscosity changes in a bulk n-dodecane fluid versus a confined n-dodecane fluid. They found that viscosity dramatically increases with decreasing shear rate in the confined fluid, whereas the viscosity of the bulk fluid remained true to the expected constant Newtonian viscosity with decreasing shear rate [37]. In 2005, Leng and Cummings also observed a dramatic increase in viscosity from MD simulations of water molecules confined between two mica walls. They ran simulations for film thicknesses of 0.92 nm, 1.65 nm, and 2.44 nm. While the shear rate vs. viscosity curves for the latter two film thicknesses were close to the bulk viscosity values, the smallest film thickness yielded dramatically larger viscosity values on any given shear rate. Compared to the bulk value, the smallest film thickness resulted in viscosities between roughly 1 cP to 40 cP higher than the simulation and experimental bulk viscosity [38]. Wang and Zhao also reported a dramatic increase in viscosity in 2011. They conducted MD simulations of confined n-hexadecane between two solid gold

walls for film thicknesses ranging from 0.6 nm to 105 nm. They found that the viscosity began to increase exponentially for film thicknesses under 15 nm [39].

In addition to exponential increase in viscosity at decreasing film thicknesses, a few articles report oscillation in viscosity as well. In 1998, Jabbarzadeh et al. observed that there was slight oscillatory behavior in their NEMD simulations of confined n-hexadecane. They ran simulations at the same shear rate of $10^{11.5}$ s^{-1} at film thicknesses between 0.975 nm to 19.492 nm [40]. In 2006, Martini et al. found that viscosity oscillates with film thickness for very thin films in their all atom model of decane confined between two solid walls. They found that the viscosity oscillated for film thicknesses between 0.5 and 3.0 nm where the simulations' wall speeds were at 1, 5, and 10 m/s [41]. Similarly, in 2008, Eslami et al. observed oscillation in viscosities from liquid Argon confined between two solid Argon surfaces at film thicknesses between 0.5 nm and 2.75 nm [42]. Later on, in 2010, Eslami and Müller-Plathe observed a slight viscosity oscillation for confined polyamide-6,6 oligomers between two graphene monolayers with film thicknesses from 0.5 nm to 5.0 nm [43].

1.4.3 Summary of Observations

The trends observed from analyzing the relationship between viscosity and film thickness in both experimental and simulation work are that the shear thinning transition shifts to lower shear rates for thinner films, the viscosity increases exponentially as film thickness decreases, and oscillation of viscosity has been observed with decreasing film thickness. Previously, we discussed EHL studies that found a deviation of film thickness obtained from classical methods and experimental measurements. The film thicknesses obtained from experiments were shown to be larger at slower speeds than what was predicted using classical theory. As shown in Figure 1.6, for cyclohexane and squalane, the film thickness begins to deviate from the classical theory at roughly 20 nm, where the theoretical film thickness is lower than the experimental film thickness. Similarly, Figure 1.7 showed an increase in viscosity at roughly 40 nm. This indicates that the length scale at which viscosity increases with film thickness is comparable to that where the deviation from EHL predictions is observed. Therefore, accounting for a film thickness dependence of viscosity may lead to thicker film thickness predictions.

1.5 Research Objective & Thesis Outline

Friction is a prevalent issue in many applications. Previously, it was found that systems with thick lubricant films contribute to viscous frictional loss. As technology for components advances towards higher energy efficiency and less energy loss from viscous friction, lubrication films are decreasing towards thicknesses on the nanometer scale. As mentioned previously, various literature references have shown that there is a dependence of viscosity on film thickness. Therefore, it is important to

understand how film thickness affects viscosity and how that affects EHL predictions of lubricant interfaces.

Our research aims to characterize the relationship between film thickness and viscosity for a given lubricant using molecular dynamics simulations so that we may observe how the lubricant behaves in thin films. In addition, our research also aims to implement a film thickness dependent viscosity model into a traditional EHL numerical simulation so that our EHL simulation can make predictions that are closer to what is expected from experiments.

This first chapter of this thesis provides the motivation and background on this work. Chapter two will describe the simulations that were designed and used for this study. Chapter three will present the results of our research. Finally, Chapter four will be the last chapter, summarizing the conclusions from our study and ideas for future work.

Chapter 2

METHODS

2.1 Molecular Dynamics

The method of molecular dynamics (MD) simulations has been widely used in modeling and calculating the physics of fluids for decades [44]. In MD, Newton's equations of motion are solved for each molecule, conserving momentum and energy [45]. For systems where atoms or molecules are represented by spherical particles, MD calculates the potential energy by using the Lennard-Jones potential [44]. The Lennard-Jones potential is given by the following equation,

$$\Phi(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{2.1}$$

where ϵ and σ are constants, and r is the distance that separates a pair of atoms [46].

However, there are a variety of other, more complex, potentials that MD uses. For atomistic molecular fluid systems, we will refer to potentials as force fields. Force fields contain information on harmonic bond stretching, angle bending, Coulomb terms, Lennard–Jones terms, and the torsional energetics that are calculated by the Fourier series [47]. Our focus will be on fluid systems, where the Optimized Potentials for Liquid Simulations (OPLS) force field is often used [48]. OPLS contains all of the parameters mentioned formerly, which is characterized by the following equations for obtaining the energies associated with the bond stretching, angle bending, and torsional energy,

$$\begin{aligned} E_{bond} &= \sum_{bonds} K_l(l - l_{eq})^2 \\ E_{angle} &= \sum_{angles} K_\theta(\theta - \theta_{eq})^2 \\ E_{torsion} &= \sum_{i} \frac{V_1^i}{2}[1 + \cos(\phi_i)] + \frac{V_2^i}{2}[1 - \cos(2\phi_i)] + \frac{V_3^i}{2}[1 + \cos(3\phi_i)] \end{aligned} \tag{2.2}$$

where K is a harmonic spring force, l is the bond length, the subscript eq represents equilibrium values, θ is the bond angle, ϕ_i represents the i^{th} dihedral angle, and V_j^i is the j^{th} coefficient in the Fourier series [49].

2.1.1 Simulation Platform

The platform we use to run MD simulations is called the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS). LAMMPS is an open-source code that models a system of particles in solid, liquid or gaseous states. It can model a wide range of systems by integrating Newton's equations of motion, interaction forces, and boundary conditions to simulate how a system will behave realistically under the given conditions. The system types that LAMMPS supports are atomic, polymeric, biological, metallic, granular and coarse-grained. Written in C++ and designed for parallel computers, it can model systems from a few particles to billions of particles [46].

2.1.2 Molecular Construction Platform

LAMMPS is a simulation platform but not a molecular construction platform, meaning that it cannot define the coordinates of atoms and bonds needed to create molecules, even though it can construct simple Lennard–Jones systems and crystalline structures. For constructing molecules, there are many different platforms. The platform that we use is called Moltemplate [50]. Moltemplate is an open–source general text-based atomistic molecule builder for LAMMPS that can be run on any operating system that has a command–line interpreter such as the Bourne shell and Python 2.7 or higher. Moltemplate supports all of the force fields used in LAMMPS and gives the freedom of user customization of force field parameters. Moltemplate can be used to design molecular models by creating individual all atom subunits of a given molecule [50]. For example, for a hydrocarbon molecule consisting of hydrogen and carbon atoms, a user can specify all of the possible subunits or components of the molecule, CH, CH_2, and CH_3. Then, to create the molecule, the user would simply specify how the components are arranged in the moltemplate file.

2.1.3 Simulations of Confined Fluids

In our work, there are two types of models, the simple model of a canonical fluid used as a starting point, and the molecular model of a real lubricant fluid, squalane. Both models will be discussed in detail in the following sections.

2.1.3.1 Simple Model

Spherical particles interacting with Lennard–Jones potentials have been extensively used to model fluids, alloys and glasses over the past two decades [51]. The Lennard–Jones potential treats fluid molecules as spherical particles. The Lennard–Jones potential serves as a reliable simulation starting point before embarking on more complicated systems [51]. We begin our investigation by simulating a simple confined Lennard–Jones fluid with varying film thicknesses, as shown in Figure 2.1. Refer to Appendix A.1 for an example of the LAMMPS input file used.

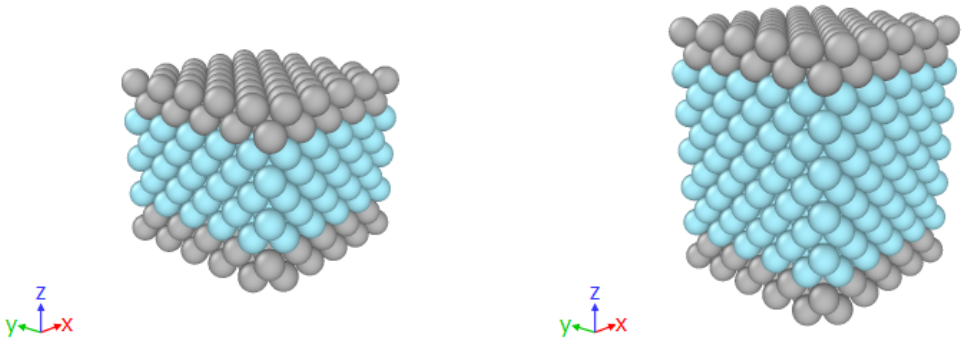

Figure 2.1: Model of Lennard–Jones fluid at film thicknesses of $h' = 5$ (Left) and $h' = 9$ (Right) generated by a free visualization software called OVITO [53]. The walls are represented by gray spheres and the fluid is represented by blue spheres.

In the simulation, the system of Lennard–Jones particles are created and defined as either wall or fluid particle groups. The reduced density of the fluid, ρ', is 0.849. In the x and y directions, the boundaries are periodic. In the z–direction where the film thickness is defined, the boundary condition is non–periodic in order to obtain a finite value for the film thickness. Three of the same system configurations are run with three different seed numbers. Upon simulation initialization, the specified seed number generates random velocities that are applied to the fluid particles until the simulations are equilibrated [52]. The fluid group is equilibrated using NVT integration, where the number of atoms (N), volume (V) and temperature (T) are all kept constant using the Berendsen thermostat where temperature is 0.7 in reduced units [46]. After equilibration, the fluid is then subjected to a shear by imposing a velocity in the x–direction on the upper wall. The shear stress and velocity profile data are recorded after the equilibration step. All values for the simple model simulation are in reduced Lennard–Jones units, denoted with a prime symbol ($'$).

2.1.3.2 Molecular Model

The molecule that we selected to model is a branched C_{30} hydrocarbon, 2,6,10,15,19,23-hexamethyltetracosane also known as squalane, as shown in Figure 2.2. Squalane is a known reference fluid in EHL and confined molecular dynamics simulations [21]. Not only has there been a plethora of work done on squalane as a lubricant, but squalane is a fluid for which EHL film thickness calculations can be very accurate [5], especially since it is considered as a Newtonian fluid under moderate contact [21]. We constructed the squalane molecule and simulation setup using Moltemplate. In order to create the squalane molecule shown in figure 2.2,

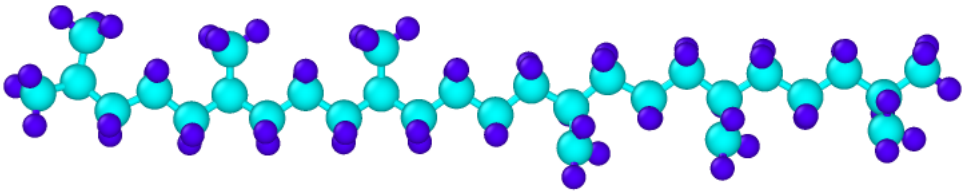

Figure 2.2: The squalane molecule illustrated using OVITO [53]. The carbon atoms are represented in blue and the hydrogen atoms are represented in purple.

we first define each type of carbon present in the molecule. The molecule is a fully atomistic model with each atom modeled explicitly.

A squalane molecule consists of the following types of components, a carbon with one hydrogen, a carbon with two hydrogens, and a carbon with three hydrogens, which we will refer to as CH, CH_2, and CH_3 respectively. To begin constructing the squalane molecule, the coordinates and bonds of each carbon and hydrogen atom are defined. The codes for defining CH_2 and CH_3 already exist upon installation of Moltemplate. Refer to Appendix B.1-B.3 for the codes used to define CH, CH_2, and CH_3. Once we define the code for CH, we place all of the hydrocarbon subunits together spaced 1.25 Å apart and defined the bonds to match the structure in Figure 2.2. Refer to Appendix B.4 for the Moltemplate code used to create the squalane molecule.

Next, we modified the pre-existing OPLS force field file, "oplsaa.lt" to include the charge, bond, angle and dihedral information for a molecule consisting of CH, CH_2 and CH_3. Then, we identified all of the possible combinations of identification numbers of CH, CH_2, and CH_3 that would make up bonds, angles and dihedrals in a squalane molecule. We organized them by placing their identification numbers into pairs, triples and quadruples for the each possible combination of bonds, angles and dihedrals, respectively. For example, one angle is created by the left-most three hydrocarbons in Figure 2.2: CH_3–CH–CH_3, which corresponds to the identification number triple, 80–82–80 (See line 62 in Appendix B.5). Then, we added lines containing the pairs, triples and quadruples of identification numbers to the force field file (see Appendix B.5). Once the molecule and force field file compiled successfully, we defined a system of squalane molecules that Moltemplate outputs into separate files containing the molecule data, charges, settings, and initialization settings as shown in Appendices B.6-B.8.

These files are then read into LAMMPS where the simulation box can be adjusted and walls are added. The system is first minimized, relaxed at high temperature of 1000 K and equilibrated to 298.15 K. Force is added to the wall to compress the fluid until the film thickness stabilizes after 2 ns and a restart file is outputted; see Appendix A.2. Then, a separate LAMMPS input file reads in the

restart file, equilibrates the fluid for 0.5 ns and shears the fluid for 10 ns. The process is shown in Figure 2.3 and the code is shown in Appendix A.3. Simulations consisting of different film thicknesses were constructed by altering the number of molecules generated in Moltemplate. Then, each film thickness is subjected to a number of different wall velocities, producing data that is then fitted to the Carreau viscosity model described in Equation 1.6.

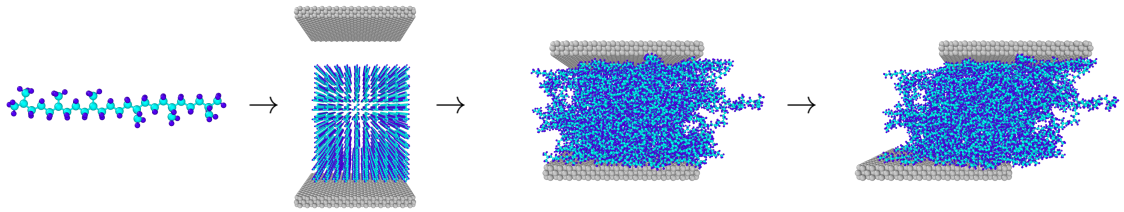

Figure 2.3: The process for creating a confined squalane simulation is shown. The carbon atoms are represented in blue and the hydrogen atoms are represented in purple. The first step is to create the squalane molecule in Moltemplate. Next, the simulation is set up in LAMMPS. Then, system is equilibrated and compressed. Lastly, the system is sheared by applied velocities to the walls. All images were generated using OVITO [53].

2.2 Numerical EHL Simulation

To computationally solve for the film thickness under EHL conditions, an algorithm developed by Hu and Zhu is used as shown in Figure 2.4. We will be simulating an EHL ball–on–disc experiment using a steel ball on glass disc, which was previously shown in Figure 1.5. Circular contact between the ball and disc will be assumed in our simulation.

2.2.1 Input Parameters

The input parameters consist of the physical, material and lubricant properties of the system. The physical properties are the physical variables pertaining to our system, such as the load, temperature and speeds of both surfaces. The material properties are properties pertaining to the materials used, such as the ball radius, and the Young's moduli and Poisson ratios of the surface and the ball. The lubricant properties are properties inherent to the lubricant, such as the ambient viscosity and the pressure–viscosity coefficient.

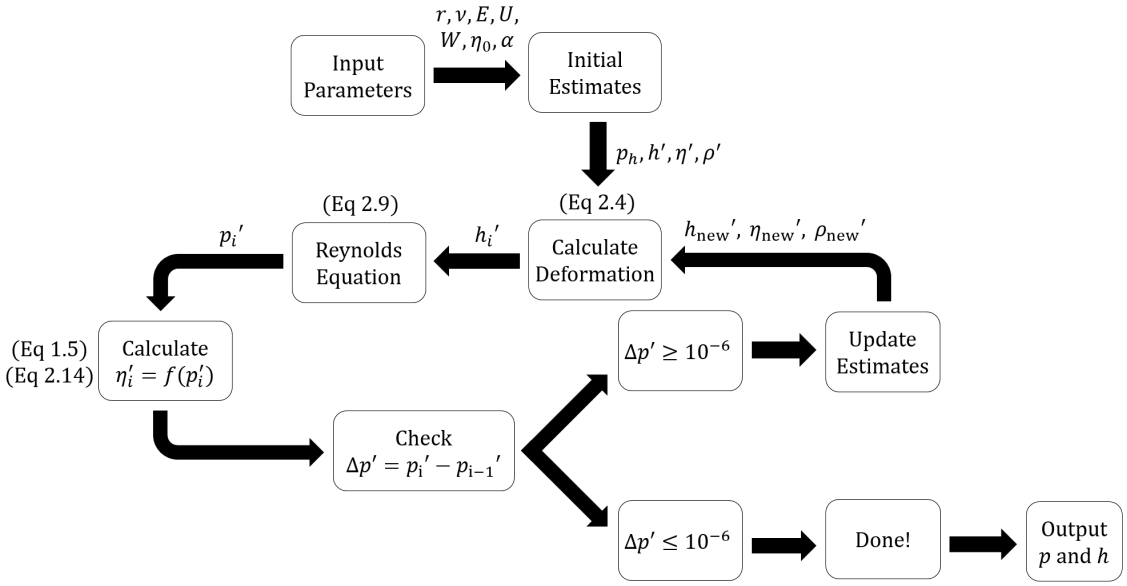

Figure 2.4: Diagram of algorithm for classical EHL simulation solver developed by Hu and Zhu in 2000 where the prime symbols ($\prime$) indicate normalized values [54].

2.2.2 Initial Estimates

The first step is to create initial estimates of the pressure, film thickness, viscosity, and density. One of the first estimates that needs to be made is the pressure. The pressure is calculated from the Hertzian contact theory of elliptical contact between two elastic bodies:

$$p_h = \frac{3W}{2\pi a^2} \tag{2.3}$$

where p_h is the Hertzian contact pressure, W is the load and a is the Hertzian contact radius [1].

Under EHL conditions, the elastic deformation characteristic of the system must be taken into account. The elastic deformation is determined by the discrete convolution and fast Fourier transform method that calculates the dimensionless elastic deformation, V, using Boussinesq integration:

$$V(x,y) = \frac{2p_h}{\pi E'} \iint_{\Omega} \frac{p(\overline{\xi},\overline{\zeta})}{\sqrt{(x-\overline{\xi})^2 + (y-\overline{\zeta})^2}} d\overline{\xi} d\overline{\zeta} \tag{2.4}$$

where E' is the reduced Young's modulus, Ω is the computational domain, $\overline{\xi}$ and $\overline{\zeta}$ are the corresponding x and y values in the computational domain [54].

The initial film thickness is determined by using the following film geometry equation that relies on elastic deformation V:

$$h = h_0 + \frac{a}{2r}(x^2 + y^2) + V(x, y) \tag{2.5}$$

where r is the ball radius and h_0 is the normal approach of two surfaces [23]. h_0 is also known as the compliance at the center of EHL contact [55].

The initial viscosity calculated from the Hertzian pressure using a pressure–viscosity model such as the Barus viscosity model, Equation 1.4, which was previously defined to be the following,

$$\eta_p = \eta_0 \exp^{\alpha p}$$

where p is the given pressure, η_p is the viscosity at pressure p, η_0 is the viscosity at atmospheric pressure, and α is the pressure viscosity coefficient [1]. Equation 1.4 is a general pressure-viscosity equation, but to obtain the initial viscosity, the Hertzian pressure would be used as the given pressure. The initial density is calculated from the Dowson–Higginson equation of state:

$$\rho = 1 + \frac{0.6 \times 10^{-9} p_h p}{1 + 1.7 \times 10^{-9} p_h p} \tag{2.6}$$

where ρ, p_h is the Hertzian pressure, and p is the pressure [56].

2.2.3 Reynolds Equation

Pressure is calculated by using the Navier–Stokes equations, given by:

$$\begin{aligned}
\rho\frac{Du}{Dt} &= \rho X_a - \frac{\partial p}{\partial x} + \eta_0 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + \frac{\eta_0}{3}\frac{\partial \xi_a}{\partial x} \\
\rho\frac{Dv}{Dt} &= \rho Y_a - \frac{\partial p}{\partial y} + \eta_0 \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + \frac{\eta_0}{3}\frac{\partial \xi_a}{\partial y} \\
\rho\frac{Dw}{Dt} &= \rho Z_a - \frac{\partial p}{\partial z} + \eta_0 \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + \frac{\eta_0}{3}\frac{\partial \xi_a}{\partial z}
\end{aligned} \tag{2.7}$$

where ρ is the density, u, v, w are the velocity components, X_a, Y_a, Z_a are the body forces needed to accelerate a fluid element, p is the pressure, η_0 is the ambient viscosity, and ξ_a is dilatation, otherwise known as the measure of fluid expansion [4]. However, the pressure distribution in fluid film lubrication is a simplification of the

Navier-Stokes Equations, called the general Reynolds Equation, given by:

$$\begin{aligned} 0 = &\frac{\partial}{\partial x}\left(-\frac{\rho h^3}{12\eta}\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial y}\left(-\frac{\rho h^3}{12\eta}\frac{\partial p}{\partial y}\right) + \frac{\partial}{\partial x}\left(\frac{\rho h(u_a+u_b)}{2}\right) \\ &+ \frac{\partial}{\partial y}\left(\frac{\rho h(v_a+v_b)}{2}\right) + \rho(w_a - w_b) - \rho u_a \frac{\partial h}{\partial x} - \rho v_a \frac{\partial h}{\partial y} + h\frac{\partial \rho}{\partial t} \end{aligned} \tag{2.8}$$

where h is the film thickness, η is the viscosity, p is the pressure, u_a and u_b are the velocities of the surfaces in the x–direction, v_a and v_b are the velocities of the surfaces in the y–direction, and w_a and w_b are the velocities of the surfaces in the z–direction. The Reynolds Equation can be described as a combination of the net flow rates due to pressure gradients (Poiseuille terms), the net entraining flow rates due to surface velocities (Couette terms), the net flow rates due to a squeezing motion (broken down into the density wedge, stretch and physical wedge terms), and the net flow rate due to local expansion [4].

However, our numerical simulation assumes that there are no velocities in the y and z directions and density is not a function of time. In addition, for ease of computation, our simulation works with normalized variables, which are indicated with the prime ($'$) symbol. Therefore, the Reynolds equation is simplified to the following form that is used in our simulation:

$$\frac{\partial}{\partial x}\left[\left(\frac{ap_h}{12U\eta_0}\right)\left(\frac{\rho' h^3}{\eta'}\right)\left(\frac{\partial p}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left[\left(\frac{ap_h}{12U\eta_0}\right)\left(\frac{\rho' h^3}{\eta'}\right)\left(\frac{\partial p}{\partial y}\right)\right] = \frac{\partial(\rho' h)}{\partial x} \tag{2.9}$$

where a is the semi–major axis of the contact ellipse, U is the entrainment speed, ρ' is the dimensionless density normalized from the density at ambient conditions ρ_0 and η' is the dimensionless viscosity normalized from ambient viscosity η_0 [23].

2.2.4 Pressure–Viscosity Model

There are a large number of empirical pressure–viscosity models, however, in the context of EHL simulations, the Barus and Roelands viscosity models are the most widely used [23] [57]. The Barus viscosity model is given by Equation 1.4, and the Roelands model is given by the following:

$$\eta = \exp\left\{(\ln\eta_0 + 0.67)\left[\left(1 + (5.1\times10^{-9}[\mathrm{GPa}^{-1}])p\right)^{\frac{\alpha}{(5.1\times10^{-9}[\mathrm{GPa}^{-1}])(\ln\eta_0+9.67)}} - 1\right]\right\} \tag{2.10}$$

where 0.67 and 9.67 are unitless and viscosity is a function of pressure with the ambient viscosity and pressure–viscosity coefficient as constants [58]. The issue with the Barus and Roelands models is that both assume linear pressure–viscosity responses of lubricants on the logarithmic scale, which is not true since most lubrications behave non–linearly [59]. In addition, both Barus and Roelands models are known

to underestimate viscosities, therefore leading to underpredicted film thickness values in EHL simulations [23] [60] [61]. Liu et al. found that using the Yasutomi free–volume model in their EHL simulations yielded high film thicknesses that were closer to experimental values [62]. The Yasutomi free–volume model is given by the following:

$$\eta' = \frac{\eta_g}{\eta_0} \exp\left[\frac{-2.3C_1(T-(T_{g_0}+A_1\ln(1+A_2p)))(1-B_1\ln(1+B_2p)}{C_2+(T-(T_{g_0}+A_1\ln(1+A_2p)))(1-B_1\ln(1+B_2p))}\right] \tag{2.11}$$

where η_g is the glass transition viscosity, T is the temperature in $^\circ C$, and A_1, A_2, B_1, B_2, C_1, and C_2 are all Yasutomi constants. Even though the Yasutomi free–volume viscosity model gives a better EHL film thickness predictions, the model is limited in the sense that there is often numerical instability when implementing the model in EHL simulations [23]. Therefore, we will use the Barus viscosity model given that it yields better EHL film thicknesses than the Roelands viscosity model [23].

2.2.5 Pressure–Viscosity Coefficient

Initially, our code used the pressure–viscosity coefficient (α) value determined in the conventional method by extracting the value from viscometer data on squalane, and obtaining α from fitting the data to the Barus equation (Equation 1.4). Liu et al. conducted a study of three pressure–viscosity coefficient definitions to see which yielded a better film thickness prediction. They compared the original value, α, with α^* and α_{film}. α^* is Blok's reciprocal asymptotic isoviscous pressure coefficient:

$$\alpha^* = \frac{1}{\int_0^\infty \frac{\eta_0}{\eta(p)}dp} \tag{2.12}$$

Liu et al. found a new definition denoted by α_{film} based on Blok's α^*:

$$\alpha_{film} = \frac{1-\exp^{-k}}{\int_0^{k/\alpha^*} \frac{\eta_0}{\eta(p)}dp} \tag{2.13}$$

k/α^* is the maximum inlet pressure where the high viscosity caused by the pressure has no contribution to the film thickness formation [23]. k is defined as a constant with the value of 3 and validated by Vergne and Bair in 2014 [63]. Viscosity is a function of pressure, $\eta(p)$, which can be obtained by fitting viscometer data to a pressure–viscosity model. We fitted the viscometer data for squalane to the Barus equation (Equation 1.4) to obtain $\eta(p)$ and the pressure–viscosity coefficient α [23]. Therefore, Equation 2.13 has the following form:

$$\alpha_{film} = \frac{1-\exp^{-3}}{\int_0^{3/\alpha^*} \frac{\eta_0}{\eta_0 \exp^{\alpha p}}dp} \tag{2.14}$$

Liu et. al found that their new definition of the pressure–viscosity coefficient, α_{film}, which combined the traditional α and Blok's α^*, helped yield good agreement between their numerical simulation and film thickness calculated from the Hamrock–Dowson formula. The value of α_{film} was found to be 20.15 GPa^{-1}, which matches the value that Liu et al. obtained [23]. Therefore, we incorporated $\alpha_{film} = 20.15$ GPa^{-1} in our simulation as the pressure–viscosity coefficient for our pressure–viscosity model.

2.2.6 Numerical Procedure

In an EHL simulation, the equations are discretized and the Reynolds equation is solved with an iterative procedure predominantly using the 1st-order backward scheme. The 2nd-order backward scheme is used for solving the wedge term since simulations with denser mesh sizes have been found to converge faster than using the 1st-order backward scheme [23]. In Figure 2.4, the criterion for determining convergence is if $\Delta p' \leq 1.0 \times 10^{-6}$, where $\Delta p' = p'_i - p'_{i-1}$. If the criterion is not satisfied, then the values of film thickness, viscosity and density are updated accordingly with the new pressure and placed back into the Reynolds Equation. Due to the numerical aspect of the simulation, each iteration produces a pressure value that is closer to the final pressure [54].

2.2.7 Equation for Viscosity Dependence on Film Thickness

Viscosity has been shown to increase exponentially with decreasing film thickness, as described in the previous chapter. To determine an initial relationship of viscosity as a function of film thickness, we used the polybutadiene data provided by Luengo et al. in Figure 1.7. We fit an exponential decay function to the data, which had the following form:

$$\eta(h) = A \exp^{-h/t} + \eta_0 \tag{2.15}$$

where A is the amplitude constant and t is the decay constant. The values from our polybutadiene fit were $A = 196.924$ and $t = 4.914$. We used these values as starting points for determining what A and t would give central film thicknesses with the least error for our EHL simulation setup using squalane. To incorporate Equation 2.15 into our algorithm, we replaced the ambient viscosity (η_0) in our Barus equation (Equation 1.4) with Equation 2.15 as shown below

$$\eta(p, h) = (A \exp^{-h/t} + \eta_0) \exp^{\alpha_{film} p} \tag{2.16}$$

The method for determining the values of A and t to fit our squalane simulation will be discussed in further detail in the next section.

2.2.8 Enhanced EHL Simulation for Thin Films

As previously discussed, numerical simulations produce film thickness data that agrees well with classical theory, but deviates at thin film thicknesses. The classical methods of prediction give film thicknesses that are lower than what is expected from experiments. To modify our EHL simulation to better predict film thicknesses at slower speeds, we made an adjustment to our algorithm. As shown in Figure 2.5, we modify our viscosity model to not only depend on pressure, but to also incorporate a film thickness dependence, as given by Equation 2.16.

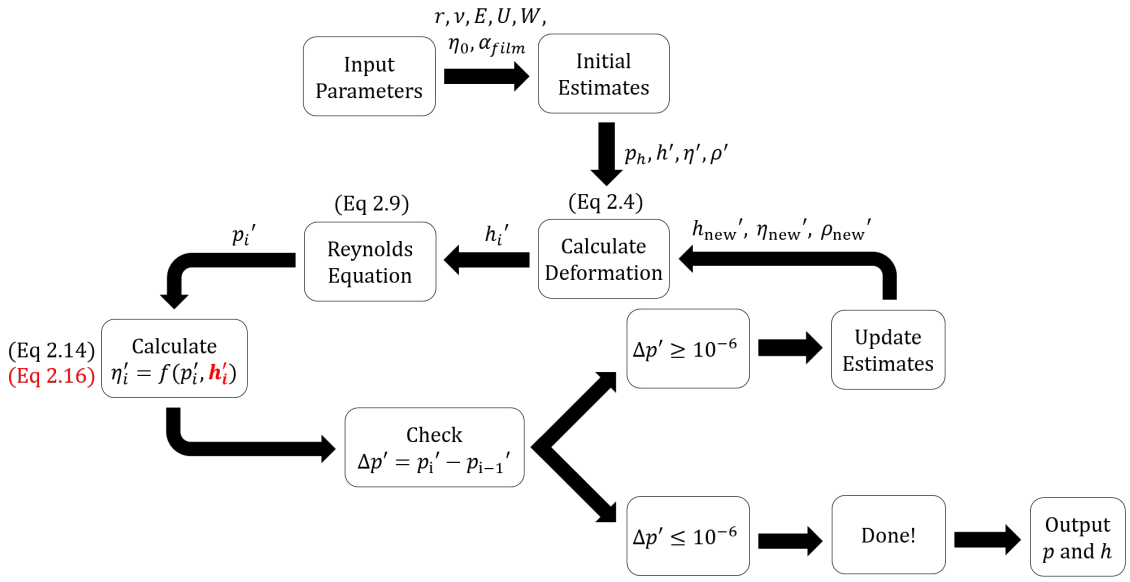

Figure 2.5: Diagram of algorithm for enhanced EHL simulation incorporating film thickness dependence on viscosity (Equation 2.16).

EHL simulations incorporating Equation 2.16 were run with different values of A and t at a speed of 0.003 m/s to determine which values gave film thickness results with the least error. This speed was selected because it had the largest deviation from experimental data [23]. The results will be shown and discussed in the next chapter. Once we determined values for A and t for Equation 2.16, we ran simulations with different speeds. Since we are using the data reported by Liu et al. for comparison, our simulations using our enhanced EHL algorithm were run at speeds of 0.003, 0.01, 0.03, 0.1, 0.3 and 1.0 m/s. We used 256x256 and 512x512 meshes for comparison [23]. The results will be discussed in the next chapter.

Chapter 3

RESULTS

3.1 MD Predictions of Viscosity vs. Film Thickness

In the following sections, we will discuss the results obtained from our molecular dynamics simulations of a simple model of a Lennard–Jones fluid and an atomistic model of squalane. Our goal with our simple and molecular models is to replicate the trends proposed by Yamada in Figure 1.8. Two distinct trends that are observed in this figure are the confined fluid has a larger ambient viscosity than a bulk fluid and a confined fluid begins shear thinning at a lower shear rate than its bulk counterpart.

3.1.1 Simple Model Viscosity vs. Film Thickness Predictions

The results for our initial configuration of a simple confined Lennard–Jones fluid using molecular dynamics simulations are shown in Figure 3.1. We obtained viscosity and shear rate data for two different film thicknesses, which are in Lennard–Jones reduced units. The error bars in Figure 3.1 reflect the standard deviation of the viscosity calculated from three independent simulations using different seed numbers as mentioned in Chapter 2. Similar to what has been seen in the literature (Figure 1.8), we observed that the simple model with a reduced film thickness of $h' = 5$ showed overall larger viscosity and the fluid begins to shear thin at a slower shear rate than the thicker film, where $h' = 9$.

In order to quantitatively compare our data, we used the Carreau viscosity model, described by Equation 1.6. We fit both sets of viscosity vs. shear rate data for the two different film thicknesses to Equation 1.6, which is represented by the dashed lines in Figure 3.1. The Lennard–Jones reduced fit parameters are shown in Table 3.1. Table 3.1 contains the fitted ambient viscosity values, relaxation (characteristic) time constant, and the power law constant.

Table 3.1 shows that the ambient viscosity for the thicker film is in good agreement with the literature (relative error is 4.6%) which is expected for a bulk Lennard–Jones fluid [28]. This table also shows that the ambient viscosity for the smaller film thickness is higher than the thicker film thickness. In addition, the relaxation time constant for the smaller film is much larger than that of the larger film. This means that the smaller film begins to shear thin sooner than the larger

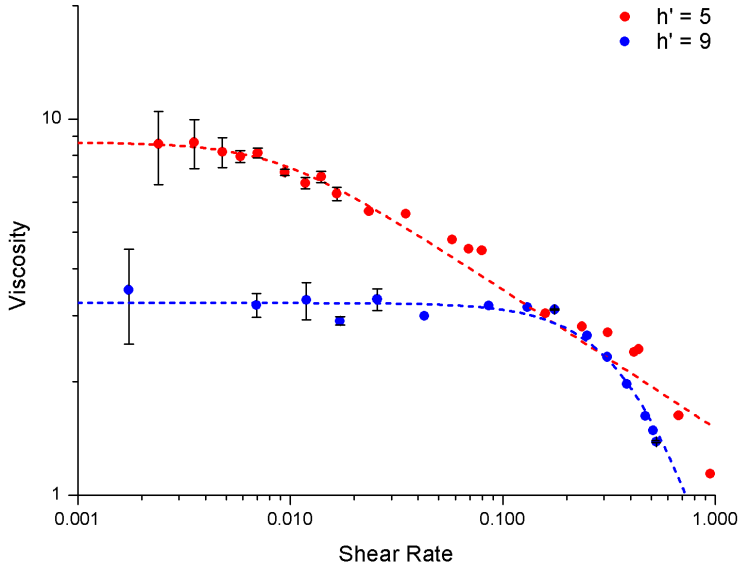

Figure 3.1: Viscosity as a function of shear rate for two film thicknesses. The data points were obtained from MD simulations and the dashed lines indicate fits to the Carreau viscosity model. All values are in reduced Lennard–Jones units. See Appendix C.1 for the Python script used to post–process the data.

film. Lastly, the power law constant for the thicker film is much larger than the smaller film, indicating that the fluid shear thinning decline is much quicker than the thinner film. This is consistent with Reference [33].

3.1.2 Molecular Model Viscosity vs. Film Thickness Predictions

The results for our molecular model of squalane are shown in Figures 3.2 and 3.3. In both figures, the red symbols represent the system with the smallest film thickness of 0.83 nm, the blue represents the system with a film thickness of 1.29 nm, the purple symbols represent the system with a film thickness of 2.25 nm, and the black dashed line represents the ambient bulk viscosity. The error bars are obtained from running each film thickness at each speed three times with different initial velocity seed numbers applied to the fluid molecules. The difference between the two figures is that one shows effective viscosity vs. effective shear rate and the other shows actual viscosity vs. actual shear rate. The effective shear rate is calculated from the ratio of the speed imposed of the walls by the lubricant film thickness, and

Film Thickness	η_0'	λ'	n'
$h' = 5$	8.661	115.447	0.184
$h' = 9$	3.244	2.061	1.0

Table 3.1: Reduced parameters obtained from fitting our simulation data of two different film thicknesses to the Carreau viscosity model.

the effective viscosity is calculated using the effective shear rate. The actual shear rate values are obtained differently from the effective shear rate values. The actual shear rate is obtained by fitting a linear curve to the time averaged velocity profile outputted at the last timestep, and the actual viscosity is calculated using the actual shear rate. Figure 3.2 shows the effective viscosity with respect to effective shear rate and Figure 3.3 shows the actual viscosity with respect to actual average shear rate.

As shown in Figure 3.2 for the effective viscosity vs. effective shear rate, the shear thinning region is captured for all three film thicknesses. Since no transition into the Newtonian viscosity has been observed, we do not have sufficient data to fit the Carreau viscosity model with. The overall viscosity of the thickest film is larger than the other two film thicknesses. However, this does not imply that the ambient viscosity of the thickest film is larger than the others. Some overlap in the viscosity values can be seen between the 2.25 nm and 1.29 nm systems at shear rates around 1×10^{11} 1/s. The smallest film thickness seems to exhibit faster shear thinning behavior than the other two film thicknesses. However, since the Newtonian viscosity regime was not captured by our simulations, it is possible that the behavior of our systems may exhibit different trends at slower shear rates.

As shown in Figure 3.3, the actual viscosity values only correspond to shear rates between 1×10^9 and 1×10^{12} 1/s. Since the actual shear rate values are averaged between three different simulation seed numbers, there are larger variations between the final actual shear rate values. Different actual shear rate values may be obtained depending on the correlation goodness, or R^2 value, of the linear fit. A high R^2 value results in a reliable actual shear rate value, whereas a low R^2 value results in a less reliable actual shear rate value. Figure 3.3 shows larger error bars in the calculated viscosity values than Figure 3.2. Unlike Figure 3.2, shear rates slower than 1×10^9 1/s were not reached in Figure 3.3. However, there are similarities between the two figures: (i) the overall viscosity of the thickest film is larger than the other two film thicknesses, (ii) the thickest film's viscosity appears to slightly overlap at 1×10^{11} 1/s, and (iii) the smallest film thickness exhibits faster shear thinning behavior than the other two film thicknesses. Since the Newtonian region was not reached using our average actual shear rates, we do not have sufficient data to fit the Carreau viscosity model to in order to obtain the ambient viscosities for each film thickness. Our attempt to obtain a viscosity vs. film thickness plot similar to Figure 1.7 was

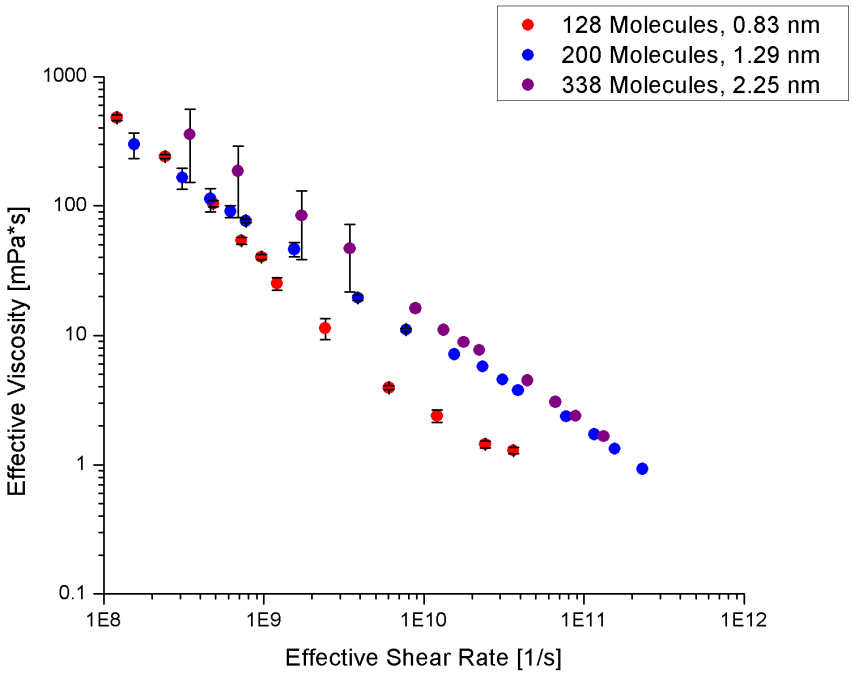

Figure 3.2: Effective viscosity with respect to effective shear rate. See Appendix C.2 for the Python script used to post–process the data.

unsuccessful.

3.2 Numerical EHL Simulation

The results obtained from our EHL simulations are discussed in the following sections. We compared our traditional EHL simulation results to the literature to validate our algorithm [23]. Then, we modified our EHL simulation to incorporate a film thickness dependence on viscosity, and show the improvement in the data compared to experimental values [20].

3.2.1 Validation of the Traditional Model

In order to verify that our numerical EHL simulation is producing reliable data, we compared our results to the data reported by Liu et. al [23]. We validated our simulation by using the parameters specified in the EHL simulation setup that Liu et al. used, which were chosen to be the same as the experiment by Hartl et al. We used the new definition of the pressure–viscosity coefficient, α_{film} by Liu et. al and we also compared a couple of pressure–viscosity models. In addition, we fit our

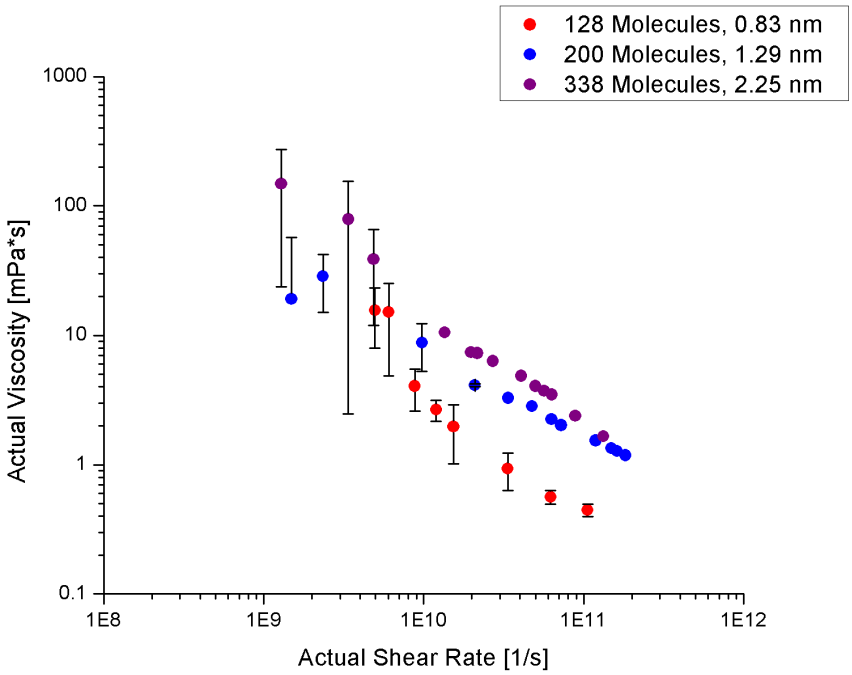

Figure 3.3: Actual viscosity with respect to actual shear rate. See Appendix C.2 for the Python script used to post–process the data.

data and the literature data to the Hamrock Dowson equation (Equation 1.1) and compared the fit parameters. All three validations will be discussed below.

As mentioned in Chapter 2, Liu et al. found a new definition for the pressure–viscosity coefficient α, denoted as α_{film}. They found that α_{film} resulted in film thickness values that were in good agreement with the Hamrock–Dowson predictions [23]. To validate this new pressure–viscosity coefficient, where $\alpha_{film} = 20.15$ GPa^{-1}, we ran three EHL simulations on a 512×512 mesh at speeds of 1 m/s, 0.1 m/s, and 0.01 m/s using the both the Roelands and Barus viscosity model. Table 3.2 shows our central and minimum film thickness data obtained using the Roelands model in our EHL simulations in comparison with the literature data provided by Liu et al. With the relative errors between our simulations and the literature data under 1.2%, this indicates that our EHL simulation algorithm produces reliable results. Table 3.3 shows our central and minimum film thickness data obtained from using the Barus model in our simulations along side the literature data provided by Liu et al. also using the Barus viscosity model. The relative errors between our simulations using the Barus model and the literature data are also under 1.2%. This also shows that our EHL simulation algorithm produces reliable results.

Speed [m/s]	Literature Simulation h_{cen} [nm]	Our Simulation h_{cen} [nm]	Literature Simulation h_{min} [nm]	Our Simulation h_{min} [nm]
1.0	216.8	214.9	114.6	114.1
0.1	46.2	45.7	717.8	17.7
0.01	9.4	9.3	2.6	2.6

Table 3.2: Validation of our EHL simulation to the literature data provided by Liu et al. using the Roelands viscosity model and a pressure–viscosity coefficient $\alpha_{film} = 20.15$ GPa^{-1} [23].

Speed [m/s]	Literature Simulation h_{cen} [nm]	Our Simulation h_{cen} [nm]	Literature Simulation h_{min} [nm]	Our Simulation h_{min} [nm]
1.0	223.1	221.3	116.4	115.9
0.1	47.3	46.7	17.8	17.7
0.01	9.5	9.5	2.6	2.6

Table 3.3: Validation of our EHL simulation to the literature data provided by Liu et al. using the Barus viscosity model and a pressure–viscosity coefficient $\alpha_{film} = 20.15$ GPa^{-1} [23].

To further validate our pressure viscosity model, we ran our EHL simulations at additional speeds, consisting of 1.0 m/s, 0.3 m/s, 0.1 m/s, 0.01 m/s and 0.003 m/s, on a coarser 256×256 mesh using the Barus viscosity model and pressure–viscosity coefficient $\alpha_{film} = 20.15$ GPa^{-1}. Additional speeds were run to directly compare to the results that Liu et al. obtained. We selected a coarser mesh than the one used previously due to the shorter simulation run time that comes with using coarser meshes. We observed that the difference in the central and minimum film thickness values reported by Liu et al. calculated using the 256×256 and 512×512 meshes were between 0.2–1 nm [23]. Our film thickness results are shown in Table 3.4. The relative errors in our central film thickness results and the simulations by Liu et al. ranged from 1.4%–3.3%. The relative errors in our minimum film thickness results and the simulations by Liu et al. ranged from 0%–5.4%. This shows that using the Barus viscosity model with the pressure–viscosity coefficient α_{film} gave reliable results.

As discussed previously in the introduction, film thickness has been characterized by its relationship to entrainment speed by the Hamrock–Dowson theory. Since all of the other variables in the Hamrock–Dowson equations (Equation 1.1) are

Speed [m/s]	Literature Simulation h_{cen} [nm]	Our Simulation h_{cen} [nm]	Literature Simulation h_{min} [nm]	Our Simulation h_{min} [nm]
1.0	225.3	222.1	120.5	116.0
0.3	101.2	99.3	45.8	43.6
0.1	47.6	46.5	18.4	17.4
0.03	20.4	19.8	6.4	6.1
0.01	9.0	8.7	2.1	2.0
0.003	3.1	3.0	0.0	0.0

Table 3.4: Validation of our EHL simulation to the literature data provided by Liu et al. using a 256×256 mesh, the Barus viscosity model and a pressure–viscosity coefficient $\alpha_{film} = 20.15$ GPa^{-1} [23].

	Using the Literature h_{cen} Data	Using our Simulation h_{cen}	Using the Literature h_{mim} Data	Using our Simulation h_{min} Data
a	225.7	222.5	120.7	116.1
b	0.676	0.679	0.815	0.822

Table 3.5: Fitted parameters to the simplified Hamrock–Dowson equation.

constant, the Hamrock–Dowson equations can be simplified to the following form:

$$h = aU^b \tag{3.1}$$

where h is film thickness, a is the amplitude constant, and b is the linearity constant. To further validate our results from Table 3.4, we fit both our results and the literature data to Equation 3.1. The fitted values for constants a and b are shown in Table 3.5. Relative errors were calculated between the constants fitted to our data and the constants fitted to the data obtained by Liu et al. The relative error between the constant a fitted to the central film thickness was 1.4%. The relative error between the constant a fitted to the minimum film thickness data was 3.8%. The constant b had relative errors of 0.5% and 0.9% for the central and minimum film thickness data, respectively. This shows that our algorithm produces data that fits the classical theory well.

3.2.2 Enhanced EHL Simulation for Thin Films

To determine the constants of the film thickness dependent viscosity equation (Equation 2.16), we first ran simulations with varying values of A and t at a speed of 0.003 m/s. We selected 0.003 m/s because it yields the largest central film thickness

deviation from the experimental data from the results obtained by Hartl et al. in the right hand plot of Figure 1.6. To obtain the best fit parameters, we calculated the relative error between the experimental central film thickness value and our simulation predicted central film thickness value. We plotted our results on a contour plot. Figure 3.4 is a contour plot depicting the relative error percentages between our central film thickness values and the experimental film thickness values with respect to varying values of A and t. This figure shows that there are optimal values of A and t that give good central film thickness agreement within 3% of the experimental data where $185 \leq A \leq 210$ and $1.8 \leq t \leq 4.0$.

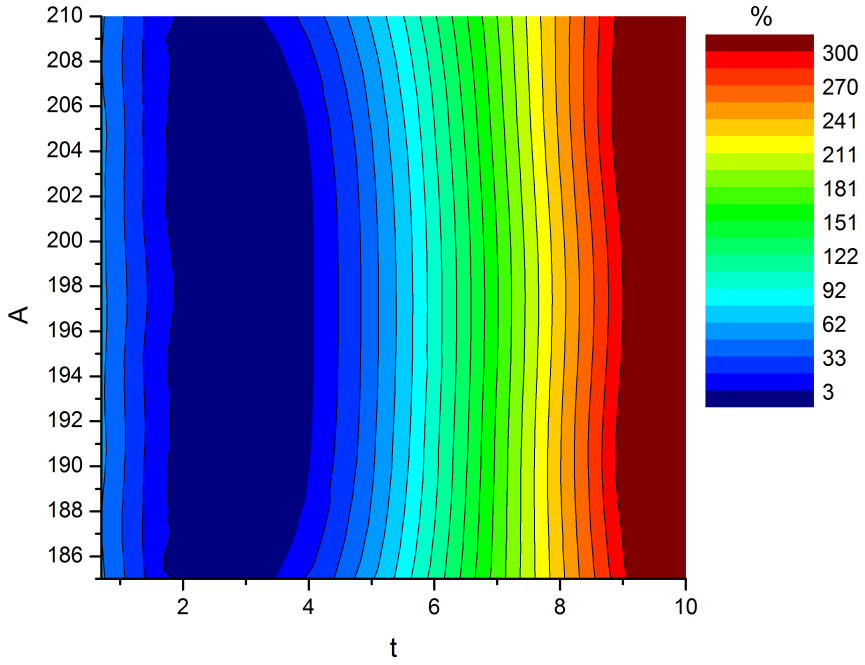

Figure 3.4: Contour plot of the relative errors of the central film thickness obtained from our enhanced EHL simulation compared to the expected experimental film thickness value.

The values we selected for A and t were 205 and 1.6, respectively. To test the accuracy of our enhanced EHL simulation with the film thickness dependent viscosity model, we ran speeds of 0.003, 0.01, 0.03, 0.1, 0.3 and 1 m/s, replicating the speeds we used in Table 3.4. Figure 3.5 is a plot consisting of the experimental film thickness data from Hartl et al., the film thickness data obtained from our original traditional EHL simulation, and the film thickness values obtained from our enhanced EHL simulation. This plot shows that there is improvement in the central

film thickness, indicating that our enhanced simulation predicts film thickness that is closer to experimental measurement than the traditional simulation.

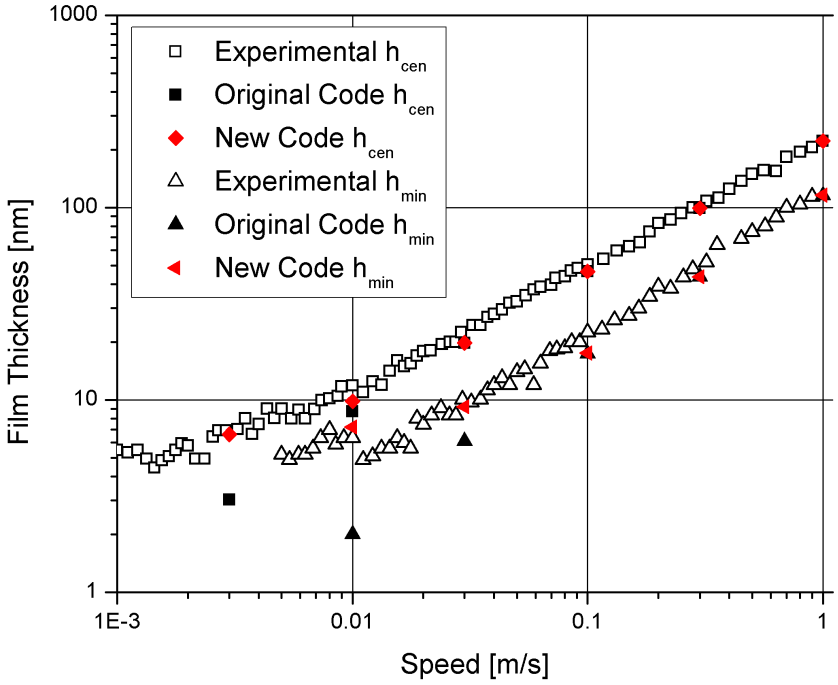

Figure 3.5: Comparison of the film thicknesses produced by our EHL solver, the values obtained by Hartl et al. and the original experimental values [20]. The open symbols represent the experimental data obtained by Liu et al. The black symbols represent the film thickness values obtained from our original traditional EHL simulation. Lastly, the red symbols represent the film thickness values obtained from our enhanced EHL simulation that accounts for film thickness dependence on calculating viscosity [23].

Table 3.6 shows the calculated percent relative errors between the experimental film thicknesses and the film thicknesses obtained from both the traditional EHL simulations and the enhanced EHL simulations. The percent relative errors for the minimum film thicknesses at 0.003 m/s were left blank since there was no experimental data reported for this speed. As shown, the traditional model may be used to predict central film thicknesses created from EHL interfaces when the film thickness is 20 nm or higher. However, for the central film thickness of our system, the error in the traditional model becomes more apparent with film thicknesses that

Speed [m/s]	Traditional Simulation h_{cen} % Error	Enhanced Simulation h_{cen} % Error	Traditional Simulation h_{min} % Error	Enhanced Simulation h_{min} % Error
1.0	1.3	1.3	3.7	3.7
0.3	4.6	4.6	12.7	12.7
0.1	7.9	7.9	22.6	22.1
0.03	12.0	12.0	39.3	8.5
0.01	26.9	17.4	68.5	13.7
0.003	56.5	4.8		

Table 3.6: Calculated percent relative errors between our simulation film thicknesses and the experimental data [20]. h_{cen} refers to the central film thickness and h_{min} refers to the minimum film thickness.

are below 20 nm, which are caused by slower speeds of 0.01 m/s and lower. Using the enhanced EHL simulation for the thinner films showed a 9.5% and 51.7% relative error decrease in the central film thickness for 0.01 m/s and 0.003 m/s, respectively. For these two speeds, the central film thicknesses obtained from the experiments were roughly 12 nm and 7 nm, respectively. The percent relative error in the minimum film thicknesses began decreasing at a film thickness of roughly 17 nm. The enhanced simulation saw a dramatic improvement in the minimum film thickness percent error at speeds 0.03 m/s and 0.01 m/s, where the relative error decrease was 30.8% and 54.8%, respectively. The minimum film thicknesses corresponding to these relative error changes were at experimental values of 10 nm and 6.5 nm, respectively [20]. Our results show that, for faster speeds that result in thicker films, the traditional EHL simulation gives results with relative errors under 10%. However, we also discovered that if the resulting film thicknesses are below 20 nm, the enhanced EHL simulation gives results with lower relative errors than the traditional simulation.

Chapter 4

CONCLUSIONS AND RECOMMENDATIONS

4.1 Research Summary

The goal of this project was divided into two parts, (i) use molecular dynamics (MD) simulations to characterize the relationship between film thickness and viscosity for squalane so that we may observe how squalane behaves in thin films, and (ii) implement a film thickness dependent viscosity model into a traditional EHL numerical simulation so that our EHL simulation can make predictions using squalane that are closer to what has been reported from experiments [20].

Our results from our simple model successfully portrayed the trends seen in Figure 1.8 by Reference [33]. Our thin film simple model had a larger ambient viscosity than the thick film model and began shear thinning at a lower shear rate than the thick film. However, our molecular model simulations gave inconclusive results. Not only did we need a larger number of film thicknesses, we also needed viscosities of each film thickness at the Newtonian region characterized by slower shear rates. Our results for three film thicknesses were only able to capture the non–Newtonian shear thinning region, which did not give sufficient data to fit the Carreau viscosity model to. Hence, we were unable to obtain the ambient viscosities for any of our film thicknesses, and subsequently unable to create a viscosity vs. film thickness plot similar to one shown in Figure 1.7.

While the MD simulation goal of this project was only partially successful, we were able to obtain convincing results from the EHL half of this project. We found that the traditional model gave accurate results for film thicknesses above roughly 20 nm. We modified the traditional numerical EHL simulation algorithm to incorporate film thickness dependence. Our enhanced EHL simulation algorithm gave central and minimum film thickness results that saw much better agreement with the experimental data by Hartl et al. at film thicknesses below 20 nm.

4.2 Areas for Further Study

In this section, we will summarize some of the areas that could be explored in this project going forward. The topics associated with the MD simulations will be discussed as well as the topics associated with the EHL experiments and simulations.

4.2.1 MD Simulation Issues & Areas of Improvement

We encountered quite a few issues with our MD simulations of squalane. The areas that will be discussed here are the ultra–thin films and computational limitations, the signal–to–noise ratio and RNEMD, and alternative lubricant selection.

4.2.1.1 Ultra–Thin Films & Computational Limitations

Our results were only able to capture the shear thinning region of confined squalane. All lubricants experience constant viscosity to shear thinning. Bair et al. summarized the experimental and simulation viscosity and shear rate data for bulk squalane in their work as shown in Figure 4.1. They found that the Newtonian region was obtained predominantly with experiments using rheometer measurements. Even though the MD simulations can give good shear thinning results, they can also capture viscosity values at the transition point and at the Newtonian regime [64].

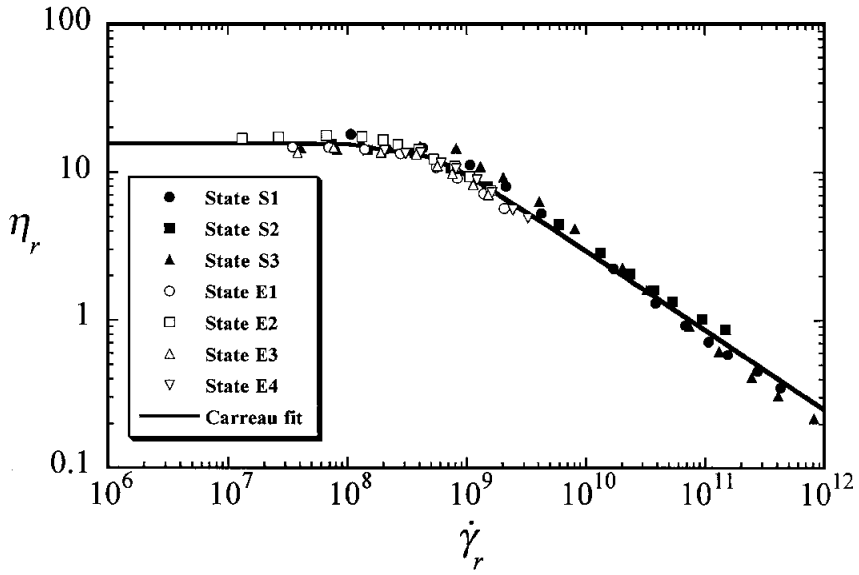

Figure 4.1: Plot of the reduced viscosity versus reduced shear rate for experimental and simulation data of squalane from Reference [64].

The data reported by Bair et al. is for bulk squalane, which we consider as a thick film. The film thicknesses of our MD simulations were 0.83 nm, 1.29 nm and 2.25 nm, which are all extremely thin. In the work by Luengo et al., the smallest film thickness in their viscosity vs. film thickness plot for polybutadiene occurs at roughly 20 nm (Figure 1.7), which is much larger than what we modeled with squalane using molecular dynamics [32]. We were unable to reach the Newtonian regime for all three film thicknesses. Is it likely that the transition point from

constant viscosity to shear thinning viscosity occurs at much slower shear rates. To investigate this issue, the logical step would be to create larger systems with larger film thicknesses.

However, the largest system we were able to create contained 338 molecules (2.25 nm). Any systems that were larger would result simulation failure with error messages stating "domain too large for neighbor bins," indicating that the simulation was too large [52]. We attempted to fix the error by changing the number of cores in which to run the problem simulations on, from 16 to 48. However, we were unsuccessful and were not able to create systems with film thicknesses that were larger than 2.25 nm. In addition, we were limited by the duration to which we can run our simulations. The cluster or supercomputer used to run our simulations had a walltime or timecap of fourteen days. For our smallest film thickness, the simulation would require approximately four days to complete, but our two larger film thicknesses, would require six or eight days, which already consumed half of the allowed walltime. To further explore this issue of creating thicker films, more cores can be used to better distribute the domain and decrease runtime, or a cluster with longer walltime may be used.

4.2.1.2 Signal–to–Noise Ratio & RNEMD

In our results for the actual viscosity and actual shear rate in Figure 3.3, we were unable to obtain shear rates slower than 1×10^9 1/s. It is known that a challenge in using molecular dynamics simulations to determine viscosity is that only very fast shear rates can be captured accurately. Liu et al. stated that the low signal–to–noise ratio intrinsic to MD simulations causes inaccurate viscosity calculations at moderate shear rates and so MD is only capable of capturing the viscosities in the shear thinning regime. The minimum achievable shear rate is roughly 1×10^8 1/s, which is still faster than the shear rates found in engineering applications and mechanical components [65]. This is illustrated in Figures 4.2 and 4.3 as plots of the velocity profile in the shearing direction with respect to the position in the film thickness direction, x. Figure 4.3 shows a linear velocity profile with a goodness value of $R^2 = 99.1\%$, resulting in a shear rate of 2.97×10^9 1/s for an imposed speed of 200 m/s. However, when imposing a much slower speed of 0.8 m/s, Figure 4.2 shows that the velocity profile is not linear and only results in a goodness value of $R^2 = 1.6\%$.

A method for obtaining accurate moderate shear rates in LAMMPS was developed by Müller-Plathe in 1999 for Lennard–Jones fluids, called Reverse Nonequilibrium Molecular Dynamics (RNEMD) [66]. In our simulations, which consisted of fluid molecules sandwiched between two walls, we applied a shear rate to our fluid by imposing speeds on both walls. We calculated viscosity based on the ratio between the shear stress outputted by LAMMPS and the shear rate. However, in RNEMD, a momentum flux is imposed and transferred within the system. The momentum

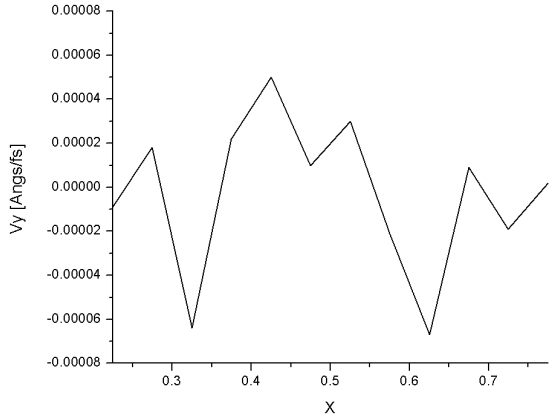

Figure 4.2: Velocity profile vs. x for the 200 molecule system (1.29 nm) with an imposed wall speed of 0.8 m/s.

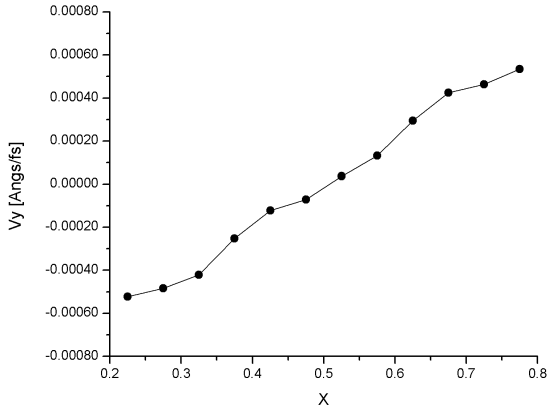

Figure 4.3: Velocity profile vs. x for the 200 molecule system (1.29 nm) with an imposed wall speed of 200 m/s.

transfer results in a velocity profile with a nonconventional V–shape. Figure 4.4 shows the schematic of the RNEMD method by Müller-Plathe [66]. In 2002, Bordat and Müller-Plathe further developed RNEMD to work for atomistic systems [67]. Since we were unable to obtain reliable actual shear rates slower than 1×10^9, investigation into RNEMD may be an option to circumvent the low signal–to–noise ratio.

4.2.1.3 Alternative Lubricant Selection

As previously mentioned, we used molecular dynamics simulations to attempt to create a viscosity vs. film thickness relationship for squalane. To circumvent this issue, a follow on project would be to select a lubricant that has existing viscosity vs. film thickness data and fit an exponential decay curve to that lubricant. For example, Luengo et al. found a viscosity vs film thickness relationship for polybutadiene [32]. A potential follow on project would be to reproduce EHL simulations using polybutadiene as the lubricant in EHL simulations rather than squalane.

4.2.2 EHL Simulation Issues & Areas of Improvements

While our attempt to enhance the traditional EHL numerical simulation algorithm was successful, there are issues and areas where the work could be extended. Particularly, we will discuss convergence, divergence, error, experimental accuracy and issues.

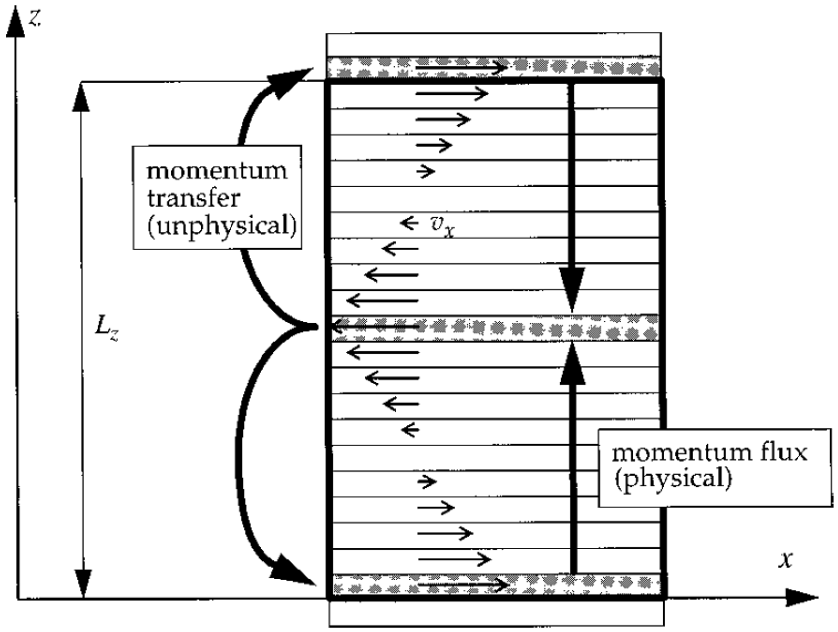

Figure 4.4: Schematic of the RNEMD method from [66].

4.2.2.1 Convergence & Error

As shown in Figure 3.4, we have a region where the error is minimal, under 3%. However, our simulation algorithm did not converge for values of A and t that gave relative errors under 3%. Therefore, we selected 205 and 1.6 for A and t, respectively, based on the minimum relative error with experimental data and the number of timesteps needed for the simulation to complete. By incorporating a film thickness dependence in our viscosity calculation, this caused oscillations in the criterion that we based convergence on, $\Delta p'$. We also observed that the central and minimum film thickness values oscillated as well. To illustrate the oscillations in $\Delta p'$ and central film thickness h_{cen}, both variables are plotted with respect to simulation timestep in Figures 4.5 and 4.6 for values of $A = 205$ and $t = 1.5$. These variables resulted in simulation convergence at 112 timesteps, which is much sooner than the 1320 timesteps needed for an enhanced EHL simulation with $A = 205$ and $t = 1.6$. Figures 4.5 and 4.6 show that the oscillation in $\Delta p'$ still results in simulation convergence. However, as mentioned previously, for central film thickness relative errors under 3%, our simulations would diverge. We categorized them as divergent after observing that $\Delta p'$ oscillated between two values consistently for at least 500 timesteps. There are a couple areas where this divergence issue may be investigated. It is possible that the simulations are converging, but at a slower rate. $\Delta p'$ is

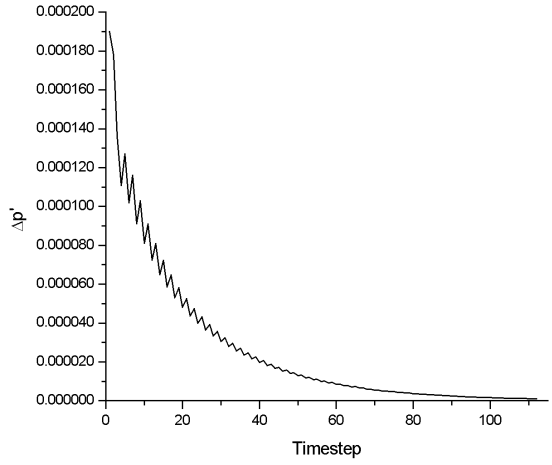

Figure 4.5: For values of $A = 205$ and $t = 1.5$, $\Delta p'$ converges to 9.96×10^{-7}.

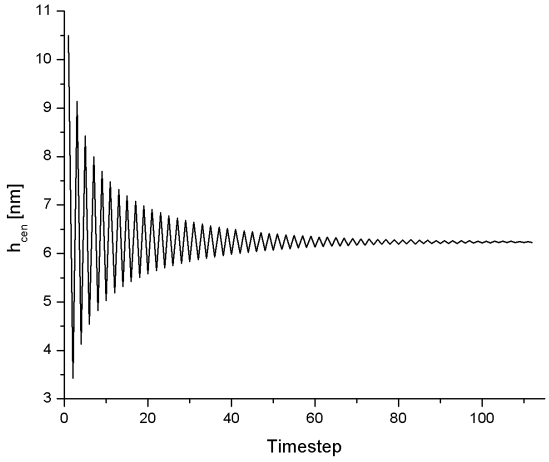

Figure 4.6: For values of $A = 205$ and $t = 1.5$, h_{cen} converges to 6.224 nm.

outputted with three significant digits. Adjusting the significant digits may show slow convergence. It is also possible that the fluctuations in $\Delta p'$ resulting from the numerical solution to the Reynolds equation may be causing the divergence. Further investigation in the numerical solution algorithm may also be explored.

4.2.2.2 Optical Interferometry Accuracy

Optical interferometry is a widely used technique for measuring lubricant film thickness in elastohydrodynamic lubrication experiments. However, one of the initial limitations of the method is that the it cannot accurately measure film thicknesses less than roughly 100 nm [68]. In 1991, Johnston et al. developed an extension to optical interferometry that allows for improved accuracy in measuring film thicknesses from 100 nm to 15 nm. Their experiment involved testing a number of lubricants, including squalane. Johnston et al. found that for squalane, there were anomalous film thickness effects that were not seen in the other lubricants tested. On a logarithmic scale of their film thickness vs. speed plot, they observed a change from linear to nonlinear behavior for film thicknesses below 15 nm. They believe that squalane is a lubricant that is dependent on speed rather than EHL theory when the film thickness in the range of 5–15 nm [68]. The possible causes for the anomalous behavior that can be investigated are the speed limitation of the experimental rig, viscosity effects related to molecular alignment of ultrathin films, load effects on the surfaces' elasticities, and surface roughness.

4.2.2.3 Optical Interferometry Phase Change

In 2001, Hartl et al. stated that their film thicknesses for squalane were thicker than the EHL theory prediction, which is consistent with the predictions made by Johnston et al. [20] [68]. In ball–on–disc EHL experiments using optical interferometry, ϕ is determined by calculating the reflection of two beams of light. The first beam of light reflects directly off of the glass, while the second beam passed through the lubricant film before being reflected off of steel ball. The beams of light then combine constructively or destructively to give the net phase change and the combination of light is processed by the researcher [68]. The inaccuracy of the experimental data may be due to the constant value given to ϕ, the net phase change that occurs upon reflection. Spikes and Cann stated that the value of ϕ is typically a constant between 0 and 0.2, indicating that the minimum film thickness that can be measured for any fluid using visible light is 45–75 nm [69]. Johnston et al. used a phase change value of 0.28 [68]. We compared our film thickness results to the experimental data obtained by Hartl et al. and the value for ϕ used was not reported. We hypothesize that, at the time that Hartl et al. conducted their EHL experiments using squalane, it is possible that they used a constant value of ϕ with white light. The issue with white light is that the resulting phase change is much more complex than a constant value. Upon reflection, all wavelengths that create the white light undergo partial or complete interference where the researcher perceives light as a single color when it is in reality much more complex [69]. Additional research in optical interferometry and EHL experiments may be conducted to investigate this hypothesis.

4.3 Conclusion

Our research aimed to characterize the relationship between film thickness and viscosity for a given lubricant using MD simulations so that we may observe how the lubricant behaves in thin films. In addition, our research also aimed to implement a film thickness viscosity model into a traditional EHL simulation to better predict film thicknesses closer to what is expected in experiments. In our MD simulations, our simple Lennard–Jones model successfully showed the trends described by Yamada et al. in Reference [33]. We observed that the thin film simple model had a larger ambient viscosity than the thick film model and began shear thinning at a lower shear rate than the thick film. Our molecular model of squalane gave only partially successful results where only the shear thinning regions were able to be captured with the three different film thicknesses that we constructed. However, in our EHL simulations, we were able to obtain convincing results. We validated our traditional numerical EHL algorithm against the film thickness results by Liu et al. in Reference [23]. Lastly, we modified the traditional algorithm to incorporate a film thickness dependence on viscosity, resulting in film thickness

results that saw much better agreement with the experimental data by Hartl et al. at thinner films [20].

BIBLIOGRAPHY

[1] G. W. Stachowiak, A. W. Batchelor, Engineering Tribology, Vol. 24 of Tribology Series, Elsevier Science Publishers B.V., 1993.

[2] K. Holmberg, P. Andersson, A. Erdemir, Global Energy Consumption Due to Friction in Passenger Cars, Tribology International 47 (2012) 221–234.

[3] K. Holmberg, R. Siilasto, T. Laitinen, P. Andersson, A. Jsberg, Global Energy Consumption Due to Friction in Paper Machines, Tribology International 62 (2013) 58–77.

[4] B. J. Hamrock, S. R. Schmid, B. O. Jacobson, Fundamentals of Fluid Film Lubrication, 2nd Edition, Marcel Dekker, Inc., 2004.

[5] S. S. Bair, High Pressure Rheology for Quantitative Elastohydrodynamics, 1st Edition, Tribology and Interface Engineering Series, No. 54, Elsevier, 2007.

[6] C. M. Mate, Tribology on the Small Scale: A Bottom Up Approach to Friction, Lubrication, and Wear, Mesoscopic Physics and Nanotechnology, Oxford University Press, 2008.

[7] Y. Wang, Q. J. Wang, Stribeck Curves, in: Q. J. Wang, Y. W. Chung (Eds.), Encyclopedia of Tribology, Springer US, 2013, pp. 3365–3370.

[8] R. Larsson, Point Contact EHL, in: Q. J. Wang, Y. W. Chung (Eds.), Encyclopedia of Tribology, Springer US, 2013, pp. 2549–2550.

[9] D. Zhu, H. S. Cheng, B. J. Hamrock, Effect of Surface Roughness on Pressure Spike and Film Constriction in Elastohydrodynamically Lubricated Line Contacts, Tribology Transactions 33 (2) (1990) 267–273.

[10] H. A. Spikes, Sixty Years of EHL, Lubrication Science 18 (4) (2006) 265–291.

[11] B. Hamrock, D. Dowson, Ball Bearing Lubrication: The Elastohydrodynamics of Elliptical Contacts, John Wiley & Sons, 1981.

[12] D. Dowson, Elastohydrodynamic and Micro–Elastohydrodynamic Lubrication, Wear 190 (2) (1995) 125–138.

[13] J. F. Archard, M. T. Kirk, Lubrication at Point Contacts, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 261 (1307) (1961) 532–550.

[14] A. Cameron, R. Gohar, Theoretical and Experimental Studies of the Oil Film in Lubricated Point Contact, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 291 (1427) (1966) 520–536.

[15] J. Luo, S. Wen, P. Huang, Thin Film Lubrication. Part I. Study on the Transition Between EHL and Thin Film Lubrication Using a Relative Optical Interference Intensity Technique, Wear 194 (12) (1996) 107–115.

[16] L. Ma, C. Zhang, Discussion on the Technique of Relative Optical Interference Intensity for the Measurement of Lubricant Film Thickness, Tribology Series 36 (3) (2009) 239–245.

[17] G. Popovici, C. Venner, P. Lugt, Effects of Load System Dynamics on the Film Thickness in EHL Contacts During Start Up, Journal of Tribology 126 (2) (2004) 258–266.

[18] H. Spikes, M. Ratoi, Molecular Scale Liquid Lubricating Films, in: D. Dowson, M. Priest, C. Taylor, P. Ehret, T.H.C, Childs, G. Dalmaz, A. Lubrecht, Y. Berthier, L. Flamand, J. Georges (Eds.), Thinning Films and Tribological Interfaces: Proceedings of the 26th Leeds-Lyon Symposium on Tribology, Vol. 38, Elsevier, 2000, pp. 359–370.

[19] M. Shen, J. Luo, S. Wen, Effects of Surface Physicochemical Properties on the Tribological Properties of Liquid Paraffin Film in the Nanoscale, Surface and Interface Analysis 32 (1) (2001) 286–288.

[20] M. Hartl, I. Krupka, R. Poliscuk, M. Liska, J. Molimard, M. Querry, P. Vergne, Thin Film Colorimetric Interferometry, Tribology Transactions 44 (2) (2001) 270–276.

[21] C. J. Pierre, D. Gérard, P. Vergne, Experimental Results and Analytical Predictions of EHL Film Thickness, 33rd Leeds Lyon Symposium on Tribology "Tribology at the Interface" (2007) 1–19.

[22] J. Guegan, A. Kadiric, H. Spikes, A Study of the Lubrication of EHL Point Contact in the Presence of Longitudinal Roughness, Tribology Series 59 (1) (2015) 1–18.

[23] Y. Liu, Q. J. Wang, W. Wang, Y. Hu, D. Zhu, I. Krupka, M. Hartl, EHL Simulation Using the Free–Volume Viscosity Model, Tribology Letters 23 (1) (2006) 27–37.

[24] B. R. Munson, T. H. Okiishi, W. W. Huebsch, A. P. Rothmayer, Fundamentals of Fluid Mechanics, 7th Edition, John Wiley & Sons, Inc., 2012.

[25] M. Masjedi, M. Khonsari, An Engineering Approach for Rapid Evaluation of Traction Coefficient and Wear in Mixed EHL, Tribology International 92 (2015) 184–190.

[26] H. A. Barnes, Shear–Thickening ("Dilatancy") in Suspensions of Nonaggregating Solid Particles Dispersed in Newtonian Liquids, Journal of Rheology 33 (2) (1989) 329–366.

[27] J. Koszkul, J. Nabialek, Viscosity Models in Simulation of the Filling Stage of the Injection Molding Process, Journal of Materials Processing Technology 157-158 (2004) 183–187.

[28] C. M. Tenney, E. J. Maginn, Limitations and Recommendations for the Calculation of Shear Viscosity using Reverse Nonequilibrium Molecular Dynamics, Journal of Chemical Physics 132 (1) (2010) 014103.

[29] M. L. Gee, P. M. McGuiggan, J. N. Israelachvili, A. M. Homola, Liquid to Solidlike Transitions of Molecularly Thin Films Under Shear, The Journal of Chemical Physics 93 (3) (1990) 1895–1906.

[30] H. W. Hu, G. A. Carson, S. Granick, Relaxation Time of Confined Liquids Under Shear, Physical Review Letters 66 (21) (1991) 2758–2761.

[31] Y. Zhu, S. Granick, Reassessment of Solidification in Fluids Confined Between Mica Sheets, Langmuir 19 (20) (2003) 8148–8151.

[32] G. Luengo, F. Schmitt, R. Hill, J. Israelachvili, Thin Film Rheology and Tribology of Confined Polymer Melts: Contrasts with Bulk Properties, Macromolecules 30 (8) (1997) 2482–2494.

[33] S. Yamada, General Shear–Thinning Dynamics of Confined Fluids, Tribology Letters 13 (3) (2002) 167–171.

[34] P. Padilla, S. Toxvaerd, Fluid Alkanes in Confined Geometries, The Journal of Chemical Physics 101 (2) (1994) 1490–1502.

[35] E. Manias, I. Bitsanis, G. Hadziioannou, G. ten Brinke, On the Nature of Shear Thinning in Nanoscopiacally Confined Films, Europhysics Letters 33 (5) (1996) 371–376.

[36] M. J. Stevens, M. Mondello, G. S. Grest, S. T. Cui, H. D. Cochran, P. T. Cummings, Comparison of Shear Flow of Hexadecane in a Confined Geometry and in Bulk, The Journal of Chemical Physics 106 (17) (1997) 7303–7314.

[37] S. T. Cui, C. McCabe, P. T. Cummings, H. D. Cochran, Molecular Dynamics Study of the Nano-Rheology of n-Dodecane Confined Between Planar Surfaces, The Journal of Chemical Physics 118 (19) (2003) 8941–8944.

[38] Y. Leng, P. T. Cummings, Fluidity of Hydration Layers Nanoconfined Between Mica Surfaces, Physical Review Letters 94 (2) (2005) 026101.

[39] F. Wang, Y. Zhao, The Unique Properties of the Solid-Like Confined Liquid Films: A Large Scale Molecular Dynamics Simulation Approach, Acta Mechanica Solida Sinica 24 (2) (2011) 101–116.

[40] A. Jabbarzadeh, J. D. Atkinson, R. I. Tanner, Nanorheology of Molecularly Thin Films of n–Hexadecane in Couette Shear Flow by Molecular Dynamics Simulation, Journal of Non–Newtonian Fluid Mechanics 77 (1-2) (1998) 53–78.

[41] A. Martini, Y. Liu, R. Snurr, Q. J. Wang, Molecular Dynamics Characterization of Thin Film Viscosity for EHL Simulation, Tribology Letters 21 (3) (2006) 217–225.

[42] H. Eslami, F. Mozaffari, J. Moghadasi, F. Müller-Plathe, Molecular Dynamics Simulation of Confined Fluids in Isosurface–Isothermal–Isobaric Ensemble, The Journal of Chemical Physics 129 (19) (2008) 194702.

[43] H. Eslami, F. Müller-Plathe, Viscosity of Nanoconfined Polyamide–6,6 Oligomers: Atomistic Reverse Nonequilibrium Molecular Dynamics Simulation, The Journal of Physical Chemistry B 114 (1) (2010) 387–395.

[44] Y. Y. Gotlib, N. K. Balabaev, A. A. Darinskii, I. M. Neelov, Investigation of Local Motions in Polymers by the Method of Molecular Dynamics, Macromolecules 13 (3) (1980) 602–608.

[45] G. S. Grest, K. Kremer, Molecular Dynamics Simulation for Polymers in the Presence of a Heat Bath, Physical Review A 33 (1986) 3628–3631.

[46] S. Plimpton, Fast Parallel Algorithms for Short–Range Molecular Dynamics, Journal of Computational Physics 117 (1) (1995) 1–19.

[47] W. Jorgensen, D. Maxwell, J. Tirado-Rives, Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids, Journal of the American Chemical Society 118 (1996) 11225–11236.

[48] G. A. Kaminski, R. A. Friesner, J. Tirado-Rives, W. L. Jorgensen, Evaluation and Reparametrization of the OPLS–AA Force Field for Proteins via Comparison with Accurate Quantum Chemical Calculations on Peptides, The Journal of Physical Chemistry B 105 (2001) 6474–6487.

[49] W. L. Jorgensen, D. S. Maxwell, J. Tirado-Rives, Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids, Journal of the American Chemical Society 118 (45) (1996) 11225–11236.

[50] A. Jewett, Moltemplate Manual, University of California, Santa Barbara Shea Lab (August 2015).

[51] A. Lima, A. Martins, J. S. Martins, Lennard–Jones Binary Fluids: A Comparative Study Between the Molecular Dynamics and Monte Carlo Descriptions of Their Structural Properties, Physica A: Statistical Mechanics and its Applications 391 (18) (2012) 4281–4289.

[52] S. Plimpton, A. Thompson, P. Crozier, LAMMPS Documentation, Sandia National Laboratories (January 2016).

[53] A. Stukowski, Visualization and Analysis of Atomistic Simulation Data with OVITO-The Open Visualization Tool, Modelling and Simulation in Materials Science and Engineering 18 (1) (2010) 015012.

[54] Y. Z. Hu, D. Zhu, A Full Numerical Solution to the Mixed Lubrication in Point Contacts, Journal of Tribology 122 (1) (2000) 1–9.

[55] J. F. Lin, H. Y. Chu, A Numerical Solution for Calculating Elastic Deformation in Elliptical–Contact EHL of Rough Surface, Journal of Tribology 113 (1991) 12–21.

[56] D. Dowson, G. Higginson, Elasto–hydrodynamic Lubrication: The Fundamentals of Roller and Gear Lubrication, The Commonwealth and international library, Pergamon Press, 1966.

[57] S. Bair, Y. Liu, Q. J. Wang, The Pressure–Viscosity Coefficient for Newtonian EHL Film Thickness with General Piezoviscous Response, Journal of Tribology 128 (3) (2006) 624–631.

[58] C. J. A. Roelands, Correlational Aspects of the Viscosity–Temperature–Pressure Relationship of Tubricating Oils, Ph.D. thesis, TU Delft, Delft University of Technology (1966).

[59] S. Bair, The High Pressure Rheology of Some Simple Model Hydrocarbons, Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology 216 (3) (2002) 139–149.

[60] S. Bair, Roelands' Missing Data, Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology 218 (3) (2004) 237–237.

[61] V. Petrone, A. Senatore, V. D'Agostino, Effect of an Improved Yasutomi Pressure–Viscosity Relationship on the Elastohydrodynamic Line Contact Problem, ISRN Tribology 2013 (149451) (2013) 1–7.

[62] S. Bair, Y. Liu, Q. J. Wang, The Pressure–Viscosity Coefficient for Newtonian EHL Film Thickness with General Piezoviscous Response, Journal of Tribology 128 (3) (2006) 624–631.

[63] P. Vergne, S. Bair, Classical EHL Versus Quantitative EHL: A Perspective Part I—Real Viscosity–Pressure Dependence and the Viscosity–Pressure Coefficient for Predicting Film Thickness, Tribology Letters 54 (1) (2014) 1–12.

[64] S. Bair, C. McCabe, P. T. Cummings, Comparison of Nonequilibrium Molecular Dynamics with Experimental Measurements in the Nonlinear Shear–Thinning Regime, Phys. Rev. Lett. 88 (5) (2002) 058302.

[65] P. Liu, H. Yu, N. Ren, F. E. Lockwood, Q. J. Wang, Pressure–Viscosity Coefficient of Hydrocarbon Base Oil through Molecular Dynamics Simulations, Tribology Letters 60 (3) (2015) 1–9.

[66] F. Müller-Plathe, Reversing the Perturbation in Nonequilibrium Molecular Dynamics: An Easy Way to Calculate the Shear Viscosity of Fluids, Physical Review E 59 (5) (1999) 4894–4898.

[67] P. Bordat, F. Müller-Plathe, The Shear Viscosity of Molecular Fluids: A Calculation by Reverse Nonequilibrium Molecular Dynamics, The Journal of Chemical Physics 116 (8) (2002) 3362–3369.

[68] G. Johnston, R. Wayte, H. A. Spikes, The Measurement and Study of Very Thin Lubricant Films in Concentrated Contacts, Tribology Transactions 34 (2) (1991) 187–194.

[69] H. A. Spikes, P. M. Cann, The Development and Application of the Spacer Layer Imaging Method for Measuring Lubricant Film Thickness, Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology 215 (3) (2001) 261–277.

APPENDIX A

LAMMPS INPUT FILES

A.1 run.nemdljwall

```
# Input file to create and shear a confined Lennard-Jones fluid

# -------------- Parameter Definition -----------------------------

variable        rho equal 0.849
variable        t equal 0.722
variable        rc equal 2.5

variable        x equal 3
variable        y equal 3
variable        z equal 3.50
variable        diff equal 0.25
variable        zlo equal -${diff}
variable        zhi equal ${z}+0.25

# -------------- Simulation Setup ---------------------------------

units           lj
dimension       3
atom_style      atomic
neigh_modify    delay 0 every 1

lattice         fcc ${rho}
region          simbox block 0 ${x} 0 ${y} ${zlo} ${zhi}
create_box      3 simbox
create_atoms    1 box
change_box      all boundary p p s

mass            * 1.0

pair_style      lj/cut ${rc}
pair_coeff      * * 1 1

# -------------- Create Walls -------------------------------------

region          lower block INF INF INF INF INF 0.0
region          upper block INF INF INF INF ${z} INF
group           lower region lower
group           upper region upper
set             group lower type 2
set             group upper type 3
group           wall union lower upper
```

```
43 group           flow subtract all wall
44
45 # -------------- Equilibration ----------------------------------
46
47 velocity        flow create $t 498756
48 fix             move upper move linear 0.01 NULL NULL units box
49 fix             1 upper setforce 0.0 0.0 0.0
50 fix             2 lower setforce 0.0 0.0 0.0
51 fix             3 all nve
52 fix             4 all temp/berendsen ${t} ${t} 0.1
53 dump            1 all atom 10000 LJ_walls_NEMD.lammpstrj
54 compute         uspeed upper reduce ave vx
55 compute         com upper com
56 thermo_style    custom step temp pe c_com[2] c_uspeed
57 thermo          1000
58 variable        eq1time equal 25000
59 run             ${eq1time}
60
61 # -------------- Data Gathering ---------------------------------
62
63 fix             6 all ave/spatial 1 10000 10000 z center 0.125 vx units
       reduced ave running file vxprofile.running
64 thermo_style    custom step pe temp pxz
65 run             40000000
```

A.2 run.sqaequilibrate

```
 1 # -- Initialization Section -------------------------------------
 2
 3 include         system.in.init
 4
 5 # -- Atom Definition Section -------------------------------------
 6
 7 read_data       system.data
 8
 9 # -- Settings Section --------------------------------------------
10
11 include         system.in.settings
12 include         system.in.charges
13
14 # -- Create walls -----------------------------------------------
15
16 region          lower block INF -5 INF INF INF INF
17 region          upper block 85 INF INF INF INF INF
18 lattice         fcc 3.625
19 create_atoms    5 region lower
20 create_atoms    5 region upper
21
22 change_box      all boundary s p p
23
24 group           lower region lower
25 group           upper region upper
26 group           fluid subtract all upper lower
27 fix             1 lower setforce 0 0 0
28 fix             2 upper setforce 0 0 0
29 #fix            2 upper rigid single torque 1 off off off
30
```

```
31 # -- Equilibration Part 1: Initial minimization -------------------
32
33 velocity        fluid create 298.15 482748 dist gaussian mom no rot no
34 thermo_style    custom step temp pe ke vol press
35 timestep        1.0
36 dump            1 all custom 10 sqa_min.lammpstrj id mol type xu yu zu
37
38 minimize        1.0e-16 1.0e-17 100000 400000
39 undump          1
40
41 # -- Equilibration Part 2: Relax molecules at high temperature ----
42
43 neigh_modify    every 10 delay 20 check yes
44 thermo          1000
45 dump            1 all custom 1000 sqa_nve1000.lammpstrj id mol type xu
       yu zu
46 fix             3 fluid langevin 1000.0 1000.0 100.0 482748
47 fix             4 fluid nve
48 compute         2 upper com
49 thermo_style    custom step temp pe c_2[1] c_2[2] c_2[3]
50 run             500000
51 undump          1
52
53 # -- Equilibration Part 3: Move the upper wall down ---------------
54
55 unfix           2
56 fix             2 upper rigid single torque 1 off off off
57 unfix           3
58 fix             3 fluid langevin 298.15 298.15 100 482748
59 fix             5 upper addforce -0.15 0 0
60 dump            1 all custom 50000 squish_u.lammpstrj id mol type xu yu
       zu
61 dump            2 all custom 50000 squish_w.lammpstrj id mol type x y z
62 compute         1 upper reduce min x
63 thermo_style    custom step temp pe c_1 c_2[1] c_2[2] c_2[3]
64
65 run             2000000
66
67 write_restart   squish.restart
```

A.3 run.sqashearwalls

```
 1 # -- Read restart and other files ---------------------------------
 2
 3 units           real
 4 dimension       3
 5 boundary        s p p
 6 neigh_modify    every 10 delay 20 check yes
 7
 8 atom_style      full
 9 #bond_style     hybrid harmonic
10 #angle_style    hybrid harmonic
11 #dihedral_style hybrid opls
12 pair_style      hybrid lj/cut/coul/cut 10.0
13 pair_modify     mix geometric
14 special_bonds   lj/coul 0.0 0.0 0.5
15 kspace_style    none
```

```
16
17 read_restart    squish.restart
18
19 include         system.in.settings
20 include         system.in.charges
21
22 # -- Recreate fixes and computes from previous run ---------------
23
24 compute         newTemp fluid temp/profile 1 0 1 y 20
25 fix             1 fluid nve
26 fix             2 fluid langevin 298.15 298.15 100 482748
27 fix_modify      2 temp newTemp
28 fix             3 lower setforce 0 0 0
29 fix             4 upper move linear 0 0 0 units box
30
31 # -- Let system equilibrate -------------------------------------
32
33 thermo          1000
34 compute         1 upper reduce min x
35 compute         2 upper com
36 thermo_style    custom step temp pxy c_1 c_2[1] c_2[2] c_2[3]
37 dump            1 all custom 10000 shear_u.lammpstrj id mol type xu yu
       zu
38 dump            2 all custom 10000 shear_w.lammpstrj id mol type x y z
39 run             500000
40
41 # -- Move walls laterally ---------------------------------------
42
43 unfix           4
44 fix             10 upper move linear 0 0.0015 0 units box
45 fix             11 lower move linear 0 -0.0015 0 units box
46 fix             8 fluid ave/spatial 100000 1 100000 x lower 0.05 vy
       units reduced ave running file vyprofile.fluid
47 run             10000000
```

APPENDIX B

MOLTEMPLATE FILES

B.1 ch1group.lt

```
import "oplsaa.lt"    # <-- defines the "OPLSAA" force field

CH1 inherits OPLSAA {

  # Format: atom-id mol-id atom-type charge x y z

  write("Data Atoms") {
    $atom:C  $mol:... @atom:82  0.00   0.000  0.000             0.000
    $atom:H1 $mol:... @atom:83  0.00   0.000  0.63104384422426
       0.892430762954
  }

  # Specify which pairs of atoms are bonded:
  write('Data Bond List') {
    $bond:CH1 $atom:C $atom:H1
  }

} # CH1

CH1.move(0,0.4431163,0)
```

B.2 ch2group.lt

```
import "oplsaa.lt"    # <-- defines the "OPLSAA" force field

CH2 inherits OPLSAA {

  # Format: atom-id mol-id atom-type charge x y z

  write("Data Atoms") {
    $atom:C  $mol:... @atom:81  0.00   0.000  0.000             0.000
    $atom:H1 $mol:... @atom:85  0.00   0.000  0.63104384422426
       0.892430762954
    $atom:H2 $mol:... @atom:85  0.00   0.000  0.63104384422426
       -0.892430762954
  }

  # Specify which pairs of atoms are bonded:
  write('Data Bond List') {
    $bond:CH1 $atom:C $atom:H1
    $bond:CH2 $atom:C $atom:H2
  }

```

```
} # CH2

CH2.move(0,0.4431163,0)
```

B.3 ch3group.lt

```
import "oplsaa.lt"    # <-- defines the "OPLSAA" force field

CH3 inherits OPLSAA {

  # Format: atom-id mol-id atom-type charge x y z

  write("Data Atoms") {
    $atom:C  $mol:... @atom:81  0.00   0.000  0.000             0.000
    $atom:H1 $mol:... @atom:85  0.00   0.000  0.63104384422426
       0.892430762954
    $atom:H2 $mol:... @atom:85  0.00   0.000  0.63104384422426
       -0.892430762954
        $atom:H3 $mol:... @atom:85  0.00  -0.8924307629540046
           -0.6310438442242609 0.000000
  }

  # Specify which pairs of atoms are bonded:
  write('Data Bond List') {
    $bond:CH1 $atom:C $atom:H1
    $bond:CH2 $atom:C $atom:H2
    $bond:CH3 $atom:C $atom:H3
  }


} # CH3

CH3.move(0,0.4431163,0)
```

B.4 squalane.lt

```
import "oplsaa.lt"    # load the "OPLSAA" force-field information
import "ch1group.lt"  # load the definition of the "CH1" object
import "ch2group.lt"  # load the definition of the "CH2" object
import "ch3group.lt"  # load the definition of the "CH3" object

Squalane inherits OPLSAA {

  create_var {$mol}

  # Create an array of 30 "CH2" objects distributed along the x-axis

  monomers = new CH2 [30].rot(180,1,0,0).move(1.2533223,0,0)

  # Each monomer is rotated 180 degrees with respect to the previous
  # monomer, and then moved 1.2533223 Angstroms down the x-axis.

  # Modify CH2 with CH3

  delete monomers[0]
  delete monomers[23]

  monomers[0] = new CH3
```

```
23  monomers[23] = new CH3
24
25  delete monomers[24]
26  delete monomers[25]
27  delete monomers[26]
28  delete monomers[27]
29  delete monomers[28]
30  delete monomers[29]
31
32  monomers[24] = new CH3
33  monomers[25] = new CH3
34  monomers[26] = new CH3
35  monomers[27] = new CH3
36  monomers[28] = new CH3
37  monomers[29] = new CH3
38
39  # Modify CH2 with CH
40
41  delete monomers[1]
42  delete monomers[5]
43  delete monomers[9]
44  delete monomers[14]
45  delete monomers[18]
46  delete monomers[22]
47
48  monomers[1] = new CH1
49  monomers[5] = new CH1
50  monomers[9] = new CH1
51  monomers[14] = new CH1
52  monomers[18] = new CH1
53  monomers[22] = new CH1
54
55  # Move the CH3 groups to their correct locations:
56
57  monomers[0].rot(180,0,0,1).move(1,0,0)
58  monomers[23].rot(180,0,0,1).move(29.8264129,0,0)
59
60  monomers[24].rot(180,0,0,1).move(2.2533223,-1.2533223,0)
61  monomers[25].rot(180,0,0,1).move(7.2666115,-1.2533223,0)
62  monomers[26].rot(180,0,0,1).move(12.2799007,-1.2533223,0)
63  monomers[27].rot(180,0,0,1).move(18.5465122,1.2533223,0)
64  monomers[28].rot(180,0,0,1).move(23.5598014,1.2533223,0)
65  monomers[29].rot(180,0,0,1).move(28.5730906,1.2533223,0)
66
67  # Move the CH groups to their correct locations:
68
69  monomers[1].rot(180,0,0,0).move(2.2533223,0,0)
70  monomers[5].rot(180,0,0,0).move(7.2666115,0,0)
71  monomers[9].rot(180,0,0,0).move(12.2799007,0,0)
72  monomers[14].rot(0,0,0,0).move(18.5465122,0,0)
73  monomers[18].rot(0,0,0,0).move(23.5598014,0,0)
74  monomers[22].rot(0,0,0,0).move(28.5730906,0,0)
75
76  # Create a list of bonds connecting the carbon atoms together:
77
78  write('Data Bond List') {
```

```
    $bond:b1  $atom:monomers[0]/C $atom:monomers[1]/C
    $bond:b2  $atom:monomers[1]/C $atom:monomers[2]/C
    $bond:b3  $atom:monomers[2]/C $atom:monomers[3]/C
    $bond:b4  $atom:monomers[3]/C $atom:monomers[4]/C
    $bond:b5  $atom:monomers[4]/C $atom:monomers[5]/C
    $bond:b6  $atom:monomers[5]/C $atom:monomers[6]/C
    $bond:b7  $atom:monomers[6]/C $atom:monomers[7]/C
    $bond:b8  $atom:monomers[7]/C $atom:monomers[8]/C
    $bond:b9   $atom:monomers[8]/C $atom:monomers[9]/C
    $bond:b10  $atom:monomers[9]/C $atom:monomers[10]/C
    $bond:b11  $atom:monomers[10]/C $atom:monomers[11]/C
    $bond:b12  $atom:monomers[11]/C $atom:monomers[12]/C
    $bond:b13  $atom:monomers[12]/C $atom:monomers[13]/C
    $bond:b14  $atom:monomers[13]/C $atom:monomers[14]/C
    $bond:b15  $atom:monomers[14]/C $atom:monomers[15]/C
    $bond:b16  $atom:monomers[15]/C $atom:monomers[16]/C
    $bond:b17  $atom:monomers[16]/C $atom:monomers[17]/C
    $bond:b18  $atom:monomers[17]/C $atom:monomers[18]/C
    $bond:b19  $atom:monomers[18]/C $atom:monomers[19]/C
    $bond:b20  $atom:monomers[19]/C $atom:monomers[20]/C
    $bond:b21  $atom:monomers[20]/C $atom:monomers[21]/C
    $bond:b22  $atom:monomers[21]/C $atom:monomers[22]/C
    $bond:b23  $atom:monomers[22]/C $atom:monomers[23]/C
    $bond:b24  $atom:monomers[1]/C $atom:monomers[24]/C
    $bond:b25  $atom:monomers[5]/C $atom:monomers[25]/C
    $bond:b26  $atom:monomers[9]/C $atom:monomers[26]/C
    $bond:b27  $atom:monomers[14]/C $atom:monomers[27]/C
    $bond:b28  $atom:monomers[18]/C $atom:monomers[28]/C
    $bond:b29  $atom:monomers[22]/C $atom:monomers[29]/C
  }

} # Squalane
```

B.5 oplsaa.lt

```
OPLSAA {

  write_once("Data Masses"){
    @atom:80 12.011 #CT "Alkane CH3-"  6 partial charge=-0.18
    @atom:81 12.011 #CT "Alkane -CH2-" 6 partial charge=-0.12
    @atom:82 12.011 #CT "Alkane -CH-"  6 partial charge=-0.06
    @atom:85 1.008  #HC "Alkane H-C"   1 partial charge= 0.06
  } #(end of atom masses)

  write_once("In Settings"){
    pair_coeff @atom:80 @atom:80 lj/cut 0.066 3.5
    pair_coeff @atom:81 @atom:81 lj/cut 0.066 3.5
    pair_coeff @atom:82 @atom:82 lj/cut 0.066 3.5
    pair_coeff @atom:83 @atom:83 lj/cut 0.03  2.5
  } #(end of pair coeffs)

  write_once("In Charges"){
    set type @atom:80 charge -0.18
    set type @atom:81 charge -0.12
    set type @atom:82 charge -0.06
    set type @atom:83 charge  0.06
  } #(end of atom charges)
```

```
23
24  write_once("In Settings") {
25    bond_coeff @bond:80-80 harmonic 268.0 1.529
26    bond_coeff @bond:80-81 harmonic 268.0 1.529
27    bond_coeff @bond:81-80 harmonic 268.0 1.529
28    bond_coeff @bond:81-81 harmonic 268.0 1.529
29    bond_coeff @bond:80-83 harmonic 340.0 1.09
30    bond_coeff @bond:81-83 harmonic 340.0 1.09
31    bond_coeff @bond:82-82 harmonic 268.0 1.529
32    bond_coeff @bond:80-82 harmonic 268.0 1.529
33    bond_coeff @bond:81-82 harmonic 268.0 1.529
34    bond_coeff @bond:82-83 harmonic 340.0 1.09
35  } #(end of bond_coeffs)
36
37  write_once("Data Bonds By Type") {
38    @bond:80-80 @atom:80 @atom:80
39    @bond:80-81 @atom:80 @atom:81
40    @bond:81-80 @atom:81 @atom:80
41    @bond:81-81 @atom:81 @atom:81
42    @bond:80-83 @atom:80 @atom:83
43    @bond:81-83 @atom:81 @atom:83
44    @bond:82-82 @atom:82 @atom:82
45    @bond:80-82 @atom:80 @atom:82
46    @bond:81-82 @atom:81 @atom:82
47    @bond:82-83 @atom:82 @atom:83
48  } #(end of bonds by type)
49
50  write_once("Data Angles By Type"){
51    @angle:80-80-80 @atom:80 @atom:80 @atom:80
52    @angle:80-80-81 @atom:80 @atom:80 @atom:81
53    @angle:80-81-80 @atom:80 @atom:81 @atom:80
54    @angle:80-81-81 @atom:80 @atom:81 @atom:81
55    @angle:81-80-80 @atom:81 @atom:80 @atom:80
56    @angle:81-80-81 @atom:81 @atom:80 @atom:81
57    @angle:81-81-80 @atom:81 @atom:81 @atom:80
58    @angle:81-81-81 @atom:81 @atom:81 @atom:81
59    @angle:83-80-83 @atom:83 @atom:80 @atom:83
60    @angle:83-81-83 @atom:83 @atom:81 @atom:83
61    @angle:80-80-83 @atom:80 @atom:80 @atom:83
62    @angle:80-81-83 @atom:80 @atom:81 @atom:83
63    @angle:81-80-83 @atom:81 @atom:80 @atom:83
64    @angle:81-81-83 @atom:81 @atom:81 @atom:83
65    @angle:80-82-80 @atom:80 @atom:82 @atom:80
66    @angle:80-82-81 @atom:80 @atom:82 @atom:81
67    @angle:81-82-81 @atom:81 @atom:82 @atom:81
68    @angle:80-82-83 @atom:80 @atom:82 @atom:83
69    @angle:81-82-83 @atom:81 @atom:82 @atom:83
70  } #(end of angles by type)
71
72  write_once("In Settings" ){
73    angle_coeff @angle:80-80-80 harmonic 58.35 112.7
74    angle_coeff @angle:80-80-81 harmonic 58.35 112.7
75    angle_coeff @angle:80-81-80 harmonic 58.35 112.7
76    angle_coeff @angle:80-81-81 harmonic 58.35 112.7
77    angle_coeff @angle:81-80-80 harmonic 58.35 112.7
78    angle_coeff @angle:81-80-81 harmonic 58.35 112.7
```

```
  angle_coeff @angle:81-81-80 harmonic 58.35 112.7
  angle_coeff @angle:81-81-81 harmonic 58.35 112.7
  angle_coeff @angle:83-80-83 harmonic 33.0 107.8
  angle_coeff @angle:83-81-83 harmonic 33.0 107.8
  angle_coeff @angle:80-80-83 harmonic 37.5 110.7
  angle_coeff @angle:80-81-83 harmonic 37.5 110.7
  angle_coeff @angle:81-80-83 harmonic 37.5 110.7
  angle_coeff @angle:81-81-83 harmonic 37.5 110.7
  angle_coeff @angle:80-82-80 harmonic 58.35 112.7
  angle_coeff @angle:80-82-81 harmonic 58.35 112.7
  angle_coeff @angle:81-82-81 harmonic 58.35 112.7
  angle_coeff @angle:80-82-83 harmonic 37.5 110.7
  angle_coeff @angle:81-82-83 harmonic 37.5 110.7
} #(end of angle_coeffs)

write_once("Data Dihedrals By Type") {
  @dihedral:0-80-80-80 @atom:* @atom:80 @atom:80 @atom:80
  @dihedral:0-80-80-81 @atom:* @atom:80 @atom:80 @atom:81
  @dihedral:0-80-81-80 @atom:* @atom:80 @atom:81 @atom:80
  @dihedral:0-80-81-81 @atom:* @atom:80 @atom:81 @atom:81
  @dihedral:0-81-80-80 @atom:* @atom:81 @atom:80 @atom:80
  @dihedral:0-81-80-81 @atom:* @atom:81 @atom:80 @atom:81
  @dihedral:0-81-81-80 @atom:* @atom:81 @atom:81 @atom:80
  @dihedral:0-81-81-81 @atom:* @atom:81 @atom:81 @atom:81
  @dihedral:80-80-80-80 @atom:80 @atom:80 @atom:80 @atom:80
  @dihedral:80-80-80-81 @atom:80 @atom:80 @atom:80 @atom:81
  @dihedral:80-80-81-80 @atom:80 @atom:80 @atom:81 @atom:80
  @dihedral:80-80-81-81 @atom:80 @atom:80 @atom:81 @atom:81
  @dihedral:80-81-80-80 @atom:80 @atom:81 @atom:80 @atom:80
  @dihedral:80-81-80-81 @atom:80 @atom:81 @atom:80 @atom:81
  @dihedral:80-81-81-80 @atom:80 @atom:81 @atom:81 @atom:80
  @dihedral:80-81-81-81 @atom:80 @atom:81 @atom:81 @atom:81
  @dihedral:81-80-80-80 @atom:81 @atom:80 @atom:80 @atom:80
  @dihedral:81-80-80-81 @atom:81 @atom:80 @atom:80 @atom:81
  @dihedral:81-80-81-80 @atom:81 @atom:80 @atom:81 @atom:80
  @dihedral:81-80-81-81 @atom:81 @atom:80 @atom:81 @atom:81
  @dihedral:81-81-80-80 @atom:81 @atom:81 @atom:80 @atom:80
  @dihedral:81-81-80-81 @atom:81 @atom:81 @atom:80 @atom:81
  @dihedral:81-81-81-80 @atom:81 @atom:81 @atom:81 @atom:80
  @dihedral:81-81-81-81 @atom:81 @atom:81 @atom:81 @atom:81
  @dihedral:80-80-80-83 @atom:80 @atom:80 @atom:80 @atom:83
  @dihedral:80-80-81-83 @atom:80 @atom:80 @atom:81 @atom:83
  @dihedral:80-81-80-83 @atom:80 @atom:81 @atom:80 @atom:83
  @dihedral:80-81-81-83 @atom:80 @atom:81 @atom:81 @atom:83
  @dihedral:81-80-80-83 @atom:81 @atom:80 @atom:80 @atom:83
  @dihedral:81-80-81-83 @atom:81 @atom:80 @atom:81 @atom:83
  @dihedral:81-81-80-83 @atom:81 @atom:81 @atom:80 @atom:83
  @dihedral:81-81-81-83 @atom:81 @atom:81 @atom:81 @atom:83
  @dihedral:83-80-80-83 @atom:83 @atom:80 @atom:80 @atom:83
  @dihedral:83-80-81-83 @atom:83 @atom:80 @atom:81 @atom:83
  @dihedral:83-81-80-83 @atom:83 @atom:81 @atom:80 @atom:83
  @dihedral:83-81-81-83 @atom:83 @atom:81 @atom:81 @atom:83

  @dihedral:80-82-81-81 @atom:80 @atom:82 @atom:81 @atom:81
  @dihedral:81-81-81-82 @atom:81 @atom:81 @atom:81 @atom:82
  @dihedral:81-82-81-81 @atom:81 @atom:82 @atom:81 @atom:81
```

```
    @dihedral:80-82-80-83 @atom:80 @atom:82 @atom:80 @atom:83
    @dihedral:80-82-81-83 @atom:80 @atom:82 @atom:81 @atom:83
    @dihedral:81-82-80-83 @atom:81 @atom:82 @atom:80 @atom:83
    @dihedral:81-82-81-83 @atom:81 @atom:82 @atom:81 @atom:83
    @dihedral:82-81-81-83 @atom:82 @atom:81 @atom:81 @atom:83
    @dihedral:83-80-82-83 @atom:83 @atom:80 @atom:82 @atom:83
    @dihedral:83-81-82-83 @atom:83 @atom:81 @atom:82 @atom:83
  } #(end of Dihedrals by type)

  write_once("In Settings") {
    dihedral_coeff @dihedral:0-80-80-80 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:0-80-80-81 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:0-80-81-80 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:0-80-81-81 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:0-81-80-80 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:0-81-80-81 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:0-81-81-80 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:0-81-81-81 opls 1.711 -0.5 0.663 0.0
    dihedral_coeff @dihedral:80-80-80-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-80-80-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-80-81-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-80-81-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-81-80-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-81-80-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-81-81-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-81-81-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-80-80-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-80-80-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-80-81-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-80-81-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-81-80-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-81-80-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-81-81-80 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-81-81-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-80-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:80-80-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:80-81-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:80-81-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:81-80-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:81-80-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:81-81-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:81-81-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:83-80-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:83-80-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:83-81-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:83-81-81-83 opls 0.0 0.0 0.3 0.0

    dihedral_coeff @dihedral:80-82-81-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-81-81-82 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:81-82-81-81 opls 1.3 -0.05 0.2 0.0
    dihedral_coeff @dihedral:80-82-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:80-82-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:81-82-80-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:81-82-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:82-81-81-83 opls 0.0 0.0 0.3 0.0
    dihedral_coeff @dihedral:83-80-82-83 opls 0.0 0.0 0.3 0.0
```

```
191     dihedral_coeff @dihedral:83-81-82-83 opls 0.0 0.0 0.3 0.0
192   } #(end of dihedral_coeffs)
193
194   write_once("Data Impropers By Type (opls_imp.py)") {
195   } #(end of impropers by type)
196
197   write_once("In Settings") {
198   } #(end of improp_coeffs)
199
200   write_once("In Init") {
201     units real
202     atom_style full
203     bond_style hybrid harmonic
204     angle_style hybrid harmonic
205     dihedral_style hybrid opls
206     improper_style hybrid harmonic
207     pair_style hybrid lj/cut 10.0
208     pair_modify mix geometric
209     special_bonds lj/coul 0.0 0.0 0.5
210     kspace_style none
211   } #end of init parameters
212
213 } # OPLSAA
```

B.6 system.in.charges

```
1     set type 1 charge -0.18
2     set type 2 charge -0.12
3     set type 3 charge -0.06
4     set type 4 charge  0.06
```

B.7 system.in.init

```
1     units real
2     atom_style full
3     bond_style hybrid harmonic
4     angle_style hybrid harmonic
5     dihedral_style hybrid opls
6     pair_style hybrid lj/cut/coul/cut 10.0
7     pair_modify mix geometric
8     special_bonds lj/coul 0.0 0.0 0.5
9     kspace_style none
```

B.8 system.in.settings

```
1     pair_coeff 1 1 lj/cut/coul/cut 0.066 3.5
2     pair_coeff 2 2 lj/cut/coul/cut 0.066 3.5
3     pair_coeff 3 3 lj/cut/coul/cut 0.066 3.5
4     pair_coeff 4 4 lj/cut/coul/cut 0.03 2.5
5     pair_coeff 5 5 lj/cut/coul/cut 6.463 2.242
6     bond_coeff 1 harmonic 268.0 1.529
7     bond_coeff 2 harmonic 268.0 1.529
8     bond_coeff 3 harmonic 268.0 1.529
9     bond_coeff 4 harmonic 268.0 1.529
10    bond_coeff 5 harmonic 340.0 1.09
11    bond_coeff 6 harmonic 340.0 1.09
12    bond_coeff 7 harmonic 268.0 1.529
13    bond_coeff 8 harmonic 268.0 1.529
14    bond_coeff 9 harmonic 268.0 1.529
```

```
bond_coeff 10 harmonic 340.0 1.09
angle_coeff 1 harmonic 58.35 112.7
angle_coeff 2 harmonic 58.35 112.7
angle_coeff 3 harmonic 58.35 112.7
angle_coeff 4 harmonic 58.35 112.7
angle_coeff 5 harmonic 58.35 112.7
angle_coeff 6 harmonic 58.35 112.7
angle_coeff 7 harmonic 58.35 112.7
angle_coeff 8 harmonic 58.35 112.7
angle_coeff 9 harmonic 33.0 107.8
angle_coeff 10 harmonic 33.0 107.8
angle_coeff 11 harmonic 37.5 110.7
angle_coeff 12 harmonic 37.5 110.7
angle_coeff 13 harmonic 37.5 110.7
angle_coeff 14 harmonic 37.5 110.7
angle_coeff 15 harmonic 58.35 112.7
angle_coeff 16 harmonic 58.35 112.7
angle_coeff 17 harmonic 58.35 112.7
angle_coeff 18 harmonic 37.5 110.7
angle_coeff 19 harmonic 37.5 110.7
dihedral_coeff 1 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 2 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 3 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 4 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 5 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 6 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 7 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 8 opls 1.711 -0.5 0.663 0.0
dihedral_coeff 9 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 10 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 11 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 12 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 13 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 14 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 15 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 16 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 17 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 18 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 19 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 20 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 21 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 22 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 23 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 24 opls 1.3 -0.05 0.2 0.0
dihedral_coeff 25 opls 0.0 0.0 0.3 0.0
dihedral_coeff 26 opls 0.0 0.0 0.3 0.0
dihedral_coeff 27 opls 0.0 0.0 0.3 0.0
dihedral_coeff 28 opls 0.0 0.0 0.3 0.0
dihedral_coeff 29 opls 0.0 0.0 0.3 0.0
dihedral_coeff 30 opls 0.0 0.0 0.3 0.0
dihedral_coeff 31 opls 0.0 0.0 0.3 0.0
dihedral_coeff 32 opls 0.0 0.0 0.3 0.0
dihedral_coeff 33 opls 0.0 0.0 0.3 0.0
dihedral_coeff 34 opls 0.0 0.0 0.3 0.0
dihedral_coeff 35 opls 0.0 0.0 0.3 0.0
dihedral_coeff 36 opls 0.0 0.0 0.3 0.0
```

```
71      dihedral_coeff 37 opls 1.3 -0.05 0.2 0.0
72      dihedral_coeff 38 opls 1.3 -0.05 0.2 0.0
73      dihedral_coeff 39 opls 1.3 -0.05 0.2 0.0
74      dihedral_coeff 40 opls 0.0 0.0 0.3 0.0
75      dihedral_coeff 41 opls 0.0 0.0 0.3 0.0
76      dihedral_coeff 42 opls 0.0 0.0 0.3 0.0
77      dihedral_coeff 43 opls 0.0 0.0 0.3 0.0
78      dihedral_coeff 44 opls 0.0 0.0 0.3 0.0
79      dihedral_coeff 45 opls 0.0 0.0 0.3 0.0
80      dihedral_coeff 46 opls 0.0 0.0 0.3 0.0
```

APPENDIX C

POST–PROCESSING SCRIPTS

C.1 ljviscshear.py

```
from numpy import linspace, std
from pylab import *
from scipy import stats
import matplotlib.pyplot as plt
import os

# Utility Function ------------------------------------------------

def eformat(f, prec, exp_digits):
  s = "%.*e"%(prec, f)
  mantissa, expn = s.split('e')
  # add 1 to digits as 1 is taken by sign +/-
  return "%sE%+0*d"%(mantissa, exp_digits+1, int(expn))

# Calculate Shear Stress ------------------------------------------

logfiles = ['log.lammps.NEM.0.01.0.9',\
            'log.lammps.NEM.0.04.0.9',\
            'log.lammps.NEM.0.07.0.9',\
            'log.lammps.NEM.0.10.0.9',\
            'log.lammps.NEM.0.15.0.9',\
            'log.lammps.NEM.0.25.0.9',\
            'log.lammps.NEM.0.50.0.9',\
            'log.lammps.NEM.0.75.0.9',\
            'log.lammps.NEM.1.00.0.9',\
            'log.lammps.NEM.1.25.0.9',\
            'log.lammps.NEM.1.50.0.9',\
            'log.lammps.NEM.1.75.0.9',\
            'log.lammps.NEM.2.00.0.9',\
            'log.lammps.NEM.2.25.0.9',\
            'log.lammps.NEM.2.50.0.9',\
            'log.lammps.NEM.0.01.0.5',\
            'log.lammps.NEM.0.015.0.5',\
            'log.lammps.NEM.0.02.0.5',\
            'log.lammps.NEM.0.025.0.5',\
            'log.lammps.NEM.0.03.0.5',\
            'log.lammps.NEM.0.04.0.5',\
            'log.lammps.NEM.0.05.0.5',\
            'log.lammps.NEM.0.06.0.5',\
            'log.lammps.NEM.0.07.0.5',\
            'log.lammps.NEM.0.10.0.5',\
            'log.lammps.NEM.0.15.0.5',\
```

```
43              'log.lammps.NEM.0.25.0.5',\
44              'log.lammps.NEM.0.30.0.5',\
45              'log.lammps.NEM.0.35.0.5',\
46              'log.lammps.NEM.0.50.0.5',\
47              'log.lammps.NEM.0.75.0.5',\
48              'log.lammps.NEM.1.00.0.5',\
49              'log.lammps.NEM.1.25.0.5',\
50              'log.lammps.NEM.1.50.0.5',\
51              'log.lammps.NEM.1.75.0.5',\
52              'log.lammps.NEM.2.00.0.5',\
53              'log.lammps.NEM.2.25.0.5',\
54              'log.lammps.NEM.2.50.0.5']
55
56 nemdpxz = []
57 lz = []
58 z  = []
59 seed = [0,1,2]
60
61 for x in range(len(logfiles)):
62
63   logfile = logfiles[x]
64
65   if (x == 16) or (x == 18):
66     z.append(int(logfile[23:]))
67     head = logfile[:21]
68   else:
69     z.append(int(logfile[22:]))
70     head = logfile[:20]
71
72   temppxz = []
73
74   for k in range(len(seed)):
75
76     if os.path.isfile(head+str(seed[k])+'.'+str(z[x])):
77
78       myfile = head+str(seed[k])+'.'+str(z[x])
79
80       flux = []
81
82       with open(myfile, 'r') as f1:
83         data = f1.readlines()
84
85       for i in range(0, len(data)):
86         if 'Created orthogonal box' in data[i]:
87           temp = data[i].split()
88           lz.append(float(temp[-1][:-1]))
89         if 'Step PotEng Temp Pxz' in data[i]:
90           i = i + 1
91           while 'Loop' not in data[i]:
92             temp = data[i].split()
93             flux.append(float(temp[3])) # [kg/m-s]
94             i = i + 1
95
96     # Append the avg pxy to a temp
97     temppxz.append(sum(flux[len(flux)/2:])/len(flux[len(flux)/2:])) # [
       kg/m*s^2]
```

```
 98
 99   nemdpxz.append(temppxz) # [kg/m*s^2]
100
101 # Calculate Actual Shear Rate ------------------------------------
102
103 vxfiles = ['vxprofile.running.NEM.0.01.0.9',\
104            'vxprofile.running.NEM.0.04.0.9',\
105            'vxprofile.running.NEM.0.07.0.9',\
106            'vxprofile.running.NEM.0.10.0.9',\
107            'vxprofile.running.NEM.0.15.0.9',\
108            'vxprofile.running.NEM.0.25.0.9',\
109            'vxprofile.running.NEM.0.50.0.9',\
110            'vxprofile.running.NEM.0.75.0.9',\
111            'vxprofile.running.NEM.1.00.0.9',\
112            'vxprofile.running.NEM.1.25.0.9',\
113            'vxprofile.running.NEM.1.50.0.9',\
114            'vxprofile.running.NEM.1.75.0.9',\
115            'vxprofile.running.NEM.2.00.0.9',\
116            'vxprofile.running.NEM.2.25.0.9',\
117            'vxprofile.running.NEM.2.50.0.9',\
118            'vxprofile.running.NEM.0.01.0.5',\
119            'vxprofile.running.NEM.0.015.0.5',\
120            'vxprofile.running.NEM.0.02.0.5',\
121            'vxprofile.running.NEM.0.025.0.5',\
122            'vxprofile.running.NEM.0.03.0.5',\
123            'vxprofile.running.NEM.0.04.0.5',\
124            'vxprofile.running.NEM.0.05.0.5',\
125            'vxprofile.running.NEM.0.06.0.5',\
126            'vxprofile.running.NEM.0.07.0.5',\
127            'vxprofile.running.NEM.0.10.0.5',\
128            'vxprofile.running.NEM.0.15.0.5',\
129            'vxprofile.running.NEM.0.25.0.5',\
130            'vxprofile.running.NEM.0.30.0.5',\
131            'vxprofile.running.NEM.0.35.0.5',\
132            'vxprofile.running.NEM.0.50.0.5',\
133            'vxprofile.running.NEM.0.75.0.5',\
134            'vxprofile.running.NEM.1.00.0.5',\
135            'vxprofile.running.NEM.1.25.0.5',\
136            'vxprofile.running.NEM.1.50.0.5',\
137            'vxprofile.running.NEM.1.75.0.5',\
138            'vxprofile.running.NEM.2.00.0.5',\
139            'vxprofile.running.NEM.2.25.0.5',\
140            'vxprofile.running.NEM.2.50.0.5']
141
142 nemdshrrte = []
143 effshrrte = []
144
145 for x in range(len(vxfiles)):
146
147   myfile = vxfiles[x]
148   speed = float(myfile[22:26])
149   tempshrrte = []
150   tempeffshrrte = []
151
152   for k in range(len(seed)):
153
```

```
154     if x == 16 or x == 18:
155       head = myfile[:28]
156     else:
157       head = myfile[:27]
158
159     if os.path.isfile(head+str(seed[k])+'.'+str(z[x])):
160
161       vxfile = head+str(seed[k])+'.'+str(z[x])
162
163       with open(vxfile, 'r') as f:
164         data = f.readlines()
165
166       timestep = [10020000, 20020000, 40020000]
167       t10020000 = []
168       t20020000 = []
169       t40020000 = []
170       coord = []
171
172       if x == 0 or x == 23:
173         checktmstep = timestep[1]
174       elif x >= 15 and x <= 22:
175         checktmstep = timestep[2]
176       else:
177         checktmstep = timestep[0]
178
179       # Extract the last timestep's velocity
180       for i in range(0, len(data)):
181         if str(checktmstep) in data[i]:
182           temp = data[i].split()
183           numbins = int(temp[-1])
184           for j in range (1, numbins+1):
185             temp = data[i+j].split()
186             coord.append(float(temp[1]))
187             if x == 0 or x == 23:
188               t20020000.append(float(temp[3]))
189             elif x >= 15 and x <= 22:
190               t40020000.append(float(temp[3]))
191             else:
192               t10020000.append(float(temp[3]))
193
194       tslope = 0
195       bslope = 0
196       tr_value = 0
197       br_value = 0
198
199       # We're going to be using the last timestep, hence index = -1
200       if x == 0 or x == 23:
201         t = t20020000
202       elif x >= 15 and x <= 22:
203         t = t40020000
204       else:
205         t = t10020000
206
207       # Linear regression to fit a linear curve to our velocity profile
              data
208       if x < 15:
```

```
209         if x >= 9:
210           if x >= 12:
211             chop = 7
212             slope, yint, r_value, p_value, std_err = stats.linregress(
                    coord[chop:-2],  t[chop:-2])
213           elif x == 11:
214             chop = 6
215             slope, yint, r_value, p_value, std_err = stats.linregress(
                    coord[chop:-2],  t[chop:-2])
216           else:
217             chop = 5
218             slope, yint, r_value, p_value, std_err = stats.linregress(
                    coord[chop:-2],  t[chop:-2])
219         else:
220           chop = 2
221           slope, yint, r_value, p_value, std_err = stats.linregress(
                 coord[chop:-2],  t[chop:-2])
222       else:
223         if x >= 32:
224           if x >= 34:
225             if x >= 35:
226               chop = 5
227               slope, yint, r_value, p_value, std_err = stats.linregress
                     (coord[chop:],  t[chop:])
228             else:
229               chop = 4
230               slope, yint, r_value, p_value, std_err = stats.linregress
                     (coord[chop:-1],  t[chop:-1])
231           else:
232             chop = 3
233             slope, yint, r_value, p_value, std_err = stats.linregress(
                    coord[chop:-2],  t[chop:-2])
234         else:
235           chop = 2
236           slope, yint, r_value, p_value, std_err = stats.linregress(
                 coord[chop:-2],  t[chop:-2])
237
238       # Calculate the slope and resulting shear rate
239       tempshrrte.append(slope/lz[x]) # [1/s]
240
241       # Calculate the effective shear rate
242       tempeffshrrte.append(speed/lz[x])
243
244   # Append the list of shear rates
245   nemdshrrte.append(tempshrrte)
246
247   # Append the list of effective shear rates
248   effshrrte.append(tempeffshrrte)
249
250 # Calculate NEMD Viscosity ---------------------------------------
251
252 nemdviscty  = []
253 effviscty   = []
254 avgnemdvisc = []
255 avgeffvisc  = []
256 avgnemdshrt = []
```

```
avgeffshrrt = []
nemdetaerr = []

for i in range(len(nemdpxz)):

  tempnemdviscty  = []
  tempeffviscty   = []

  for j in range(len(nemdshrrte[i])):
    tempnemdviscty.append(-nemdpxz[i][j]/nemdshrrte[i][j])
    tempeffviscty.append(-nemdpxz[i][j]/effshrrte[i][j])
  nemdviscty.append(tempnemdviscty)
  effviscty.append(tempeffviscty)

  avgnemdvisc.append(sum(nemdviscty[i])/len(nemdviscty[i]))
  avgeffvisc.append(sum(effviscty[i])/len(effviscty[i]))
  avgnemdshrt.append(sum(nemdshrrte[i])/len(nemdshrrte[i]))
  avgeffshrrt.append(sum(effshrrte[i])/len(effshrrte[i]))
  nemdetaerr.append(std(nemdviscty[i]))

with open('nemdshearvisc.csv', 'w') as f:
  f.write('Shear Rate Viscosity\n')
  for i in range(len(avgnemdshrt)):
    f.write('%s,%s,%0.6f'%(eformat(avgnemdshrt[i],9,2),eformat(
        avgnemdvisc[i],9,2),nemdetaerr[i]))
    if i == 14:
      f.write('\n')
    f.write('\n')


# Create a Carreau SLLOD fit line (Eq 10 in Tenney Maginn 2010) ---

gdot = array([1.00e-03, 1.50e-03, 2.00e-03, 2.50e-03, 3.00e-03, \
              3.50e-03, 4.00e-03, 4.50e-03, 5.00e-03, 5.50e-03, \
              6.00e-03, 6.50e-03, 7.00e-03, 7.50e-03, 8.00e-03, \
              8.50e-03, 9.00e-03, 1.00e-02, 2.50e-02, 3.00e-02, \
              3.50e-02, 4.00e-02, 4.50e-2, 5.00e-02, 6.00e-02,  \
              7.00e-02, 8.00e-02, 9.00e-02, 1.00e-01, 1.00e-01, \
              1.50e-01, 2.00e-01, 2.50e-01, 3.00e-01, 4.00e-01, \
              5.00e-01, 6.00e-01, 7.00e-01, 8.00e-01, 9.00e-01, \
              1.00e00])

eta0 = 3.4
lamda = 7.1
p = 0.11

sllodfit = []
for i in range(len(gdot)):
  sllodfit.append(eta0/(1+(lamda*gdot[i])**2)**p)

carreaufit10 = []
eta0 = 3.2435
lamda = 2.06086
p = 1.0
for i in range(len(gdot)):
  carreaufit10.append(eta0/(1+(lamda*gdot[i])**2)**p)
```

```
312
313 carreaufit6 = []
314 eta0 = 8.66145
315 lamda = 115.44655
316 p = 0.1843
317 for i in range(len(gdot)):
318   carreaufit6.append(eta0/(1+(lamda*gdot[i])**2)**p)
319
320 # Plot Data ------------------------------------------------------
321
322 figure(figsize=(16,9))
323 plot(avgnemdshrt[:15], avgnemdvisc[:15], 'ro', markersize=10)
324 plot(avgnemdshrt[15:], avgnemdvisc[15:], 'bo', markersize=10)
325 plot(gdot, carreaufit10, '--r', markersize=10, linewidth=2)
326 plot(gdot, carreaufit6, '--b', markersize=10, linewidth=2)
327 text(avgnemdshrt[15]+0.005, avgnemdvisc[15]+1.0, '$\eta=\\frac
       {8.661}{(1+(115.447\dot{\gamma})^2)^{0.184}}$', fontsize=32, color
       ='b')
328 text(avgnemdshrt[3], avgnemdvisc[3]-0.75, '$\eta=\\frac
       {3.244}{(1+(2.061\dot{\gamma})^2)^{1.0}}$', fontsize=32, color='r')
329 xlabel('Shear Rate', fontsize=36)
330 ylabel('Viscosity', fontsize=36)
331 xscale('log')
332 yscale('log')
333 legend(['h = 9', 'h = 5', 'Carreau Fit for h = 9', 'Carreau Fit for h =
       5'], numpoints=1, fontsize=24,loc='upper right')
334 xlim([gdot[0],gdot[-1]])
335 ylim([1.0,15.0])
336 tick_params(labelsize=24, length=10, which='both')
337 savefig('nemdvisc_z5z9_carreau.png', bbox_inches='tight', dpi=100)
```

C.2 sqaviscshear.py

```
 1 from matplotlib import colors
 2 from numpy import linspace, std, average
 3 from pylab import *
 4 from scipy import stats
 5 import matplotlib
 6 import matplotlib.pyplot as plt
 7 import os
 8
 9 # Utility Function ----------------------------------------------
10
11 def eformat(f, prec, exp_digits):
12   s = "%.*e"%(prec, f)
13   mantissa, expn = s.split('e')
14   # add 1 to digits as 1 is taken by sign +/-
15   return "%sE%+0*d"%(mantissa, exp_digits+1, int(expn))
16
17 # Begin Post-Processing Code -------------------------------------
18
19 logfile = [ 'log.lammps.shearwalls.128.0.000001.0',\
20                         'log.lammps.shearwalls.128.0.000002.0',\
21                         'log.lammps.shearwalls.128.0.000004.0',\
22                         'log.lammps.shearwalls.128.0.000006.0',\
23                         'log.lammps.shearwalls.128.0.000008.0',\
24                         'log.lammps.shearwalls.128.0.000010.0',\
```

```
                        'log.lammps.shearwalls.128.0.000020.0',\
                        'log.lammps.shearwalls.128.0.000050.0',\
                        'log.lammps.shearwalls.128.0.000100.0',\
                        'log.lammps.shearwalls.128.0.000200.0',\
                        'log.lammps.shearwalls.128.0.000300.0',\
                        'log.lammps.shearwalls.200.0.000002.0',\
                        'log.lammps.shearwalls.200.0.000004.0',\
                        'log.lammps.shearwalls.200.0.000006.0',\
                        'log.lammps.shearwalls.200.0.000008.0',\
                        'log.lammps.shearwalls.200.0.000010.0',\
                        'log.lammps.shearwalls.200.0.000020.0',\
                        'log.lammps.shearwalls.200.0.000050.0',\
                        'log.lammps.shearwalls.200.0.000100.0',\
                        'log.lammps.shearwalls.200.0.000200.0',\
                        'log.lammps.shearwalls.200.0.000300.0',\
                        'log.lammps.shearwalls.200.0.000400.0',\
                        'log.lammps.shearwalls.200.0.000500.0',\
                        'log.lammps.shearwalls.200.0.001000.0',\
                        'log.lammps.shearwalls.200.0.001500.0',\
                        'log.lammps.shearwalls.200.0.002000.0',\
                        'log.lammps.shearwalls.200.0.003000.0',\
                        'log.lammps.shearwalls.338.0.000010.0',\
                        'log.lammps.shearwalls.338.0.000020.0',\
                        'log.lammps.shearwalls.338.0.000050.0',\
                        'log.lammps.shearwalls.338.0.000100.0',\
                        'log.lammps.shearwalls.338.0.000200.0',\
                        'log.lammps.shearwalls.338.0.000300.0',\
                        'log.lammps.shearwalls.338.0.000400.0',\
                        'log.lammps.shearwalls.338.0.000500.0',\
                        'log.lammps.shearwalls.338.0.001000.0',\
                        'log.lammps.shearwalls.338.0.001500.0',\
                        'log.lammps.shearwalls.338.0.002000.0',\
                        'log.lammps.shearwalls.338.0.003000.0',\
]

speed = []
effectiveshearrate = []
p = []            # All the pxy data
pxys = [[],[],[]] # The averaged pxy values that we want
seed = [0,1,2]
mytimecheck = []
lx = [[],[],[]]

# Calculate pxy ------------------------------------------------

for x in range(len(logfile)):

  if int(logfile[x][22:25]) == 128:
    mol = 0
  elif int(logfile[x][22:25]) == 200:
    mol = 1
  elif int(logfile[x][22:25]) == 338:
    mol = 2

  pxy = []

```

```
  for m in range(len(seed)):

    pathname = logfile[x][:-1]+str(seed[m])

    if logfile[x][26:-2] == '0.003000':
      stringmatch = 'run             10000000'
        else:
          stringmatch = 'run             20000000'

    if os.path.isfile(pathname):

      p = []
      templx = []
      mytimecheck = []

      with open(pathname, 'r') as f:
        data = f.readlines()

      for i in range(0, len(data)):
        if stringmatch in data[i]:
          i = i + 3
        while 'Loop' not in data[i]:
          temp = data[i].split()
          mytimecheck.append(int(temp[0])*0.000001) # fs --> ns
          p.append(float(temp[2])*101325*1000) # atm --> mPa
          templx.append(float(temp[3])) # Ang
          i = i + 1

      pxy.append(average(p[len(p)/2:])) # Average last 1/2 of pxy data
      lx[mol].append(average(templx))

  # Append temporary pxy to the main pxys list
  pxys[mol].append(pxy) # [mPa]

# Calculate Shear Rate ------------------------------------------

vyfile = ['vyprofile.128.0.000001.running.0',\
          'vyprofile.128.0.000002.running.0',\
          'vyprofile.128.0.000004.running.0',\
          'vyprofile.128.0.000006.running.0',\
          'vyprofile.128.0.000008.running.0',\
          'vyprofile.128.0.000010.running.0',\
          'vyprofile.128.0.000020.running.0',\
          'vyprofile.128.0.000050.running.0',\
          'vyprofile.128.0.000100.running.0',\
          'vyprofile.128.0.000200.running.0',\
          'vyprofile.128.0.000300.running.0',\
          'vyprofile.200.0.000002.running.0',\
          'vyprofile.200.0.000004.running.0',\
          'vyprofile.200.0.000006.running.0',\
          'vyprofile.200.0.000008.running.0',\
          'vyprofile.200.0.000010.running.0',\
          'vyprofile.200.0.000020.running.0',\
          'vyprofile.200.0.000050.running.0',\
          'vyprofile.200.0.000100.running.0',\
          'vyprofile.200.0.000200.running.0',\
```

```
137           'vyprofile.200.0.000300.running.0',\
138           'vyprofile.200.0.000400.running.0',\
139           'vyprofile.200.0.000500.running.0',\
140           'vyprofile.200.0.001000.running.0',\
141           'vyprofile.200.0.001500.running.0',\
142           'vyprofile.200.0.002000.running.0',\
143           'vyprofile.200.0.003000.running.0',\
144           'vyprofile.338.0.000010.running.0',\
145           'vyprofile.338.0.000020.running.0',\
146           'vyprofile.338.0.000050.running.0',\
147           'vyprofile.338.0.000100.running.0',\
148           'vyprofile.338.0.000200.running.0',\
149           'vyprofile.338.0.000300.running.0',\
150           'vyprofile.338.0.000400.running.0',\
151           'vyprofile.338.0.000500.running.0',\
152           'vyprofile.338.0.001000.running.0',\
153           'vyprofile.338.0.001500.running.0',\
154           'vyprofile.338.0.002000.running.0',\
155           'vyprofile.338.0.003000.running.0',\
156 ]
157
158 r2s = [[],[],[]]
159 shearrates = [[],[],[]]
160 effshearrates = [[],[],[]]
161
162 for x in range(len(vyfile)):
163
164   if int(vyfile[x][10:13]) == 128:
165     mol = 0
166   elif int(vyfile[x][10:13]) == 200:
167     mol = 1
168   elif int(vyfile[x][10:13]) == 338:
169     mol = 2
170
171   effshearrate = []
172   shearrate = []
173   r2 = []
174
175   for m in range(len(seed)):
176
177     pathname = vyfile[x][:-1]+str(seed[m])
178
179     if float(vyfile[x][14:22]) == 0.003000:
180       timestep = linspace(3100000, 13000000, 100)
181       timestep_nano = linspace(3.1, 13.0, 100)
182     else:
183       timestep = linspace(3100000, 23000000, 200)
184       timestep_nano = linspace(3.1, 23.0, 200)
185
186     if os.path.isfile(pathname):
187
188       speed = float(vyfile[x][14:22]) # [Ang/fs]
189       effshearrate.append(speed/lx[mol][seed[m]]*1.0e15) # [1/s]
190
191       with open(pathname, 'r') as f:
192         contents = f.readlines()
```

```
193
194       data = []
195
196       for i in range(len(contents)):
197         if '#' not in contents[i]:
198           data.append(contents[i])
199
200       t = []
201       j = 0
202       for i in range(0, len(data), 21):
203         if (j < len(timestep)):
204           try:
205             if (timestep[j] == timestep[-1]):
206               t.append([])
207               for k in range (1, 21):
208                 temp = data[i+k].split()
209                 if float(temp[2]) > 349:
210                   t[-1].append(float(temp[3]))
211                 else:
212                   j = j + 1
213           except IndexError:
214             print 'Sorry, index exception.'
215
216       if len(t[-1]) == 10:
217         coord = linspace(0.275, 0.725, 10)
218       elif len(t[-1]) == 12:
219         coord = linspace(0.225, 0.775, 12)
220       elif len(t[-1]) == 14:
221         coord = linspace(0.175, 0.825, 14)
222       elif len(t[-1]) == 16:
223         coord = linspace(0.125, 0.875, 16)
224       elif len(t[-1]) == 18:
225         coord = linspace(0.075, 0.925, 18)
226       else:
227         coord = linspace(0.025, 0.975, 20)
228
229       slope, intercept, r_value, p_value, std_err = stats.linregress(
              coord, t[-1])
230
231       r2.append(r_value**2)
232       shearrate.append(slope/lx[mol][seed[m]]*1.0e15) # Units: [1/fs]
              --> [1/s]
233
234   # Append the average shear rates and corresponding R^2 values
235   r2s[mol].append(r2)
236   shearrates[mol].append(shearrate)
237   effshearrates[mol].append(effshearrate)
238
239 # Calculate Actual and Effective Viscosity ------------------------
240
241 viscositys = []
242 visc_error = []
243 for i in range(len(pxys)):
244   viscositys.append([])
245   visc_error.append([])
246   for j in range(len(pxys[i])):
```

```
247     viscosity = []
248     for k in range(len(pxys[i][j])):
249       viscosity.append(-pxys[i][j][k]/shearrates[i][j][k])
250       viscositys[i].append(average(viscosity))
251       visc_error[i].append(std(viscosity))
252
253 # Average the shear rates
254 avgshearrates = []
255 for i in range(len(shearrates)):
256   avgshearrates.append([])
257   for j in range(len(shearrates[i])):
258     avgshearrates[i].append(average(shearrates[i][j]))
259
260 # Calculate effective viscosity
261 effviscositys = []
262 effvisc_error = []
263 for i in range(len(pxys)):
264   effviscositys.append([])
265   effvisc_error.append([])
266   for j in range(len(pxys[i])):
267     effviscosity = []
268     for k in range(len(pxys[i][j])):
269       effviscosity.append(-pxys[i][j][k]/effshearrates[i][j][k])
270     effviscositys[i].append(average(effviscosity))
271     effvisc_error[i].append(std(effviscosity))
272
273 # Average the effective shear rates
274 avgeffshearrates = []
275 for i in range(len(effshearrates)):
276   avgeffshearrates.append([])
277   for j in range(len(effshearrates[i])):
278     avgeffshearrates[i].append(average(effshearrates[i][j]))
279
280 # Output Data in .csv Files ---------------------------------------
281
282 # Output all effective viscosity shear rate data
283 outfile = open('output_eff.csv','w')
284 outfile.write('Shrrte[1/s],Visc0[mPa-s],Visc1[mPa-s],Visc2[mPa-s]\n\n')
285 for i in range(len(pxys)):
286   for j in range(len(pxys[i])):
287     outfile.write('%s, ' % eformat(average(effshearrates[i][j]),9,2))
288     for k in range(len(pxys[i][j])):
289       outfile.write('%.3f, ' % (-pxys[i][j][k]/effshearrates[i][j][k]))
290     outfile.write('\n')
291   outfile.write('\n')
292 outfile.write('EffShrrte[1/s],EffAvgVisc[mPa-s],EffViscErr[mPa-s]\n\n')
293 for i in range(len(avgeffshearrates)):
294   for j in range(len(avgeffshearrates[i])):
295     outfile.write('%s, %.3f, %.3f\n' % (eformat(avgeffshearrates[i][j
          ],9,2),effviscositys[i][j],effvisc_error[i][j]))
296   outfile.write('\n')
297 outfile.close()
298
299 # Output all actual viscosity shear rate data
300 outfile = open('output_act.csv','w')
301 outfile.write('Shrrte[1/s],Visc0[mPa-s],Visc1[mPa-s],Visc2[mPa-s]\n\n')
```

```
302 for i in range(len(pxys)):
303   for j in range(len(pxys[i])):
304     outfile.write('%s, ' % eformat(average(shearrates[i][j]),9,2))
305     for k in range(len(pxys[i][j])):
306       outfile.write('%.3f, ' % (-pxys[i][j][k]/shearrates[i][j][k]))
307     outfile.write('\n')
308   outfile.write('\n')
309 outfile.write('Shrrte[1/s],AvgVisc[mPa-s],ViscErr[mPa-s]\n\n')
310 for i in range(len(avgshearrates)):
311   for j in range(len(avgshearrates[i])):
312     outfile.write('%s, %.3f, %.3f\n' % (eformat(avgshearrates[i][j
          ],9,2),viscositys[i][j],visc_error[i][j]))
313   outfile.write('\n')
314 outfile.close()
315
316 # Create Literature Ambient Viscosity Line -----------------------
317
318 gdot = array([1.00e08, 1.50e08, 2.00e08, 3.00e08, 4.00e08, \
319               5.00e08, 6.00e08, 7.00e08, 8.00e08, 9.00e08, \
320               1.00e09, 1.50e09, 2.00e09, 3.00e09, 4.00e09, \
321               5.00e09, 6.00e09, 7.00e09, 8.00e09, 9.00e09, \
322               1.00e10, 1.50e10, 2.00e10, 3.00e10, 4.00e10, \
323               5.00e10, 6.00e10, 7.00e10, 8.00e10, 9.00e10, \
324               1.00e11, 5.00e11, 1.00e12])
325
326 ambient = []
327 for i in range(len(gdot)):
328         ambient.append(27.6) # Ref Liu et al 2006
329
330 # Plots ----------------------------------------------------------
331
332 # Plot actual viscosity vs actual shear rate
333 figure(figsize=(16,9))
334 errorbar(avgshearrates[0], viscositys[0], yerr=visc_error[0], \
335          fmt='ko', mfc='m', markersize=10, linewidth=3)
336 errorbar(avgshearrates[1], viscositys[1], yerr=visc_error[1], \
337          fmt='ko', mfc='c', markersize=10, linewidth=3)
338 errorbar(avgshearrates[2], viscositys[2], yerr=visc_error[2], \
339          fmt='ko', mfc='b', markersize=10, linewidth=3)
340 loglog(gdot, ambient, '--r', markersize=10, linewidth=3)
341 xlabel('Shear Rate [1/s]', fontsize=36)
342 ylabel('Viscosity [mPa$\cdot$s]', fontsize=36)
343 legend(['128 Molecules, 8.3$\AA$', '200 Molecules, 12.9$\AA$', \
344         '338 Molecules, 22.5$\AA$', 'Bulk $\eta_0$'], \
345         numpoints=1, fontsize=24,loc='upper right')
346 tick_params(labelsize=24)
347 plt.minorticks_on()
348 savefig('viscshear.png', bbox_inches='tight', dpi=100)
349
350 # Plot effective viscosity vs effective shear rate
351 figure(figsize=(16,9))
352 errorbar(avgeffshearrates[0], effviscositys[0], yerr=effvisc_error[0],\
353          fmt='ko', mfc='m', markersize=10, linewidth=3)
354 errorbar(avgeffshearrates[1], effviscositys[1], yerr=effvisc_error[1],\
355          fmt='ko', mfc='c', markersize=10, linewidth=3)
356 errorbar(avgeffshearrates[2], effviscositys[2], yerr=effvisc_error[2],\
```

```
357          fmt='ko', mfc='b', markersize=10, linewidth=3)
358 loglog(gdot, ambient, '--r', markersize=10, linewidth=3)
359 xlabel('Effective Shear Rate [1/s]', fontsize=36)
360 ylabel('Effective Viscosity [mPa$\cdot$s]', fontsize=36)
361 legend(['128 Molecules, 8.3$\AA$', '200 Molecules, 12.9$\AA$', \
362         '338 Molecules, 22.5$\AA$', 'Bulk $\eta_0$'], \
363         numpoints=1, fontsize=24,loc='upper right')
364 tick_params(labelsize=24)
365 rc('font', size=24) # Scientific notation font matches tick_params font
366 plt.minorticks_on()
367 savefig('effviscеffshear.png', bbox_inches='tight', dpi=100)
368
369 show()
```

C.3 plotEHLph.py

```
 1 from pylab import *
 2
 3 # Utility Function ------------------------------------------------
 4
 5 def eformat(f, prec, exp_digits):
 6   s = "%.*e"%(prec, f)
 7   mantissa, expn = s.split('e')
 8   # add 1 to digits as 1 is taken by sign +/-
 9   return "%sE%+0*d"%(mantissa, exp_digits+1, int(expn))
10
11 # -----------------------------------------------------------------
12
13 with open('consol_A205_t1.5.txt', 'r') as f:
14   data = f.readlines()
15
16 error_p1 = []
17 timestep1 = []
18 hcen1 = []
19
20 for i in range(0, len(data)):
21   if 'Error_p' in data[i]:
22     temp = data[i].split()
23     error_p1.append(float(temp[5]))
24     timestep1.append(int(temp[2]))
25   if 'Hcen[nm]' in data[i]:
26     temp = data[i].split()
27     hcen1.append(float(temp[2]))
28
29 # Plot error_p (Delta p)
30 figure(figsize=(16,9))
31 plot(timestep1, error_p1, '-co', markersize=6, linewidth=2)
32 xlabel('Timestep', fontsize=36)
33 ylabel('$\Delta p$', fontsize=36)
34 title('$A = 205, \; t = 1.5$', fontsize=36)
35 tick_params(labelsize=24)
36 xlim([timestep1[0], timestep1[-1]])
37 ylim([1e-6, 1e-3])
38 yscale('log')
39 plt.minorticks_on()
40
41 # Plot h_cen
```

```
figure(figsize=(16,9))
plot(timestep1, hcen1, '-bo', markersize=6, linewidth=2)
xlabel('Timestep', fontsize=36)
ylabel('H$_{cen}$ [nm]', fontsize=36)
title('$A = 205, \; t = 1.5$', fontsize=36)
tick_params(labelsize=24)
xlim([timestep1[0], timestep1[-1]])
plt.minorticks_on()

# Output data to .csv file
outputfile = open('A205t1..5csv','w')
outputfile.write('Timestep, p_error, hcen\n\n')
for i in range(len(timestep1)):
  outputfile.write('%i, %s, %.3f\n' % (timestep1[i], eformat(error_p1[i
      ],3,2), hcen1[i]))

show()
```