Predicting static friction in a molecular dynamic system using machine learning

Interjunctional Asperity Relations

Anders Bråte

Thesis submitted for the degree of
Master in Computational Science: Physics
60 credits

Department of Physics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2022

Predicting static friction in a molecular dynamic system using machine learning

Interjunctional Asperity Relations

Anders Bråte

Predicting static friction in a molecular dynamic system using machine learning

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Friction is immensely important, whether the concern is moving or still objects, energy loss, heat transfer or wear and tear. Despite its obvious importance friction is not fully understood, due to small scale behaviours both effecting and being effected by large scale behaviours, making a scalable friction theory difficult to define. In this thesis we wish to study these small scale interactions at the junctions where objects meet, and how the placement of junctions relative to each other can tell us about their interactions. We study a molecular dynamical system of 3c-SiC crystals in the form of a rectangular block with dodecahedron asperities underneath , the shape of which is picked due to it being close to the equilibrium shape of SiC nanoparticles. This top plate meets another stationary bulk crystal plate on which it slides with the asperities forming the junctions. The choice of system is made as to reduce any effects that might conceal the effects of junction-junction interactions. Preliminary studies are done varying physical aspects of the system such as the sizes of various parts of the system, normal force, velocity of the top plate in the push phase etc, to find a physically and computationally feasible system before proceeding with large scale simulations varying the placement of the aforementioned asperities. The placement of the asperities form the input into a neural network predicting the maximum static friction and the slope of the load curves as experienced by the top plate. We found that the system we studied did not show an effect of the asperity configuration on the friction force. A large grid search over hyperparameters of the networks was done to thoroughly examine any network which might be able to learn from the simulations data. These methods were also applied to a dataset of random value to create a baseline to compare our actual data to. The networks were able to confirm the insensitivity of the friction to variations in configurations.

Acknowledgements

After a year of working on this thesis I am left with a fleeting sense of a year which has passed quickly, yet which is full of events and memories. I am very grateful to the whole community at the Center for Computing in Science Education, many of whom I have become close friends with. I also want to thank all my friends from the bachelors programme at the University of Oslo. Without you I would never get here.

I would also like to thank my supervisors, who have in a great deal contributed to this thesis, and whom i have a lot of respect for. Even Nordhagen whom I have worked very close with throughout the year, and who's work I in a sense inherited once i started my thesis has been of great help. I would not have gotten as far as I did without your help. Henrik Andersen Sveinsson has from the beginning steered my work and guided my progress in ways I could never have done without his help. I also have to thank Anders Malthe-Sørenssen who provided valuable feedback.

A special mention to Halvard Sutterud who had the pleasure of teaching me c++ in my younger days. My makefiles still compile as 'halvard.c' in your honor.

Finally i must thank my girlfriend and partner Stine Langen Wiik. I cannot thank you enough. Hopefully I will succeed in being at least half as helpful to you when its your turn to finish a masters, as you were to me during mine.

Contents

Chapter 1

Introduction

Friction is a term for forces which resist relative motion. It is very important in anything involving moving parts, stationary parts, wear and similar. Friction as a term encompasses a number of physical effects and attributes, yet is itself a part of the larger field, namely tribology, the science of surfaces in relative motion. This field spans across many disciplines and sciences and is as expansive in size as it is important. Despite its importance, and the fact that research into friction dates back hundreds of years [13] to the time of Leonardo da Vinci, it is not yet fully understood. The way junctions interact affects the macroscopic movements and vice versa meaning models rarely scale well. This means that we to this day do not have a bottom up multi-scale friction theory which starts at a microscopic junction level and explains the macroscopic movements. Therefore there is still knowledge to be gained from studying this field, which is exactly what we will attempt in this thesis.

Early work by the likes of da Vinci and Charles-Augustin de Coulomb [9] around the 17th century found friction to be independent on the size of the object sliding, but rather the normal force, the static and dynamic friction coefficient and the direction of travel. This is the type of friction many is still taught early in school today. Though a good approximation in many setting, reality is not as simple. In fact, Coulomb himself found later some of what we today refer to as *rate-and-state* friction [10]. This states that the friction of an object is dependent on a number of things, including how long the two objects have been in contact (ageing), how fast the objects moves relative to each other once sliding and the state of the surfaces. Important work in realizing this was done by Andy Ruina in 1983 [30].

This is a good empirical description for many applications, however later experimental work found that yet again, the reality is not as simple as that. The Fineberg group [5, 29] found through experiments that the often used friction coefficient is not a material constant, and varies largely based on so called rupture conditions [4]. There are junctions at which materials meet and form covalent bonds, which do not break uniformly and instead behaves in rather complex ways. Slow and fast rupture fronts where bonds break are discovered, and a more dynamic interaction between bonds and junctions in the material is theorized. This lack of insight into the behaviour of these junctions is what

prompts us to further delve into the topic.

We use molecular dynamic simulations, which provides an insight into the behaviour of single atoms. This provides a more controlled environment and more information than previous studies which focus on the scalability of the models by approximating the junctions between materials in a much simpler way [40]. Thought there is an argument for developing simple models which scale well, there can still be macroscopic effects discovered using microscopic methods [16]. The interesting effects of ageing, faceting, diffusion and similar are made more so by our use of SiC, which has a high reliance on the shape of the surface and surface energy. An example is the shape of a sphere, which as it relaxes undergoes faceting, forming a dodecahedron [37]. This effect is faster and more prevalent in SiC crystals. This is also useful as we wish to study the effects of junctions, and not effects like asperity deformation due to normal force ageing and similar. If we introduce shapes which already exist at their lowest energy level we can reduce effects like faceting and deformation, leading to less noise in our data.

The system we will study is that of a rectangular slab, with asperities in the shape of the aforementioned dodecahedra underneath. This moving top plate will be placed on a still rectangular block. In keeping other variables such as relax time for ageing, normal force and other physical properties of the system constant we can reduce the problem to one where only the placement of asperities affects the results. Thus we can attempt to study only the interjunctional interactions, which can possibly be used in another more complex system later. The first part of doing this involves finding a system which we deem realistic, yet which has as few effects as possible originating from topics out of the scope of this thesis. We do this by performing simulations varying these variables to find the ones we deem to be best. The main simulations will be those where we vary the placements of asperities on the top plate. We start by varying the position of two asperities as this is a simpler system where we can get an overview over how all systems perform. Then we can proceed with a larger eight asperity system.

An eight asperity system provides a very large number of unique configurations, but this provides an opportunity to apply machine learning. This is done using a dense neural network, and a convolutional neural network on a large dataset of simulations. Applying machine learning to get a better understanding of our physical system is a method which has proven itself to be useful for certain applications [17].

The question to be answered is essentially, how does the relative placement of junctions between objects affect the static friction and stiffness of the interaction when other factors are kept constant.

1.1 Goals

The goals we wish to achieve through this thesis is listed below:

- Make a system in which we can vary the placements of asperities, and in which effects from ageing and other factors are minimized.

- Develop methods of describing and analyzing the system, with which we can compare our findings with that of previous findings.
- Make a pipeline for simulating the systems, and study the ways in which placement of asperities affects the behaviour of the system
- Explore ways in which the results of these simulations can be used to gain useful insight into other similar systems by using machine learning.

1.2 Contributions

With the listed goals in mind, we succeed in the following ways:

- Made a pipeline for finding and making systems to our liking, with asperities placed in any configuration. The scripts also allows for any system size, and various crystal orientations.
- Performed simulations and measured the various static friction forces using techniques such as the sigmoid curve fit to find the maximum static friction.
- Employing machine learning methods with both dense neural networks and convolutional neural networks in robust grid searches.
- Developed a way to confirm the insensitivity of the system with respect to variations in asperity configuration by attempting to train neural networks with a random dataset as a baseline.

1.3 Thesis Structure

Starting at the beginning in part I, Chapter 2 summarizes some of the ways friction has been described mathematically, which we later use to compare with our simulation results.

In chapter 3 we describe and state the necessary background for our molecular dynamical simulations. These include, but are not limited to, potentials and forces which describe the interaction between the particles, the way in which we integrate the movements and some ways in which we make the system more realistic keeping in mind computational limitations. Chapter 4 is a brief introduction to the machine learning methods which we employ after simulations. The brief theory behind an artificial neural network, which leads to a description of a dense neural network and finally convolutional neural networks.

Part II describes the way we implement the methods for finding and simulating our system. Chapter 5 introduces the very important packages used in this thesis. With all our theory now behind us we describe how we design our final system, as well as methods for describing the resulting simulations in chapter 6.

Part III consists of plots and results from preliminary simulations whilst designing our system, before handling the two asperity case and finally the eight asperity case using our final system parameters. The simulations of the eight asperity case is then analyzed using various varieties of machine learning.

Part I

Theory

Chapter 2

Friction

The inherent complex and multi-scale properties of friction makes it a poorly understood field, despite being of immense importance in all aspects of our daily lives. Microscopic effects at the atomic level both affects and is affected by large scale behaviours which means making a theory which scales up from micro to macro very hard.

Some of the earliest work on friction came as far back as in 1699 and 1801 when Amontons and later Coulomb penned a single degree of freedom friction law known commonly as the Amontons-Coulomb laws [3],

$$\begin{aligned} f \leq & \mu_s F_n & , v = 0 \\ f = & - sign(v)\mu_d F_N & , v \neq 0. \end{aligned} \tag{2.1}$$

These relate the friction f to the normal force F_n with a threshold for static friction in which the objects begins to slide and the friction coefficient μ, all very intuitive. Perhaps not so intuitive however is the fact that friction according to this equation does not depend on the surface or indeed the shape of the object what so ever. This, along with the infinite sharpness of the transition between dynamic and static friction, ageing effects and the dynamic friction varying with velocity which are known effects that is not considered in this simple friction law, all opens up for further studies within the field of friction. Coulomb did actually discover the effects of ageing and the variations of dynamic friction with velocity in a 1821 paper [10] however equation 2.1 was the one that stuck.

2.1 Mohr-Coulomb

A similar equation penned in 1777 by Mohr also relates friction and normal force, however an important discovery was made in the edge case of zero normal force.

$$\mu_s = F_n \tan(\phi) + c, \tag{2.2}$$

This states that the shear strength μ, is equal the normal force F_n multiplied with the so called slope of the failure envelope, where ϕ is the angle of internal friction and a

constant denoted by c. The internal angle and failure envelope is much like the original Coulomb equation far too simplistic, however the constant c tells us that even without normal force we still have friction due to the aptly named cohesion variable.

2.2 Rate-and-State Friction

The effects of the *rate*, i.e. the sliding velocity of the centre of mass, and the *state*, which can to some degree be interpreted as the time spent in the static position, have turned out to be essential in the understanding of friction which was left out of equation 2.1. The contact area in which two surfaces meet, which according to Amontons and other at the time does not affect the friction force does indeed have an effect, which interestingly also ties into the time the two objects spend in contact. James H. Dieterich made important progress in the field of rate and state friction, which later was improved upon by Andy Ruina in his 1983 paper [30] in which he presented the following equation,

$$\mu = \mu_0 a \ln\left(1 + \frac{1}{v_0}\right) + b \ln\left(\frac{v_0 \theta(t)}{d_0}\right). \tag{2.3}$$

Here μ_0 is the initial static friction coefficient, v_0 is the reference velocity, d_0 is the characteristic decay length, $\theta(t)$ is the time dependent state variable and a and b are coefficients found empirically.

The essence of this equation is that at any point the surface in between two objects always has a state θ, which changes only dependent on the normal stress, the slip rate and the state itself. If this explanation seems vague, it is because the theory stated by Ruina is itself general, and can be viewed as perhaps more of a framework to encompass many different friction laws.

Much has been said, and a lot of work has been done to arrive at a description of friction which mostly describe the behaviours of friction well, however we are not yet at the point where a complete and full description of the phenomena exists. This we know thanks to experimental work by groups such as the Fineberg group [5, 29] who found that reality is yet even more complex. The discovery is that of slow and fast rupture fronts. When applying force to an object with the intention of sliding it the movement depends on the behaviours of the junctions at which the objects meet. These do not break simultaneously, and instead act more dynamically, sharing the load and interacting. The result is section of junctions breaking at various speeds and times, and in general a quite complex behaviour known as rupture fronts.

In fact, the Fineberg group published a paper in 2011 [4] titled 'Static Friction Coefficient Is Note a Material Constant', which proves just what the title says. In the paper they argue that large variations can be found for the static friction coefficient, dependent on the rupture dynamics of the junctions separating two surfaces.

2.3 Ageing

An effect discovered a lot earlier than rupture fronts and similar is that of ageing, which Coulomb as mentioned observed as far back as 1821. On an intuitive level, it might seem strange that objects will 'grow' more attached over time, and though it might seem as a minute detail the effects of ageing can increase the coefficient of friction 10-20% [5]. There are two ways that surfaces grow more attached. One is through the increase in covalent bonds between the surfaces as they are in contact[26, 27], and the other is the increase in contact area between the surfaces [12]. The increase in total surface area in the junction between two surfaces is due to two effects; diffusion and creep. Diffusion is the effect of free atoms wandering around on a rough surface and settling in the junction of two surfaces in an attempt to reduce its energy level. This increases the total contact area, and the atoms do to some degree help in sharing the load once the surfaces experience forces. Creep, which we also refer to as normal force ageing, is the effect of asperities, which form the junctions between surfaces to compress outwards as it is under a normal force. This also increases the contact area if the material in question is relatively incompressible, and thus must expand outwards upon experiencing force from above. Both of these effects which increase the contact area also increases the number of covalent bonds between the surfaces.

2.4 Thermal Activation

Previous work by by many research groups have shown a dependence on a velocity in friction, especially looking at single asperity in the form of a friction force microscope [6]. The energy level of a given bond oscillates a little bit all the time, and dependent on the conditions, the energy level it needs to reach in order to break the bond will every now and then be fulfilled. In the case that the bonds between atoms are strained, as they are when say an asperity attempts to slide across a surface, the energy level of the atoms are a bit closer to reaching the breaking point. The aforementioned oscillation of the energy levels in the atoms are relatviley periodic, and do not under very short timeframes depend on the velocity of the top plate. This mean that if the velocity is greater, then the system will make it a larger distance before a random oscilation causes a bond to break and for the rupture front to form. This means that suddenly a system will be able to produce a higher static friction before the system has time to break. This velocity dependent statc friction is modeled by Sang et. al. (2001) [33] in the following equation.

$$F \propto c - T^{2/3} |\ln v/T|^{2/3}. \tag{2.4}$$

Here, friction force F is related to a constant c, a constant temperature T and the velocity v.

2.5 Diffusion Creep And Healing

In crystals such as SiC which is very dependent on surface energy there are a lot of interesting effects that take place. One such example relating to our choice in asperity shape is that of faceting. The relaxation of a sphere takes on the shape of a dodecahedron as this shape is that which lowers the surface energy and thus the energy of the sphere [37]. Another important effect is that of healing. Say a portion of a relaxed surface somehow gets the energy to come loose and go elsewhere, then this leaves a portion of the surface in which atoms can reduce their energy. This gap is filled in and 'healed'. Somewhat related is the effect called Diffusion creep, or just creep. This effect is when a surface moves relative to another, but instead of a slip-stick behaviour diffusion causes 'free' atoms to catch the load and cause the surface to creep rather than slip. These effects are more commonly described in larger systems such as geological studies [2].

Chapter 3

Molecular Dynamics

Classical molecular dynamics (MD) is a numerical method for the purpose of studying the interactions of atoms forming larger materials. The atoms interact using a variety of potentials depending on the system, and the movement is integrated according to Newton's laws using a numerical integration scheme. The small scale interactions lead to systems that replicate many physical properties, and MD has become a common method of modelling, especially as computing power increases.
In this chapter we will briefly introduce the theoretical groundwork, and then explain the practical applications.

3.1 Potentials and Forces

Looking at a quantum mechanical system, though it is the most accurate description of the interaction between particles, it is also extremely computationally demanding. Therefore, instead of handling the full wave function using Schrödinger, we assume an interaction potential that only depends on the relative position of point particle representation of the atoms. The potential is written as the following,

$$U(\{\vec{r}_i\}), \tag{3.1}$$

Where $\{r_i\}$ is a set of particle coordinates with i denoting the particle of the index. The gradient of the potential gives us the force denoted as

$$\vec{F}_j = -\nabla U(\vec{r}_{i,j}). \tag{3.2}$$

Here, $r_{i,j}$ is the distance between two particles i and j. From here we can easily find the acceleration at time j using Newton's equations,

$$\vec{a}_i = \frac{1}{m_i} \sum_j \nabla U_{i,j}, \tag{3.3}$$

where m_i is the mass of particle i.

When looking at potentials and the functions describing them a common, and perhaps the most widely used interaction potential is the Lennard-Jones potential[23].

$$V(r_{i,j}) = 4\epsilon \left[(\frac{\sigma}{r_{i.j}})^{12} - (\frac{\sigma}{r_{i,j}})^{6} \right] \tag{3.4}$$

This equation developed in the 1920s by a John Lennard-Jones does a good job describing the two main potentials in inter atomic interactions of nonionic particles.

This function of the distance between particle i and j combines van der Waals equation and the Pauli repulsion equation. The parameter σ known as the characteristic length effectively describes the distance at which the potential is zero, and the parameter ϵ known as the characteristic energy acts as a simple multiplier, describing the depth of the potential well.

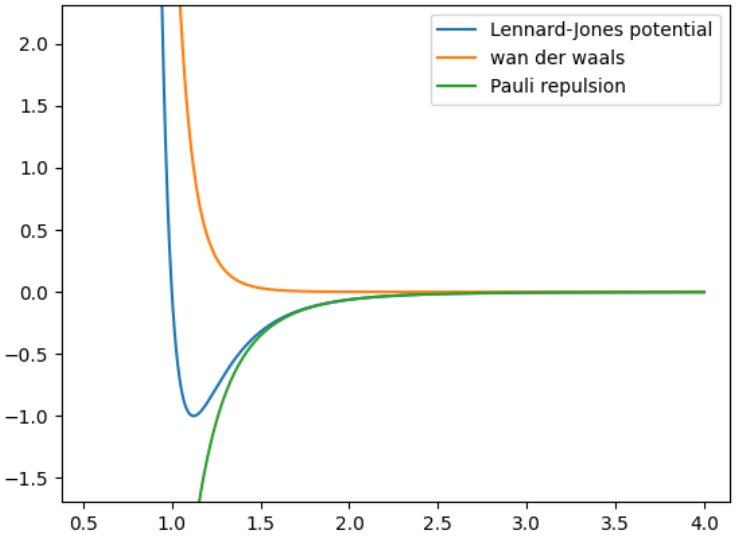

Figure 3.1: A plot of the Lennard-Jones potential, which consists of van der Waals equation and the Pauli repulsion as a function of distance from a given atom.

The Lennard-Jones equation (LJ) has been shown to hold true in the case of many noble gases, since these can be modeled with a spherically symmetric potential which is only dependent on the distance between particles [32]. This is not the case however for crystals with covalent bonds, such as silicon carbide which we study in this thesis. In a 1985 paper by Frank Stillinger and Thomas Weber [35] they proposed a new equation for the potential which adds a angle-dependent three-body potential on the following form

$$U(\{\vec{r}_i\}) = \sum_{i<j} U_2(r_{i,j}) + \sum_{i<j<k} U_3(r_{i,j,k}). \tag{3.5}$$

Here the three body potential U_3, which is dependent on both distance and angle of the three particles, is added to the two body potential U_2. The two expressions were presented as the following

$$U_2(r_{i,j}) = A_{i,j}\epsilon_{i,j}\left[B_{i,j}\left(\frac{\sigma_{i,j}}{r_{i,j}}\right)^{p_{i,j}} - \left(\frac{\sigma_{i,j}}{r_{i,j}}\right)^{q_{i,j}}\right]\exp\left(\frac{\sigma_{i,j}}{r_{i,j} - a_{i,j}\sigma_{i,j}}\right), \tag{3.6}$$

$$\begin{aligned} U_3(r_{i,j}, r_{i,k}, \theta_{i,j,k}) = \lambda_{i,j,k}\epsilon_{i,j,k}\big[\cos(\theta_{i,j,k}) - \cos(\theta_{i,j,k,0})\big]^2 \\ \times \exp\left(\frac{\gamma_{i,j}\sigma_{i,j}}{r_{i,j} - a_{i,j}\sigma_{i,j}}\right)\exp\left(\frac{\gamma_{i,k}\sigma_{i,k}}{r_{i,k} - a_{i,k}\sigma_{i,k}}\right). \end{aligned} \tag{3.7}$$

Here A, B, q, p, a, γ and λ are positive variables which are tweaked to give a stable behaviour.

The addition of the angular dependency in triplets in the third term introduces an equilibrium angle $\theta_{i,j,k,0}$. In the case of deviation from this the potential energy will increase with a square factor. The two body term U_2 has many similarities to the LJ potential and thus adds the Pauli repulsion and van der Waals forces as we saw in equation 3.4. The additional exponential term adds a cutoff range for $a \leq r/\sigma$.

The additional complexity, and accuracy of the three particle equation is all well and good, but another consideration to make is that of the charges within the elements and steric interactions. This occurs in materials of multiple elements, like our silicon carbide. The Vashishta potential [42] introduced in 1990 was originally a way of studying silicon dioxide (SiO2) which does have charge transfer, and thus Coulomb interactions.

For our applications, we make use of the following equation for SiC [43],

$$U_2(r_{i,j}) = \frac{H_{i,j}}{r_{i,j}^{\eta_{i,j}}} + \frac{Z_i Z_j}{r_{i,j}}\exp(-r_{i,j}/\lambda) - \frac{D_{i,j}}{2r_{r_i,j}^4}\exp(-r_{i,j}/\xi) - \frac{W_{i,j}}{r_{i,j}^6}. \tag{3.8}$$

The presence of both van der Waals and steric forces is described by the factors W and H respectively, where η determines the strength of the former. Z describes the charge of a given atom, and the parameters λ and ξ describe the screening lengths of the Coulomb and charge dipole interactions, respectively. Thus we can see that Coulomb interactions is handled in the second term, and the third term describes the charge dipole interactions.

In order to include the interactions arising from covalent bonds a third term is added similarly to the Lennard-Jones equation 3.4,

$$U_3(r_{i,j}, r_{i,k}, \theta_{i,j,k}) = B_{i,j,k}\frac{[\cos(\theta_{i,j,k}) - \cos(\theta_{i,j,k,0})]^2}{1 + C_{i,j,k}[\cos(\theta_{i,j,k}) - \cos(\theta_{i,j,k,0})]^2}, \tag{3.9}$$

where $\theta_{i,j,k}$ is the bond angle between particle $r_{i,j}$ and $r_{i,k}$ with equilibrium angle $\theta_{i,j,k,0}$. In our work we use the parameters by Vashishta et al., 2007 [43].

3.2 Numerical Integration

In applying the potentials described previously in a useful way, we make use of numerical integration methods. In a classical sense one can describe the movement of a given particle with a simple equation of motion,

$$\frac{\partial^2}{\partial t^2}\vec{R}(\{\vec{r}_i\},t) = -1/m_i \nabla U(\vec{R}(\{\vec{r}_i\},t)). \tag{3.10}$$

The basic scheme of numerical integration is having the particle follow a given path for a set time step, then calculating new forces and rinse and repeat. In doing this there are a few important considerations to make, most of which are about finding the sweet spot between high accuracy results, calculation time and numerical stability. Additionally, the integration scheme needs to satisfy physical conditions like energy conservation and reversibility. weighing all these considerations we choose a middle ground integration scheme called Verlet integration [44]. The fairly simple method of finding an updated position is simply a second order Taylor expansion of the position at at time t, with a small positive and negative change Δt.

$$r(t \pm \Delta t) = r(t) \pm v(t)\Delta t + 1/2 a(t)\Delta t^2 + \mathcal{O}(\Delta t^3). \tag{3.11}$$

These two equations combined give us a velocity independent equation for the position given by

$$r(t + \Delta t) = 2r(t) - r(t - \Delta t) + a(t)\Delta t^2 + \mathcal{O}(\Delta t^4). \tag{3.12}$$

from equation 3.11 we can also find the velocity as

$$v(t) = \frac{1}{2\Delta t}[r(t + \Delta t) - r(t - \Delta t) + a(t)\Delta t^2 + \mathcal{O}(\Delta t^4)]. \tag{3.13}$$

These equations are all dependent on $t + \Delta t$, which can be inconvenient in certain cases. These equations lead us to a velocity dependent Verlet integration known appropriately as velocity Verlet. The velocity Verlet scheme reads,

$$\begin{aligned} v_i(t) + \frac{\Delta t}{2} &= v_i(t) + \frac{1}{2}a_i(t)\Delta t, \\ r_i(t + \Delta t) &= r_i(t) + v_i(t + \frac{\Delta t}{2})\Delta t, \\ a_i(t + \Delta t) &= -1/m_i \sum_j \nabla U_i(r_{i,j}(t + \Delta t)), \\ v_i(t + \Delta t) &= v_i(t + \frac{\Delta t}{2}) + \frac{1}{2}a_i(t + \Delta t)\Delta t, \end{aligned} \tag{3.14}$$

in its entirety.

Despite the very unfamiliar look, this has been shown to be mathematically equivalent to the original equation [43]. This method is also more numerically stable, as a typical place where instabilities occur is subtraction of similar floating point numbers, unlike the previous rendition of the method this avoids that. It is also quicker, as there is only one calculation of the forces acting upon a given atom.

3.3 Ensemble Averages

During a simulation we only compute the atomic level position and velocities of atoms. We need these however to express the more usable often macroscopic quantities. Typical measurement such as pressure and temperature can be derived statistically using the velocity of atoms, as will be shown here. As these are statistical properties, a certain time average is assumed.

First we find the system energy,

$$E = \sum_i 1/2 m_i |\vec{v}_i|^2 + U_i. \tag{3.15}$$

This equation is simply a combination of kinetic energy in a classical sense, and the potential energy from earlier sections.

The kinetic energy goes directly into finding the temperature using the equipartition theorem,

$$T = \frac{2\langle E_k \rangle}{dNk_b}. \tag{3.16}$$

Here, N is the number of atoms, d is the spatial dimensionality and number of particles in the system and k_b is the Boltzmann constant.
The pressure needs a bit more thorough examination, as it is not as simple as just using the equation of state for an ideal gas, which might be tempting. Thompson et. al. [38] gives us an equations which takes into account both periodic boundary conditions and interatomic potentials.

$$P = \frac{Nk_bT}{V} + \frac{\sum_i \vec{r}_i \cdot \vec{F}_i}{dV}. \tag{3.17}$$

As before, $\vec{r}_i$ is the positional vector, and $\vec{F}_i$ are the forces acting upon the given atom.

3.4 The Langevin Thermostat

When applying velocity Verlet integration in a system as described until now we get a microcanonical ensemble, since energy, particles and volume is conserved. We do however also wish to look at simulations of relatively constant temperature, and so in achieving this in a system in which forces are introduced we use thermostats to keep the temperature relatively stable. Thermostats can be useful in simulating real life experiments, as these often are run in thermostatized systems rather than isolated ones, or simply to handle changes in total energy due to numerical errors. [21]

There are many ways of implementing thermostats, however the one we will use is the Langevin thermostat [11]. The thermostat removes or adds energy by a drag force which is proportional to the velocity of the particle with a dampening factor. The exact expression and algorithm is as given by the LAMMPS documentation [1],

$$F = F_c + F_f + F_r, \tag{3.18}$$

[1] https://docs.lammps.org/fix_langevin.html

$$F_f = -\frac{m}{damp}v, \tag{3.19}$$

$$F_r \propto \sqrt{\frac{k_B T m}{dt \cdot damp}}. \tag{3.20}$$

In equation 3.18, F_c is the force from the particle interactions as described by the Vashishta potential, F_f is the drag force applied to each particle and F_r is the force due to atoms colliding in accordance with the fluctuation/dissipation theorem [7]. In addition, m is the particle mass, K_B is the Boltzmann constant, T is the temperature of the bath, dt is the timestep and $damp$ is the damping factor.

An important factor to note is the possibility of undesirable effects due to the stochastic nature of the thermostat. Many repeated drawings of pseudo-random numbers may lead to series of similar numbers causing unwanted effects [8]. There is also the flying ice cube problem [19], where thermostats alter the proportions of energy in the different degrees of freedom. The origin of the name comes from the case where one does not take care to remove the net translational and/or rotational kinetic energy of the centre of mass of the system. A thermostat could in this case convert all the energy into these two forms of energy, leaving no energy in the internal degrees of freedom, forming a flying ice cube.

3.4.1 Bounding Box

When simulating relatively small systems one often finds that we get a lot of boundaries, that if not handled well will lead to strange artifacts and unrealistic behaviour. A seemingly simple yet surprisingly effective trick for making a more realistic system is having periodic boundary conditions in the x and y directions for our system. This increases the size of our system by allowing particles that venture beyond one edge of the simulation box to appear on the opposite side. Naturally the forces experienced by atoms near the edge also has to adhere to this. In practice this means that two particles both δx from the edge is at a distance $2\Delta x$ from each other instead of the distance $L - 2\Delta X$, where L is the system length in a given direction. Certain considerations must be made as to not have nonphysical behaviours occur, such as avoiding a particle being able to interact with itself through the periodic boundary. Our simulation box is, though small, is not nearly small enough for this to ever be a problem.

3.4.2 Cutoff Distance

When defining our potentials, one finds that in theory the distance at which particles interact is infinite. With computational limitations in mind this is clearly not feasible, however the potential tends quickly towards zero over relatively short distances. Using this to our advantage we define a cutoff distance that balances the physical properties

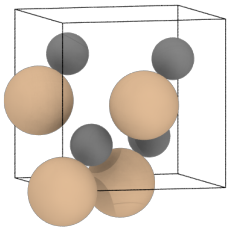

Figure 3.2: A face-centered cubic lattice unit cell of SiC in a 3C-polytype crystal arrangement. Visualized using Ovito [36]

without computational needs,

$$U(r)^* = \begin{cases} U(r) - U(r_c) & \text{if } r \leq r_c, \\ 0 & \text{if}, r > r_c. \end{cases} \tag{3.21}$$

Here $U(r)^*$ is truncated to the cutoff radius of r_c, and also shifted so that the function is continuous and zero at $r = r_c$. This approximation brings us a great advantage, as the computation per atom is more or less fixed, and the scaleability now sits around $\mathcal{O}(N)$, rather than $\wr N^2$ or more.

3.4.3 Initialization

When making a system for simulations the starting positions of atoms needs to be considered, and is naturally chosen to resemble the actual structure of the material in question. We use a *unit cell* (see fig 3.2), which is a three dimensional box in which atoms are placed, and in the case of bulk crystals these unit cells repeat to form the material.

The exact placement of the atoms within the unit cells vary depending on the material, however the simplest known as the simple cubic lattice is an atom in the origin corner of the unit cells. In Lennard-Jones solids, however, the energetically optimal placement of atoms is a so called *face-centered cubic* lattice. This structure consists of four atoms distributed within the cell in a vaguely pyramid looking structure where the peak is at the origin of the cell. Closely related to this, a version of silicon carbide can be represented in a similar fashion, however with the addition of four carbon atoms with a slight twisted orientation with respect to the silicon atoms. This 3C-SiC modification is the only type of SiC one can simulate with a cubic lattice hence the 'C' for cubic, as other types require hexagonal or rhombohedral [1].

3.5 Miller Indices

When describing atomic structures, especially crystals, it is often handy to refer to Miller indices when describing planes relative to an origin. The index is described as the distance

from an origin to a plane, in the x-y and z-direction One then takes the reciprocal of this distance and reduces it to the lowest integer. If a plane is parallel to an axis the distance is infinite, and the reciprocal is zero. This is then written in parentheses without any commas.

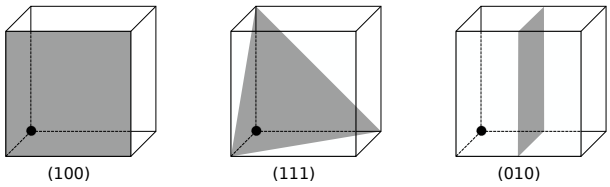

Figure 3.3: Three different uses of Miller indices. The reciprocal of the distance along each axis is reduced to the nearest integer, and written in parentheses without commas. The figure on the right has an y-and-z axis parallel to the plane, thus the distance is infinite and the reciprocal zero. In the left most figure the plane is parallel to the x-axis. Made using inkscape https://inkscape.org/

Chapter 4

Machine Learning

> A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
>
> -Tom Mitchell, 1997 [15]

As Tom puts it, machine learning is an algorithm gaining or learning from data, without explicitly being programmed. This has some major advantages over traditional programming and algorithms, and its use has seen a massive rise lately due to this fact. The way the algorithms learn from data is to map a set of data features to a set of targets, where features and targets is the input and output, or the data and what to predict, respectively. Usually this is done by fitting a set of weights using known input and output data to learn which set of input data predicts what outcome. This is known as *supervised learning* and is very common.

There exists a plethora of methods of which to set up a machine learning algorithm, a very common one of which is the Artificial Neural Network (ANN). Another method of machine learning which is especially popular in image classification tasks in Convolutional Neural Networks (CNN). In this section we will provide a brief introduction to both of these, and explain the reasons for applying these to our simulation data.

4.1 Artificial Neural Networks

The artificial neural network is inspired by the actual neurons of a biological brain, and is a very common form of neural network. The network consists of layers containing nodes, the amount of which can vary. The input passes through the network, going from layer to layer and often from one node in a given layer, to multiple nodes in the next. Each node

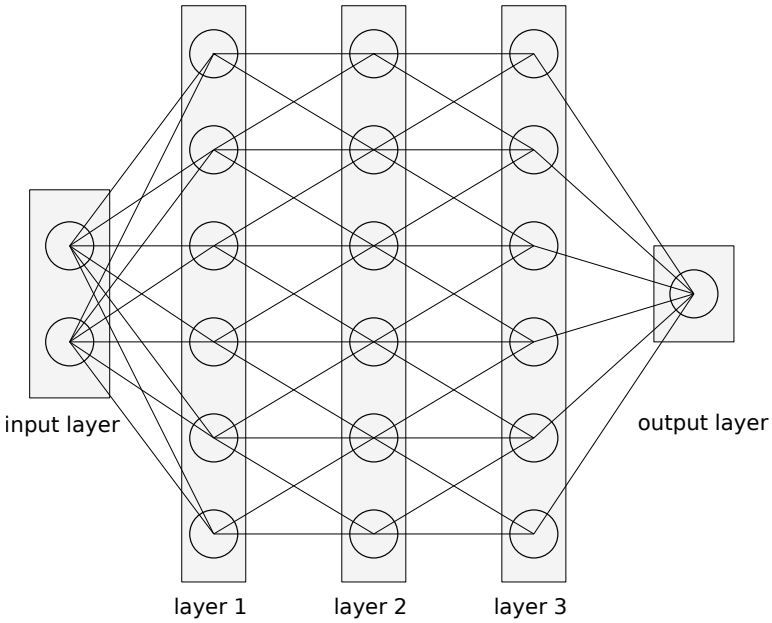

Figure 4.1: A dense neural network with two imput features, three fully connected layers containing six nodes each, and one output.
Made using inkscape https://inkscape.org/

performs a prediction based on the features passed on to it by the previous layer, and then passes its predictions forward. A set of weights determine how important a certain node is and is ultimately the thing that is altered in order for the network to learn. The final layer reduces the input features it receives and produces the final predictions.

Many different variations in networks have different properties, so things like amount of layers, nodes, how connected the nodes are and how the prediction is performed at each node has a major impact on performance. These are commonly referred to as 'hyperparameters'. This means that a network has to be tailor made for each dataset and setting. One common type of network that we will utilize in this thesis is the Dense Neural Network (DNN), named such due to each node being connected to every other node (see fig 4.1).

4.1.1 Feed Forward Propagation

The way one produces a prediction using a network and the first part of training a neural network is the feed forward propagation which passes data from the input layer through the network to the output layer. At each node the input to that node is multiplied with the weights to form the output from that node, and the input to the next layer. The data is 'fed forward' until the output layer is reached and a prediction or 'label' is produced.

Each operation on a layer can be written as a simple matrix multiplication;

$$\vec{z_l} = \vec{W_l}\vec{a}_{l-1} + \vec{b_l}. \tag{4.1}$$

Here, $\vec{W_l}$ are the weights of layer l, $\vec{a}_{l-1}$ is the output of the previous layer and $\vec{b_l}$ is a so called bias term which along with the weights are adjusted later so that the network can learn from data.

A note should be made of the initialization of the weights $\vec{W}$, as depending on the data and network these can be very important. A 2010 paper by Xavier Glorot and Yoshua Bengio [14] proposes to draw initial weights from a uniform distribution in the range $(\pm\sqrt{6/(n_l + n_{l+1})})$ where n_l are the number of nodes in layer l. This method has proven to be useful, as it has become very common.

The output of the matrix operation in equation 4.1, $\vec{z_l}$ is then passed through an activation function ϕ in order to introduce non-linearlity. The resulting equation is the written as

$$\vec{a}_l = \phi(\vec{W_l}\vec{a}_{l-1} + \vec{b_l}). \tag{4.2}$$

There are a number of activation functions, and the choice of which can be instrumental in getting the network to learn complex data, especially in deep neural networks which has a lot of layers. A very common choice for activation function though is the ReLU, or Rectified Linear Unit,

$$\phi(z, \alpha) = \begin{cases} lr\alpha z & z < 0 \\ z & z \geq 0 \end{cases} \tag{4.3}$$

The benefits of this activation function, and why it has become the industry standard is numerous, though the main one is perhaps that it is very simple. This is a computation that in any larger network will be computed millions of times, so fractions of seconds can add up to quite some time.

4.1.2 Backward Propagation

In the forward pass the network has not learned anything, what it does however is perform a prediction which we can evaluate using training data in which we know the correct label. A cost function which evaluates the network output is used to adjust the weights minimizing the difference between the predicted label and the actual label. The types of cost function can vary but since this is a typical regression task a very common one is the mean squared error (MSE),

$$L = \frac{1}{n}\sum_{i}^{n}(y_i - y_i^*)^2. \tag{4.4}$$

Here y_i is the predicted label and y_i^* is the correct label of n samples. Using this we proceed with the backward propagation proposed by Rumelhart et. al., (1986) [31]. The derivation of the back propogation method is fairly involved, but the general idea is to

compute the gradient of the cost function for each node starting at the last layer and working your way to the front. Once all the gradients are computed the weights and other parameters such as the bias can be updated using an optimization algorithm, for instance the so called stochastic gradient decent (SGD).

At each node in each layer, an adjustment of the weights is done based on the aforementioned cost function. As the name indicates, the negative gradient is followed to reduce the cost function as much as possible. This is given as the following function,

$$\nabla_{\vec{W}} C(\vec{W}) = \sum_{i}^{n} \nabla_{\vec{W}} C_i(\vec{x}_i, \vec{W}). \tag{4.5}$$

Here the weights are denoted as $\vec{W}$, the cost function is C and the input into the node is $\vec{x}_i$. A 'step' is then performed in the direction of the gradient,

$$\vec{W}_{j+1} = \vec{W}_j - \gamma_j \nabla_{\vec{W}} C(\vec{W}). \tag{4.6}$$

The term γ_j is known as the learning rate, and is the size of the step in an iteration. A more intuitive way of thinking of this is to imagine SGD as moving in a three dimensional terrain, where the topography is determined by the cost function. We wish to find as low terrain as possible, and we simply follow the slope of the point at which we are standing, and take a step of size γ. There are certain pitfalls one might happen upon though, such as local minima which stop us from reaching the global minima. This can happen if we use a too short learning rate, or we might step over the global minima by using a too high learning rate. Most common is to use an adaptive learning rate, or a similar scheme such as Adam optimizer [25].

The calculation of node gradients is very computationally intensive, so to lessen the load the gradient is taken over an average of datapoints at a time in a so called 'minibatch' [15, p. 15], . Once the whole dataset has been evaluated and calculated we say the model has finished one 'epoch'. Both the number of minibatches, epochs and many other parameters both mentioned previously and left out are dependent on the dataset, and the network in question. How to choose the correct hyperparameters is therefore a hot topic, as it can absolutely make or break the network and its usefulness.

4.1.3 Evaluating Model Performance

In order to evaluate how the network is performing on a set of data we need a metric for doing this. The previously mentioned mean squared error 4.4 is good for regression, however another metric which has become popular is that of the R^2 score or 'coefficient of determination'.

$$R^2 = 1 - \frac{\sum_i (y_i - y_i^*)^2}{\sum_i (y_i - \overline{y})^2}, \tag{4.7}$$

here y_i is the correct label, y_i^* is the predicted label and $\overline{y_i}$ is the mean of the labels. Though a bit hard to get a grasp on, R^2 can be interpreted as how well the model explains the variance in the dataset. It does this based on a 'null hypothesis', which is a

horizontal straight line. When the R^2 is negative, it means that the model describes the worse than that horizontal straight line.

Having metrics for evaluating performance is vital, but we still have a lot to gain in applying these in a smart way. The first step is to split the data (of which we know the correct label) into training and test data. The test data will stay completely separated from the network, so as to not teach the network anything about it. It is also essential that the data is randomly shuffled before we separate into about 20% test and 80% training data, depending on the amount of data etc. Another further step in evaluating the model is to split the training data into validation and training data. This technique is known as k-fold cross-validation [15, p. 71], and it involves splitting the training data into k separate parts. One of the parts or 'folds' are used to test the network, often called validation data, and the $k-1$ folds remaining are used as training data. The data is then shuffled and another fold is used to evaluate the training performed on the remaining data. The performance of the network is then found to be the average of the k folds but also the variance and deviation of the resulting scores can be of interest. The actual performance of the model is finally found once the trained network is applied to the test set, which until now has been left untouched as to not cause a 'data leak' [24], meaning the network somehow learned from the test set leading to optimistic performance results.

Overfitting and Underfitting

An ever present problem with dear I say all machine learning is that of overfitting, especially in cases where training data is sparse. This means that the network you train is too tuned in to the training data you've provided, so it has learned specifically that data and does not perform well on general data. This can clearly be seen in the loss plots which indicate performance of the model on the training set and the test set. The network will perform arbitrarily well on the training set, but eventually worse and worse on the test set. Some methods might also be too simple to effectively describe and catch on to the complex nuances in a dataset. There are many methods for making a network perform well in general cases such as regularization, the reader is encouraged to seek sources such as 'Hands-On Machine Learning with Scikit-Learn & TensorFlow' [15] or similar.

4.2 Convolutional Neutral Networks

Another type of neural networks which has become popular lately in image recognition software is that of the Convolutional Neural Network (CNN). In principle it has much in common with a dense neural network however in a CNN the weights take the form of a *kernel*, or layers thereof which propagate across the input data in strides. The unique and very useful feature of this kind of network is that it picks up on spacial features, or features which are translationally invariant. The application of the kernel layer or filter is a $feature\ map$, which constitutes the next layer in the case that the filter has multiple layers.

In the case of an image, the input data is in the form of a matrix, in which the CNN network will start with the kernel layer in the top right corner and work through row by row, column by column with a given stride. The computation of a neuron in a 2D convolutional layer with multiple kernel layers is written as [15, p. 366]

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k}, \tag{4.8}$$

where

$$i' = i \times s_h + u, \tag{4.9}$$
$$j' = j \times s_W + v. \tag{4.10}$$

Here, $z_{i,j,k}$ is the feature map k of the neuron located on row i, column j in layer l. s_h and s_w are the vertical and horizontal strides, f_h and f_w are dimensions of the receptive field and $f_{n'}$ is the number of feature maps in layer $l-1$. $x_{i',j',k'}$ and $w_{u,v,k',k}$ is the input from the previous layer with variables x', j', k', and the weights or kernel connecting the two with k and k' being the current and previous feature map. There is also a bias term b_k such as in the DNN case, which alters the importance of the given input, and which is tweaked along with the weights. This operation is done for all i, j in the input data to form the complete feature map. The parameters in the kernel is the tweaked to predict the correct label. A large CNN will often have many layers of kernels, and feature many other techniques such as padding, and pooling depending again on the input data. The last operation on the kernel layer is often one or two fully connected layers of DNN before the final label is predicted.

The shape and size of CNN networks vary a lot, depending on the type of data and the shape. For example a typical image has the shape (n_x, n_y, c) where n_x, n_y is the shape of the image, and c is the number of channels (typically 3, RGB). In this case another summation is needed in equation 4.8 over the channels. The number of fully connected layers before producing the output labels also varies, like in figure 4.2.

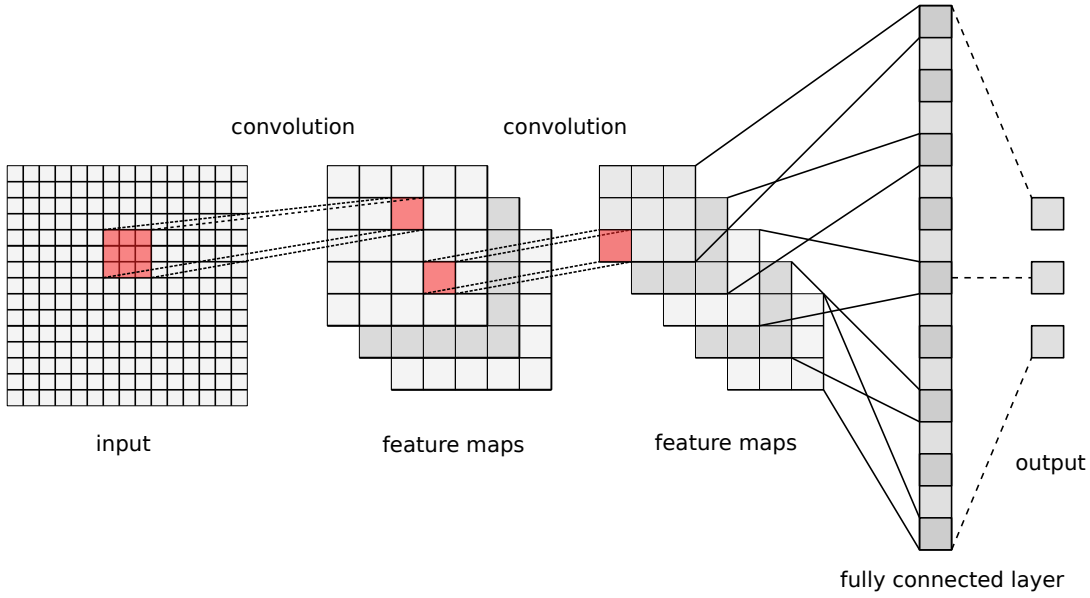

Figure 4.2: A network consisting of two convolutions then a fully connected layer. Depending on the input size and number of channels the network will look very differently. Output is three labels.

Made using inkscape https://inkscape.org/

Part II

Method

Chapter 5

Software Packages and Computational Conscideration

The ease at which one can do complex molecular dynamics simulations is not a given, yet it is so thanks to a set of software packages.

5.1 Software Packages

5.1.1 LAMMPS

The practical implementation of the ideas presented in the theory section heavily relies upon the open source software Lammps (Large-scale Atomic/Molecular Massively Parallel Simulator) [39]. The easy to use high level API makes it possible to focus on the what, rather than the how. It is also compatible with MPI, allowing for easy parallelization and CUDA for GPU-acceleration.

The LAMMPS software has many types of calculations which are useful in describing an analyzing the system, however these add computation time depending on how many and how often we wish to compute them. The two main types of files that are dispensed by LAMMPS, that is a 'dump' file, and a log file. Log files are updated at a given n time steps, and can contain any of a number of calculations we wish to include. A dump file has information regarding the atoms and particles in the system, and is mostly useful for visualizing the system in time using Ovito.

5.1.2 Molecular-builder and Atomic Simulation Environment

Another package which simplifies making molecular structures is the molecular-builder[1] package. This is a wrapper package for Atomic Simulations Environment (ASE) atoms [2], which simplifies making of bulk Crystal and other shapes such as the dodecahedron mentioned later. ASE, which also offers a graphical user interface allows for a high degree

[1] https://github.com/henriasv/molecular-builder
[2] https://wiki.fysik.dtu.dk/ase/index.html

of customizability and control. In addition to being written in Python, it also strives to be compatible with other popular Python packages such as NumPy [18], making using and understanding the package easier for those with a typical Python background.

5.1.3 Ovito

Visualizing the system is mostly done using the Ovito package [36], which is a graphical and programming interface for working with atomic structures. Operations such as splitting the system, identifying crystal structures and visualizing is made very easy using this package. Especially using the Pro version of the graphical interface provides a very simple way of manipulating the system visually, and the exporting the code to do similar operations using the programming interface. Ovito also has tools for rendering and animating simulations, providing unique insight into the behaviour of the system after performing a simulations. In addition there are implemented an array of modifiers, such as displacement vectors, crystal structure identification and clustering, many of which we use later in the thesis.

For easy use with other packages there are also ASE and LAMMPS integration, which makes transferring data trivial. To top it all off, there is a surprisingly active forum, where users of all backgrounds can ask questions, and often get a quick reply from an in-house Ovito employee. Not something one comes across often in relatively niche molecular dynamics software.

5.1.4 Atomsk

Atomsk is command line tool for manipulating and creating molecular structures. It supports many of the same data types as LAMMPS, Ovito and ASE, of which we primarily use .xsf [3] as this is used by all three packages. The Atomsk package prides itself of being the 'Swiss-army knife' of atomic simulations, yet our primary use of Atomsk is the making of a slightly tilted (110) oriented crystal structure as explained in subsection 6.2.2. Atomsk is armed with create functions in which one can specify spacegroup [22] as well as orientation in Miller indices (see section 3.5), meaning we can easily make our 3C-polytype FCC lattice in whichever orientation we need.

5.2 Computational Considerations

All simulation was done on NVIDIA A or P 100 Graphical Processor Units, or GPU cards. These have the benefit of being able to parallelize simulations with double float precision. The resulting data were of no insignificant size, and care had to be made to not exceed the alotted number of terrabyte storage. A tradeoff between what data we would like to store, and storage space were made. Examples of this is to only store dumpfiles, which can

[3] http://web.mit.edu/xcrysden_v1.5.60/www/XCRYSDEN/doc/XSF.html

Chapter 6

Practical Implementation

In this part we explain the ways in which we proceed in choosing the system, and how the simulations were done as to yield the best results. We then describe the ways we apply machine learning to further analyze the behaviour of the system. Initially a series of simulations are done looking at various systems, in search of one which we find fitting with regards to physical and computational considerations. Once our system is chosen, the remaining variable is then the placement of asperities. As in the paper by Hanakta et. al. [17] we initially look at a smaller two asperity system where the number of combinations is a manageable 10. We then proceed to a larger 8 asperity system with 767 possible configurations. A large amount of simulations are then done with various asperity placements, in order to make a dataset for machine learning.

Code accompanying the texts can be found at https://github.com/andebraa/master

6.1 Methods For Describing And Analyzing a Molecular Dynamics Simulations

The best and clearest results imaginable can be missed if the correct analytical methods are not applied. Since we apply molecular dynamical simulations we have the opportunity to actually observe how the asperities and the rest of the system behaves and moves using various visualization packages. Having ways of extracting results from simulations is also necessary to do further analysis like machine learning. This section will describe some of the ways we extract results from the simulations.

6.1.1 Asperity-Asperity Norm

In a system in which placement of asperities is the main varying factor it is natural to discern a way of describing the distance of which the asperities are located from each other. We also need to consider the periodic boundary conditions which is an important assumption in the simulations. As will be explained in section **??** the placement of asperities is limited to a 4×4 grid as shown in figure 6.4. The distance between each and every asperity is simply the norm of the index in the boolean matrix which describes

the system. Since we throughout this thesis consider a 4×4 system we know that if the distance between two asperities is more than 2 in either x or y direction, them the opposite way through the periodic boundary is the shortest. The equation below shows the algorithm applied on the x and y components of the index of a boolean 'true' in the matrix describing the system. This is commonly known as the minimum image convention and is used in systems with periodic boundary conditions.

$$dl \rightarrow dl - round(dl/4) * 4, \tag{6.1}$$

where dl is the distance between two asperities in a given direction. The function is exactly as written in Python, with *round* being the NumPy package round function.

Once the shortest distance between each and every asperity is found we simply sum up all the squared norms and average based on the number of asperities. The whole algorithm is shown here

```
matrix = np.array(matrix)
indices = np.asarray(np.where(matrix==1.0)).T

x = indices[np.newaxis,:]
y = indices[:,np.newaxis]
dist = x-y
dist = dist - (np.round(dist/4))*4

norm = np.linalg.norm(dist, axis = 2)
norm = norm**2
res = np.sum(norm)/2
ones_matrix = np.ones((len(norm[0]), len(norm[1])))
res = res/((ones_matrix.sum() - np.trace(ones_matrix))/2)
```

6.1.2 Friction Measurements

Looking at the resistance the top plate experiences as we push the top plate at a constant velocity we get what we refer to as a load curve. The forces start off at zero, and as the forces increase and the system is strained, so does the load curve before eventually slipping and forming a slight peak. The first prominent peak of the load curve corresponds to our estimate of the maximum static friction force. The aforementioned peaks are found by simply selecting the point with the highest value in the immediate area after the breakage.

The velocity of the top plate is constant, however the rate at which the forces increase in the load curve plot is not. In order to describe this slope in forces one can recognize that the general shape of the load curve from start to finish has two constant levels and a transition in between. To estimate the sharpness of the transition we fit a Sigmoid curve to these data. This we do by applying the SciPy method "curve_fit" [1] to a selected portion of the data. The slope is then found using the sigmoid curve, which is simply

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

defined as

$$y = \frac{L}{(1 + exp(-k * (x - x_0)))} + b. \tag{6.2}$$

The "curve_fit" method fits L, k, x_0 and b as to fit the data as well as possible. L scales the output to be in the interval $[0, L]$, b normalizes the data to fit the interval $[b, L + b]$, k scales the input and x_0 sets the middle point of the curve.

An alternate way of finding the maximum static friction, especially in data with a lot of noise is to simply use the highest value in the sigmoid fit. We will refer to this as the maximum sigmoid value, and we make use of this later in the results section (see figure 8.1).

6.2 Designing the System

The system is designed as two bulk crystal plates with a number of asperities in the shape of a dodecahedra placed at periodic intervals in a grid pattern in between the two plates. The specific shape of a dodecahedron is chosen as a stable result of the relaxation of a sphere [37], hopefully resulting in a reduction of diffusion in the asperities which might affect results.

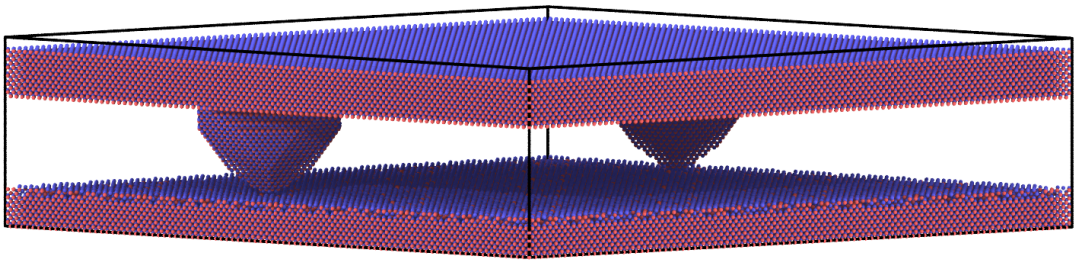

Figure 6.1: An example of a two asperity system, in between two bulk crystals. Here the top plate is 5 unit cells thick totalling 21.978 Å. The bottom plate is 17.582 Å or 4 unit cells. The placement of the asperities is limited within a 4 by 4 grid, denoted by a boolean matrix on the form [[0,0,1,0][0,0,0,0][0,0,0,0][0,0,1,0,]]

The way we describe the layout of a given system is by a boolean 4×4 matrix, for example like this 'chess' layout, which is used later as a standard layout [[0,1,0,1],[1,0,1,0][0,1,0,1],[1,0,1,0] (see fig 6.2. For the two asperity case this is visualized in figure 6.4.

In designing the system we must choose the size of the asperities, the number of grids in which these can be placed, the thickness of the upper and lower plates as well as relaxation time, to name a few. As with most simulation we choose these to be as realistic as possible, within the confines of computational cost. We are also searching for a system, in which random factors, or perhaps unforeseen effects is lessened as to

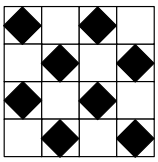

Figure 6.2: A visualization of the 'chess system'. A four by four grid occupied by eight asperities in an alternating fashion.

not conceal any effects the placement of the asperities might produce. With regards to relaxation time we simply set this at a constant, ensuring that the system has reached a relatively stable state. Throughout this thesis we choose a 1 ns relax time.

6.2.1 Top Plate Thickness

The top plate, being a dense bulk crystal makes up a large part of the total atoms, and therefore also a lot of the computation time per time step. It is in general assumed that the top plate represents a larger object, which binds the asperities together and allows these to interact via the exchange of forces. The simulated system should be large enough to be realistic with respect to the measured properties, but still small enough to be computationally feasible. In choosing the height of the upper surface we run a series of simulations to gauge the effect a thicker versus a thinner plate. As the system gets bigger and more realistic we would assume the behaviour of the system eventually reaches a point in which effects due to the system being too small would no longer plague the results, and the variance in max static is stable as the system increases. The idea is then to find this point, and in doing so finding the smallest system which still have seemingly realistic behaviours. The incremental increases in top plate thickness comes in the form of unit cells as described in chapter 3. The size increases in increments of 4.3956 Å in the z direction.

The thickest plate was plagued by some numerical stability issues, as atoms were lost far outside the bounding box. The thickest plate was also slower to simulate, having to run for over 20 hours on an NVIDIA A100 graphics card making it unfeasible for our simulations.

6.2.2 Lower Plate Thickness and Orientation

As with the top plate the bottom plate is of some significance, as it is the surface on which the asperities will slide. One thing we included in the early runs of the system is the removal of atoms on the bottom plate surface to increase surface diffusion. This is linked to the many studies of so called stick slip systems, in which it is theorized that the area around the asperities of atoms caused by diffusion affects the static friction of the asperity [26]. Since surface diffusion is so important in these systems so is the number of available atoms and hence the thickness of the bottom plate. Although we ran some initial simulations with the surface atoms removed, this has since been found

to be outside of the scope of this thesis.

As the top plate and asperity are carved out from one crystal, these naturally have the same crystal orientation. The bottom plate however is thought to be another system in which the top plate and asperities come into contact with. It would then make sense that the two objects have different orientation. The different orientations of crystals has a major effect on the behaviour of these, mainly for our purposes being the ease of which the asperities and the bottom plate form bonds. In figure 6.3 one can see the difference between a system in which upper plate, asperity and bottom plate all are aligned with respect to the crystal orientation and one in which the bottom plate has a different orientation. Specifically the bottom plate and asperities has orientation (100) in Miller indices, and the bottom plate has either (100) or (110) in Miller indices (see section 3.5). The upper panel of figure 6.3 has the same orientation (100) as the bottom plate, meaning after a few picoseconds they have managed to fuse together to form one solid crystal structure. The bottom figure has a different orientation, and so has no crystal structure throughout the relax phase. The method we used for this figure is the 'Identify diamond structure' method in Ovito [28] (appendix A), which does as the name says identify certain diamond structures.

Also of note is the fact that despite periodic boundary conditions in x-and-y direction there is a continuous diamond structure extending out through the boundaries. This is important as the beneficial effects of periodic boundary conditions is dependent on the system acting as one through these boundaries.

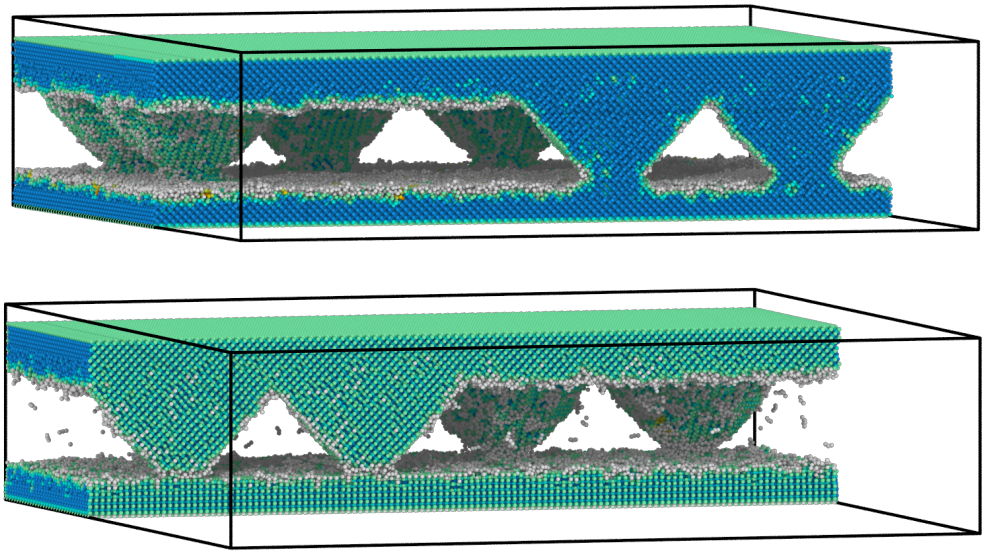

Figure 6.3: Two eight asperity systems with lower plate orientation (100) on the top, and (110) below. Both have a cross section slice shown using the Ovito function 'identify diamond structure'. Continuous diamond crystal structures corresponds to uniform uninterrupted colour, in this case variations of blue.

Making bulk crystals with orientation (110)

The aforementioned molecular-builder package which acts as a wrapper package for ASE has built in functionality for bulk crystals of orientation (100). In implementing functionality for orientation (110) we made use of the Atomsk package [20] which offers an easy to use command line tool to create and manipulate atomic structures. This allowed us to create a bulk crystal with orientation (110) and space group 216, then from this cut out a slab of the desired size to use as my bottom plate. The whole process is shown in the bash script below

```
#!/bin/bash

atomsk --create diamond 4.3596 Si C -spacegroup 216 -orient [100] [010]
    [001] [10-1] [010] [101] -duplicate 100 0 100 output.xsf

atomsk output.xsf -cut above 315 z xsf
atomsk output.xsf -cut below 295 z xsf
atomsk output.xsf -cut above 201 x xsf
```

```
atomsk output.xsf -cut below -200 x xsf

atomsk output.xsf -duplicate 0 92 0 xsf

atomsk output.xsf -shift 200 0 0 testrun_shifted.xsf
atomsk final_output.xsf -shift 0 0 -296
```

The code above simply generates a large block of SiC with the desired orientation and characteristics, then cuts it down to the desired size in x-, y- and z-directions. In order for the code to work with ASE atoms and molecular-builder some shifting of the coordinates has to be done in the last two lines. The end result is a block of atoms which fits specifically for the 4×4 grid which we look at in this thesis. If any different shape is desired all values must be adjusted accordingly.

6.2.3 Normal Force

Normal force, especially in classic friction theory, has a major effect on the friction characteristics of a system. A too high of a normal force might cause the system to be too compressed, as well as contributing to creep during the relax phase. As with top plate and other variables we perform a series of simulations with various values of normal forces, to see which have the desired characteristics. We also analyzed the system for various normal force by animating the simulation using Ovito.

6.3 Handling Symmetries

Due to periodic boundary conditions mentioned earlier we need to handle translational and mirrored symmetries that would otherwise lead to a skewed dataset for machine learning. [34] This includes symmetries in both the x and y direction, as well as systems mirrored along the x-and y-axis. We do not include rotational similarities, as the direction in which we push means a rotated system will not behave the same as the non-rotated one. Finding these symmetries in practice involves iterating thought all possible combinations of the asperities in a four by four grid, and finding which are similar to each other once shifted and/or flipped. We use the itertools package which has a function 'combinations' [2], then we pass arguments like the grid resolution and the number of asperities associated with the system that we want to find all combinations of. We iterate over all the possible configurations, checking each system for whether is is a unique system when it is flipped and translated along the x-and y-axis. If we find a system to be translational or reflectional symmetrical to a previously found system, then we discard it as to not have any duplicates.

[2] https://docs.python.org/3/library/itertools.html

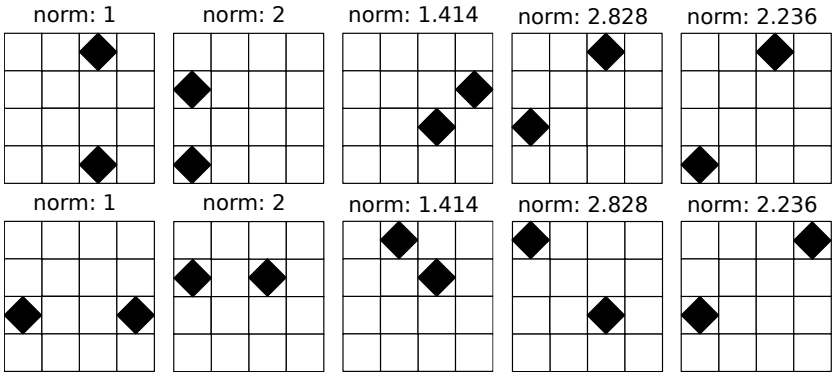

Figure 6.4: All ten unique two asperity systems with various asperity-asperity norms. Each system is expressed as a boolean matrix indicating the location of each asperity. In finding these unique systems only translational and mirrored symmetries were considered. Also note that the norm given is the shortest one considering periodic boundary conditions.

6.4 Temperature

A quick look was had on the variation of temperatures to see how this affects the system, and to what degree the effects we see in the hotter system carries over to the slightly colder system. The usual temperature in which simulations were carried out is 2300K, this temperature is close to the melting temperature of SiC of around 2700K which gives us a rather malleable system as the asperities are pushed and pulled. It also increases diffusion, which is often an essential effect in ageing and stick-slip systems. This is exacerbated as SiC crystals are very surface energy dependent, and effects such as faceting happens even faster with higher temperature and diffusion.

6.5 Machine Learning

Initially we apply both a dense neural network (DNN) and a convolutional neural network (CNN) using k-fold cross-validation in a large grid search in an effort to find a network which can predict the maximum static friction force and slope. The argument for applying a DNN, which isn't known for its spatial invariance properties, is the fact that the input to the network is so small. If we were to apply DNN to an image of regular quality we would have a harder time achieving translational invariance [3], however for an input image of 4×4, or 8×8 pixels with 2 pixels of padding in all directions, we can easily allocate a node to each input feature. This increases the chance of the network learning spatial features, which is essential in this application. It is also possible to apply DNN to MNIST datasets, with often gives good results [41].

[3] Translational invariance is when a neural network recognises a feature, despite it being located in a different spot than it is trained for.

The input to the networks is the boolean matrix described earlier, and the output is the slope of the load curve and the maximum static friction force. The variables we explore in our grid search are listed in table 6.1.

Table 6.1: The various hyperparameters for DNN on the left and CNN on the right, which become part of our grid search.

	DNN		CNN
Nodes	(4,8,16,32,64,128,256)	kernel sizes	(2,3,4)
layers	(4,8,16,32,64,128,256)	number of kernels	(8,16,32), (16,32,64)
learning rate	(1e-3, 1e-4, 1e-5)	learning rate	(1e-6, 1e-5, 1e-4)
batch size	(32, 64)	batch size	(16, 32, 64)
bias	(0,1,2)	bias	(0,1,2)
padding	(0,1,2)	padding	(0,1,2)
k-folds	(5,6)	dense layer nodes	(4,8,16,32,64,128,256,512,1024)
methds	(mse, r2)	methods	(mse, r2)
epochs	400	epochs	300

Disregarding variations in method and padding this leads to a grid search of 1458 parameters for CNN and 588 for DNN. The method variable in table 6.1 is either R^2 or MSE, since both of these evaluate data in slightly different ways it is helpful to use both in training the network. When we test the method using our test data we always apply both. An interesting aside is how important efficiency is when it comes to calculations performed many times. Using R^2 as the method in training the dataset slows the training down considerably, despite the fact that R^2 is only a little more complicated than MSE.

Part III

Results And Discussion

This part of the thesis will describe the results of the simulations described in the method chapter. Initially in chapter 7 we aim to choose a system for further analysis. We perform preliminary simulations varying top plate thickness, velocity of the top plate and the normal force on the top plate. After settling on simulation conditions deemed realistic we run simulations that are presented in section 8. The simulations chapter is divided into two and eight asperity cases, as the two asperity case can easily be simulated for all possible configurations. Section 8.3 is divided into two parts, as we in analyzing the initial machine learning results realize a different approach to finding the maximum static friction yields more stable results. The first part analyzes maximum static friction, whilst the second part uses a sigmoid fit to the loading curve to estimate the friction force.

Chapter 7

Choosing Simulation Conditions

All preliminary simulations in order to gauge the effects of various physical aspects are done with the so called 'chess system', where every other grid cell is occupied by an asperity (see fig 6.2). Results are discussed based on their physical behaviours compared to theoretical models. A final set of physical variables are decided on before we proceed with simulations in chapter 8.

In order to keep variables which are not included in our scope to a minimum we decide right away to keep to a 1 ns relax time before pushing.

7.1 Varying Thickness

In varying the thickness of the upper plate we found that this has a major effect on the computation time, as the plate is so dense a single unit cell in thickness adds about 72000 atoms, from roughly 735000 atoms to 803000 atoms in an eight asperity system. This adds hours to computation time, and the largest system took days to compute, whilst being effected by numerical instability. Keeping this in mind, we must also remember that if the system doesn't model reality well enough, there's no point in doing the computation at all.

Also note that the top plate simulations were done before our decision to implement a (110) Miller index bottom plate. This means that figure 7.1 and 7.2 both have bottom plate (100). As we considered the stability both in results and numerically, as well as computation time we believe this would not have changed the result if it was to be done with bottom plate (110).

Figure 7.1 shows the load curves of multiple systems of so called 'chess' layout. The solid line is the average of the three dotted lines, which are all identical systems with unique seeds in the velocity initialization and thermostat. Most plots look fairly similar, however the behaviour of the smallest top plate is a bit more interesting. Note that a one unit cell top plate is practically no plate at all. The behaviour of the load curves is rather staccato, almost forming a sawtooth signal. This might be due to the fact that asperities don't communicate, in a sense that they don't unload forces onto each other. When one asperity breaks, all others do as well, and the system is practically sliding with no

resistance at all, before finding grip again and continuing resistance. All the simulations of the same thickness do seem to be fairly identical, but keep in mind that the asperity layout of these are all identical. Also note the smoothness is due to a running mean of the simulation data. For these data a window of 5000 was used, as opposed to the usual 1000 which we apply in the rest of the data.

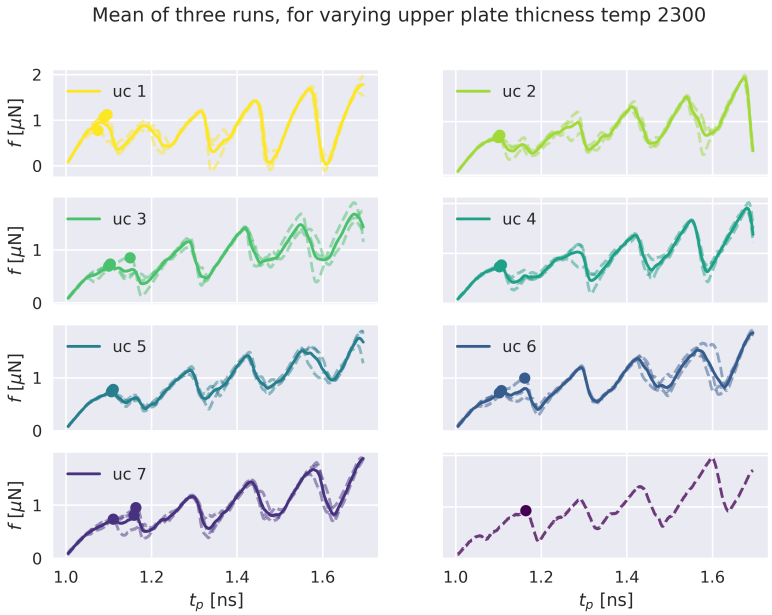

Figure 7.1: The various load curves, i.e. the resistance the top plate is experiencing, as a function of time, with varying top plate thickness. A unit cell (uc) is 4.39 Å, meaning this is the smallest top plate, ranging to 35.12 Å. This system was a four by four grid with eight asperities placed in a chessboard pattern. Note that the bottom plate has (100) orientation, as compared to all other plots which has (110). The thickest top plate was affected by a lot of numerical instabilities, resulting in too many lost atoms, hence only one full load curve is available.

Figure 7.2 is comprised of the same simulations as figure 7.1, however here we have only plotted the maximum static friction each curve reached on the first peak. As stability in simulations is important if anything is to be read out of the data, we can clearly see which systems are more sporadic than others. As expected, the one unit cell system behaves very strangely, especially for higher temperatures. It is not until we reach thicknesses of around four or five unit cells that it seems the system is behaving consistently. The colder system at 1800K was found to have less variation, and slightly different maximum static friction values. They are however not hugely different, especially as the size of the upper plate increases. At a temperature of 2300K we would theoretically have more effects such as faceting and healing, although the difference between the temperatures doesn't seem to make a massive difference. Since

the simulations are a lot more well behaved around the four five unit cell thickness we choose the five unit cell thickness as we move forward, as well as a temperature of 2300K.

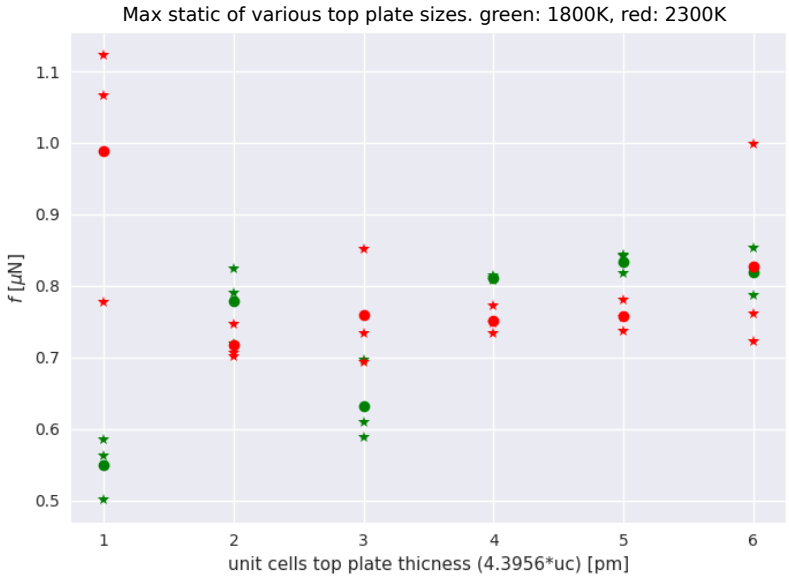

Figure 7.2: The various maximum static friction values visualized as the dots on the peak of each load curve in 7.1. Simulations are done using a chessboard pattern of eight asperities. For certain upper thicknesses the variance in the resulting maximum static friction values is higher than others. This is especially true for the thinnest upper surface.

7.2 Varying Velocity

Figure 7.3 shows the mean of three simulations for each of the four different top plate velocities. On an intuitive level, it might seem strange that the force the system exerts on the top plate increases with the speed, however this is in line with the equations for thermal activation as described in part 2.4. Figure 7.6 shows the various maximum static friction, slope and highest value of the fitted sigmoid curve. Additional simulations not included in figure 7.3 were done as to fill in the blanks and to better fit the logarithmic and linear fittings denoted as 'logfit' and 'linfit' respectively. As the R^2 score indicates there is more of a linear trend than a logarithmic one for both the maximum static friction and the highest sigmoid value, despite thermal activation theory stating otherwise. There are various reasons why this could be, yet the simplest explanation is perhaps that we are looking at too much of a limited scope in terms of velocity values. According to Aharnov et. al. (2017) [2] 'high speed' velocities are around V $\mathcal{O}(10^{-2})$, which is a lot slower than we are looking at here.

No specific velocity value seem to have any clear disadvantages, so a choice for a velocity of 5 m/s is used.

Figure 7.3: Load curves for varying constant velocity of the top plate. Each clear curve is the average of three lower opacity curves of the same system with unique seeds. The normal force is zero, with eight asperities placed in a chessboard fashion. The increasing force resistance observed as the speed increases is consistent with thermal activation. Each load curve has been averaged with a running mean with window 1000, after which each curve consists of 50000 datapoints

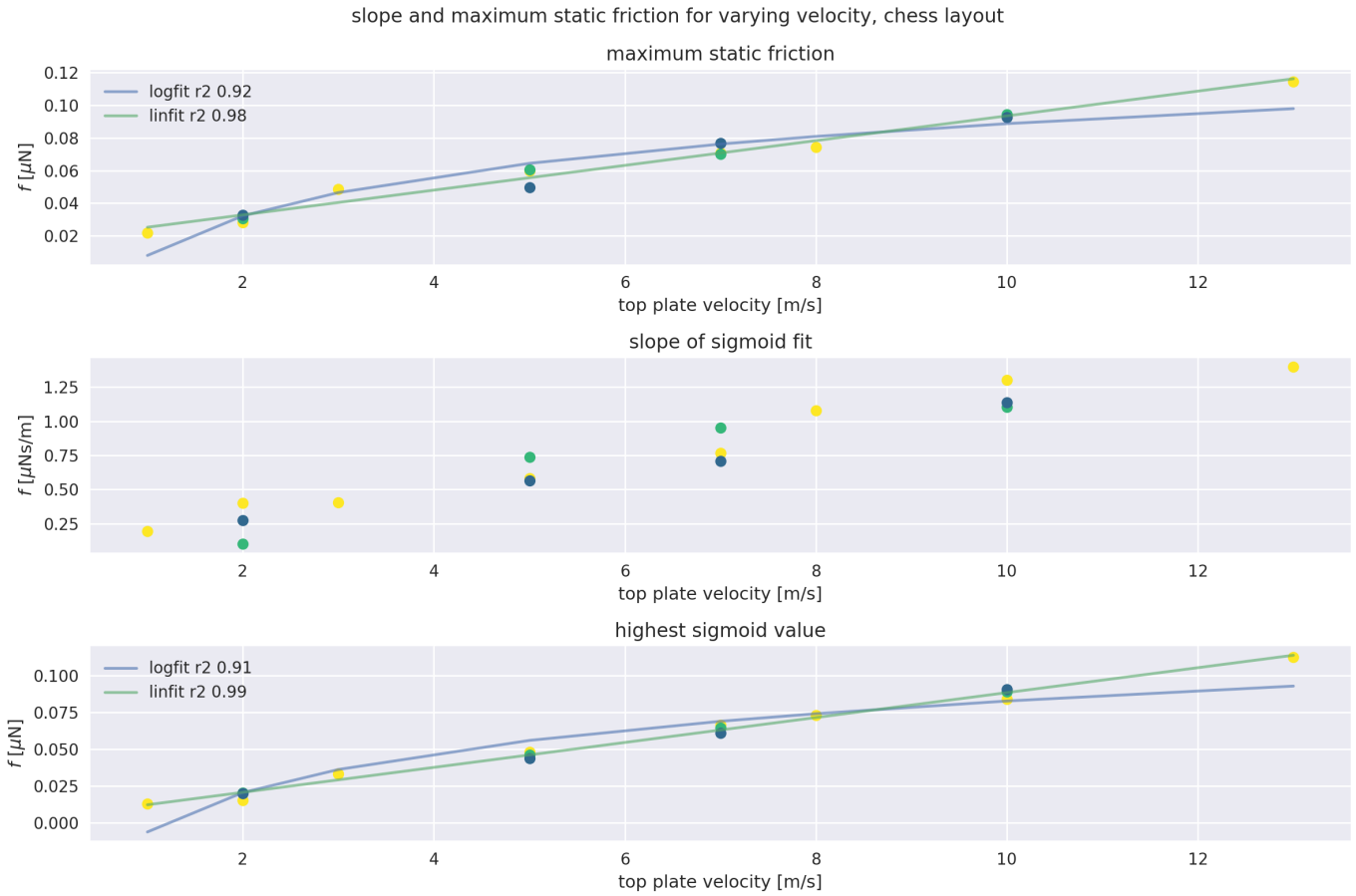

Figure 7.4: Maximum static friction, slope of the sigmoid fit and the highest sigmoid value for various top plate velocities. A linear and logarithmic fit with a computed R^2 score has been added to the maximum static friction value and the maximum sigmoid value. As the theory section states these values are expected to be logarithmic.

7.3 Varying Normal Force

As can be seen in fig 7.5 the normal force has a massive impact on the friction within the system. Like the Mohr-Coulomb equation 2.2 states we would expect to see a linear relationship between normal force and friction, with a constant factor explaining the behaviour of normal force 0. This is better shown in figure 7.6. The maximum static friction as a function of varying normal force seems fairly linear. The slight deviation could be due to inaccuracies in our simulation model, or the fact that the rather simple Mohr-Coulomb equation also has some inaccuracies.

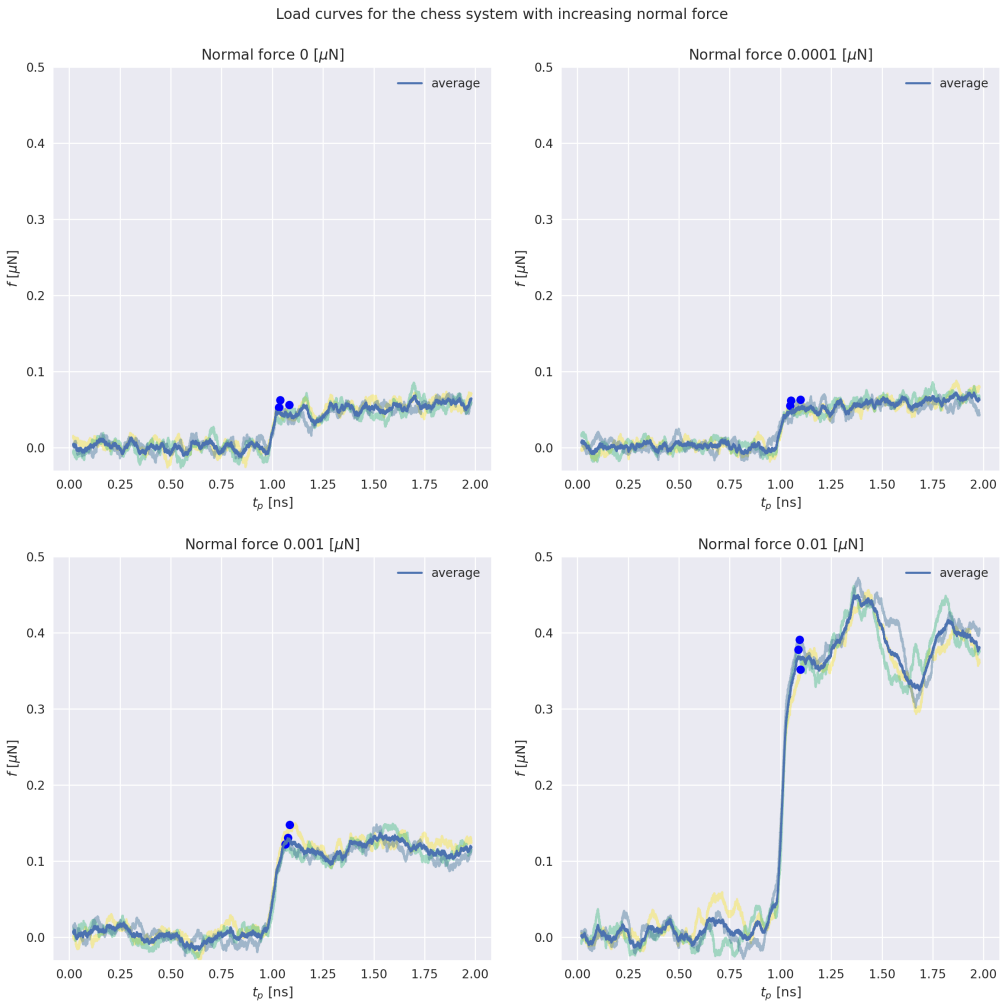

Figure 7.5: The load curves of a 'chess' system of every other grid occupied by an asperity, with varying normal force. The main curve is the average of three lower opacity plots, of which all have a unique seed.

Figure 7.6 shows the results of varying the normal force for the chess system. A linear fit shows that this corresponds well with theory as the increase in friction is fairly linear. From looking at figure 7.5 one might conclude that more is better, and that if the system had more distinct load curve peaks spotting interesting features of the static friction might be easier, however if we look at the timelapse in figure 7.7 the system even at no normal force behaves as though it was under pressure. Similaraly when we look at systems with larger normal force it tends to be crushed, especially in the push phase. The distinct cohesion effect which is very much in line with Mohr-Coulomb, with the ease at which the system is crushed once normal force is added prompts us to choose no normal force for this system going forward.

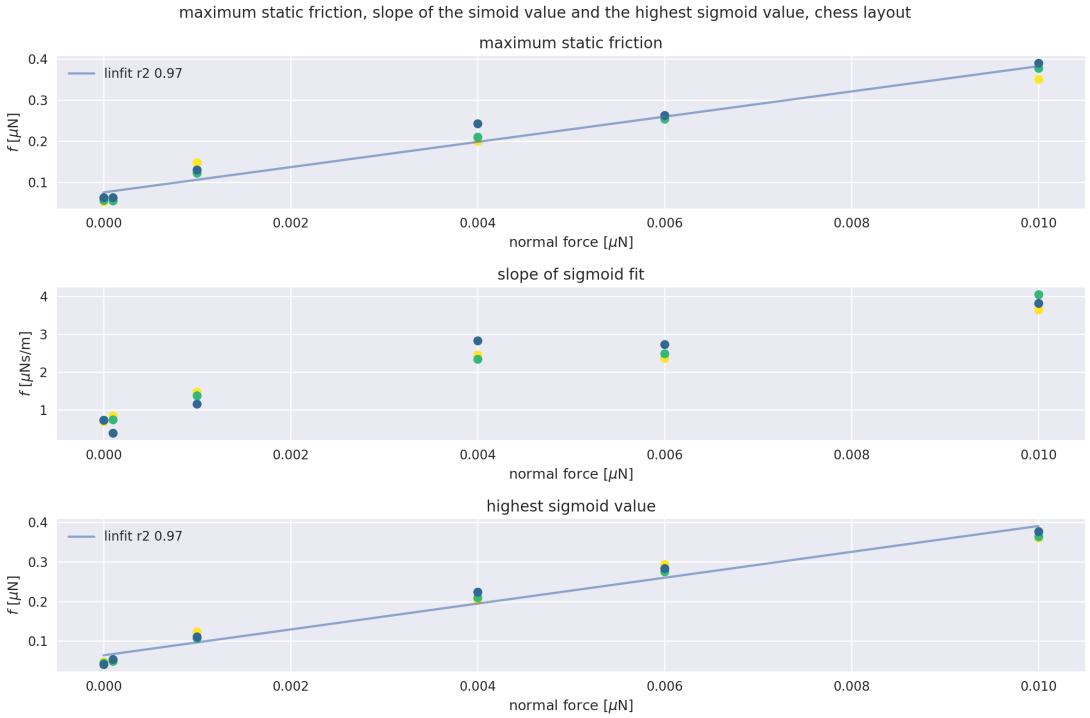

Figure 7.6: Maximum static friction, slope and highest value for sigmoid fit for various normal forces.

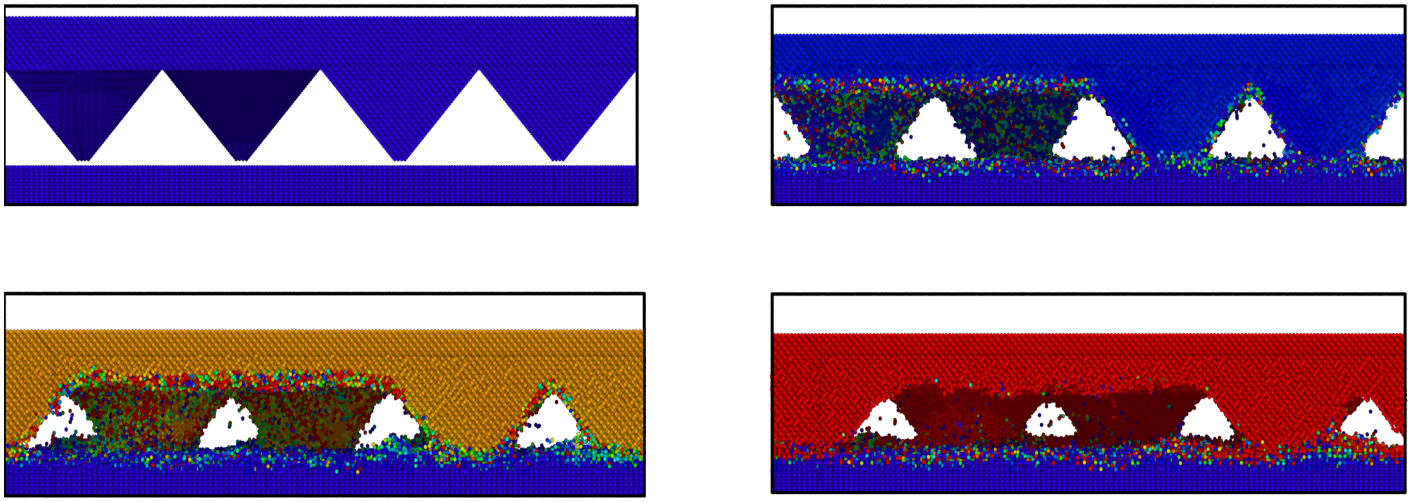

Figure 7.7: A cross section side view of an eight asperity system cut in half in the x-z plane. The top plate is 5 unit cells thick, lower plate is orientation 110, and the simulation is 1 ns relax time followed by 1 ns push with velocity 5 m/s with no normal force. The colour coding is a displacement vector, where bright red is 55 Å. The first frame in the top left is before any time has passed. The second frame top right is after approximatley 1 ns relax time with only normal force applied. The third frame in the bottom right is after 0.5 ns push time, and the last frame is at the end of the simulation. Note the periodic boundaries as the right most asperity creeps into the frame from the left.

Chapter 8

Simulations and Machine Learning Analysis

Having found a systems which seemingly satisfy our molecular-dynamical and computational needs we can proceed with simulations en masse. This chapter starts by describing the two asperity case, then the eight asperity case. This is in order to gauge the effect of asperity configurations, in a less complex system where we can simulate all configurations, before exploring the machine learning results.

8.1 The Two Asperity Case

The two asperity case has as stated previously only ten possible configurations. This allows us to easily simulate and analyze all possible configurations, which for the eight asperity case is not feasible. To what degree the two asperity case is a legitimate analog for the eight asperity case is a point of discussion however. The load bearing properties of the two asperity case is presumably a lot less than for the eight asperity case, however this will not be an issue as we decided on no applied normal force in the previous section. The ability to bear the load once the system is set in motion however is most likely affected.

Figure 8.1 shows all two asperity systems, along with the point at which maximum static friction is decided and the fitted sigmoid curve which estimates the slope. The method of finding maximum static friction by finding the largest friction value in the few picoseconds after we begin to push could prove difficult to use on account of there being a fair amount of noise in certain load curves. In later figures we see the slope of the sigmoid curve, as well as the highest sigmoid value. This is a lot more stable when it comes to fluctuations in friction force. Some do however seem to be skewed quite noticeably if there is a large drop before or after the push begins, such as for the bottom left most curve in figure 8.1.

There is also a fair amount of variations in the load curves during the relax phase.

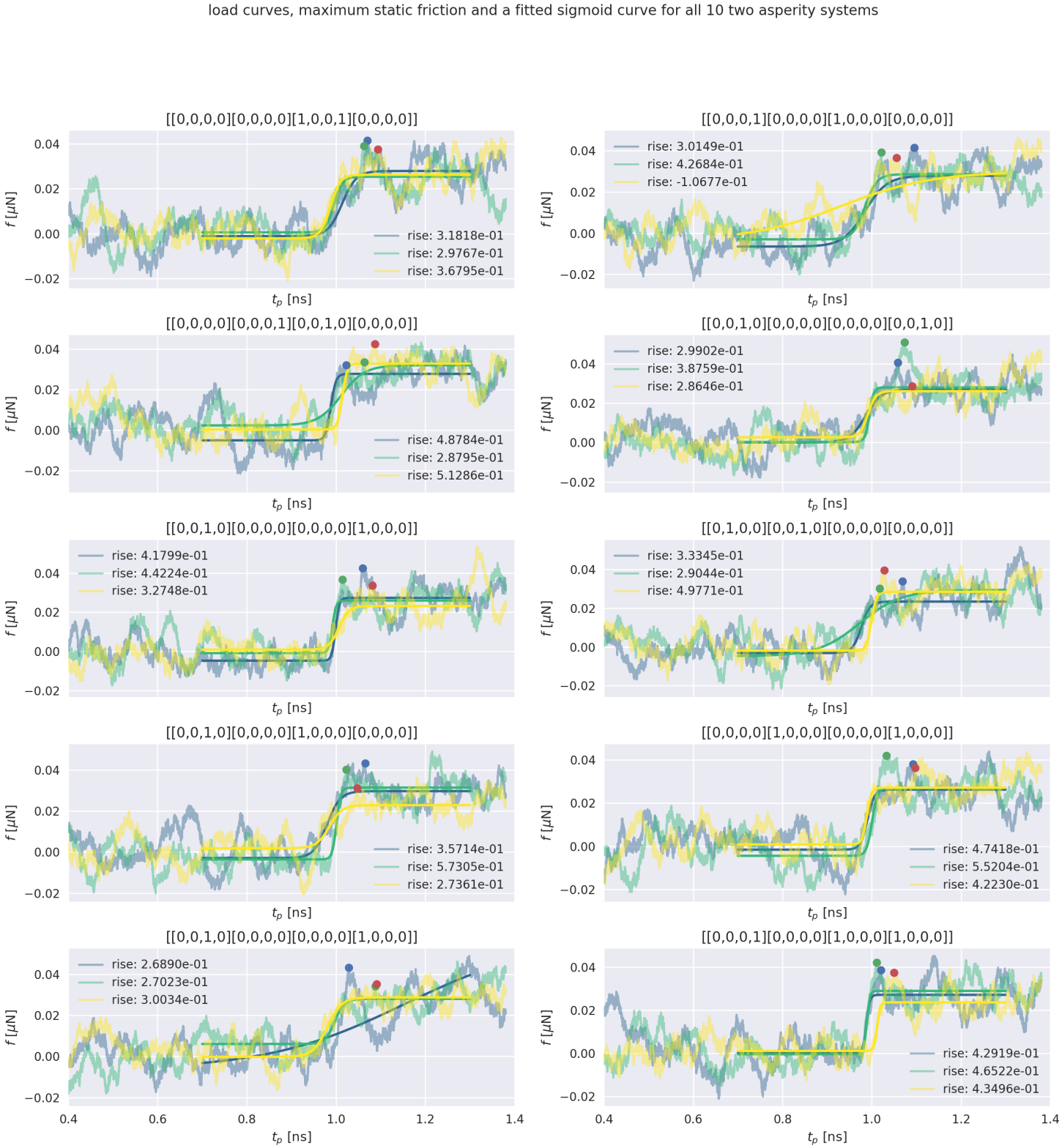

Figure 8.1: Load curves for all 10 two asperity systems. The maximum static friction is visualized as dots on each load curve. This figure also shows the fitted sigmoid curve which dictates the slope of the load curve. Simulated using variables as described in chapter 7

Using the same data as in figure 8.1 we plot the maximum static friction, slope of the fitted sigmoid curve and the highest sigmoid value for all ten configurations. The maximum static friction force values have a large amount of noise, meaning no one configuration nominates itself as the strongest or weakest. The values of the highest fitted sigmoid value are not as effected by noise, we will return to these results later.

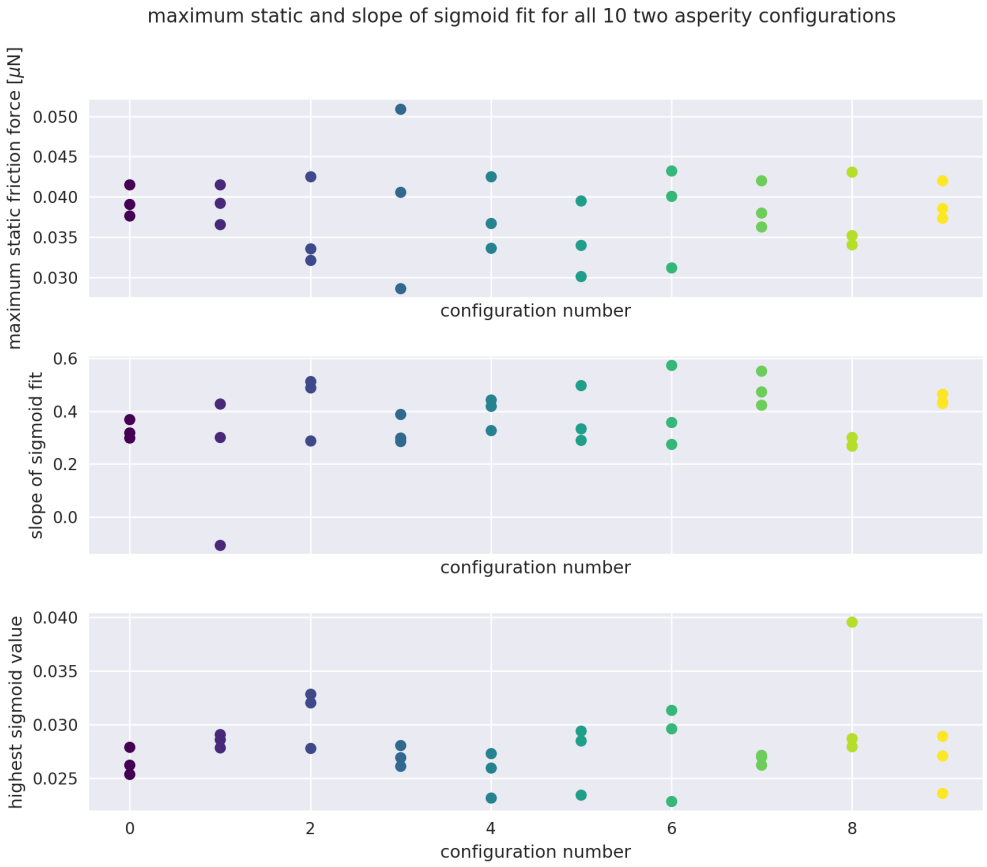

Figure 8.2: The maximum static friction, and the slope of the fitted sigmoid curve and the highest value in the sigmoid curve for all 10 two asperity cases. Data is the same as that of figure 8.1. Simulated using variables as described in chapter 7

8.2 The Eight Asperity Case

Having looked at the two asperity case we can proceed to the eight asperity case in a similar fashion.

The result of excluding symmetries as given in section 6.3 should be noted did not find all possible configurations of systems. For the two asperity case the algorithm consistently only found eight of the ten possible combinations, so a natural conclusion

then is that there are a number of systems missing from the reported 730 or so eight asperity systems. Care has been taken to make sure that there are no duplicates or invalid systems in our dataset. For the two asperity case we could easily verify that no illegitimate configurations were included, but for the eight asperity case a test function was written to confirm this.

Figure 8.3 shows the maximum static friction force, slope of the sigmoid fit and the highest sigmoid value for all the 320 eight asperity simulation data. This data is the input to the machine learning network which we will get back to later.

In section 6.1.1 we explained how we find the parameter describing the distance at which the asperities are placed, thinking this might be correlated with the friction force. Though as we see in figure 8.3 this does not seem to be the case, as neither slope or maximum static friction is affected in the slightest by asperities being distant or close in relation to each other. This could indicate that the placement of asperities does not have a big effect on the resulting slope and maximum static friction.

In figure 8.2 we found that the maximum static friction is averaged around approximately 0.038. Since that is the two asperity case we expect to see about four times as high friction force for the eight asperity case. What we see in figure 8.3 however is that the maximum static friction force is only at 0.057 though with a large variance. The cohesion which binds the two plates when we use no normal force is mostly due to close range effects from the asperity onto the bottom plate. Thus we cannot explain the difference in maximum friction by the two asperity case having fewer asperities to carry the load but the same cohesion.

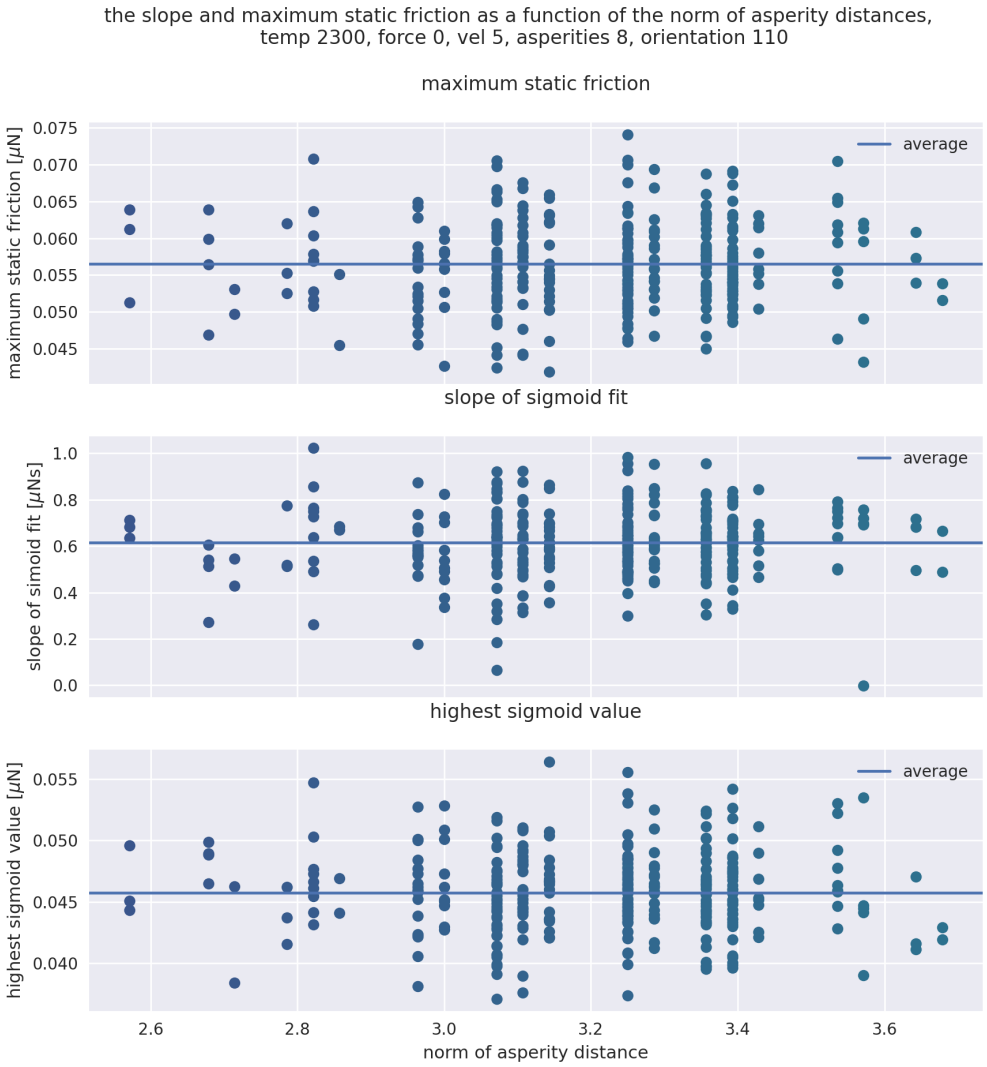

Figure 8.3: The maximum static friction, slope of the fitted sigmoid curve and the highest value of the sigmoid curve for 300 unique eight asperity systems simulated for 1 ns relax and 5 m/s push. Normal force 0, lower plate orientation 110 and 8 asperities. The norm is calculated as explained in section 6.1.1, and maximum static friction and slope is explained in section 6.1.1.

Figure 8.4 and 8.5 shows simulation data for a selected few systems. These systems were thought to be systems which might produce either low or high static friction force, however this does not look to be the case. In fact, figure 8.4 seem to vary greatly with not much to draw from in terms of high or low friction forces. Figure 8.5 also sho that whilst there is some variation in the friction in the systems, all systems have a very large

variance between runs of the same asperity configuration. The method of finding largest sigmoid value which has the benefit of averaging out noise, and which worked well for the two asperity case in figure 8.2 is not showing any trends towards any configuration being stronger or weaker.

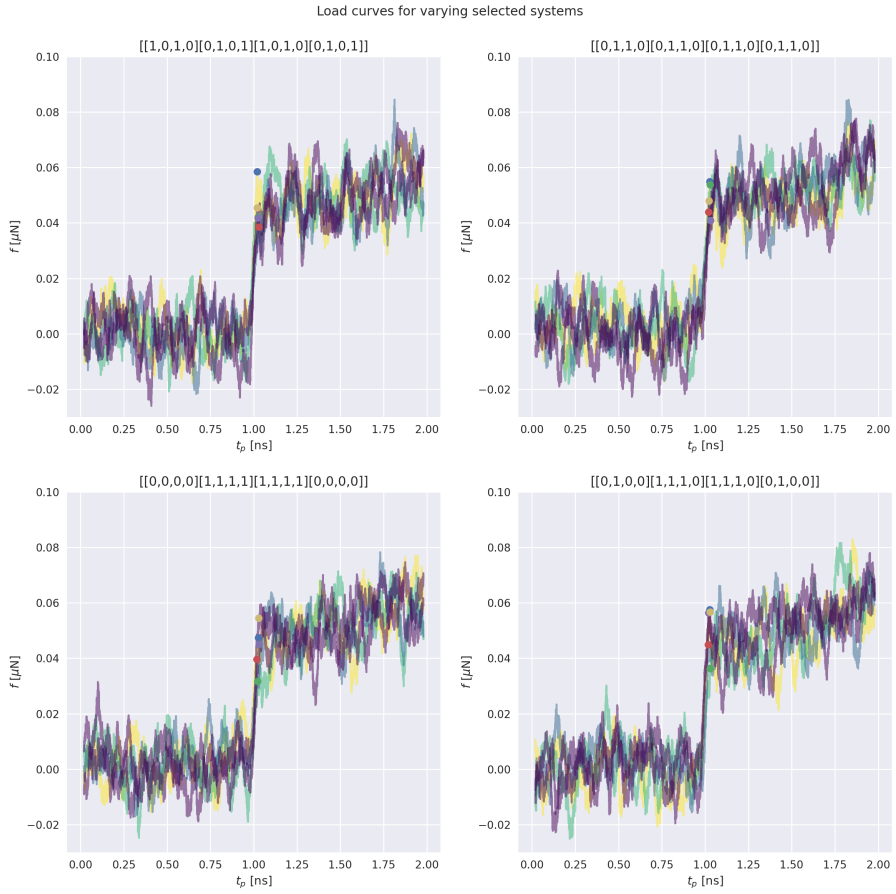

Figure 8.4: Load curves and maximum static friction of four selected systems thought to produce interesting results.

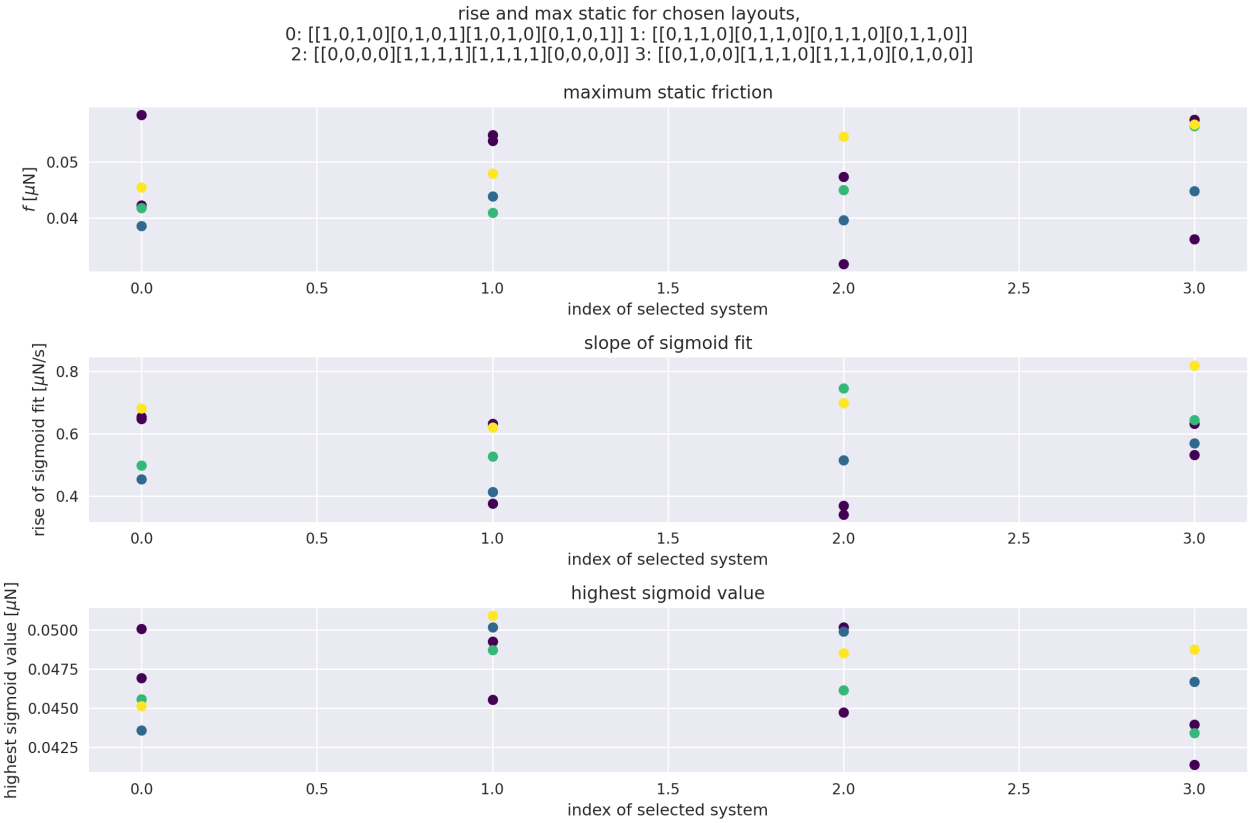

Figure 8.5: The maximum static fritcion, slope of the simoid fit and the highest sigmoid value of four selected systems. This data is the same as in figure 8.4

8.3 Machine Learning

8.3.1 Maximum Static Friction

Figure 8.4 which shows the load curves and maximum static friction for various selected systems indicates that there is no real connection between the asperity configuration and the resulting maximum static friction and the slope. This does not bode well for machine learning. In an effort to confirm the lack of a systematic effect of the asperity configuration on the friction force, we apply machine learning in a massive grid search to explore various networks ability to learn from the dataset. If we try all reasonable hyperparameters without managing to impose upon a network any knowledge what so ever, then we can conclude with certainty that the asperity placement in our system does not affect the maximum static friction.

Since it can be hard to gauge to what degree a network has learned anything, especially when we expect poor results we also apply the same strategy to a completely random dataset. We do this by attributing a random Gaussian maximum static friction

and slope, with the same mean and standard deviation as the actual dataset, to a random but legitimate boolean matrix. We can then gauge to what degree we can train a network on the actual dataset, compared to the known random Gaussian dataset.

The machine learning is applied in a cross evaluation fashion, as described in the method section 4. The method for reducing the cost function can be either MSE ogr R^2, however when having found the best performing model both R^2 and MSE is applied to the test set, which until then has been untouched as to not have any data leak. The training of these models, especially when using R^2 as the model cost function takes many hours, even on an A100 NVIDIA gpu. These are however double precision cards made for simulations, whereas a regular consumer gpu is only single precision.

figure 8.6 shows the MSE and R2 values for various best results after a grid search using dense neural networks and a convolutional neural network. All values seem to have a negative R^2, meaning the models are very poor at describing the variance in the data. The MSE is decent, but it does not make up for the poor performance of the R^2. That being said, it does seem to perform better than the random dataset especially for cnn.

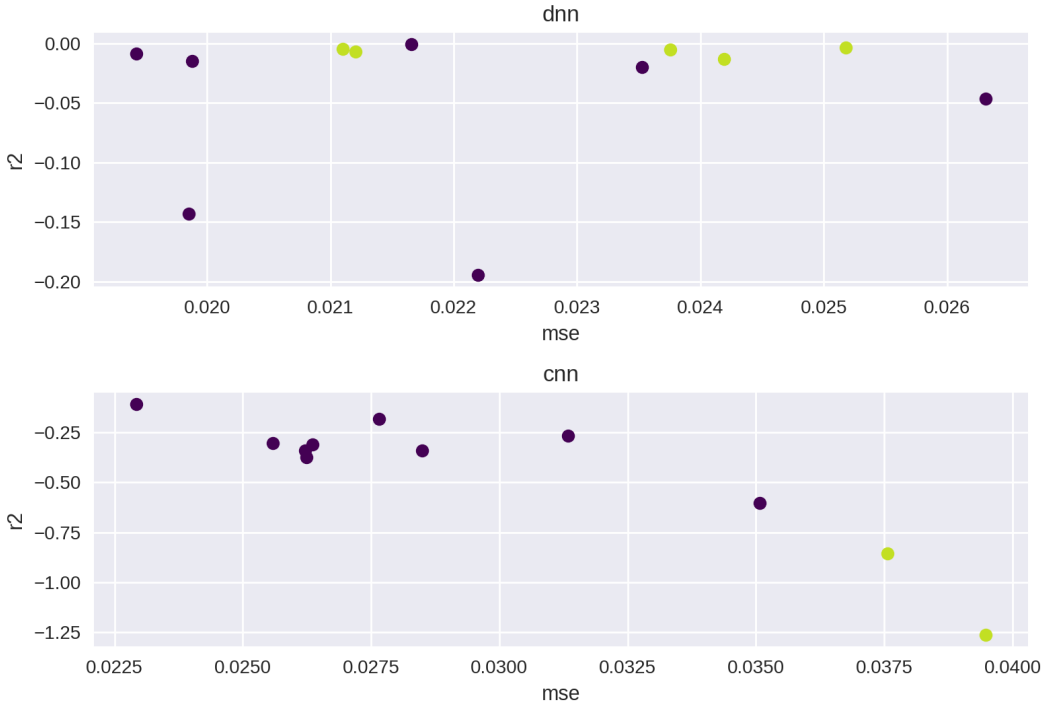

Figure 8.6: Caption

8.3.2 Highest Sigmoid Value

Looking at the load curves in the eight asperity system we see a lot of instabilities and noise in our data. This could make the maximum friction reading vary just as much,

which could limit our machine learning results. To combat this we instead employ the maximum value for the sigmoid curve fit described in the methods chapter 6.1.2, and which have been seen in earlier figures. This will serve to average out noise in the 0.3 nanoseconds after we start to push.

If we look back at figure 8.2 we do indeed see that the highest sigmoid value has both less variance in identical systems, and more of a clear difference between the configurations. This is much more promising for machine learning.

As we do see in figure 8.7 there is a trend of the real data performing better than its random counterpart. The random data is made exactly in the same way as the maximum static random data by drawing from a Gaussian distribution with the same average and deviation as the real data. Although it looks to be performing better, it still has a very bad R^2 score. This means that while it does reduce mean squared error, it does in no way explain the variance in the dataset.

Figure 8.7: Caption

It might be better, but that doesn't make it good. It is also important to remember that if we are to confidently say the asperity placement does not affect the resulting friction we need to be very thorough.

8.3.3 Why Doesn't Configuration Matter?

A reason for why the placement of asperities in relation to each other doesn't seem to have much of an effect could be due to the fact that SiC in general is fairly stiff. For single crystal cubic silicon carbide the young's modulus has been found to be around 137 GPa [1], whereas the elasticity for high carbon steel is 206 GPa [2] meaning both are very

[1] https://www.azom.com/properties.aspx?ArticleID=42
[2] https://www.azom.com/article.aspx?ArticleID=9138

stiff. We argued in the chapter on choosing our system parameters 7 that the smallest top plate didn't resist much due to the asperities not communicating. Although this got better with a thicker top plate, it might still be the case that the top plate is too stiff.

With the improved way of measuring maximum static friction through the method of fitting a sigmoid curve we see in figure 8.2 a clear trend between various systems in the two asperity case. This could be explained by the fact that there are only two asperities. Assuming that we have a stick-slip system wherein one asperity will break leaving the remaining ones to take the load, we might have saturated the eight asperity system with asperities. When one asperity breaks there is always an asperity relatively close by to compensate, so placement isn't as important. With two asperities it is more important where the other asperity is, so that it can compensate since there is no other asperity to do so.

Looking at the strongest system in figure 8.2 we see that it is system number 3. If we look at figure 6.4 we see that this corresponds to the diagonal system with norm equal to 1.414. The fact that this system is a very close one is interesting, however if we look at its rotated sister system number 6, we see in figure 8.2 that this does not perform as well.

8.3.4 Diffusion Creep and healing

The results in the previous sections all share the unexpected effect of showing none of the typical slip-stick effects. We were operating under the assumption that slip-stick effects would be the dominating ones, as in figure 7.1 where we get a very clear break, before the system settles and begins resisting the movement. Consecutive figures do not show this effect, as they have lower plate orientation of (110). The (100) system forms a very solid bond with the bottom plate during relaxation, which results in static friction a whole order of magnitude higher than the (110) system. When this strong bond breaks we get a proper break, which quickly settles and reforms.

Healing could be an explanation, in the sense that as the asperity is skewed by the moving top plate, before it reaches maximum static friction. This could lead free atoms to settle in this narrowing gap in the path of the asperity as it is beneficial to the systems surface energy. This is exacerbated by the relatively high temperature and SiC's inherent surface energy dependence. In the (100) system we have one uniform crystal structure throughout the asperity. This could lead to more atoms joining this crystal formation during the relax phase if it is beneficial energy wise. When the system is close to breaking less atoms are available to heal the asperity, and the break itself is a lot higher energy. These effects combined may result in the lack of slip-slick effects we observe.

Figure 8.8 is a load curve plot made before we made the change to an orientation of (110) on the lower block. The figure might indicate a more disruptive break, as the asperities are pushed past the maximum static friction. If we compare the static friction values to our two asperity (110) orientated system in figure 8.2 we see how the maximum static friction is a whole order of magnitude larger for the (100) system. This is sensical as the (100) system forms a cohesive crystal throughout the asperity and bottom plate.

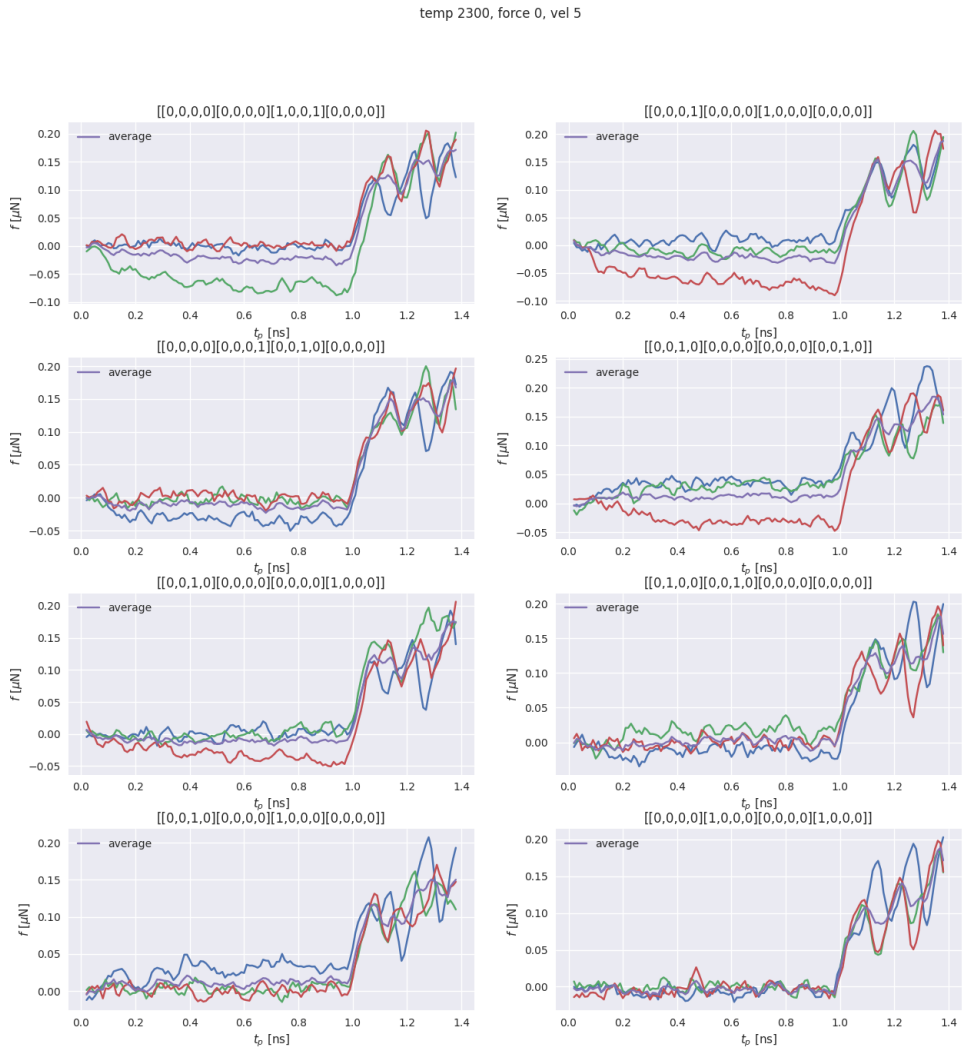

Figure 8.8: Load curves for three identical runs with unique seeds for all eight unique two asperity systems with a lower plate orientation (100).

Chapter 9

Summary and Outlook

A molecular dynamics system was designed, simulated and analyzed with theoretical models in mind. Implicit assumptions on the behaviour of the system were squandered as we discovered a wide range of effects could be the reason for an unexpected result where the behaviour of the asperities could be explained by a diffusion creep theory rather than stick-slip effects. In this chapter we summarize the process and findings, and outline various changes which would bring more clarity to the system.

9.1 Summary

9.1.1 Methods for analyzing the system

Various metrics were considered in describing the system. The norm of asperity distances was made to take periodic boundaries into account, but the variation in asperity norms was shown to have no effect on the distribution of maximum static friction, slope of the sigmoid curve or maximum sigmoid values. The method of extracting maximum static friction also had to be reevaluated as noise in the data seemed to make the data less consistent. The method of fitting a sigmoid curve proved more useful, however this also had the possibility of being disrupted by extreme fluctuations in friction force especially right after the maximum static friction point.

9.1.2 Choosing the system

In selecting our system we considered various physical aspects of our system, with the intention of keeping most simulation conditions constant and only vary the placement of asperities. This was done so that interjunctional effects of a stick-slip system could be studied. We found that the system we chose was insensitive to the spatial configuration of the asperities, leading to a healing or diffusion creep effect. This could have lead to the placement of our asperities not being significant in the resulting static friction.

9.1.3 simulations

A pipeline for creating valid systems based on translational symmetries and mirrored symmetries, initializing and relaxing before pushing was made. Simulations were done on NVIDIA A100 og P100 gpu cards, with relaxation time set to 1 ns to limit variations in the dataset not due to junction interactions. After choosing our system we made a dataset of 320 unique simulations which should be sufficient size for machine learning though the resulting insensitivity of the asperity configurations made this difficult.

9.1.4 Machine Learning

Various methods of machine learning was tested as various versions of our dataset was to be analyzed. Initially a maximum static friction output with the boolean matrix indicating asperity placement was trained using 320 simulations with various hyperparameters. Both cnn and dnn networks were applied, in a grid search fashion, exploring thousands of hyper parameter values combined. The rigidity of the grid search was paramount as our initial working theory was that maximum static friction was not related to the input in any significant way. There had to be no stone unturned in the case of possible parameters of methods before we could conclude that the network could not learn from the data. Related to this we also applied the same vigorous grid search on a fabricated random dataset to get a base line for how well a method we could get on data we knew had unrelated input and output. We found that our method of selecting maximum static friction might be too noisy for any network to learn from, so more grid searches were employed using the alternate highest sigmoid value. A reduction in MSE could be explained by reduced variance in the dataset, but a consistently negative R^2 score confirms the data is too poor to perform any useful analysis. The baseline of random data does show that the methods do perform better than completely independent data.

9.2 Outlook

A number of effects make systems like those which we studied in this thesis interesting and complex. Thus there is a number of variations that can be made to study various aspects of friction. Our high temperature system of SiC is especially interesting due to the surface energy effects and rapid healing and faceting. This however also made our system behave in a way which made it difficult to design a system with certain friction properties. A system in which healing effects is reduced by for example lowering the temperature causing less diffusion could be a way to get a more clear stick-slip effect. This might enable the system to respond more distinctly to various asperity configurations.

As discussed previously we might also have over saturated our relatively small system. A 4×4 grid with eight asperities could have smoothed out the effects we saw with two asperities. Although it might be tempting to always rely on bigger system sizes to find more realistic systems, a reduction in the number of asperities could also prove interesting.

As much of the literature, especially experimentally focus on the real contact area when discussing friction, this could also be an interesting way of describing friction in our

system. The software used for simulating and analyzing such as Ovito has the capabilities of estimating contact area, and this might be used to explore and perhaps even recreate experimental results where contact area is the focus.

As we have theorized in the discussion part of this paper we believe the effects we are seeing are that of diffusion creep, or healing as the asperities are pushed. This could be confirmed if we looked at the loads each individual asperity experiences. If all asperities creep along the bottom plate we could expect the forces to be fairly evenly divided amongst them. This would also be in agreement with our finding that asperity configurations has little effect on the friction.

Bibliography

[1] Sadao Adachi. "Cubic Silicon Carbide (3C-SiC)." In: *Optical Constants of Crystalline and Amorphous Semiconductors: Numerical Data and Graphical Information*. Ed. by Sadao Adachi. Boston, MA: Springer US, 1999, pp. 63–72. ISBN: 978-1-4615-5247-5. DOI: 10.1007/978-1-4615-5247-5_6.

[2] Einat Aharonov and Christopher H. Scholz. "A Physics-Based Rock Friction Constitutive Law: Steady State Friction." In: *Journal of Geophysical Research: Solid Earth* 123.2 (2018). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/2016JB013829, pp. 1591–1614. ISSN: 2169-9356. DOI: 10.1002/2016JB013829. (Visited on 08/18/2022).

[3] G Amontons. "De la resistance cause'e dans les machines." In: *Mem Acedemie R* A (1699), p. 257.

[4] Oded Ben-David and Jay Fineberg. "Static Friction Coefficient Is Not a Material Constant." In: *Physical Review Letters* 106.25 (June 20, 2011). Publisher: American Physical Society, p. 254301. DOI: 10.1103/PhysRevLett.106.254301.

[5] Oded Ben-David, Shmuel M. Rubinstein, and Jay Fineberg. "Slip-stick and the evolution of frictional strength." In: *Nature* 463.7277 (Jan. 2010). Number: 7277 Publisher: Nature Publishing Group, pp. 76–79. ISSN: 1476-4687. DOI: 10.1038/nature08676.

[6] T. Bouhacina et al. "Tribological behavior of a polymer grafted on silanized silica probed with a nanotip." In: *Physical Review B* 56.12 (Sept. 15, 1997). Publisher: American Physical Society, pp. 7694–7703. DOI: 10.1103/PhysRevB.56.7694. (Visited on 08/11/2022).

[7] Herbert B. Callen and Theodore A. Welton. "Irreversibility and Generalized Noise." In: *Physical Review* 83.1 (July 1, 1951). Publisher: American Physical Society, pp. 34–40. DOI: 10.1103/PhysRev.83.34.

[8] David S. Cerutti et al. "Vulnerability in Popular Molecular Dynamics Packages Concerning Langevin and Andersen Dynamics." In: *Journal of chemical theory and computation* 4.10 (Oct. 14, 2008), pp. 1669–1680. ISSN: 1549-9618. DOI: 10.1021/ct8002173.

[9] C. A. Coulomb. "Experiences Destinées à déterminer 'la cohérence des fluides et les lois de leur resistance dans les mouvemens très lents." In: *Mémoires de Mathèmatiques* A (1801), p. 246.

[10] Charles Augustin (1736-1806) Auteur du texte Coulomb. *Théorie des machines simples (Nouv. éd.) / , en ayant égard au frottement de leurs parties et à la roideur des cordages, par C.-A. Coulomb,... Nouvelle édition...* 1821.

[11] Ruslan Davidchack, Richard Handel, and Michael Tretyakov. "Langevin thermostat for rigid body dynamics." In: *The Journal of chemical physics* 130 (July 1, 2009), p. 234101. DOI: 10.1063/1.3149788.

[12] James H. Dieterich and Brian D. Kilgore. "Direct observation of frictional contacts: New insights for state-dependent properties." In: *pure and applied geophysics* 143.1 (Mar. 1, 1994), pp. 283–302. ISSN: 1420-9136. DOI: 10.1007/BF00874332.

[13] Duncan (1928-2020). Dowson and Professional Engineering Publishing. *History of tribology.* Second edition. Section: XXIV, 768 stron : ilustracje kolorowe ; 25 cm. London ; Professional Engineering Publishing, 2006. ISBN: 1-86058-070-X 978-1-86058-070-3.

[14] Xavier Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Journal of Machine Learning Research - Proceedings Track* 9 (Jan. 1, 2010), pp. 249–256.

[15] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 1st. O'Reilly Media, Inc., 2017. 568 pp. ISBN: 978-1-4919-6229-9.

[16] Marthe G. Guren et al. "Molecular dynamics study of confined water in the periclase-brucite system under conditions of reaction-induced fracturing." In: *Geochimica et Cosmochimica Acta* 294 (Feb. 1, 2021), pp. 13–27. ISSN: 0016-7037. DOI: 10.1016/j.gca.2020.11.016. (Visited on 08/15/2022).

[17] Paul Z. Hanakata et al. "Accelerated Search and Design of Stretchable Graphene Kirigami Using Machine Learning." In: *Physical Review Letters* 121.25 (Dec. 20, 2018). Publisher: American Physical Society, p. 255304. DOI: 10.1103/PhysRevLett.121.255304.

[18] Charles R. Harris et al. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020). Number: 7825 Publisher: Nature Publishing Group, pp. 357–362. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2649-2.

[19] Stephen C. Harvey, Robert K.-Z. Tan, and Thomas E. Cheatham III. "The flying ice cube: Velocity rescaling in molecular dynamics leads to violation of energy equipartition." In: *Journal of Computational Chemistry* 19.7 (1998). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291096-987X%28199805%2919%3A7%3C JCC4%3E3.0.CO%3B2-S, pp. 726–740. ISSN: 1096-987X. DOI: 10.1002/(SICI)1096-987X(199805)19:7<726::AID-JCC4>3.0.CO;2-S.

[20] Pierre Hirel. "Atomsk: A tool for manipulating and converting atomic data files." In: *Computer Physics Communications* 197 (Dec. 1, 2015), pp. 212–219. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2015.07.012.

[21] Philippe H. Hünenberger. "Thermostat Algorithms for Molecular Dynamics Simulations." In: *Advanced Computer Simulation: Approaches for Soft Matter Sciences I*. Ed. by Christian Dr. Holm and Kurt Prof. Dr. Kremer. Advances in Polymer Science. Berlin, Heidelberg: Springer, 2005, pp. 105–149. ISBN: 978-3-540-31558-2. DOI: 10.1007/b99427.

[22] *Introduction to Crystallography*. (Visited on 08/11/2022).

[23] J. E. Jones and Sydney Chapman. "On the determination of molecular fields. —II. From the equation of state of a gas." In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 106.738 (Oct. 1924). Publisher: Royal Society, pp. 463–477. DOI: 10.1098/rspa.1924.0082.

[24] Shachar Kaufman, Saharon Rosset, and Claudia Perlich. "Leakage in Data Mining: Formulation, Detection, and Avoidance." In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Vol. 6. Jan. 1, 2011, pp. 556–563. DOI: 10.1145/2020408.2020496.

[25] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980[cs].

[26] Qunyang Li et al. "Frictional ageing from interfacial bonding and the origins of rate and state friction." In: *Nature 2011 480:7376* 480.7376 (Nov. 2011). Publisher: Nature Publishing Group, pp. 233–236. DOI: 10.1038/nature10589.

[27] Yun Liu and Izabela Szlufarska. "Chemical Origins of Frictional Aging." In: *Physical Review Letters* 109.18 (Nov. 2, 2012). Publisher: American Physical Society, p. 186102. DOI: 10.1103/PhysRevLett.109.186102.

[28] E. Maras et al. "Global transition path search for dislocation formation in Ge on Si(001)." In: *Computer Physics Communications* 205 (Aug. 1, 2016), pp. 13–21. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2016.04.001.

[29] Shmuel M. Rubinstein, Gil Cohen, and Jay Fineberg. "Detachment fronts and the onset of dynamic friction." In: *Nature* 430.7003 (Aug. 2004). Number: 7003 Publisher: Nature Publishing Group, pp. 1005–1009. ISSN: 1476-4687. DOI: 10.1038/nature02830.

[30] Andy Ruina. "Slip instability and state variable friction laws." In: *Journal of Geophysical Research: Solid Earth* 88 (B12 1983). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029 pp. 10359–10370. ISSN: 2156-2202. DOI: 10.1029/JB088iB12p10359.

[31] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." In: *Nature* 323.6088 (Oct. 1986). Number: 6088 Publisher: Nature Publishing Group, pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.

[32] Gábor Rutkai et al. "How well does the Lennard-Jones potential represent the thermodynamic properties of noble gases?" In: *Molecular Physics* 115 (Nov. 3, 2016), pp. 1–18. DOI: 10.1080/00268976.2016.1246760.

[33] Yi Sang, Martin Dubé, and Martin Grant. "Thermal Effects on Atomic Friction." In: *Physical Review Letters* 87.17 (Oct. 8, 2001). Publisher: American Physical Society, p. 174301. DOI: 10.1103/PhysRevLett.87.174301.

[34] Alexander Harold Sexton. "Mechanical Characteristics and Design of Faceted Silicon Carbide Nanosystems using Molecular Dynamics and Machine Learning." Master's Thesis. 2021.

[35] Frank H. Stillinger and Thomas A. Weber. "Computer simulation of local order in condensed phases of silicon." In: *Physical Review B* 31.8 (Apr. 15, 1985). Publisher: American Physical Society, pp. 5262–5271. DOI: 10.1103/PhysRevB.31.5262.

[36] Alexander Stukowski. "Visualization and analysis of atomistic simulation data with OVITO–the Open Visualization Tool." In: *Modelling and Simulation in Materials Science and Engineering* 18.1 (Dec. 2009). Publisher: IOP Publishing, p. 015012. ISSN: 0965-0393. DOI: 10.1088/0965-0393/18/1/015012.

[37] Henrik Andersen Sveinsson et al. "Direct Atomic Simulations of Facet Formation and Equilibrium Shapes of SiC Nanoparticles." In: *Crystal Growth & Design* 20.4 (Apr. 1, 2020). Publisher: American Chemical Society, pp. 2147–2152. ISSN: 1528-7483. DOI: 10.1021/acs.cgd.9b00612.

[38] Aidan P. Thompson, Steven J. Plimpton, and William Mattson. "General formulation of pressure and stress tensor for arbitrary many-body interaction potentials under periodic boundary conditions." In: *The Journal of Chemical Physics* 131.15 (Oct. 21, 2009). Publisher: American Institute of Physics, p. 154107. ISSN: 0021-9606. DOI: 10.1063/1.3245303.

[39] Aidan P. Thompson et al. "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales." In: *Computer Physics Communications* 271 (Feb. 1, 2022), p. 108171. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2021.108171.

[40] Jørgen Trømborg et al. "Slow slip and the transition from fast to slow fronts in the rupture of frictional interfaces." In: *Proceedings of the National Academy of Sciences* 111 (June 2, 2014), pp. 8764–8769. DOI: 10.1073/pnas.1321752111.

[41] Majid Vafadar. *A Convolutional Neural Network solution for MNIST dataset.* Feb. 1, 2018.

[42] P Vashishta et al. "Interaction Potential for SiO2: A Molecular-Dynamics Study of Structural Correlations." In: *Physical review. B, Condensed matter* 41 (July 1, 1990), pp. 12197–12209. DOI: 10.1103/PhysRevB.41.12197.

[43] Priya Vashishta et al. “Interaction potential for silicon carbide: A molecular dynamics study of elastic constants and vibrational density of states for crystalline and amorphous silicon carbide.” In: *Journal of Applied Physics* 101.10 (May 15, 2007). Publisher: American Institute of Physics, p. 103515. ISSN: 0021-8979. DOI: 10.1063/1.2724570.

[44] Loup Verlet. “Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules.” In: *Physical Review* 159.1 (July 5, 1967). Publisher: American Physical Society, pp. 98–103. DOI: 10.1103/PhysRev.159.98.