Atomistic Modelling of Creep and Flow in Silica-Water Systems

Anders Johansson

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

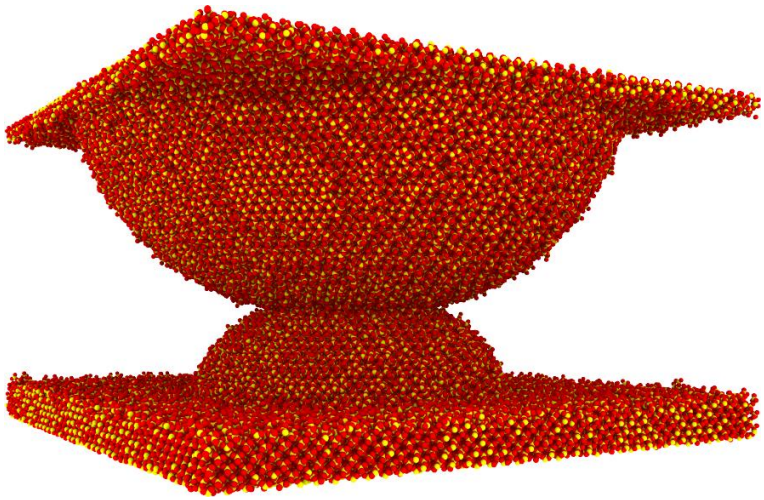
Department of Physics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

Atomistic Modelling of Creep and Flow in Silica-Water Systems

Anders Johansson



© 2019 Anders Johansson

Atomistic Modelling of Creep and Flow in Silica-Water Systems

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

Printed: Reprosentralen, University of Oslo

Contents

Abstract

I use molecular dynamics simulations and a recently developed force field to study two silica-water systems, which are relevant for many geological processes. First, I evaluate the transport properties of the force field by measuring the viscosity of water. The measured value, $\eta = 0.16(1)$ mPa s, is much smaller than the experimental value of 0.89 mPa s. Furthermore, the viscosity does not decrease with increasing temperature. I then study the flow process in a silica-water system subject to an external force. The resulting velocity field qualitatively agrees with results from continuum fluid mechanics, but the velocity magnitudes differ. These results lead to the conclusion that the force field needs to be reparametrised in order to accurately model the transport properties in water and be viable for flow simulations.

My second set of simulations model the creep process in silica and the influence of water by pushing two silica sphere caps together and measuring the system height as a function of time. The results are compared with a theoretical prediction based on a recently published model for steady state friction, and I find good agreement when the silica sphere caps are in vacuum. Filling the vacuum with water has a dramatic effect on the creep process which is not explained by my theoretical prediction. With water present between the sphere caps, the total amount of creep is increased, and the system height no longer plateaus as it does without water. I observe water diffusing into the silica spheres and dissolving the silica structure, hence the deviations from theory appear to be a chemical effect.

Acknowledgements

First and foremost, I must thank my supervisors, Anders Malthe-Sørenssen, Henrik Andersen Sveinsson, Anders Hafreager and Kjetil Thøgersen. You have been incredibly supportive and responsive, and the breadth and depth of your combined knowledge has been vital to the progress of my work on this thesis.

Secondly, I want to thank the Computing in Science Education initiative, now a Norwegian Centre of Excellence, for their effort on integrating computing and programming into the entire bachelor's programmes. When I started my materials science studies at the University of Oslo, I had zero interest in programming, and computers were mainly sources of entertainment and frustration. But after only a few minutes of listening to Hans Petter Langtangen's lectures in introductory scientific programming, I was hooked. Five years later, I have now finished my thesis in computational physics and could not be happier with my choice of programme.

I must thank Professors Rajiv Kalia at USC and Einat Aharonov at HUJI for interesting and helpful discussions which led to the choice of topics for my thesis.

I would also like to thank my fellow students at the Computational Physics group. Without our lively lunch discussions of scientific breakthroughs, Python packages, Vim extensions and everything in between, these last two years would not have been the same.

Finally, I thank my parents for continuous support throughout my studies, allowing me go all-in on courses, teaching and this thesis.

Chapter 1

Introduction

Welcome to my master's thesis! In this thesis, I present the work that I have been doing over the last year or so. There has been joy and suffering, happiness and rage, and good results and bad results — presumably fairly typical for a master's student. I hope you will enjoy reading this thesis, and maybe even learn something useful.

1.1 Topics and motivation

The overall topic of this thesis is molecular dynamics simulations of two different processes in systems consisting of silica and water. I have studied the flow of water in a nanoscale silica system and the creep process in two silica sphere caps being pushed towards each other — with and without water in between.

Molecular dynamics continues to be a relevant and widely applied method because it offers a good compromise between computational efficiency and accuracy. I know of no other simulation techniques which would be able to give me the same atomic resolution in my systems consisting of hundreds of thousands of atoms at a timescale of up to one microsecond. With today's computaional resources, molecular dynamics can be used to study systems which are directly comparable to experiments. Additionally, I simply enjoy doing molecular dynamics simulations. I find it fascinating that such a simple and intuitive method, basically "Newton's second law on atoms", can accurately model complex processes at the atomic level.

Silica systems, with or without water, are interesting for a variety of reasons. The simplest reason is that silicates are found everywhere; for example, it is estimated that the Earth's continental crust contains as much as 66 % silicates [1]. I will study the creep process in silica asperities, which is a step towards understanding the mechanism behind friction. Frictional instabilities in the Earth's crust are earthquakes [2], and a proper understanding of these is obviously important.

Furthermore, silicates are used for a wide variety of applications, such as electronics and

optics, because of its interesting functional and mechanical properties. These properties are known to be affected by water, such as hydrolytic mechanical weakening [3]. It is therefore important to have a molecular dynamics force field which accurately models the coexistence of water and silica.

In this thesis, I will apply a recently developed potential which reproduces the mechanical and chemical properties of silica and water. The first half of my simulations deals with the transport properties of water in the new potential, which have not been thoroughly studied. In particular, I will measure the viscosity of water, perform a flow simulation in a silica geometry and compare the results with continuum simulations using the same viscosity.

The second topic is to study the creep process in two spherical silica asperities being pushed against each other. This is largely based on a recent paper by Aharonov and Scholz [4] in which they develop a physics-based model for steady state friction. A major step in their work is to go from the assumption of thermally activated creep, i.e. the creep rate being proportional to an Arrhenius factor, to an analytical expression for the contact stress as a function of time in contact between two rough surfaces. I will study a simplified version, namely two asperities shaped as sphere caps, and compare my results with those of Aharonov and Scholz. Additionally, I will add water to the void between the asperities and study its effect on the creep process.

Throughout my work on these topics, I will develop whatever additional software I find necessary or useful. These may include extensions of LAMMPS, analysis software and interfaces to results. I will also find a suitable way of automating the simulation pipelines, thus making the results more reproducible. My motivation for these things is simply that I find programming an software development an enjoyable process, and these tools will also make my work less error prone and tedious.

1.2 Goals

The topics described above can be summarised in the following goals:

1. Study the transport properties of the new force field. If the result is positive, the force field can be used to model the nanoscale behaviour of water flow in silica systems.

2. Study the creep process in a single pair of silica asperities, and compare with theory. Study the effect of passivating hydroxy groups and water.

3. Develop necessary and useful software for simplifying and automating molecular dynamics simulations of creep and flow, with a focus on reproducible results and simulation pipelines.

1.3 My contributions

The goals described in the previous section have largely been achived. Unfortunately, the water force field proved to be quite inept for studying the transport properties and flow of water. I did perform one set of flow simulations and compared the results with continuum predictions. Since the underlying model seemed incorrect, I did not pursue this topic further. The creep process was studied for a variety of systems, including passivation and water. Water proved to have a significant chemical effect on the creep mechanism. Additionally, I was able to derive a theoretical estimate of the system height as a function of time with which the molecular dynamics results could be compared. Other than this, my analysis is mainly visual, and the next step would be to quantitatively characterise the creep and deformation with structural analysis.

Here is the detailed list of my scientific contributions:

- Calculating the viscosity of the new water force field.
- Devising and implementing a simulation of water flow in a silica system, performing a parameter sweep over two system sizes and ten driving forces, comparing the results with continuum simulations.
- Deriving a theoretical prediction for the system height as a function of time during thermally activated creep from the results and assumptions of Aharonov and Scholz [4].
- Devising and implementing a simulation of the creep process in an opposing pair of silica asperities. The following systems have been studied:
  - An asymmetric system with silica in vacuum, simulated for 200 ns, with two different barostat damping times.
  - A symmetric system with silica in vacuum, simulated for 1 µs.
  - A symmetric system with silica in vacuum, simulated for 50 ns, for four different temperatures.
  - A symmetric system with passivated silica in vacuum, simulated for 50 ns.
  - A symmetric system with passivated silica in water, simulated for 38 ns.

  In all cases, the system height as a function of time has been compared with the theoretical prediction. Interesting differences have been discussed qualitatively.
- Developing a LAMMPS extension for efficient high-frequency and high-resolution sampling of the velocity field in flow simulations, as well as a simple interface to the results.
- Developing a command line interface to LAMMPS log files for easy inspection of results.
- Developing a simplified, nearly boilerplate-free interface to LAMMPS dump files.
- Devising and implementing makefiles for automated simulation pipelines, including checkpointing of long-running simulations.

1.4 Thesis structure

Part I gives a brief introduction to the theoretical background and methodology of this thesis. The main focus is molecular dynamics, as this is the technique used in the entire thesis. I cover both motivation, basic principles, theoretical considerations and implementation aspects. I also choose to spend some time discussing the main tools, such as LAMMPS. Another large chunk of part I is devoted to an overview of the computational tools that I have developed myself. Part I ends with a short review of related publications.

Part II is the main chunk of the thesis, and contains all the results that I have produced. First, I present the result of two verification simulations, namely calculating the viscosity of water and creating amorphous silica. I then present my comparison of molecular dynamics and continuum fluid mechanics results for flow in a silica-water system. Finally, I move on to the study of creep processes and discuss my results. I conclude part II, and also my thesis, with a summary and discussion, followed by a conclusion and a few perspectives on improvements and extensions of my work, as well as possible long-term goals.

At the end of this thesis, in part III, I have added some information on topics which may be useful for reference and for future work, but which proved irrelevant for the particular simulations that I ended up doing. Additionally, there is a list of figures and a list of references.

1.5 Ethical aspects

The physical systems studied in this thesis are not the most ethically dubious. In the first part of this thesis, I attempt to validate a force field by measuring the viscosity of water and the velocity field in a silica-water system, and find no results that can be used for mischief. The creep system studied should be similarly unproblematic, although I am studying a physical effect rather than methodology in this part. Looking at the big picture, my creep simulations are part of the effort to get a solid understanding of the underlying mechanism of friction. Achieving this goal should mainly have positive consequences, as an understanding of friction may allow materials to be designed with specific frictional properties, and this may help reduce waste heat in production processes etc.

On the other hand, there are a few ethically problematic sides to my methodology. First and foremost, the molecular dynamics force field that I have used throughout this thesis is still unpublished. Without the force field, the majority of my results are completely unreproducible. This is clearly an ethical conundrum, but it is something that I have simply had to accept. Fortunately, there is already a LAMMPS implementation of the potential, so that it should instantly be available and usable when it is published. Additionally, one can only hope that current efforts towards generalised force fields using machine learning succeed and put an end to the problem of force field availability.

Computational studies are often easier to reproduce than experimental studies, since they

mainly require a computer and not any advanced experimental equipment. This thesis, however, is harder to reproduce because my simulations have been computationally heavy. In total, I estimate that I have used somewhere between one and two million CPU hour equivalents (including GPU computations). While a significant portion has been wasted on trial and error, serious resources are still needed to reproduce the final results. To mitigate this problem, I have published the raw data on Zenodo, so that my analysis procedure can be verified. Additionally, publishing the data enables others to extend the analysis without having to redo the simulations.

As an extra measure, I have fully automated the simulation pipelines using makefiles (see section 3.4 on page 32). With the required software installed and sufficient computing power, anyone should be able to repeat the full series of scripts and simulations by issuing a simple make command with the desired target. The checkpointing procedure described in section 3.4.3 ensures that one does not need ultra-reliable hardware to run the longest simulations.

Finally, I have stored all code at GitHub, and version control has been actively used throughout the development process. The GitHub repositories have been made public and given a permissive license to allow others to use and extend the code.

All in all, I hope that I am not a crook. The physical systems studied in this thesis are ethically unproblematic. There are issues with using an unpublished force field and doing simulations which are not trivially reproducible, but I have taken measures to mitigate the problems by openly publishing all simulation output, as well as source code, and having an automated workflow.

Part I

Theory and methods

Chapter 2

Molecular dynamics

Molecular dynamics is a common simulation technique for investigating physical processes and materials properties with atomic or molecular resolution. In this chapter, I give both a motivation for and a practical introduction to molecular dynamics, as well as some technical aspects. Additionally, I will cover the main software that I have chosen to use in my simulations.

2.1 Motivation

While the various quantum theories are our most accurate descriptions of nature, they are universally viable for all kinds of physical problems. Unfortunately, the most accurate theories are also the most complex and therefore the most computationally heavy. As an example, quantum chromodynamics is usually only viable for a few elementary particles on a timescale up to femtoseconds. This severely limits what phenomena can be modelled.

It is here that coarse-graining enters the picture. Quantum field theories can, in the most common limits, be approximated with non-relativistic quantum mechanics. Different approximations to the Schrödinger equation, such as variational Monte Carlo, enable simulations of systems up to hundreds or thousands of electrons.

The next step is to assume that atoms behave classically, i.e. according to Newton's laws of motion. This is how classical molecular dynamics arises. Molecular dynamics is the main simulation method applied in this thesis. In fact, the main theme of this thesis is to compare atomistic simulations using molecular dynamics with the predictions of a coarser step in the coarse-graining hierarchy, namely continuum theories for fluid flow and creep.

In molecular dynamics, the computational task is to solve Newton's equations of motions for all particles simultaneously. While enormously less complex than solving the quantum many-body problem, this is still a computationally heavy task which continues to push the limits of our largest computing infrastructures.

The first molecular dynamics simulations were done around 1960, with the advent of the first scientific computers. Since the pioneering work by Rahman [5], where 864 argon atoms were simulated with the Lennard-Jones potential and each timestep took 40 seconds on a state-of-the-art computer, molecular dynamics have been applied to larger and larger systems and time scales. At present, systems with millions, billions or even trillions of atoms are routinely simulated for several nanoseconds.

Over the last decade or so, we have reached the intersection of a continuous increase in computing power, allowing larger systems to be studied computationally, with the development of better and better experimental techniques, allowing smaller systems to be studied experimentally. In this thesis, I simulate systems on the order of tens of nanometres over timescales of up to 1 microsecond — a length scale which can easily be studied in an electron microscope and a time scale sufficiently long to see macroscopic processes such as creep.

2.2 Potentials and forces

As stated above, molecular dynamics is about applying Newton's laws of motion to atoms. While Newton's second law is assumed to be the same for atoms as all other particles, the question remains of what the forces on the atoms should be. Ab-initio molecular dynamics solves this by calculating what the forces *should* be directly from some quantum mechanical method, usually density functional theory. This, however, is computationally expensive and should therefore only be used whenever one does not have a better idea of what the forces should be.

This thesis focuses on a system for which a good model for the forces is known explicitly. Concretely, there exists a function $U\left(\{\vec{r}_j\}_1^N\right)$ which approximates the potential energy as a function of the positions of all the N particles. The force on particle number i is given by the derivative of the potential energy with respect to the particle's position,

$$\vec{F}_i = -\nabla_i U\left(\{\vec{r}_j\}_1^N\right), \tag{2.1}$$

giving the equations of motion as a set of N coupled differential equations,

$$\ddot{\vec{r}}_i = -\frac{1}{m_i}\nabla_i U\left(\{\vec{r}_j\}_1^N\right). \tag{2.2}$$

In systems where the interatomic forces are spherically symmetric, such as in the case of noble gases, the potential energy is a sum of pairwise contributions,

$$U\left(\{\vec{r}_j\}_1^N\right) = \sum_{i<j} U_{ij}(\vec{r}_i, \vec{r}_j) = \sum_{i<j} U_{ij}(\|\vec{r}_i - \vec{r}_j\|) = \sum_{i<j} U_{ij}(r_{ij}). \tag{2.3}$$

The equation of motion for atom i is then

$$\ddot{\vec{r}}_i = -\frac{1}{m_i}\sum_{j\neq i} \nabla_i U_{ij}. \tag{2.4}$$

The Lennard-Jones potential is a pair-wise potential which accurately models the interactions in e.g. argon. Its potential energy term for each pair of atoms is

$$U_{ij}(r_{ij}) = 4\varepsilon\left(\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right). \tag{2.5}$$

Repulsive forces between atoms too close to each other are represented by the first term, while the attractive London dispersion forces are modelled by the r_{ij}^{-6} term.

Most real systems do not have spherically symmetric interactions, as any chemical bond with a non-vanishing covalent character will have an angular dependence due to the structure of orbitals in an atom. Consequently, bond angle dependent terms must be added to the potential. This is usually achieved by extending the potential energy with one term for each triplet of particles, i.e.

$$U(\{\vec{r}_j\}) = \sum_{i<j} U_{ij} + \sum_{i\neq j\neq k} U_{ijk}, \tag{2.6}$$

where U_{ijk} depends on both the magnitudes of $\vec{r}_j - \vec{r}_i$ and $\vec{r}_k - \vec{r}_i$ *and* the angle between them.

A common example of such an extension of the potential is seen in the Stillinger-Weber potential [6], which is a reasonable model for the covalent interactions in silicon. In addition to a two-particle term similar to that of the Lennard-Jones potential, the Stillinger-Weber potential has a three-body term

$$U_{ijk}(r_{ij}, r_{ik}, \theta_{ijk}) = \exp\left(\frac{1}{r_{ij}/\sigma - a}\right)\exp\left(\frac{1}{r_{ik}/\sigma - a}\right)\left(\cos\theta_{ijk} - \cos\theta_0\right)^2, \tag{2.7}$$

where the cosine term serves to increase the potential energy whenever the angle θ_{ijk} deviates from the preferred angle, $\pm\theta_0$. The exponentials work as a cut-off (more on that later).

While one could, in theory, continue adding terms to the potential which depend on every quadruplet of particles, few molecular dynamics potentials go beyond three-body terms.

In this thesis, a more complicated, multi-element compound is studied using a fairly simple three-body potential, namely silica with the Vashishta potential [7]. In its original form (1990), the Vashishta potential has a three-body term identical to that of the Stillinger-Weber potential, but the two-body term is

$$U_{ij} = \frac{H_{ij}}{r_{ij}^{\eta_{ij}}} + \frac{Z_i Z_j}{r_{ij}} - \frac{\frac{1}{2}\left(\alpha_i Z_j^2 + \alpha_j Z_i^2\right)}{r_{ij}^4} e^{-r_{ij}/r_{4s}}. \tag{2.8}$$

Steric repulsion is represented by the first term, similarly to the first Lennard-Jones term, with η_{ij} equal to 11, 9 or 7 depending on the atom types. The second term is a simple Coulomb term, while the last term models charge-dipole interaction as an effect of polarisation.

Over the years, the Vashishta potential has been modified and extended to also work for other compounds, such as silicon carbide [8] and indium phosphide [9]. The version implemented in LAMMPS has an exponential screening of the Coulomb interaction, as added in 1994 [10]. Additionally, a van der Waals (r_{ij}^{-6}) term has been added, but it is set to zero for silica.

The latest extension of the Vashishta potential [11] enables it to model the coexistence of silica and water. As it turns out, the Vashishta potential as described above is perfectly capable of modelling water. The question is then how to deal with the interface between water and silica. In particular, oxygea-oxygen bonds behave differently depending on whether they are a part of water, silica or the interface.

Traditionally, molecular dynamics simulations of mixed systems have been carried out using some sort of mixing rule for the parameters in the potential [12]. For example, the Lorentz-Berthelot rules simply use an arithmetic mean of the distance parameters, such as σ in the Lennard-Jones potential, and a geometric mean of the energy parameters, such as ε in the Lennard-Jones potential. While these combining rules have accurately predicted many properties of mixtures, they can not be expected to be perfect models or fully describe the interface chemistry.

Instead of a parameter combining rule, the silica-water Vashishta potential combines the silica and water behaviours at the potential level. Mathematically, the oxygen bond energy is a linear combination of the water and silica energies, with the weights calculated as the fraction of silicon or hydrogen neighbours. With n_i^{Si} and n_i^{H} denoting the number of silicon or hydrogen neighbours of atom number i, the potential energy of a pair of oxygen atoms is

$$U_{ij} = U_{ij}^{\mathrm{SiO_2}}(1 - f_{ij}) + U_{ij}^{\mathrm{H_2O}} f_{ij}, \tag{2.9}$$

where the weight is defined as

$$f_{ij} = \frac{n_i^{\mathrm{H}} + n_j^{\mathrm{H}}}{n_i^{\mathrm{H}} + n_j^{\mathrm{H}} + n_i^{\mathrm{Si}} + n_j^{\mathrm{Si}}}. \tag{2.10}$$

In order to keep the potential conservative, ${n_i}^{\mathrm{H}}$ and friends are modified in such a way that neighbouring atoms smoothly go from being a neighbour to not being a neighbour as the distance increases.

2.3 Time integration

Newton's second law and the equations of motion are a set of coupled, second order, ordinary differential equations. Such equations can be solved by a variety of numerical schemes. Molecular dynamics applications usually employ the Velocity Verlet algorithm. This algorithm is one order more accurate than the Forward Euler scheme, without requiring any additional force calculations. It therefore provides a good balance between numerical accuracy and computational efficiency.

The Velocity Verlet scheme is derived with a second order Taylor expansion of both the positions and the velocities. Positions at the next timestep, $\vec{r}_i^{\,n+1}$, are found from the positions at the current timestep, $\vec{r}_i^{\,n}$, using

$$\vec{r}_i^{\,n+1} = \vec{r}_i^{\,n} + \vec{v}_i^{\,n}\Delta t + \tfrac{1}{2}\vec{a}_i^{\,n}\Delta t^2, \tag{2.11}$$

where the acceleration is calculated from Newton's second law. While there is no way of directly calculating the second derivative of the velocity, namely the derivative of the acceleration, it can be approximated using

$$\dot{\vec{a}}_i^{\,n} \approx \frac{\vec{a}_i^{\,n+1} - \vec{a}_i^{\,n}}{\Delta t}. \tag{2.12}$$

The second order Taylor expansion of the velocity is then

$$\vec{v}_i^{\,n+1} = \vec{v}_i^{\,n} + \vec{a}_i^{\,n}\Delta t + \tfrac{1}{2}\dot{\vec{a}}_i^{\,n}\Delta t^2 \tag{2.13}$$

$$= \vec{v}_i^{\,n} + \tfrac{1}{2}\left(\vec{a}_i^{\,n} + \vec{a}_i^{\,n+1}\right)\Delta t, \tag{2.14}$$

where $\vec{a}_i^{\,n+1}$ can be calculated once the position has been updated according to equation (2.11) because the forces only depend on positions, not velocities.

A small rewriting gives the following algorithm, given the positions, velocities and accelerations at one timestep:

1. Half-update velocities:
$$\vec{v}_i = \vec{v}_i + \tfrac{1}{2}\vec{a}_i\Delta t \tag{2.15}$$
2. Update positions:
$$\vec{r}_i = \vec{r}_i + \vec{v}_i\Delta t \tag{2.16}$$
3. Calculate new accelerations:
$$\vec{a}_i = -\frac{1}{m}\nabla_i U\left(\{\vec{r}_j\}_1^N\right) \tag{2.17}$$
4. Complete velocity update:
$$\vec{v}_i = \vec{v}_i + \tfrac{1}{2}\vec{a}_i\Delta t \tag{2.18}$$

Thus one avoids using more than one array for each quantity for each atom, and, more importantly, calculating the forces more than once per timestep. It is important to note that each step in the algorithm must be carried out for all atoms before moving on to the next step.

2.4 Computational considerations

2.4.1 Interaction range

By far the most computationally expensive part of molecular dynamics simulations is the calculation of forces. If one were to blindly implement the potentials as described

in section 2.2 on page 10, molecular dynamics simulations would be impossible for all but the smallest systems. As an illuminating example, consider a typical system with one million atoms. A two-particle potential such as the Lennard-Jones potential would require a loop over one trillion pairs of atoms, which would take considerable time on most modern computers. Then consider a three-particle potential — this would require a loop over one quintillion triplets of atoms!

Fortunately, there is a solution to the seemingly bad scaling of force calculations. The crucial point is the functional form of the various potentials presented in section 2.2. They all decay rapidly as the interatomic distances increase. As such, it is a reasonable approximation to say that the potential is zero beyond a certain distance, or, in other words, that the interatomic forces have a finite range.

With r_c denoting the maximum interaction distance, each atom has approximately $\rho\frac{4}{3}\pi r_c^3$ neighbours. This is a quantity which does not depend on the size of the system, and so the scaling of molecular dynamics is improved from $\mathcal{O}(N^2)$ or even $\mathcal{O}(N^3)$ to simply $\mathcal{O}(N)$. Suddenly, it is possible to simulate systems with millions, billions and even trillions [13] of atoms!

Of course, this requires an algorithm for force calculations which actually takes advantage of the finite interaction range. This is typically achieved through a combination of spatial decomposition and neighbour lists. Spatial decomposition means dividing space into regions of such a size that atoms in one region can only interact with atoms either in the same region or in a neighbouring region. Then, a list of the neighbours of each atom is created by looping over all the atoms in the regions within interaction range. With these lists already constructed, it is simple to calculate forces by looping over the list of atoms with which each atom interacts.

2.4.2 Boundary conditions

Molecular dynamics simulations are, with our current computing capabilities, usually limited to nanoscale systems. Yet we wish to simulate systems where many of the material properties are the same as in bulk. For example, one of my verification simulations is to measure the viscosity of my water model. Since viscosity is a macroscopic property of a liquid, I want the water to have mostly the same properties as bulk water. In other words, I do not want to simulate nanoparticles of water, as nanoparticles often behave completely differently than bulk materials.

Instead, I want to do a molecular scale simulation of a bulk system. This requires the atoms to not notice the very finite extent of the system. A solution to this problem is to apply periodic boundary conditions, i.e. to pretend that the system is replicated infinitely many times in each direction. In practice, this is handled by the minimum image convention, which in a one-dimensional system of length L redefines the distance $\Delta x = x_i - x_j$ between two atoms as

$$\Delta x \to \Delta x - \text{round}(\Delta x / L) \cdot L. \tag{2.19}$$

This formula ensures that an atom interacts with the closest periodic image of its neigh-

bours. Similarly, the atomic positions are kept within the simulation box with

$$x_i \to x_i - \text{floor}(x_i/L) \cdot L. \tag{2.20}$$

2.5 Statistical mechanics perspective

An important goal of statistical mechanics is to estimate macroscopic quantities in systems where obtaining an excact, microscopic description is impossible. Instead of finding the positions and momenta of e.g. 10^{23} particles and calculating macroscopic quantities from these, the microstate of the system (the configuration of positions and momenta) is seen as a stochastic variable. The set of all possible microstates is called *phase space*. Some quantity A is then estimated as the expectation value

$$\langle A \rangle = \int \mathrm{d}\omega \, \rho(p,q) A, \tag{2.21}$$

where $\mathrm{d}\omega$ is a volume element in phase space, typically proportional to $\mathrm{d}^{3N}q \, \mathrm{d}^{3N}p$ where q and p are positions and momenta and N is the number of particles, while $\rho(p,q)$ is the probability of finding the system in a microstate inside this volume element. A famous example is the canonical ensemble with the Boltzmann distribution,

$$\langle A \rangle = \frac{1}{Z} \int \frac{\mathrm{d}^{3N}q \, \mathrm{d}^{3N}p}{h^N} A \mathrm{e}^{-H(p,q)/k_\mathrm{B}T}, \tag{2.22}$$

with the probability of finding the system in a microstate being inversely proportional to the exponential of the microstate's energy.

There are two ways of viewing the role of molecular dynamics as a way of calculating the same properties. First, molecular dynamics can be seen as an attempt to use raw computational power to circumvent the underlying problem of not being able to explicitly calculate the positions and momenta of all particles. Newton's equations of motion are solved directly for all atoms, giving the complete time evolution of the system and the ability to calculate macroscopic quantities directly. Statistical mechanics can therefore be seen as an approximation of molecular dynamics. This follows the ideas of Gibbs, Boltzmann and the other pioneers who developed statistical mechanics from a classical point of view, with no knowledge of the quantum nature of the world.

Alternatively, one may view molecular dynamics as an approximation to statistical mechanics and equation (2.21). In statistical mechanics, expectation values are calculated through phase space integrals, i.e. a weighted sum over all the possible microstates. The same expectation values can be estimated in molecular dynamics simulations of a system in equilibrium under the assumption of equivalence between a phase space average and a time average,

$$\langle A \rangle = \int \mathrm{d}\omega \, \rho A = \lim_{T\to\infty} \frac{1}{T} \int_0^T \mathrm{d}t \, A. \tag{2.23}$$

This is called the *ergodic hypothesis* and says that a system in equilibrium will visit different regions of phase space according to the phase space density, ρ. In the limit of infinite simulation time, the time average becomes identical to the phase space average.

With modern physics's knowledge of quantum mechanics and how the world is inherently nondeterministic, viewing equilibrium molecular dynamics as an approximation to the phase space integral for calculating expectation values is the "truest" interpretation, although the classical perspective remains the most intuitive. In this thesis, I have mainly studied dynamic processes, i.e. properties which change with time, allowing me to stick to the more intuitive, classical approach.

2.6 Ensembles

So far, I have described molecular dynamics as time integration of Newton's second law. Since the forces come from a potential, they are conservative, and the total energy is conserved. Furthermore, the number of particles and the system volume have also been constant. This constitutes the microcanonical ensemble. While this is an important ensemble, ensembles which allow exchanging heat and/or particles and expansion are often more realistic and physically relevant than isolated systems. I have not needed to allow for an exchange of particles, but a constant temperature and/or pressure has been essential — especially for simulating creep processes.

Constant temperature and pressure are maintained via thermostats and barostats. There are many examples, with varying properties. Often, it is not sufficient to keep the temperature and/or pressure constant; in addition, the implementation should reproduce the other properties of the corresponding canonical or isothermic-isobaric ensemble. Finally, the desired ensembles should be reproduced without unrealistically altering microscopic properties such as structure and diffusivity.

2.6.1 Thermostats

Velocity scaling

The easiest method for simulating a system with constant temperature is to rescale the velocities whenever the measured temperature, T, deviates from the desired temperature, T_0. If the velocities are rescaled by the factor $\lambda = \sqrt{T_0/T}$, the velocity is kept completely constant. Alternatively, one can use a scaling factor with a characteristic time τ,

$$\lambda = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T}{T_0} - 1\right)}. \tag{2.24}$$

where τ roughly gives the time it takes to obtain the desired temperature, T_0. This is called the Berendsen thermostat [14], and allows for some temperature fluctuations. Unfortunately, this simple approach does not accurately represent a canonical ensemble.

Stochastic thermostats

Usually, minimisation of the Helmholtz free energy in a canonical ensemble is derived by studying a system with constant temperature that is part of a larger, microcanonical (isolated) system. The rest of the system acts as a heat bath with which the particles in the canonical part can interact in order to keep the temperature constant. Stochastic thermostats aim to model this.

The Andersen thermostat [15] is the simplest stochastic thermostat. At each time step, each atom draws a random number between 0 and 1. If the number if greater than $\nu\Delta t$, where ν is a frequency giving the thermostat strength, the atom's velocity is randomly reassigned according to the Maxwell-Boltzmann distribution with the desired temperature. This simple scheme accurately reproduces the canonical ensemble, although it alters transport properties such as diffusivity and viscosity.

A slightly more advanced alternative is the Langevin thermostat. The basic principle is to simply add the drag and stochastic forces from the Langevin equation [16] of Brownian motion, i.e.

$$m\vec{a} = \vec{F} - \alpha\vec{v} + \vec{R}, \tag{2.25}$$

where α and the strength of $\vec{R}$ are related through the assumption of thermal equilibrium and constant temperature. Thus the canonical ensemble is automatically reproduced, but transport properties etc. are still affected.

Transport properties are, however, conserved when using a thermostat based on dissipative particle dynamics [17]. Similarly to the Langevin thermostat, this adds drag and random forces, but the forces are calculated per *pair* of atoms and antisymmetric so that momentum is conserved. The only disadvantage is the non-deterministic, time-irreversible and non-smooth nature of the trajectories due to the stochastic approach.

Extended Hamiltonian thermostats

An issue with the stochastic thermostats is that the resulting trajectories are neither deterministic nor time-reversible, and their velocities are discontinuous. Extended Hamiltonian thermostats work by a different principle. Instead of directly altering the equations of motion, they add terms to the Hamiltonian from which the equations of motion are derived. With wisely chosen additions, this can lead to accurate simulations of the canonical ensemble.

The most important thermostat with the extended Hamiltonian approach is the Nosé-Hoover thermostat, as it is one the most widely used thermostats. In the original formulation by Nosé [18], an additional degree of freedom, s, is added to the system, together with two corresponding terms in the Hamiltonian,

$$H = \sum_i \frac{\vec{p}_i^{\,2}}{2m_i s^2} + U + \frac{p_s^2}{2Q} + g k_\mathrm{B} T \ln s. \tag{2.26}$$

Nosé showed that conservation of this Hamiltonian, i.e. a microcanonical distribution of the full set of degrees of freedom, is equivalent to a canonical distribution of positions and scaled momenta ($\vec{p}_i/s$).

Hoover [19] derived a much more intuitive formulation. It turns out that the extended Hamiltonian above is equivalent to adding a time-dependent drag term to the equations of motion,

$$\vec{a}_i = \tfrac{1}{m_i}\vec{F}_i - \xi\vec{v}_i, \tag{2.27}$$

where the change in the drag coefficient is proportional to the difference between the desired temperature and the current, instantaneous temperature,

$$\dot{\xi} = \frac{1}{Q}\left(\sum_i m_i\vec{v}_i^2 - fk_\mathrm{B}T_0\right). \tag{2.28}$$

Q is a damping factor, while f is the number of quadratic degrees of freedom, as in the equipartition theorem.

2.6.2 Barostats

The Nosé-Hoover thermostat is straightforwardly extended to also include a barostat [19]. By making the derivative of the relative change in volume proportional to the difference between the measured and desired pressure,

$$\frac{\dot{V}}{V} = D\dot{\varepsilon} \quad \text{with} \quad \ddot{\varepsilon} = \frac{V}{\tau^2 k_\mathrm{B}T}(P - P_0), \tag{2.29}$$

the pressure is kept approximately constant. This adds another drag term, so that

$$\vec{a}_i = \tfrac{1}{m_i}\vec{F}_i - (\xi + \dot{\varepsilon})\vec{v}_i, \tag{2.30}$$

where ξ still follows equation (2.28). Combined, these effects allow the Nosé-Hoover barostat to reproduce the isothermal-isobaric ensemble. Barostats have, however, been repeatedly improved for both accuracy and computational complexity. Currently, LAMMPS implements an advanced Nosé-Hoover chain based algorithm as derived by Tuckerman et al. [20].

2.7 Estimating quantities of interest

2.7.1 Velocity field and viscosity

Molecular dynamics is a powerful technique for studying the motion of atoms in a fluid. Yet its computational demands make it unviable for macroscale simulations. At macroscale, however, a fundamental assumption can be made, namely that the fluid is homogeneous. This would be absurd at the molecular scale, but at macroscale, it will generally be a good

approximation. Thus one can stop talking about the velocities of individual atoms and instead study the *velocity field* — the velocity as a function of space and time.

The equations relating the velocity field to external forces, pressure gradients etc. are the Navier-Stokes equations,

$$\rho\left(\frac{\partial}{\partial t} + \vec{v}\cdot\nabla\right)\vec{v} = \rho\vec{F} - \nabla p + \eta\nabla^2\vec{v} + \eta\nabla(\nabla\cdot\vec{v}), \tag{2.31}$$

where ρ is the density, $\vec{F}$ is an externally applied force, p is the pressure and η is the viscosity. This set of equations is not easily solvable analytically, but solving it with the finite element method is still computationally trivial when compared to doing a molecular dynamics simulation. In molecular dynamics, the velocity field must be computed as an average of the velocities of single atoms. Section 3.1 on page 27 describes my solution for doing this as efficiently as possible.

The shear viscosity can be estimated in an equilibrium molecular dynamics simulation with a Green-Kubo relation,

$$\eta = \frac{V}{k_\mathrm{B}T}\int_0^\infty \left\langle P_{ij}(t)P_{ij}(0)\right\rangle \mathrm{d}t\,. \tag{2.32}$$

In words, the viscosity is found by integrating the autocorrelation of the off-diagonal elements of the pressure tensor. The autocorrelation decays rapidly, allowing the integral to be truncated when the integral has plateaued.

Unfortunately, single autocorrelations and their integrals will fluctuate wildly and give no useful information. Consequently, the viscosity is estimated as the average of many such integrals. The different integrals can be acquired from either separate simulations or different parts of a single simulation. Furthermore, one should average over the independent pressure tensor components.

2.7.2 Radial distribution function

The radial distribution function, $g(r)$, is used to get a rough idea of the atomic structure in a material. It gives information on the relative distances between atoms, as seen from its definition,

$$g(r) = \frac{\mathrm{d}n(r)}{4\pi r^2\rho\,\mathrm{d}r}. \tag{2.33}$$

$n(r)$ is the total number of neighbours within a radius r, so that $\mathrm{d}n(r)$ is the number of neighbours inside a thin spherical shell extending from r to $r+\mathrm{d}r$, while $4\pi r^2\rho\,\mathrm{d}r$ would have been the number of neighbours within the same shell if the system were completely homogeneous.

Alternatively, the radial distribution function can be defined through its integral,

$$n(r_1 < r < r_2) = \rho\int_{r_1}^{r_2} g(r)\,4\pi r^2\,\mathrm{d}r\,, \tag{2.34}$$

which can be used to calculate the number of neighbours at a distance between r_1 and r_2, $n(r_1 < r < r_2)$. For example, the coordination number is easily estimated by integrating over the first peak of $g(r)$.

2.8 LAMMPS

The preceding sections contain a wealth of complications and computational tidbits. Implementing a straight forward, quadratically scaling molecular dynamics code for a simple Lennard-Jones system in the microcanonical ensemble is easily done in less than fifty lines of code, but problems arise when trying to make the code usable for systems of relevant sizes. Molecular dynamics is, unfortunately, not embarrassingly parallel [21], and serious effort is required to develop a library that can offer both flexibility and performance — especially with the increasing use of heterogeneous computing systems.

Fortunately, there is no need for a master's student like myself to spend months reinventing the wheel, because LAMMPS exists. LAMMPS, short for Large-scale Atomic/Molecular Massively Parallel Simulator, is a general-purpose molecular dynamics software and library which can do most tasks in classical molecular dynamics and, most importantly, do them fast, across different computing architectures. Furthermore, it is freely available, open-source and easy to extend with additional functionality (see e.g. section 3.1 on page 27).

2.8.1 Usage overview

LAMMPS can be run in two ways; either through a script of commands or through a library interface. In this thesis, I have exclusively used the scripting interface, and passing commands as strings is also the main way of using the library interface. These commands are written in the primitive LAMMPS programming language, which is a fairly complete language with variables, if-tests, loops and so on.

An input script mainly consists of two types of commands. First, there are the commands which are executed as they are read by LAMMPS. Typical examples include the creation of groups and regions, as well as the `run` command, which starts the simulation. The second group of commands consist of those which define operations to be performed during the subsequent simulations, for example outputting, adding forces etc.

As an example, my script for compressing a system may look something like this:

```
# common settings
units metal
boundary p p p
atom_style atomic

# script which determines SiO2, passivated or water based on the
# VARIANT command line argument, and sets SUFFIX_UNDERSCORE
# to either "", "_passivated" or "_water"
```

```
include in.choosevariant

log data/log.down${SUFFIX_UNDERSCORE}

read_data data/data.setup${SUFFIX_UNDERSCORE}

# include system settings
include in.common_variables
include in.common_regions
include in.common_groups

include in.potential_etc${SUFFIX_UNDERSCORE}

# fix misplaced OH groups or water molecules
minimize 1e-6 1e-6 1000 1000

# time integration
fix nvt all nvt temp ${TEMP} ${TEMP} 1.0

# interesting quantities to be computed
compute temp all temp
compute pressure all pressure temp
compute com_top_sphere top_sphere com

# writing atom positions
dump mydump all custom 100000 data/dump.down${SUFFIX_UNDERSCORE}.bin id type x
    y z vx vy vz

# writing thermodynamic info
thermo 100
thermo_style custom step time temp press pe ke etotal c_pressure[3]
    c_com_top_sphere[*] spcpu cpuremain

# initialise velocities
velocity all create ${TEMP} 277385 mom yes loop geom

# determine deformation parameters
variable indent equal $a*5.0
variable time_down equal 200
variable steps_down equal $(round(v_time_down/dt))
variable distance equal ${indent}+${initial_clearance}

# deformation fix
fix deform_z all deform 1 z vel $(-v_distance/(v_steps_down*dt)) units box

# run simulation
run ${steps_down}
```

This code example shows several useful techniques, such as

- Storing code common for multiple simulations in separate script files which are reused with the `include` command.

- Using variables and LAMMPS's built-in mathematical functions to avoid hard-coding "magical" numbers.
- Using command-line arguments to control the flow of the program, allowing this script to be used for simulating either pure SiO_2, passivated SiO_2 or passivated SiO_2 with water.

Simulations often require multiple steps to go from the initial system to some interesting result. LAMMPS is well-suited for such studies, as it is easy to write data or restart files in one simulation and continue from these in another. Typically, my simulation pipelines have consisted of a series of LAMMPS, Python and PACKMOL scripts for setting up the system, converting file formats, filling the system with water and reading off parameters which will be used later. There are many tools for managing such pipelines, but I have chosen to stick with old-fashioned makefiles, as they are perfectly capable of handling the sequences of dependencies necessary for my simulations.

2.8.2 Optimising performance

Hardware

Traditional supercomputers consist of a large number of nodes, where each node has a powerful CPU, with usually 16 to 32 cores. LAMMPS is designed to run as fast as possible on such supercomputers, by dividing the spatial domain among processors and allowing them to work in parallel on their subset of atoms. More than half of my simulations have been run on such clusters, and they work well for any type of molecular dynamics simulation.

In recent years, graphical processing units (GPUs) have emerged as a viable alternative or complement to CPUs for certain tasks. Whereas CPUs have fast cores but typically 32 or fewer, GPUs have a large number of cores (our Tesla P100s have 3584 CUDA cores each) and high memory bandwidth, but each core is a lot weaker than a CPU core. GPUs are therefore well-suited for solving many small tasks in parallel. Force calculations in molecular dynamics is a prime example.

LAMMPS supports the use of GPUs in two ways. First, the GPU package provides implementations of many potentials and will offload the force calculation to GPU. Unfortunately, this requires copying the positions and forces between the GPU and CPU for every timestep, which reduces performance somewhat.

Another option is the KOKKOS package, which uses the Kokkos library [22] to provide parallel performance across a variety of platforms. The purpose of the Kokkos library is to provide the necessary abstractions for writing programs that will perform well on different computer architectures, while being considerably higher-level than e.g. CUDA. Consequently, a larger subset of LAMMPS features has been implemented in the KOKKOS package. As long as only these features are used, the computations can run entirely on the GPU, giving excellent performance.

Since I have access to a machine with 8 of the powerful NVIDIA Tesla P100 GPUs, I have used these through the KOKKOS package whenever possible. The KOKKOS package has implemented all the features I have needed, with one exception. In some cases, it has also been possible to work around missing features, such as replacing `fix move` with `fix setforce 0` and setting the velocity. The one exception is, however, quite a big one, namely that there is no GPU or KOKKOS implementation of the force field for water. As such, all simulations containing water, or even just passivating hydroxyle groups, have been run on a conventional, CPU-based cluster.

Load balancing

If not instructed otherwise, LAMMPS will divide the simulation box into a three-dimensional, regular grid of spatial bins — one per processor. This is perfectly acceptable for many molecular dynamics simulations, namely those where the atoms are approximately uniformly distributed in space. This approximately applies to my water flow simulations, but not to the simulations of creep without water, where certain parts of the simulation box are completely empty.

Fortunately, LAMMPS has several automated ways of alleviating this problem. I have mainly used the `balance` command, which makes the grid irregular in order to minimise the differences in load per processor. The command

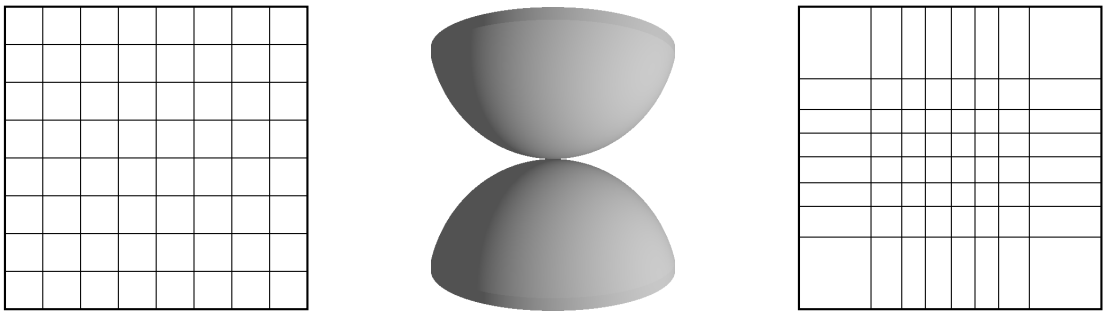
```
balance 1.1 shift xyz 30 1.1
```

tells LAMMPS to balance if the processor with the most atoms has more than 10 % more atoms than the processor with the fewest atoms. Up to 30 iterations can be performed, and the bins should be resized in all three directions. A sample output from my creep simulations on 256 cores is

```
rebalancing time: 0.0885148 seconds
iteration count = 16
initial/final max load/proc = 1248 884
initial/final imbalance factor = 2.31483 1.63967
x cuts: 0 0.238281 0.339844 0.417969 0.503906 0.582031 0.660156 0.761719 1
y cuts: 0 0.238281 0.339844 0.417969 0.503906 0.582031 0.660156 0.761719 1
z cuts: 0 0.191406 0.511719 0.816406 1
```

After 16 iterations, the ratio of the highest and lowest atom count per processor is reduced from 2.3 to 1.6, which is a considerable improvement. We see that this is achieved by making the central bins smaller in the xy-plane, and making the edge bins smaller in the z-direction.

As an alternative, LAMMPS also has the option of using a tiled layout (see illustration below), via setting `comm_style tiled` and issuing the `balance 1.1 rcb` command. Unfortunately, this communication style produced memory errors when I tried to use it, so its performance relative to an optimised grid layout has not been tested.

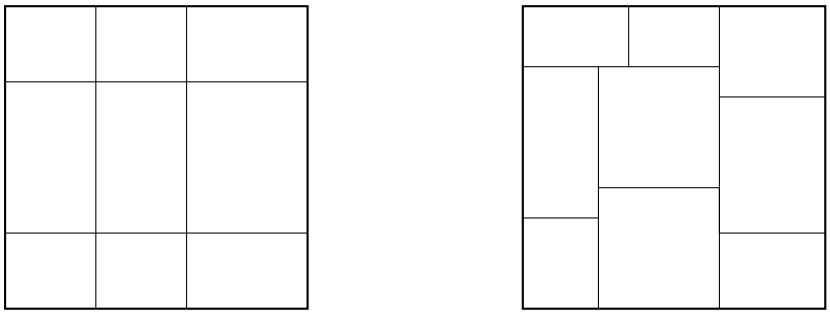


(a) Initial, regular grid.

(b) System sketch. The z-axis is vertical.

(c) Optimised grid, with a more even distribution of atoms.

Figure 2.1: Initial and final splitting of the system in the xy-plane. Since there are more atoms in the centre of the xy-plane, the bins located here are shrunk in order to reduce the load imbalance.



(a) Grid layout.

(b) Tiled layout.

Figure 2.2: Comparison of example grid and tiled processor layouts in two dimensions.

2.9 Initialising non-crystalline systems

The silica in my systems is initialised as a perfect crystal of β-cristobalite, but liquid water must be packed differently to be anywhere near an equilibrium configuration. This is where PACKMOL [23] shines. PACKMOL is, in essence, a software which packs molecules in a "reasonable" way, i.e. in such a way that the structure of each molecule is maintained, while no atoms are placed too close to each other. Additionally, the molecules can be constrained to be outside or inside of common geometric structures such as cylinders and spheres. PACKMOL reads simple input scripts from standard input, so the typical usage is to write the script in a file and invoke PACKMOL with

```
packmol < script.inp
```

My use of PACKMOL has been to fill the void in a silica system with water. There are two strategies for doing this. Since my silica systems all have a well-defined geometry (such as two sphere caps or a slab and a cylinder), the most obvious choice was to simply constrain the water molecules to be outside of these regions. This does work, although I found this approach to be slow. The faster alternative is to not give PACKMOL any geometric information on where to pack the water molecules, and instead add the silica system as a single molecule at a fixed position. An example input script is shown below. This script packs the water as shown in figure 2.3 on the next page.

```
filetype xyz
output water_data/xyz.water

# silica structure in fixed position
structure water_data/xyz.passivated
    number 1
    fixed 0. 0. 0. 0. 0. 0.
end structure

# water molecules
structure water.xyz
    number 81289
    inside box 0. 0. 0. 171.84 171.84 150.36
end structure
```

The dimensions of the `inside box` constraint for water are simply those of the simulation box in LAMMPS, and no other geometric information is given. `water_data/xyz.passivated` contains the atomic positions ouputted by the passivation step, while `water.xyz` defines the geometry of a single water molecule.

Even though PACKMOL will ensure that all atoms are placed at a reasonable distance from each other, the atomic positions may still need some tweaking. Indeed, simulations run directly from PACKMOL generated data will typically crash with the `lost atoms` error message due to too large forces. Fortunately, LAMMPS has a built-in solution, namely the `minimize` command. This command will minimise the energy and atomic forces according to the actual force field, unlike PACKMOL, which just uses interatomic distances.

The command

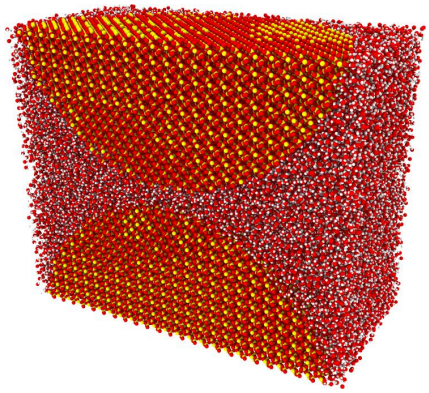


Figure 2.3: Creep system consisting of two silica sphere caps with water filling the remaining pore space, sliced in half for visualisation purposes. Silicon atoms are yellow, oxygen atoms are red and hydrogen atoms are white. PACKMOL packs the water molecules around the sphere caps, since the silica system is added as a fixed structure in the input script.

```
minimize 1e-6 1e-6 1000 1000
```

has worked reliably for me in every water-related simulation. This command will run the minimisation algorithm until the relative change in energy is less than 10^{-6}, the sum of the force magnitudes is less than 10^{-6} eV/Å, 1000 minimisation steps have been performed, *or* 1000 force evaluations have been performed. Sample output from minimisation of the system in figure 2.3 is shown below, where the minimisation loop stops when the energy requirement is fulfilled.

```
Minimization stats:
   Stopping criterion = energy tolerance
   Energy initial, next-to-last, final =
       9.1391045877e+12     -833440.995934     -833441.818441
   Force two-norm initial, final = 3.38205e+15 6.75094
   Force max component initial, final = 2.36452e+15 1.98582
   Final line search alpha, max atom move = 0.0669732 0.132997
   Iterations, force evaluations = 687 924
```

Chapter 3

New tools

In this chapter, I go over the different tools that I have developed to ease my work with LAMMPS and molecular dynamics. LAMMPS has a lot of functionality, but I found it useful to write an extension specifically for sampling and averaging the velocity field in flow simulations, as the existing solutions all have significant drawbacks. I have also developed tools which offer simplified interfaces to the various ouputs from LAMMPS, and I describe my procedure for automating simulation pipelines and checkpointing long simulations.

3.1 LAMMPS-fix for velocity profiles

3.1.1 Motivation

Continuum fluid mechanics gives the velocity profile $\vec{v}(t, \vec{r})$ as a smooth function, but the velocities in molecular dynamics are simply the instantaneous velocities of the atoms in the fluid. These fluctuate wildly, in both space and time. Consequently, the velocities must be averaged in order to obtain results which can be compared with continuum predictions.

The question is then how to perform this averaging. A simple approach is to save the velocities of all particles to a file (e.g. a `dump`) every once in a while, and then average the velocities over both space and time when the simulation has finished. Unfortunately, a reasonable resolution requires too large data files to be written. With e.g. 500 000 atoms in the system and 8 properties per atom (id, type, $\vec{r}$ and $\vec{v}$), each time step will require

$$500\,000 \text{ atoms} \cdot 8 \text{ properties/atom} \cdot 8 \text{ bytes/property} = 32 \text{ MB}. \tag{3.1}$$

Saving velocities every 1000th step over the course of 10^6 steps results in a data file of 32 GB for every simulation. Trial and error have shown that this is not a sufficient resolution, so even more data is necessary. I have therefore deemed this solution impractical.

Another direction is then to do the averaging "on the fly" while the simulation is running, reducing the need for frequent writing of large amounts of data. LAMMPS has built-in

functionality for this through the use of chunking, achieved through the `compute chunk/atom` command. The velocity in a chunk can then be averaged over space using `compute vcm/chunk` over time with `fix ave/time`. Alternatively, `fix ave/chunk` can average over both, so I chose this solution.

Two drawbacks of `fix ave/chunk` became evident. First and foremost, performance took a major hit. The number of timesteps per second was reduced by a factor of approximately 1.5 in one of my flow simulations, thus increasing the total run time by two days. This is most likely due to the general nature of `fix ave/chunk`, which can average arbitrary quantities over complex chunks. Secondly, `fix ave/chunk` writes text (ASCII) files. These are read more slowly than binary files because the reading program needs to interpret the text and not just dump the zeroes and ones into memory. Even though the text files from `fix ave/chunk` are simply formatted, the analysis time was reduced from 10 minutes to 40 seconds when instead reading binary files. The quality of life while writing and debugging analysis programs is drastically improved when testing is done in a matter of seconds rather than minutes.

Since pure post-processing requires too much data and LAMMPS's built-in velocity averaging had performance issues, I chose to write an extension of LAMMPS whose sole purpose is to average velocities over time and a three-dimensional Cartesian grid in space. Such an extension should not affect the overall performance, as the spatial binning of atoms is practically free when compared to the calculation of forces. Additionally, synchronisation between processors only needs to occur when the results are written to file — not every time the quantities are sampled. Finally, such a single-purpose fix can store its data in a primitive binary format which can quickly be read for post-processing.

3.1.2 Implementation

A major advantage of LAMMPS is its easy extensibility. Even though its most common usage is through its scripting language, it is relatively easy to add functionality due to the object-oriented design of LAMMPS. Most types of commands can be added, such as `pair_style` and `compute`, which define new force fields and computed quantities. The most general commands are `fix`es. These can be called at any stage of the computation and do whatever they want.

As my main goal was to average velocities over space and time and write the results to file, I chose to implement a `fix` which is called at the end of each time step, when all forces have been calculated and the positions have been updated. Consequently, I needed to write a class that inherits from LAMMPS's `Fix` class and overrides the method being called at the end of a time step.

The code is available at github.com/anjohan/lammps-velocityaverage. It can be added to LAMMPS simply by dropping the `.cpp` and `.h` files into LAMMPS's `src/` directory and recompiling.

Defining the command

Making LAMMPS aware of the new command is achieved by making sure that the correct macro is called at the right time,

```
#ifdef FIX_CLASS
FixStyle(velocityaverage, FixVelocityAverage)
#else
// Declare FixVelocityAverage class
#endif
```

The `FixStyle` macro adds the command `fix velocityaverage` to LAMMPS's scripting language and connects it to the class `FixVelocityAverage`.

Declaring the class

Integration of the `FixVelocityAverage` class into the LAMMPS code structure is achieved through polymorphism, with `FixVelocityAverage` inheriting from LAMMPS's `Fix` class and overriding the relevant virtual methods.

```
class FixVelocityAverage : public Fix {
    public:
        FixVelocityAverage(class LAMMPS*, int, char **); // Constructor
        ~FixVelocityAverage();      // Destructor (close files, free memory)
        virtual int setmask();      // Specify when to call this fix
        virtual void end_of_step(); // Method called at end of step
        virtual void setup(int);    // Method called before each run

    protected:
        int Nx, Ny, Nz, Ntot;
        // Other parameters
};
```

Implementing the methods

The constructor needs to set the `nfreq` attribute equal to the frequency specified by the `fix velocityaverage` command, which tells LAMMPS how often this fix should be invoked, while

```
int FixVelocityAverage::setmask(){
    int mask = 0;
    mask |= FixConst::END_OF_STEP;
    return mask;
}
```

informs LAMMPS that the fix should be invoked at the end of the step, after all forces have been applied and all positions have been updated.

Averaging of the velocities happens inside the `end_of_step()` method. A loop over all

atoms belonging to the current processor registers the number of atoms inside each bin and adds their velocities to an accumulated sum.

```
for(int a = 0; a < nlocal; a++){ // loop over atoms
    if (x[a][0] < xmin || x[a][0] >= xmax || \
        x[a][1] < ymin || x[a][1] >= ymax || \
        x[a][2] < zmin || x[a][2] >= zmax || \
        !(mask[a] & groupbit)) {
        continue; // atom outside specified region or
                  // not in the specified group
    }

    // determine bin
    int i = (x[a][0] - xmin)/dx;
    int j = (x[a][1] - ymin)/dy;
    int k = (x[a][2] - zmin)/dz;
    int idx = index(i,j,k,0);

    data[idx] += 1; // register atom
    data[idx + 1] += v[a][0]; // add velocity
    data[idx + 2] += v[a][1];
    data[idx + 3] += v[a][2];
}
```

At another frequency provided in the `fix velocityaverage` command, the results are summed from all processors using `MPI_Reduce` and written to file by the processor with rank 0.

3.1.3 Interface to results

To make the `fix velocityaverage` command easy to use, I also provide a high-level interface to the resulting files. Thus one can analyse the resulting data without having to think of the file structure or explicit error handling.

The interface is provided in the form of a Fortran module `mod_velocity_reader` containing the class `velocity_reader`. Fortran was chosen due to its combination of high performance with easy handling of multi-dimensional arrays. An object-oriented approach allows encapsulation and abstraction, giving a simple interface to the results. With this class, the structure of an analysis program can be

```
program foo
    use mod_velocity_reader
    implicit none

    class(velocity_reader), allocatable :: reader

    reader = velocity_reader("filename.bin")

    do while (.not. reader%EOF)
        call reader%read_step()
```

```
        ! do work on reader%values
    end do

    deallocate(reader)
end program
```

A `velocity_reader` object has the attributes $N_{\{x,y,z\}}$, $\{x,y,z\}_{\{\min,\max\}}$, `step` and `values`. The latter is an array with dimension $4 \times N_z \times N_y \times N_x$ (due to the column-major memory layout of Fortran). `reader%values(:,k,j,i)` refers to a vector of length 4, where the first element is the average number of atoms registered in the bin with index (i,j,k), and the three last elements are the average velocity of these atoms.

3.2 Command line tool for log files

LAMMPS log files are perhaps the most important type of output from LAMMPS, as they give a quick overview of system-wide properties such as time and temperature. Such information can give a strong indication of whether the simulation is going to give meaningful or interesting results. I have therefore found it immensely useful to have a simple one-liner command for plotting quantities from log files. It is a simple Python-script (available at github.com/anjohan/mdtools).

The basic usage is

```
logplotter.py -i log.file -x Time -y Temp
```

which will plot temperature as a function of time from the file `log.file`. Useful features include concatenating data from multiple log files and plotting log files from simulations which are still running. There are also arguments for saving the plot, writing the results to a text file, etc. See `logplotter.py --help` for an overview.

Additionally, log files can be read systematically in Python by importing the `find_data` function,

```
from matplotlib.pyplot import plot, show
from logplotter import find_data

results = find_data("log.file")

plot(results["Time"], results["Temp"])
show()
```

3.3 Interface to binary dump files

Text-based dump files have the nice feature of being human-readable, i.e. they can be opened in a text editor and inspected visually, but they are incredibly slow to read because the reading software has to interpret all of the text. The alternative is to use binary dump

files, which are read simply by copying the bytes directly from the file into memory. This can lead to a speedup of one or two orders of magnitude.

The format of LAMMPS's binary dump files is not documented anywhere, but it can be inferred from the source code of their `binary2txt` tool. There is already one library available for reading these binary dump files, namely the scripting interface to OVITO. This is a powerful tool which can read a variety of formats, and it can do many different types of analysis on the acquired data. I, on the other hand, am mainly interested in reading either velocities or positions and doing simple averaging. Consequently, OVITO becomes a bit of an overkill with an unnecessary amount of boilerplate code.

My solution has been to write a simple, object-oriented interface to the binary dump files which allows direct access to the contained values with minimal boilerplate. Since the main goal was to combine an easy-to-use interface with good performance, Fortran was chosen as a programming language for the implementation.

The source code is available at github.com/anjohan/lammps-binary-dump-reader. Using this code, the typical structure of an analysis program is

```
program analysis
    implicit none

    use mod_lammps_reader

    type(lammps_reader) :: reader

    call reader%open_file("mydump.bin")

    do while(reader%has_next_step)
        call reader%read_step()

        ! do work on reader%values
    end do
end program
```

Here, `reader%values` is a matrix where the number of rows is equal to the number of per-atom properties and the number of columns is equal to the number of particles (due to the column-major memory ordering of Fortran). Additionally, metadata such as the current step, boundaries etc. can be accessed through the `reader%header` object. `reader%next_header` gives the same information for the next timestep (if `reader%has_next_step` is `.true.`).

3.4 Makefiles for checkpointed simulation workflows

3.4.1 Motivation

Molecular dynamics simulations have shown a tendency to end up with a long and complicated workflow. As an example, my flow simulations require the following steps:

1. Create a box of SiO_2, make it amorphous through heating and cooling.
2. Create the desired geometry of the system by cutting out regions (separate step because this is fast but error-prone).
3. Add the hydrogen atom type to the data file.
4. Passivate the system (separate step because it only works on one core).
5. Write a PACKMOL input file based on the system dimensions.
6. Run PACKMOL to add water.
7. Convert the resulting `.xyz`-file to a `.data`-file.
8. Flow simulation (for each external force — 10 in this case).

In practice, each of these steps require running a program from the command line. Manually running this workflow is not a viable option, as it might lead to considerable downtime between steps if a step finishes in the middle of the night or there is a queue on the cluster. Additionally, I want the security of knowing that all the required simulations will be rerun whenever I have changed the input file of one of the simulations. Furthermore, manual work is error-prone and boring.

The easiest alternative would be to just dump the required commands in a script and run it whenever something has changed. This would be perfectly acceptable with small simulations, but my simulations have taken days or weeks on hundreds of cores, making it vital to not waste time doing unnecessary work. Hence I need a tool capable of handling dependencies and checking for updated input files. There are many alternatives, but I chose to go with the old-fashioned approach of makefiles. They are intended for compiling programs, not for running simulations, but their basic

```
target: dependencies
    command
```

syntax is the simplest to use out of all the tools that I have looked at and perfectly adequate for defining simulation workflows as well.

The disadvantage to such a makefile-automated workflow is that care must be taken to not unnecessarily update the timestamp on input files. If this occurs, the simulation will be started from scratch and results from previous simulations will be overwritten.

3.4.2 Iteration and parameter sweeps

With iteration of the type

```
all: $(foreach param,$(listofparams),target_$(param))

target_%: input_%
    command -parameter $*
```

one can do parameter sweeps directly in the makefile. A major advantage is the possibility of adding to the list of parameters, since the simulation will not be repeated for the old parameters unless the input dependencies have changed.

For example, my flow simulations were first run with forces equal to odd multiples of 10^{-5} eV/Å, but it was trivial to later add the even multiples. The odd multiples were ignored, since the data files already existed and the input scripts were unchanged. A more advanced tool, e.g. one that checks for changes in file contents rather than timestamps, would remove this problem, but it has not caused me any trouble.

3.4.3 Loops of continued simulations

The workflow example in section 3.4.1 on page 32 already has checkpoints between each step. However, the main part of the simulation happens in the last step of the list presented in the previous section. This is even more visible in the creep simulations, where there is no parameter sweep — only a few simulations which last at least one week each on either a 320-core CPU cluster or an 8-card Tesla P100 GPU machine.

There are several advantages to splitting these simulations into multiple steps which start from the final positions of the previous step. First and foremost, the queue system at our local supercomputer only allows jobs to run for one week at the time. Secondly, the computers may crash or become unavailable, and losing a week or more of progress due to this is simply not acceptable. Finally, it is difficult to know how long simulations are both required and realistic. Interesting results may appear after 1 ns, or it may take 200 ns before anything of interest happens. With a checkpointing system, one can run for e.g. 50 ns and run an extra 25 ns if required without starting from zero.

Fortunately, it is possible to implement this in makefiles, although it is slightly less trivial than in the previous examples. The magic trick is to use the `eval` function, which evaluates the result of a function as an ordinary makefile rule. Specifically, my flow makefile contains the function

```
define defflow =
data/restart.flow_%_$(1): data/restart.flow_%_$$(shell expr $(1) - 1) flow.in
        mpirun $(lammps) -var FORCE $$* -var ITERATION $(1) -in flow.in
endef
```

This function takes in one argument (referenced with `$(1)`) which is the iteration variable. `data/restart.flow_%_$(1)` (where `%` and `$$*` are later replaced with a force) depends on the output of the previous iteration, which can be calculated by calling the `shell` command `expr` as `data/restart.flow_%_$$(shell expr $(1)- 1)`. Note the double dollar signs, since the function will later be evaluated, and a double dollar sign will then evaluate to a single dollar sign. Additionally, the result depends on the input script, such that the simulation will start from scratch if the input script is changed.

The input script has the structure

```
log data/log.water_${FORCE}_${ITERATION}
read_restart data/restart.flow_${FORCE}_$(v_ITERATION-1)
```

```
if "${ITERATION} == 1" then "reset_timestep 0"
...
write_restart data/restart.flow_${FORCE}_${ITERATION}
```

and takes as its input the `FORCE` and `ITERATION` variables, as also seen in the makefile function.

A simple loop is used to evaluate the makefile function and actually define the makefile rules, namely

```
$(foreach i, $(shell seq 1 $(ITERATIONS)), $(eval $(call defflow,$(i))))
```

which lets the variable `i` run from `1` to `ITERATIONS` and calls the function for each value of `i`. `ITERATIONS ?= 10` gives `ITERATIONS` a default value of `10`, but the question mark means it will be overwritten if the makefile is invoked with e.g. `make ITERATIONS=20`.

The default target is the one first defined in the makefile, usually named `all`,

```
all: $(foreach force, $(forces), data/restart.flow_$(force)_$(ITERATIONS))
```

Unless `make` is invoked with a specific target, it will loop over the list of forces and run `ITERATIONS` iterations for each force. If it crashes or the job is cancelled, `make` will start off from the last completed iteration (for each force, separately).

Chapter 4

Related work

In this chapter, I go over a selection of papers which simulate systems that are somewhat similar to mine. I focus on computational papers, as my main contribution is simulating systems which have not previously been simulated.

4.1 Silica-water force field, viscosity and flow

There are a lot of water force fields, but only a few which aim at modelling the coexistence of silica and water. Tang [24], Cruz-Chu et al. [25] and Hassanali and Singer [26] have all proposed silica-water force fields which are nondissociative, i.e. they treat water molecules as fixed entities which move as single bodies. With this approach, there is no hope of studying chemical reactions between silica and water.

Mahadevan and Garofalini [27] developed a dissociative water model which they use for simulating amorphous silica surface structure and the adsorption of water [28] with good results. This potential appears to be a good candidate for my simulations containing amorphous silica. However, the potential is explicitly parametrised for amorphous silica and has not been tested on crystalline systems.

Based on the above considerations, I choose to use the recently developed extended Vashishta-potential [11] as described in section 2.2. This builds upon the Vashishta silica potential, which is mature and known to accurately model both crystalline and amorphous silica. Shekhar et al. [29] used this potential to study the collapse of a water nanobubble on a silica surface, and Shekhar et al. [30] (different al.) studied relaxation in nanoconfined water. These are mainly structural studies, and I have not found any investigation of the water potential's transport properties. The fact that the force field has not yet been published is presumably the main cause. My contribution is therefore to test the transport properties by measuring viscosity and performing a flow simulation.

My viscosity measurements are not new developments in any way — they simply use the Green-Kubo relation very straightforwardly. Zhang et al. [31] have devised a method

for fully automating the viscosity measurements. As I also observed myself, viscosity measurements can be tricky because of the wildly fluctuating pressures and pressure autocorrelations. The procedure of Zhang et al. would remove any need for visually determining when to cut the simulations, which data to use, how to weight the data etc. This would be vital e.g. if I were to measure the viscosity for a large number of temperatures. Since I only do a few measurements and not a systematic study, I found it faster to just generate more than enough data and do a straightforward estimate.

Flow simulations are also not uncommon in molecular dynamics. For example, Priezjev and Troian [32] simulated the slip behaviour of a liquid flowing past a surface with periodically varying wettability and compared with continuum simulations, while Hafreager [33] studied flow in a cylindrical pore as a function of pore size and confirmed the Knudsen correction for permeability at small length scales. However, these simulations are performed using the simple Lennard-Jones potential, which is only reasonable for noble gases. In particular, the varying wettability modelled by Priezjev and Troian is achieved simply by tuning the parameters (ε and σ), which does not model a real system. My departure from previous flow simulations is the use of a realistic, physically and geologically relevant potential.

4.2 Creep

The creep half of my thesis is almost entirely based on the paper by Aharonov and Scholz [4], in which they derive a physics-based model for steady state friction. Their model is capable of predicting frictional properties over a large span of slip rates, thus being more general than the well-known rate-and-state friction law which applies to slower (literally glacial) slip rates. It is interesting to note that the new model can be Taylor expanded to reproduce the rate-and-state model in the limit of slow slip rates.

Contact between surfaces is assumed by Aharonov and Scholz to be through asperities, as seen in their figure 2 and my figure 7.1. Studying a system with many nanometre scale asperities is still outside the computational reach of molecular dynamics, and I therefore choose to study the creep process in a pair of opposing asperities.

Previously, Luan and Robbins [34] simulated a rigid asperity being pushed into a slab using the Lennard-Jones potential for different asperity shapes. Larsen [35] repeated the simulation for a nonrigid spherical asperity in a β-cristobalite system using the Vashishta-potential, and studies contact force distributions and friction properties.

Molecular dynamics creep simulations have previously been performed on bulk materials. For example, Jiao and Kulkarni [36] performed a molecular dynamics study of creep mechanisms in nanotwinned copper at high temperature and find that the creep resistance is greater for nanotwinned materials than for nanocrystalline materials. To the best of my knowledge, there are no reported results from molecular dynamics simulations of creep in asperities. In particular, no one seems to have compared creep in asperities with an analytical expression based on Aharonov and Scholz's friction model.

Part II

Simulations, results and discussion

Chapter 5

Verifications

5.1 Creating amorphous silica

5.1.1 Implementation

The system was initialised as a block of $30 \times 30 \times 18$ unit cells of β-cristobalite, giving a system with 388 800 atoms and dimensions 214.8 Å $\times$ 214.8 Å $\times$ 128.88 Å. Amorphous silica was then created through a sequence of rapid heating followed by stepwise cooling. The temperatures and times were taken from the supporting material provided by Shekhar *et al*. [29], giving the following procedure:

1. Heat from 300 K to 4000 K during 30 ps.
2. Equilibrate for 20 ps.
3. Stepwise cooling to 4000 K, 1900 K, 1400 K, 950 K, 600 K, 375 K and 300 K with a cooling time of 10 ps and equilibration for 10 ps at each temperature.

During equilibration, the thermostat damping time was 1 ps as in the other simulations, but it was reduced to 0.1 ps during heating and cooling after 1 ps proved inadequate for achieving the desired heating and cooling rates.

The radial distribution function, $g(r)$, was used as a simple measure to verify that the correct structure was created. It was calculated using OVITO's built-in "Coordination analysis" modifier. Due to the large number of atoms, no averaging was necessary to obtain a smooth plot and only the final time step was used.

5.1.2 Results

Figure 5.1 on the following page shows the temperature as a function of time resulting from the melting and cooling procedure outlined above. With the reduced thermostat damping,

the temperature closely follows the desired curve. The resulting radial distribution functions are shown in figure 5.2 and figure 5.3. As expected when using the same procedure, the results are very similar to those of Shekhar *et al*. [29]. Only the nearest-neighbour peaks differ slightly.

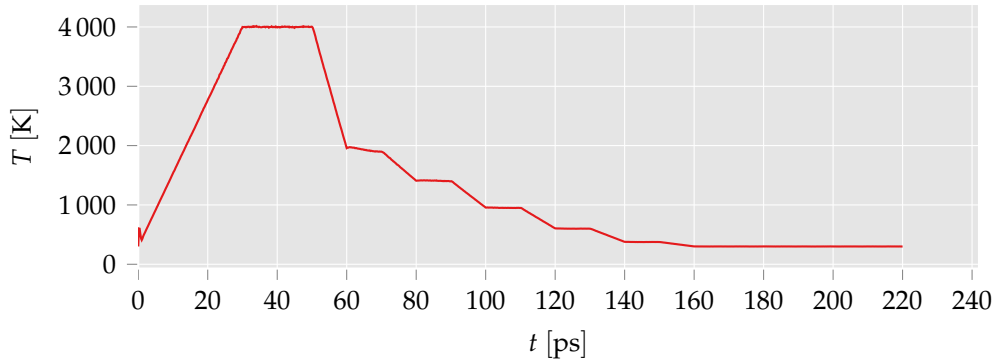


Figure 5.1: Temperature in the silica block as a function of time. The system is rapidly heated to 4000 K before being cooled to room temperature, based on the supporting material provided by Shekhar *et al*. [29].

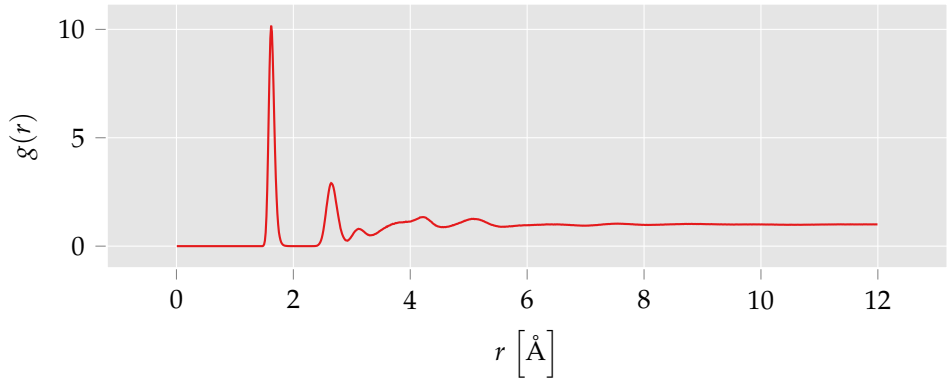


Figure 5.2: Total radial distribution function of silica after the melting procedure, calculated by OVITO.

5.2 Viscosity

The viscosity of water is well-known, but that does not mean that the viscosity of the water in my simulations is known. Since I am using a force field that is not very well-established, I choose to measure the force field's viscosity with the parameters used in the flow simulations — 300 K and 1 g/cm^3. The water portion of the force field has been fitted to symmetric bond stretching and bending in a single water molecule, which does not guarantee a good fit with the experimental viscosity of water. In particular, the authors of the force field have measured a diffusivity which is four to five times higher than the experimental value.

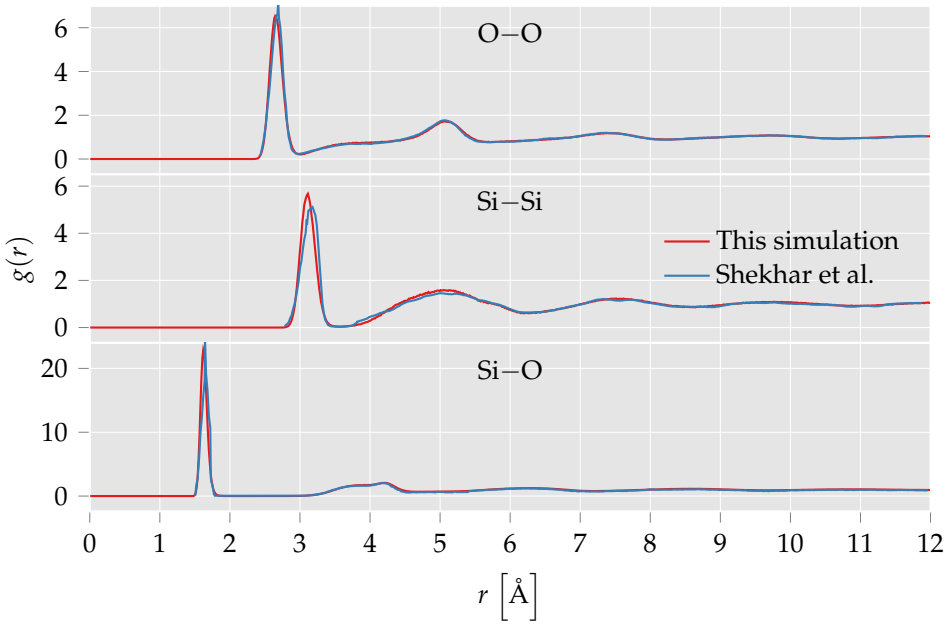


Figure 5.3: Partial radial distribution function of silica after the melting procedure, compared with data from figure 5 of Shekhar et al. [7] extracted with WebPlotDigitizer [37]. Apart from slight deviations in the nearest-neighbour peaks, the results overlap well.

5.2.1 Implementation

To measure the viscosity of water, a cubic system containing 2000 water molecules with a density of 1.0 g/cm^3 was simulated at 300 K. The initial positions were generated with PACKMOL, and the water molecules were given an initial temperature of 500 K and cooled to 300 K during 1 ps. Pressures were then sampled every timestep for 15 ns.

The results were analysed and interpreted using the Green-Kubo relation for pressure and viscosity, equation (2.32) on page 19. It turns out that the finite-time integrals fluctuate wildly, so that many integrals are required to achieve a reliable estimate of the viscosity. Since the pressure autocorrelation function decays very rapidly, the full time series was used to generate many shorter time intervals over which to perform the Green-Kubo integral, and the average of these was taken.

Zooming in on the autocorrelation plot in figure 5.4 on the following page showed that intervals which start 300 fs or more apart are essentially independent. With independent measurements, the uncertainty in the mean is the sample uncertainty divided by the square root of the number of samples. The calculated viscosity is therefore the average of all time intervals which start on multiples of 300 fs, and the uncertainty is the standard deviation of all the samples, divided by the square root of the number of samples.

5.2.2 Results

Figure 5.4 shows the results of the simulation. In the top half, the pressure autocorrelation, averaged over both time intervals and components, is shown. The autocorrelation fluctuates rapidly around zero, and the magnitude decays rapidly so that the integral converges. After 300 fs the autocorrelation is approximately zero, so that time intervals starting more than 300 fs apart can be viewed as statistically independent measurements.

The viscosity is then calculated as the time integral of the autocorrelation, as seen in the bottom half of the plot. After a converging phase of approximately 0.4 ps the integral has plateaued. The final estimate for the viscosity of this water model is thus

$$\eta = 0.16(1)\ \text{mPa s}. \tag{5.1}$$

This is considerably smaller than the experimental value [38] of 0.89 mPa s, but not entirely unexpected. Previous measurements of the diffusivity [11] have shown a diffusion coefficient which is four to five times as large as the experimental value, whereas the experimental viscosity is approximately five and a half times as large as the one measured by me. Since the diffusivity of spheres in a liquid with a small Reynolds number is inversely proportional to the viscosity (the Stokes-Einstein relation [39]), these results are in good agreement.

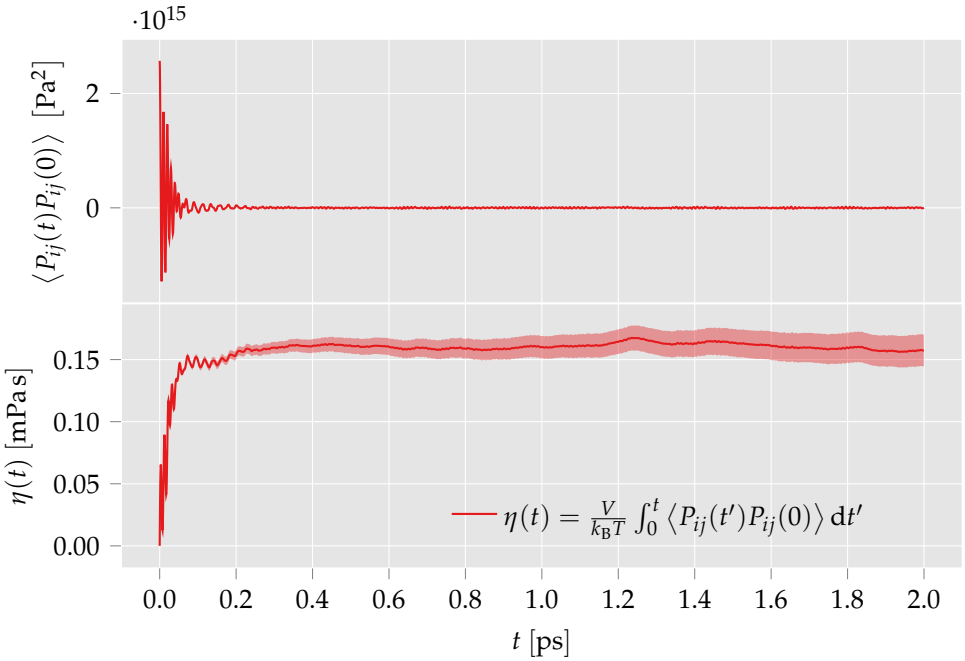


Figure 5.4: Autocorrelation of the off-diagonal pressure tensor elements and Green-Kubo integral estimate of viscosity from a water simulation at 300 K and 1.0 g/cm^3. The estimated viscosity of 0.16(1) mPa s is considerably smaller than the literature value of 0.89 mPa s, although it is consistent with observations of a high diffusivity from the authors of the force field.

Chapter 6

Flow

The viscosity measurements in the previous section showed that the transport properties of the force field are not accurate for water. Nonetheless, I have chosen to perform a single set of flow simulations with the same force field and see what happens. In this chapter, I describe the setup, implementation and results of simulating water flow through a silica system. My selected geometry consists of two silica slabs connected by cylinders. Flow is induced by applying a force to the water molecules. The force is parallel to the slabs and perpendicular to the cylinders. I compare the resulting velocity field with continuum results in order to further evaluate the aptitude of the force field for flow simulations.

6.1 Water flow around a silica nanopillar between slabs

6.1.1 Setup and implementation

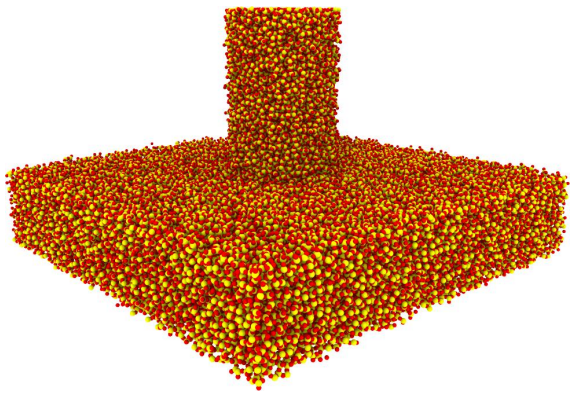


Figure 6.1: Rendering of the system used in the flow simulations. Silicon atoms are yellow, oxygen atoms are red. The direction of flow is perpendicular to the cylinder axis. Since there are periodic boundary conditions in all directions, the water is flowing between two slabs of silica. The slab has dimensions equal to $30 \times 30 \times 6$ unit cells of β-cristobalite ($a = 7.16$ Å), and the height and radius of the cylinder are 12 and 4 unit cells.

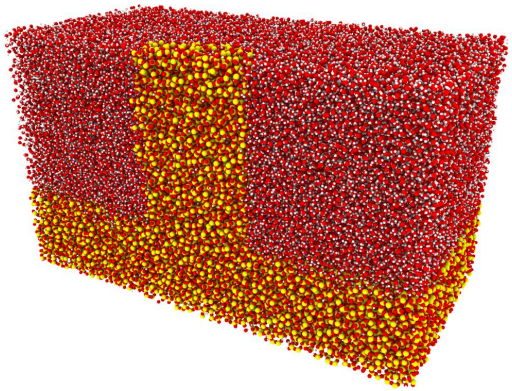


Figure 6.2: with water

Figure 6.1 on the preceding page shows the geometry used for most of the flow simulations. The idea is simply to pack the system with water, add a force to the water molecules and study the resulting velocity field. Additionally, I will compare the results from the molecular dynamics simulations with continuum calculations.

The system shown in figure 6.1 is cut out of a block of amorphous silica created with the rapid melting and cooling procedure described in section 5.1 on page 41. There are 143 937 atoms in the rendered system. Before adding water, hydroxy groups and hydrogen atoms are added to undercoordinated atoms on the surfaces of the system. This makes the system more realistically reactive, since no real systems in nature have a large number of dangling bonds on the surface.

124 513 water molecules were added to the vacuum around the region with PACKMOL, giving a density of 1.0 g/cm^3. Figure 6.2 shows the resulting configuration. During the flow simulations, the water molecules are integrated without a thermostat, while a constant temperature of 300 K is maintained by applying a Nosé-Hoover thermostat to the top and bottom thirds of the silica slab. The middle third of the slab is frozen so that the system is kept in place.

6.1.2 Results

Approach to equilibrium

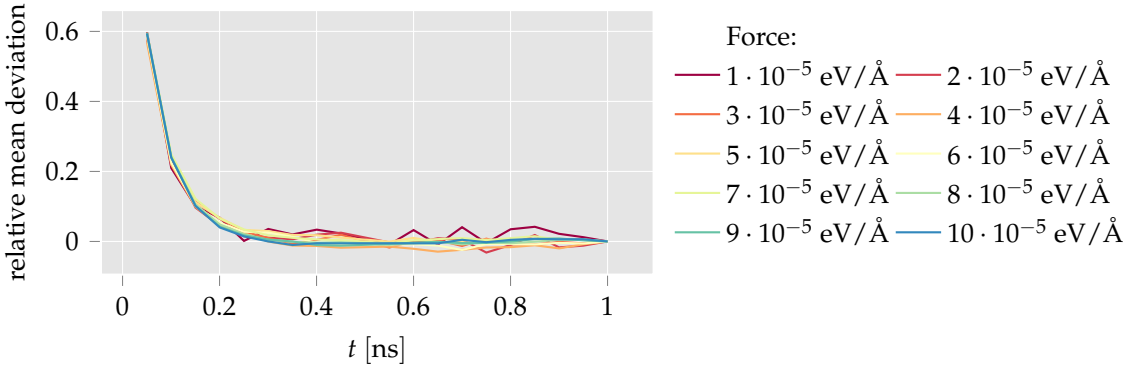
I am mainly interested in the steady state velocity field, as the molecular dynamics results require too much averaging to be able to show the time dependence of the velocity field and its approach to steady state. The velocity fields reported are therefore the average velocity fields after the system has reached steady state. I have chosen to study the relative mean deviation from the final velocity field as a way to determine when the velocity field is ready to be sampled, using the results from the LAMMPS fix described in section 3.1 on page 27.

The simulations were run for $t_1 = 1$ ns, so the measure is defined as

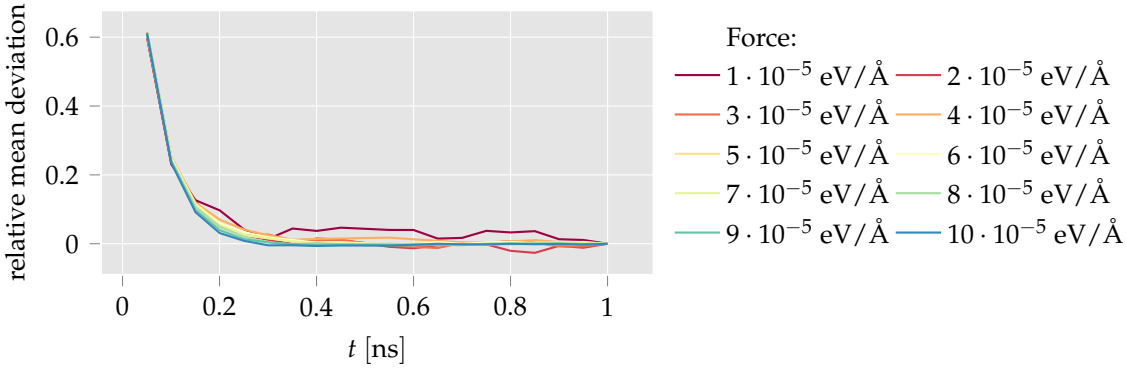
$$\text{relative mean deviation}(t) = \frac{\int v(\vec{r}, t_1) - v(\vec{r}, t) \mathrm{d}^3\vec{r}}{\int v(\vec{r}, t_1)\, \mathrm{d}^3\vec{r}} \approx \frac{\sum_{i=1,j=1,k=1,l=1}^{N_x,N_y,N_z,3} v_{i,j,k,l}(t_1) - v_{i,j,k,l}(t)}{\sum_{i=1,j=1,k=1,l=1}^{N_x,N_y,N_z,3} v_{i,j,k,l}(t_1)}, \tag{6.1}$$

where $\vec{v}_{i,j,k}$ denotes the average velocity vector in a spatial bin through the preceding time interval.

Figure 6.3 shows the results. After 0.5 ns, the water flow appears to have reached its steady state. The velocities presented in the remaining parts of this chapter are therefore averaged over the final 0.5 ns of the simulation.



(a) $L_x = 30a = 214.8$ Å



(b) $L_x = 60a = 429.6$ Å

Figure 6.3: Mean relative difference between the velocity profile at a time t and the final time, $t_1 = 1$ ns, which measures the extent to which the system has reached equilibrium. Equilibrium is reached slightly quicker with the stronger forces than with the weaker forces, but all systems reach equilibrium within 0.5 ns.

Continuum results

In order to evaluate the quality of my molecular dynamics results, I need something to which they can be compared. There are obviously no experimental results for velocity fields at nanoscale, so I must compare my velocity fields with those of continuum fluid mechanics. In order to obtain these, I must solve the Navier-Stokes equations. These are notoriously difficult to solve analytically, and my geometry is non-trivial, so the finite element method was used.

COMSOL Multiphysics® was used to calculate the continuum velocity field, with help from Kjetil Thøgersen. Since continuum mechanics is not a focus of my thesis, I opted for this simplified solution.

In these continuum simulations, the silica slabs and cylinder have been replaced by no-slip boundary conditions. Ideally, I would want periodic boundary conditions in the x- and y-direction as in the molecular dynamics simulations, but the solver would not converge with more than one set of periodic boundary conditions. Instead, the $y = 0$ and $y = L_y$ boundary conditions were set as symmetric, which should not affect the results. Figure 6.4 shows the finite element mesh used by the solver. The mesh is finer where the velocity field is expected to be less uniform.

Figure 6.5 on the facing page shows the velocity magnitude as a function of x and y halfway up on the cylinder. The velocity field is clearly asymmetric along the direction of flow (the x-axis), while it is symmetric about the cylinder in the y-direction. I also show continuum velocity profiles in figure 6.8 on page 52 in comparison with the molecular dynamics results. See the next section for details.

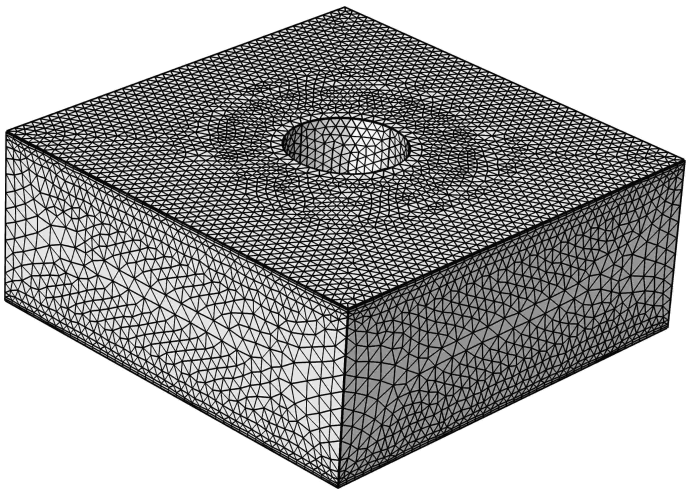


Figure 6.4: Finite element mesh used for the continuum simulation.

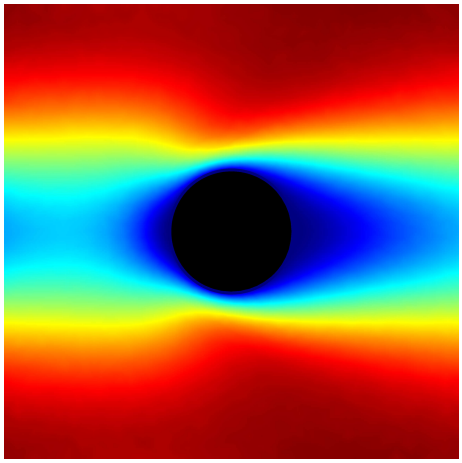


Figure 6.5: Velocity magnitude in the xy-plane at half the cylinder height. The x-axis, i.e. the direction of flow, points to the right. Blue means low velocity, red means high velocity.

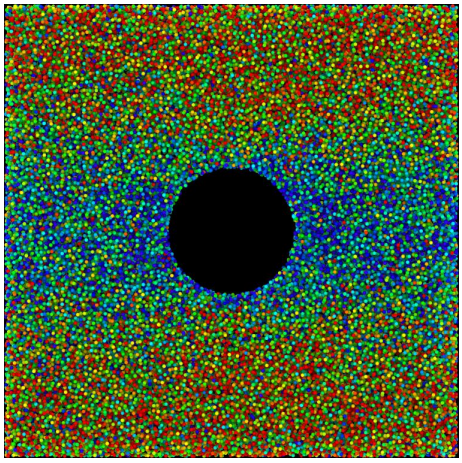


Figure 6.6: x-component of the velocity field in the xy-plane, i.e. the same results as in figure 6.5. Only oxygen atoms are shown. Blue atoms are moving slowly and red atoms are moving quickly, but the colour scale is not the same as in figure 6.5. This figure also shows that the velocities are larger further away from the cylinder, but the fluctuations are so large that these single atom velocities can barely be used for a qualitative inspection. More averaging is necessary in order to properly compare with continuum results.

Molecular dynamics results

A continuum simulation gives the steady state velocity field with little noise, as seen in figure 6.5. A molecular dynamics simulation, on the other hand, only gives the velocities of the individual particles, and these fluctuate wildly, as seen in figure 6.6 on the preceding page. My LAMMPS fix described in section 3.1 on page 27 does time averaging and some spatial averaging, but the averaging over $400 \times 400 \times 50$ spatial bins proved insufficient for obtaining anything other than noise. I have therefore chosen to take the averaging much further.

My choice of measure is to look at how the x-component of the velocity varies as a function of x, when averaged over all z positions and a selection of y positions near $L_y/2$, specifically those with $y/L_y \in [0.45, 0.55]$. The selection is visualised in figure 6.7 on the facing page.

Figure 6.8 on page 52 shows the resulting velocity profiles for molecular dynamics and continuum fluid mechanics simulations. The simulations have been carried out for two different system lengths (note that L_y is the same in both cases) and ten different driving forces. Although the graph shapes of the molecular dynamics and continuum velocity profiles are very similar, a discrepancy of approximately 20 % is seen in the magnitude of the velocities.

Molecular dynamics cannot be expected to exactly reproduce the velocity fields of fluid mechanics (or vice versa) — otherwise, this part of my thesis would be pointless. For example, velocity fields in molecular dynamics should not completely fulfill the no-slip boundary conditions assumed in the continuum simulations. Nevertheless, the observed differences are too large for this to be a satisfactory explanation without further investigation.

Instead, it seems that the continuum simulations may have been given the wrong viscosity, due to a problem with my thermostatting procedure. The viscosity estimated in section 5.2 on page 42 appears to be accurate, but it was measured at a temperature of 300 K. Ideally, this should also be the temperature in the flowing water, since it is in contact with silica that is thermostatted to the same temperature. The velocity distributions shown in figure 6.9 on page 53 can be used to estimate an effective temperature in the flowing water.

From the Maxwell-Boltzmann distribution, the velocity components of non-flowing water in equilibrium should be normally distributed around zero with a standard deviation of $\sqrt{k_\mathrm{B}T/m}$. Figure 6.9 shows that all three velocity components are indeed normally distributed, although v_x is shifted since the water is flowing in the x-direction. v_y and v_z are therefore used to estimate the temperature, and the results are

$$T_y = 367.6\ \mathrm{K} \qquad \text{and} \qquad T_z = 366.4\ \mathrm{K}, \tag{6.2}$$

giving an effective temperature of approximately 367 K. This is a much higher temperature than the one used to estimate the viscosity, which one might be tempted to think explains why the molecular dynamics velocities are so much smaller than the continuum velocities. However, viscosities decrease and velocities increase with an increased temperature, so the differences should have been the other way around.

Figure 6.10 on page 53 shows the result of recalculating the viscosity at the effective temperature, $T = 367$ K. The viscosity should be lower at higher temperatures, but this is clearly not the case for this potential — the new viscosity is approximately the same as (or slightly higher than) the one measured at 300 K. This further indicates that the transport properties of the potential are not quite as they should be. Additionally, the slight increase in viscosity is nowhere near enough to explain the 20 % difference in velocities between continuum and molecular dynamics.

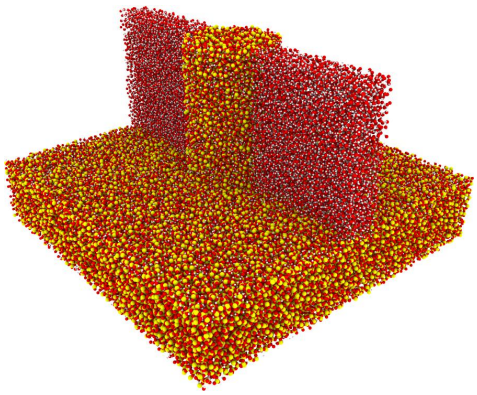


Figure 6.7: Water selection with $y/L_y \in [0.45, 0.55]$ used to measure the x-component of the velocity field as a function of x. $v_x(x)$ is obtained by averaging over all atoms in both the y- and the z-direction.

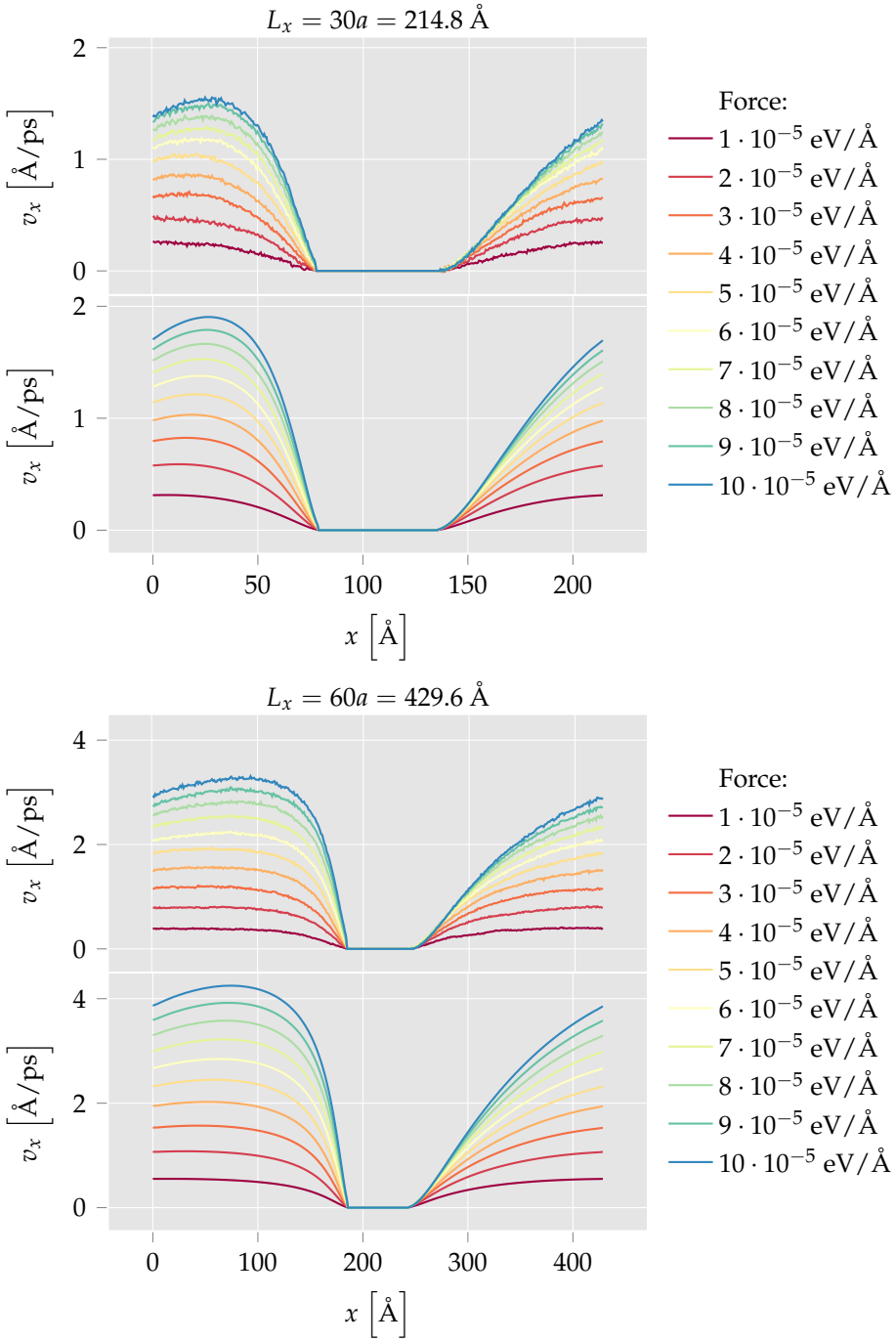


Figure 6.8: Velocity profile of the x-component as a function of x for different forces and system sizes. The top panels show molecular dynamics results, while the bottom panels show continuum results. In both cases, the velocities have been averaged over the water molecules or region shown in figure 6.7 on the preceding page. The molecular dynamics velocities are consistently smaller than the continuum velocities, and the difference is largest for large forces.

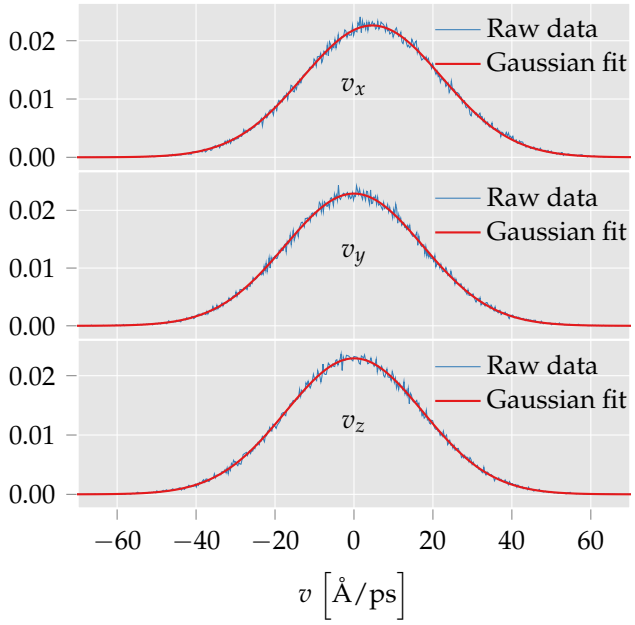


Figure 6.9: Velocity distributions of the hydrogen atoms in the flow simulation with $L_x = 30a$ and $F = 10^{-4}$ eV/Å. All graphs closely follow a Gaussian, as expected from the Maxwell-Boltzmann distribution, but the x-component is shifted since this is the direction of flow.

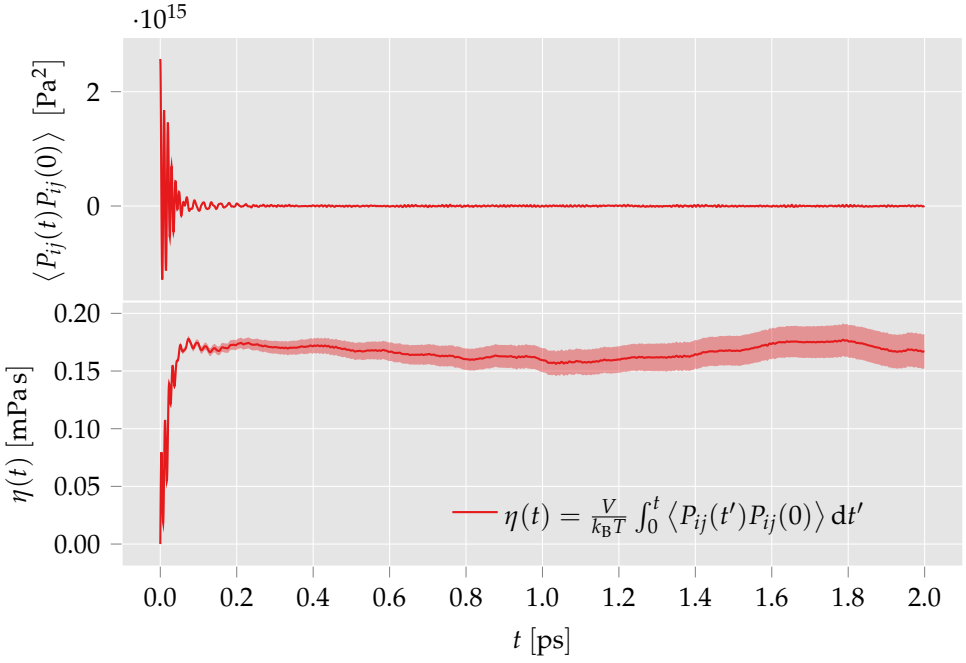


Figure 6.10: Recalculation of the viscosity as in figure 5.4 on page 44, but this time at $T = 367$ K and with thrice the number of water molecules (6000) since this simulation was run on a GPU. Despite the significantly higher temperature, the viscosity is approximately the same or even slightly larger, which does not agree with expectations.

Chapter 7

Creep

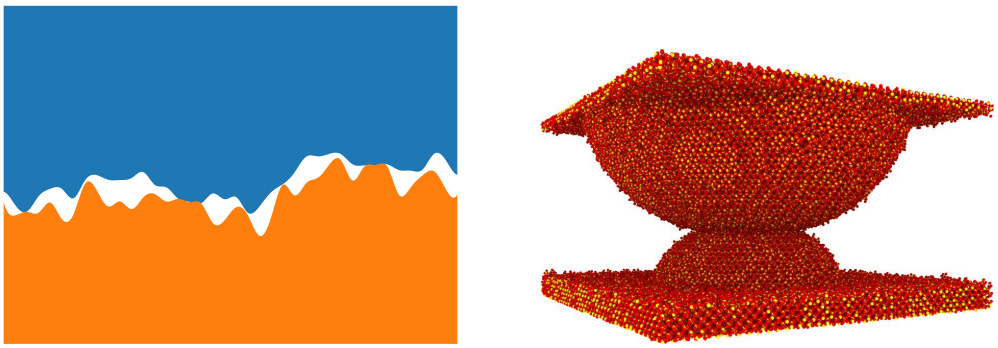


Figure 7.1: Illustration of the idea behind simulating two opposing sphere caps. These are viewed as a small part of a large system consisting of two rough slabs in contact, as seen in the left image. The right image shows two asperities in contact from my molecular dynamics simulations.

In this chapter, I describe my simulations of creep in a system consisting of two spherical asperities, as seen in the right image of figure 7.1. All surfaces in nature are rough, and rough surfaces consist of many asperities. When two slabs of a material are pushed together, contact is formed through opposing asperities, as shown in the left panel of figure 7.1. More and more contact is formed as the asperities are deformed. My goal here is to study this deformation process, called *creep*, in a system of two opposing asperities.

7.1 Analytical work

Aharonov and Scholz [4] start off from the assumption of thermally activated creep, meaning that the creep rate should be proportional to an Arrhenius factor,

$$\frac{\mathrm{d}h}{\mathrm{d}t} = -V_0 \exp\left(-\frac{Q_V - N_\mathrm{A}\Omega_V\sigma_\mathrm{c}}{RT_\mathrm{c}}\right), \tag{7.1}$$

where V_0 is some velocity, Q_V is an activation energy, σ_c is the contact stress, N_A is Avogadro's constant and Ω_V is an activation volume. h is the height of the system under stress, which consists of two blocks which are in contact through asperities. After some manipulations and assumptions, they end up with the contact stress as a function of time,

$$\sigma_c(t) = \sigma_c^0 \left[1 - b' \ln\left(1 + \frac{t}{t_c} \right) \right]. \tag{7.2}$$

In previous work, σ_c^0 was assumed to fulfil $N_A \Omega_V \sigma_c^0 = Q_V$, but Aharonov and Scholz lift this restriction and introduce the dimensionless constant $B \in [0, 1]$, with $N_A \Omega_V \sigma_c^0 = B Q_V$. Additionally, the following variables will prove useful:

$$b' = \frac{RT_c}{BQ_V}, \quad E_{tc} = Q_V - N_A \Omega_V \sigma_c^0 = (1 - B) Q_V \quad \text{and} \quad V_0' = V_0 e^{-\frac{E_{tc}}{RT}}. \tag{7.3}$$

My idea is to insert the derived stress, equation (7.2), into the equation for the creep velocity, equation (7.1) on the previous page, giving

$$\frac{dh}{dt} = -V_0 \exp\left(-\frac{Q_V}{RT_c} + \frac{N_A \Omega_V \sigma_c^0}{RT_c} \left[1 - b' \ln\left(1 + \frac{t}{t_c} \right) \right] \right) \tag{7.4}$$

$$= -V_0 \exp\left(-\frac{Q_V - N_A \Omega_V \sigma_c^0}{RT_c} - \frac{N_A \Omega_V \sigma_c^0}{RT_c} b' \ln\left(1 + \frac{t}{t_c} \right) \right). \tag{7.5}$$

This equation does not look particularly pretty, but it simplifies nicely since

$$\frac{N_A \Omega_V \sigma_c^0}{RT_c} b' = \frac{BQ_V}{RT_c} b' = 1, \tag{7.6}$$

so that

$$\frac{dh}{dt} = -V_0 e^{-\frac{E_{tc}}{RT_c}} \exp\left(-\ln\left(1 + \frac{t}{t_c} \right) \right) = -\frac{V_0'}{1 + \frac{t}{t_c}}. \tag{7.7}$$

Finally, I integrate this from $t = 0$ with $h(0) = h_0$ and find

$$h(t) = h_0 - V_0' t_c \ln\left(1 + \frac{t}{t_c} \right). \tag{7.8}$$

The quantity h and its time dependence is simple to measure when performing a molecular dynamics simulation where creep might be expected. Consequently, it serves as a good first indication of whether these microphysical but continuum-scale assumptions and derivations manifest on the atomic scale. In my simulations, I study $h(t)$ for a simple system of two spherical asperities in contact and compare the results for silica in vacuum, passivated silica and silica in water.

7.2 General simulation procedure

Figure 7.2 on the facing page shows a typical setup in the creep simulations. The general idea is simply to push two sphere caps together, monitor the system height as a function of time and compare with the analytical prediction of equation (7.8). All simulations follow the same procedure:

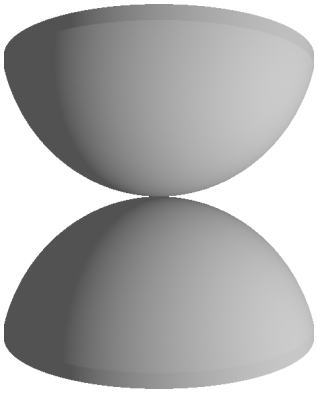


Figure 7.2: Sketch of one system used in creep simulations.

1. Compress the system by gradually reducing the system box size. The sphere caps start a small distance apart and are moved towards each other until they are significantly compressed. `fix deform` is used, as it removes the need to keep parts of the system artificially rigid. Additionally, this approach produces the least noise out of all those tested. `fix deform` simply changes the dimensions of the box, and the forces across the periodic boundaries compress the system.
2. Keep the system at constant compression. The system is integrated with a constant volume, so that constant compression is maintained. This allows the system to "settle down". When the system has reached equilibrium, the stress required to maintain the compression is measured.
3. Apply a constant normal stress equal to the equilibrium stress measured in the previous step. I have used `fix npt` with the `z` keyword. Under this constant stress, the system is slowly further compressed and $h(t)$ is easily measured.

7.3 Simulations

7.3.1 Asymmetric system with silica in vacuum

Specifications

Figure 7.3 on the following page shows the geometry of the system. In total, the system consists of 463 058 atoms. Initially, when the sphere caps are one unit cell (7.16 Å) apart, the dimensions of the system are 286.4 Å $\times$ 286.4 Å $\times$ 207.64 Å, although L_z is reduced to approximately 186 Å during compression. The system is initialised in a β-cristobalite structure. The two spheres have curvature radii of 12 and 20 unit cells, and the heights of the sphere caps are 75 % of the radii. While the system has periodic boundary conditions in all directions, the upper sphere caps have a spacing of one unit cell in the x- and y-directions such that they do not interact.

The sphere caps start off with a vertical spacing of one unit cell and are moved 6 unit cells

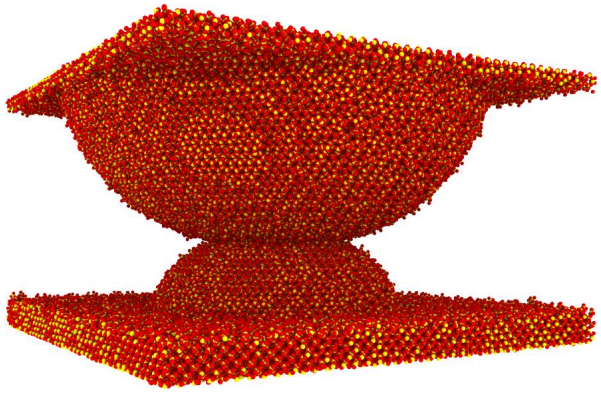


Figure 7.3: Rendering of the system before the constant stress is applied. Yellow atoms are silicon, red atoms are oxygen. The z-axis points ↑.

towards each other during 200 ps for a total "compression" of 5 unit cells, 35.8 Å. This gave a measured and applied normal stress of 7.1 GPa. Having reached the compressed configuration (shown in figure 7.3), the system is equilibrated for another 200 ps. The z-component of the normal stress is averaged during another 40 ps and used as the external pressure for the barostat in the next step.

Finally, the creep process is studied by applying a constant normal stress and recording L_z as a function of time. L_z is fitted to equation (7.8) on page 56 with the parameters h_0, V_0' and t_c via Scipy's `curve_fit`, which uses the Levenberg-Marquardt algorithm. This simulation was run for 200 ns, using the checkpointing trick outlined in section 3.4.3 on page 34. The damping time of the barostat was initially set to 1 ps (although 5 ps was also tested — see figure 7.5 on the facing page).

Results with $t_{\mathrm{damp}} = 1$ ps

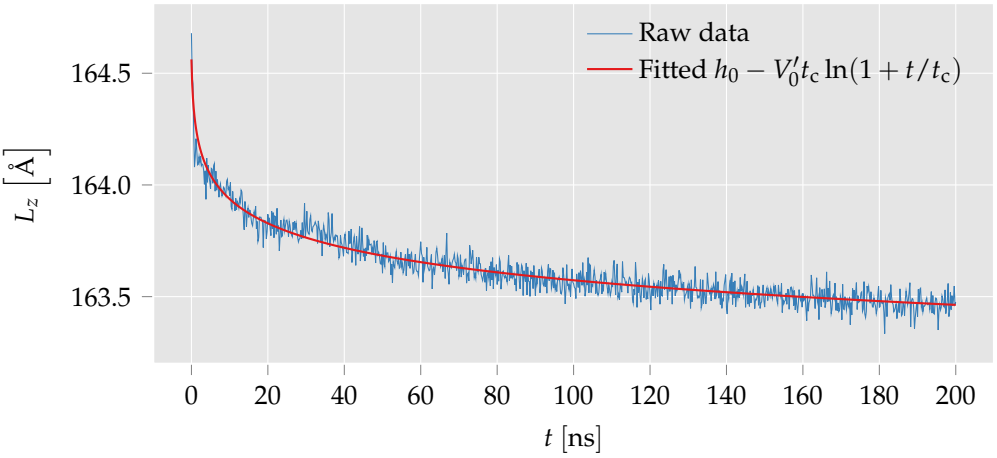


Figure 7.4: Vertical height of the system shown in figure 7.3 when a constant normal stress is applied. The result is compared with the analytical prediction of equation (7.8) on page 56, with which it shows good agreement.

Figure 7.4 on the preceding page shows the vertical height of the system as a function of time after the constant normal stress has been applied. The creep rate decays rapidly. Agreement with equation (7.8) on page 56 is good, except a small "bump" around 25 – 40 ns. The fitting parameters found by `scipy.optimize.curve_fit` are

$$t_c = 0.20(5)\ \text{ns}, \qquad h_0 = 164.56(4)\ \text{Å} \qquad \text{and} \qquad V_0' = 0.8(2)\ \text{Å/ns}. \tag{7.9}$$

Results with $t_{\text{damp}} = 5$ ps

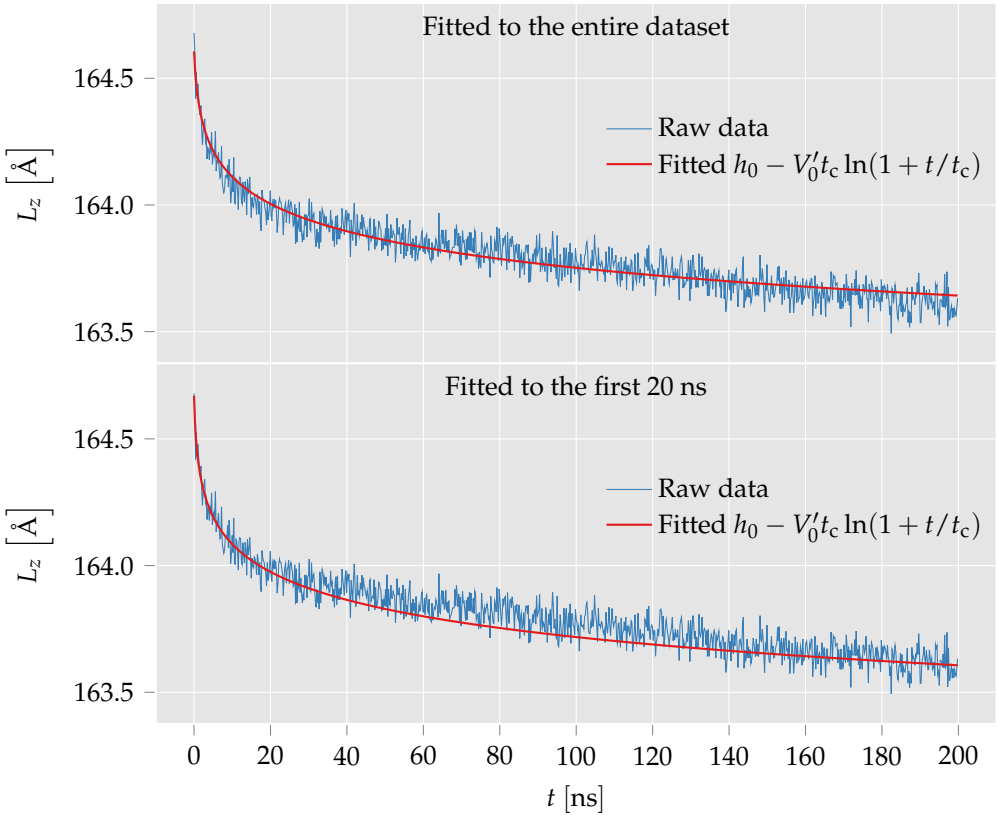


Figure 7.5: Vertical height of the system shown in figure 7.3 on the facing page when a constant normal stress is applied and the barostat damping time is $t_{\text{damp}} = 5$ ps. The result is compared with the analytical prediction of equation (7.8) on page 56. While the overall trend is similar to that of the theoretical prediction, the fit is significantly better in figure 7.4 on the facing page, where the barostat damping time is $t_{\text{damp}} = 1$ ps.

Figure 7.5 shows the vertical height of the system as a function of time, after the constant normal stress has been applied. The fitting parameters found by `scipy.optimize.curve_fit` are

$$t_c = 0.46(12)\ \text{ns}, \qquad h_0 = 164.60(3)\ \text{Å} \qquad \text{and} \qquad V_0' = 0.34(8)\ \text{Å/ns}. \tag{7.10}$$

The creep graphs clearly deviate from the theoretical prediction, unlike the ones in figure 7.4 with a shorter barostat damping time. A longer damping time means a "weaker" barostat, so that the pressure is allowed to fluctuate more and the applied normal stress

becomes less constant. I therefore choose to regard the results of figure 7.4 as more relevant and use a damping time of 1.0 ps for the remainder of the creep simulations, although this is something that should be studied more closely.

It may be that the observed deviation is just a longer, more damped version of the $25 - 40$ ns "bump" in the previous result, and that the fit would be as good if the simulation were continued for e.g. a microsecond. Indeed, fitting the analytical expression to only the first 20 ns gives a plot similar to the first 50 ns of figure 7.4. The parameters of this fit are

$$t_c = 0.22(1) \text{ ns}, \qquad h_0 = 164.68(1) \text{ Å} \qquad \text{and} \qquad V_0' = 0.69(2) \text{ Å/ns}, \tag{7.11}$$

which are similar to those found with $t_{\text{damp}} = 1.0$ ps, equation (7.9). Since the 200 ns simulation took a week, I leave the exploration of this as future work.

7.3.2 Symmetric silica-vacuum system, long-timescale behaviour

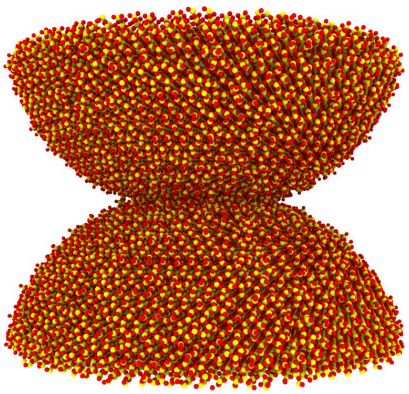


Figure 7.6: Rendering of the symmetric silica-vacuum system. This geometry is used for the remainder of the creep simulations.

Specifications

Figure 7.6 shows the simulated system, which is mostly similar to the asymmetric system in figure 7.3 on page 58. The radii of the sphere caps are 12 unit cells, and the height of each sphere cap is 75 % of its radius. Additionally, there is a disk with thickness 1 unit cell at the top and bottom of the system, giving a total of 130 408 atoms. There are periodic boundary conditions in all directions, but the box is large enough that the sphere caps do not interact with each other across the x- and y-boundaries (so that water can later be added to the system).

The simulation procedure is otherwise identical — move the spheres together, measure normal stress, apply this stress and observe creep. Since the system is smaller, the spheres are only moved 3 unit cells towards each other, from an initial spacing of 1 unit cell. This resulted in a normal stress of 7.16 GPa. Additionally, the creep process is studied for an entire microsecond.

Results

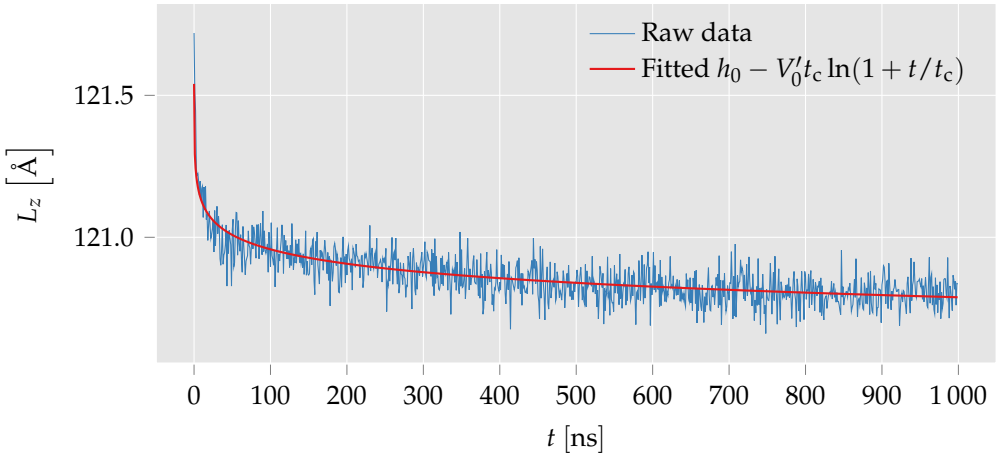


Figure 7.7: Long-timescale behaviour of the vertical height of the system shown in figure 7.6 on the facing page when a constant normal stress is applied. The result is compared with the analytical prediction of equation (7.8) on page 56. The fit is quite good, although there are deviations, the fit is unstable and the required parameters are not entirely realistic. See the text for details.

Figure 7.7 shows the creep in the symmetric system in figure 7.6 on the preceding page when it is run for a whole microsecond. The agreement with the theoretical prediction looks fairly good, with the parameters

$$t_c = 0.035(7) \text{ ns}, \qquad h_0 = 121.54(1) \text{ Å} \qquad \text{and} \qquad V_0' = 2.1(4) \text{ Å/ns}. \tag{7.12}$$

Unfortunately, the fitting procedure proved quite troublesome. The algorithm would not converge unless given a smoothed version of the data, and even then the uncertainty in the parameters is fairly large. Finally, there was a clear tendency to force a fit by reducing the critical time t_c and cranking up the initial creep rate V_0'. This presumably happens because the theoretical prediction is a logarithm, which will never truly converge, while the measured L_z seems to plateau after approximately 800 ns.

7.3.3 Effect of temperature

Specifications

The underlying assumption has been thermally activated creep. Consequently, the creep should have a very clear temperature dependence. To study this, I have taken the simulation from the previous section, shortened it to 50 ns and repeated it for four different temperatures (400 K, 450 K, 500 K and 500 K) by varying the temperature of the thermostat.

Results

Figure 7.8 on the facing page shows the resulting system heights as a function of time. Since the creep process is assumed to be thermally activated, it should be faster for higher temperatures. This is also seen in the figure, *except* for the creep simulation at 400 K. The $T = 400$ K results also show the worst agreement with the analytical expression, together with $T = 450$ K. The fit is notably better with 550 K.

The 400 K results were very unexpected. Even before the surge around $3-4$ ns, the 400 K creep is keeping up with the 550 K creep. My hypothesis is that this is caused by my choice of crystal structure, β-cristobalite. β-cristobalite is a high temperature polymorph of SiO_2, meaning it is only stable at high temperatures. At the temperatures used in these simulations, it is unstable and should *slowly* transform into one of the lower temperature forms, such as α- or β-quartz or β-tridymite. These are all denser than β-cristobalite, and thus a beginning phase transition will reduce the system height.

It may therefore be that 400 K is a low enough temperature for the phase transition to be a dominant creep process, and that this is why the $T = 400$ K shows the most creep. Figure 7.9 on the next page visually compares the structure for $T = 400$ K and $T = 500$ K. For $T = 400$ K, the structure looks different in the top half of the bottom asperity.

7.3.4 Effect of passivation and water

No systems in nature have dangling bonds on their surface, and you also find very little vacuum. In this section, I add passivating hydroxy groups to the surface of the sphere caps and water to the void between them and study how the results change when the system is made more realistic.

Specifications

The silica part of the system is identical to the one in the long-timescale, symmetric simulations (section 7.3.2 on page 60), and the results for the silica-vacuum system are the same as for the first 50 ns of that simulation. This silica-vacuum system is then passivated by adding hydroxy groups to the surface. Finally, water is added to the void in the passivated system via PACKMOL. Each of the three resulting systems is used in a creep simulation identical to the one in section 7.3.2, except for the water system.

My intention is not to simulate the compression of water confined to a nanopore, but rather the creep process in silica asperities with water between them. It is therefore important that the system allows the water to escape the region between the sphere caps. My solution has been to also add a barostat in the x- and y-directions during the initial compression phase. The pressure was set to 100 MPa, as this should keep water liquid at the set temperature, 500 K.

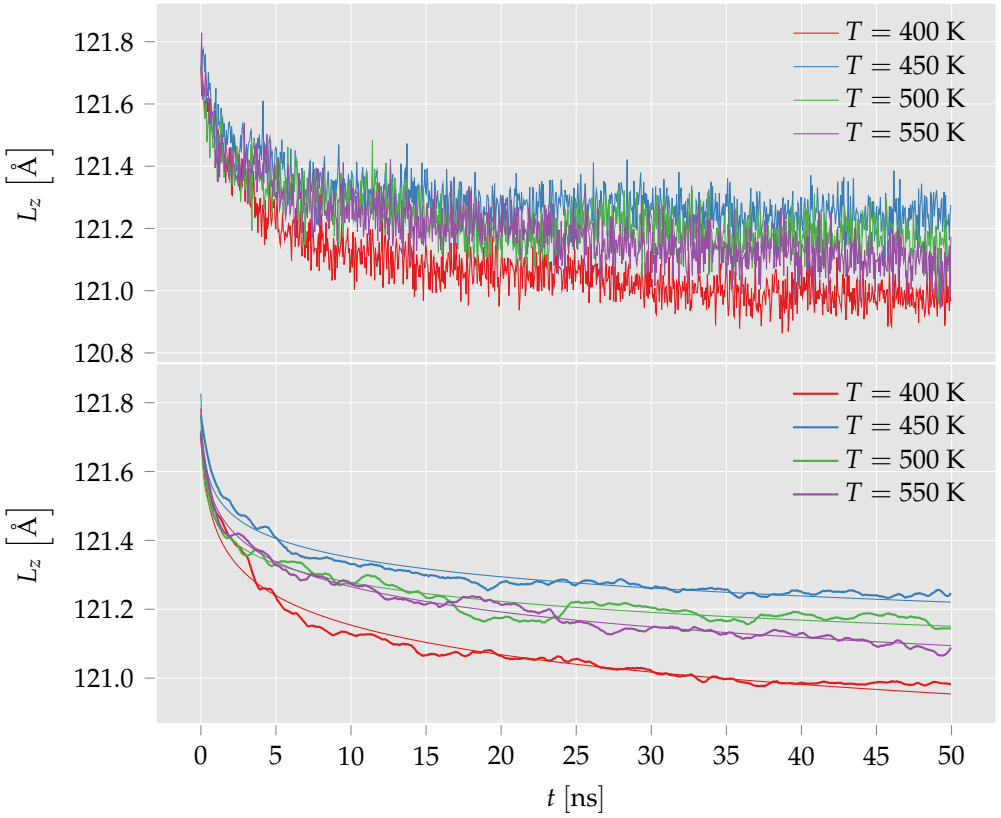
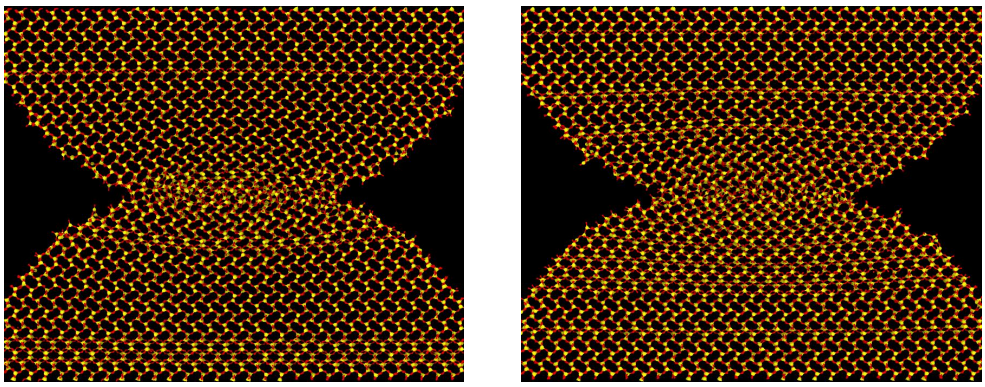


Figure 7.8: Creep curves in the symmetric silica-vacuum system for four different temperatures. The top panel shows the raw data, while the bottom panel shows smoothed versions and their fits to the theoretical prediction. The creep should increase with temperature, which fits with the observed results except for $T = 400$ K.

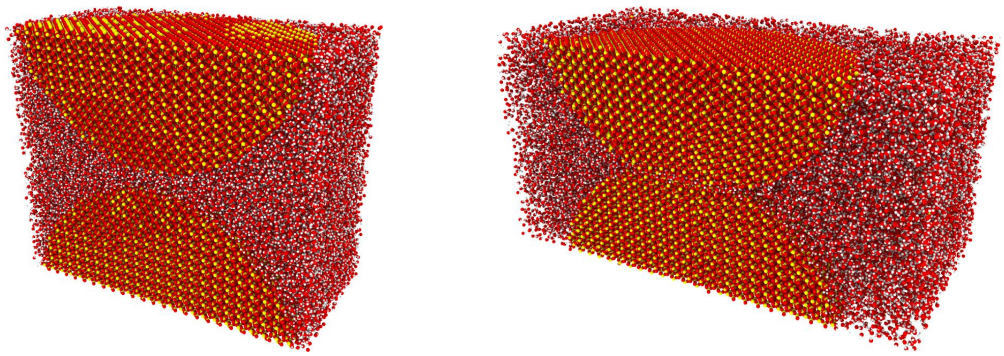


(a) $T = 500$ K. **(b)** $T = 500$ K.

Figure 7.9: Comparison of structure in silica-vacuum creep system at two different temperatures. Silicon-oxygen bonds have been added to highlight the structure. The view is along the [110] zone axis. The structure just below the contact point looks different for $T = 400$ K.

In the final phase, when the constant normal stress is applied, an xyz-barostat proved unstable when the difference in xy- and z-stress is so large. Consequently, the barostat was replaced by constant L_x and L_y in this step. The bulk modulus of water is 2.2 GPa [40], so the added pressure when $L_z \approx 120$ Å and $\Delta L_z \approx 1$ Å is approximately 18 MPa. This is considerably less than the 100 MPa used to equilibrate the system, hence the results should not be severely affected.

Results



(a) Initial configuration. **(b)** Final configuration.

Figure 7.10: Rendering of half the initial and final state of the compression procedure. In the initial configuration, water has been tightly packed around the sphere caps with PACKMOL. The system is then compressed in the z-direction while an xy-barostat allows it to expand horizontally to acommodate the water that has been pushed out from between the sphere caps.

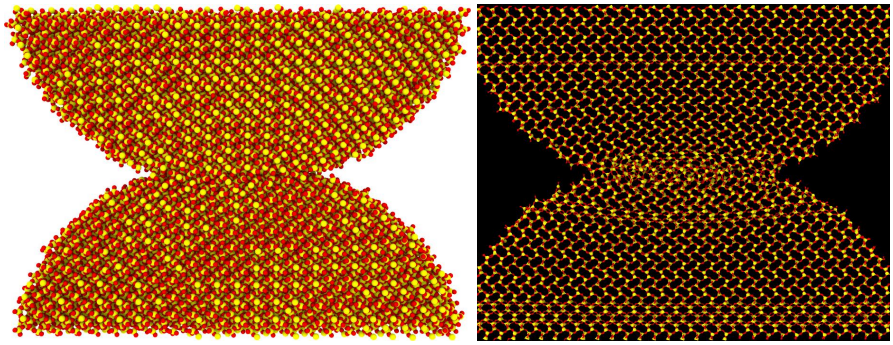
Figure 7.10 shows the result of the barostatting procedure (sliced in half). In the initial configuration, water is neatly packed around the sphere caps on all sides. The extra layer of water in the xy-plane separates the sphere caps so that they do not interact across the periodic xy boundary conditions. As the sphere caps are moved towards each other by `fix deform`, the barostat increases L_x and L_y, thereby allowing the water to escape from between the sphere caps. Even in the final configuration, a layer of water is visible between the sphere caps despite significant compression.

The resulting creep plots are shown and compared with equation (7.8) in figure 7.12 on page 67. Fitting the silica-vacuum results continues to difficult, but the passivated creep process appears to agree very well with the theoretical prediction. The water system shows decent agreement during the initial phase, but the creep rate appears to approach a constant, which is not reproducible with the logarithmic prediction.

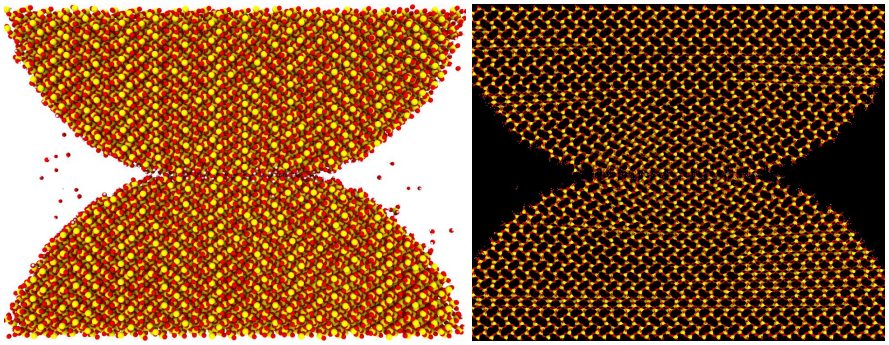
Figure 7.13 on page 68 directly compares the three creep processes, showing clear differences. The silica-vacuum and passivated silica systems show creep with a similar functional form, but the passivated system experiences significantly less creep. Visual inspection of figure 7.11 on page 66 shows that the passivated sphere caps are able to maintain their separate shapes to a larger extent than the silica-vacuum system in which the sphere caps have dangling covalent bonds.

The hydrated system shows clear differences from the other two. In figure 7.13 the hydrated system lacks the converging behaviour of the other systems, as the creep plot becomes increasingly linear instead of approaching some constant. As a consequence, the fit with the theoretical prediction is poor. Additionally, the total creep distance is much larger in the hydrated system. Previous studies have shown similar results on silica in contact with water, such as a 25 % decrease in fracture toughness [41] and general hydrolytic weakening [3].

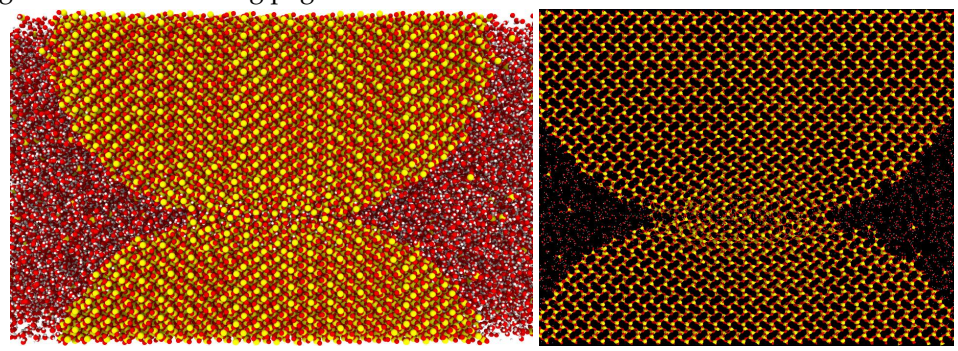
Figure 7.11 on the next page indicates the reason for the differences, namely that the layer of water between the sphere caps reacts with and diffuses into the silica, while some of the silica is dissolved in water as orthosilic acid. This leads to a compressible zone which allows for the observed increase in creep distance and rate. The solution of silica further increases the observed change in system height, and figure 7.14 on page 68 shows that the solution process is linear, possibly explaining the system height's convergence towards linearity.



(a) Silica-vacuum system. Since the surface atoms of the sphere caps have dangling bonds, the sphere caps willingly connect with each other. This allows for a greater change of shape, a higher creep rate and larger total compression than the passivated system.



(b) Passivated system. The main goal of the passivation procedure is to connect hydrogens and hydroxy groups to undercoordinated surface atoms, giving less reactive surfaces. The figure shows a smaller contact area and less deformation, giving the reduced creep rate in figure 7.12 on the facing page.



(c) Hydrated system. In figure 7.10 on page 64, there was a layer of water left between the sphere caps after the initial compression procedure. In this figure, on the other hand, it is clear that some of the water molecules have been pushed into the silica sphere caps. The silica at the interface looks to have lost its crystalline order, thus making it possible for water to enter the silica. Additionally, some silica has been dissolved in water.

Figure 7.11: The left panels show the final configurations of the creep processes, sliced in half for visualisation. The right panels highlight the silicon-oxide bonds in a slice with [110] zone axis for easier assesment of structure. There are clear differences at the interface of the sphere caps, possibly explaining the differences shown in figure 7.13 on page 68.

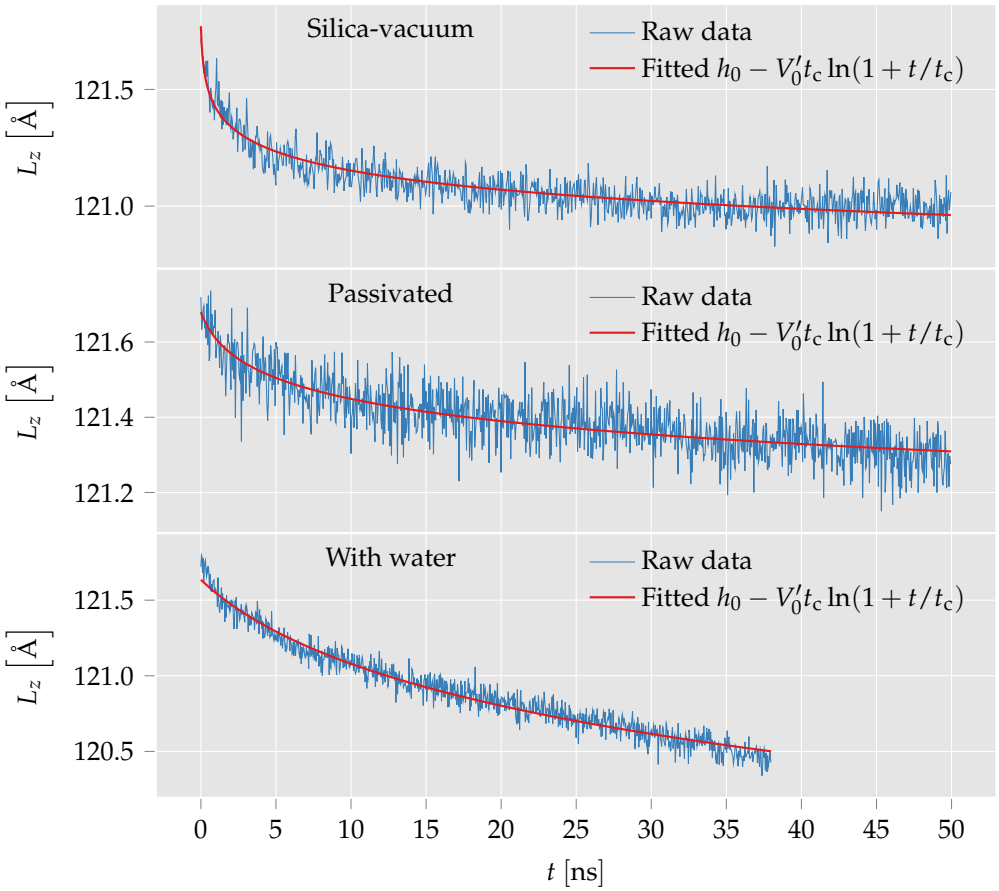


Figure 7.12: Creep in a symmetric system between a silica-vacuum system, a system where the silica has been passivated with hydroxy groups, and a passivated system where the vacuum has been filled with water. The top two graphs agree reasonably well with the theoretical prediction, although the fit of the silica-vacuum data continues to prove troublesome. The water graph approaches a linear trend rather than plateau. See also the comparison in figure 7.13 on the following page.

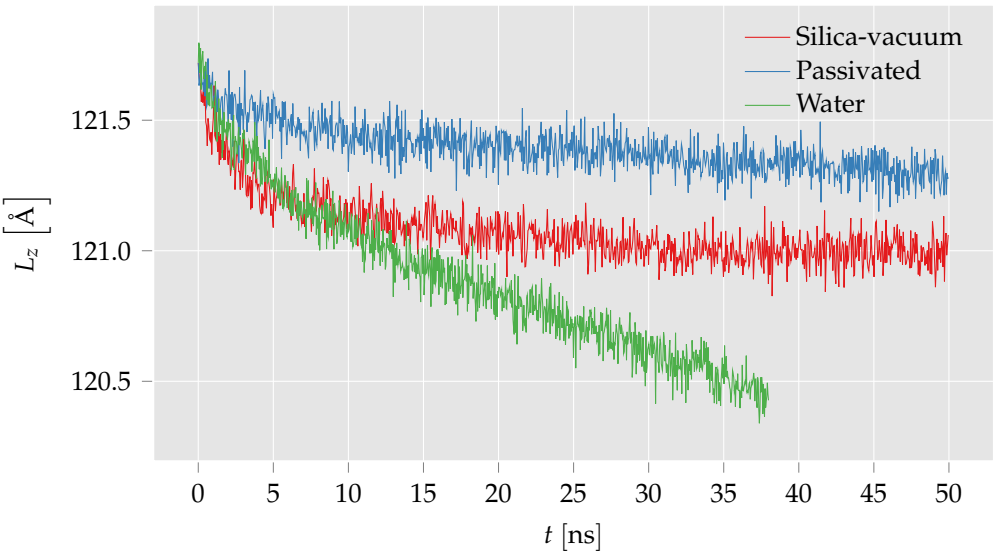


Figure 7.13: Comparison of the vertical heights shown in figure 7.12 on the previous page. The passivated system plateaus much more rapidly than the other two, while the water system shows the most creep after the initial burst of the vacuum system. See also the visualisations in figure 7.11 on page 66.

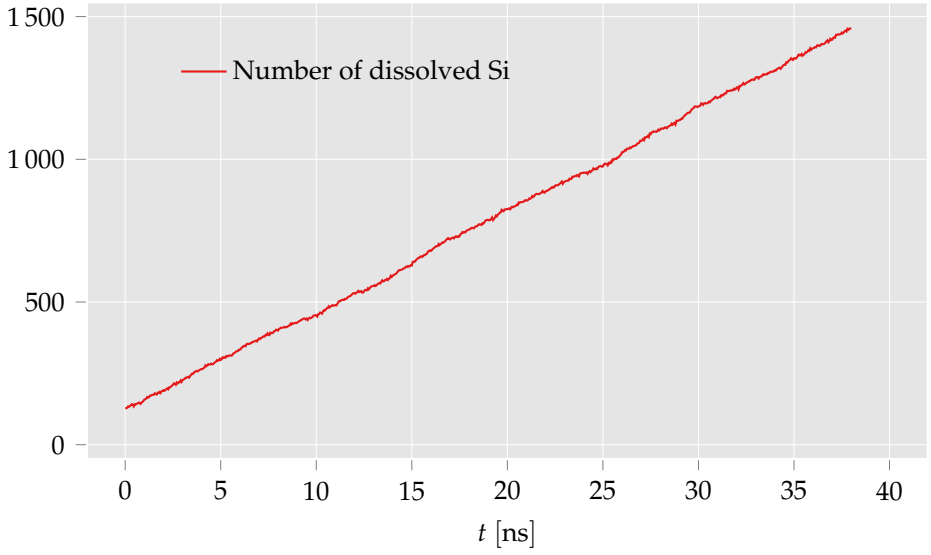


Figure 7.14: The number of silicon atoms which have been dissolved in water as orthosilic acid as a function of time. A silicon atom is defined as dissolved when its silicon coordination number is less than 2. Since the solution rate is approximately constant, this may explain why the water creep rate also approaches a constant.

Chapter 8

Summary

This thesis contains a variety of simulation techniques, molecular systems and results. What do they all mean? In this chapter, I try to summarise the main findings and draw some conclusions. Additionally, I will present my thoughts on possible extensions of this work and molecular dynamics as a method for studying transport and creep processes.

8.1 Summary and discussion

I have performed molecular dynamics simulations of two silica-water systems using LAMMPS and a recently developed force field.

First, I studied the transport properties of the force field. The viscosity of the water was measured to be $\eta = 0.16(1)$ mPa s at $T = 300$ K, which is clearly smaller than the experimental value of 0.89 mPa s. Furthermore, the viscosity was nearly unchanged or slightly increased at $T = 367$ K, while the viscosity of water is known to be a decreasing function of temperature. Combined with previous measurements which show a similar discrepancy in the diffusion coefficient, I am forced to conclude that the potential does not accurately model the transport properties of water. Since the potential has been fitted to structural properties rather than transport properties, this is not entirely unexpected.

I then studied water flow in a silica system consisting of two silica slabs connected by a cylinder. Flow was induced by adding an external force, and the simulation was repeated for two different system sizes and ten different driving forces. The resulting velocity profiles were compared with continuum results obtained with the measured viscosity. Overall, I found fairly good agreement and the shapes of the velocity profiles were very similar. There were, however, some differences in the velocity magnitudes which may be attributed to the problems with the viscosity.

Second, I studied the creep process in a silica system and the effect of adding passivating hydroxy groups and water. Based on a recent paper on a new steady state friction law for a system consisting of two slabs with asperities, I derived an expression for the system

height as a function of time. I then studied the creep process a system with two opposing spherical asperities and compared the height as a function of time with the theoretical prediction. Agreement was good, especially when using an asymmetric geometry. In the symmetric geometry, the initial burst of creep complicated the fit, although the overall trend was similar. I simulated the symmetric system for an entire microsecond and found that the system height plateaued after approximately 800 ns, which could not be reproduced by the logarithmic term in the theoretical expression.

I then added passivating hydroxy groups to the silica asperities and water to the void between the asperities, which clearly affected the creep process. When only the hydroxy groups were added, the shape of the system height as a function of time remained approximately unchanged and remained consistent with the derived prediction, but the total amount of creep was greatly reduced. In particular, the initial ($t < 5$ ns) creep rate was much smaller than without the hydroxy groups. Visually, the spherical asperities were able to preserve their shape to a larger extent with the hydroxy groups present, as they fill the dangling bonds at the silica surface and disallow bonding between the asperities, leading to reduced creep.

Water had the opposite effect and appears to profoundly alter the creep process. With water filling the void between the asperities, the system height as a function of time no longer agreed with my theoretical prediction. In particular, the height approached a linear function of time rather than plateau. Additionally, there was significantly more creep than in both the silica-vacuum system and the passivated system. I observed a strong effect of the water at the interface between the two asperities being pushed together. The structure of the silica at the interface had broken down, and water had started diffusing into the silica asperities. The hydrolytic weakening effect clearly influences the creep process, and the prediction based on thermally activated creep was unable to explain the results.

Along the way, I have developed several tools to simplify molecular dynamics simulations. I have written a LAMMPS extension which enables high-frequency and high-resolution sampling of the velocity field without affecting performance or requiring large amounts of storage space. Additionally, I have developed simplified interfaces to both log files and dump files, as well as the results from the LAMMPS extension. Finally, I have outlined and implemented a makefile-based approach to reproducible simulation pipelines, including parameter sweeps. In particular, I have demonstrated how this approach can be used to checkpoint long-running simulations and automate the continuation of a crashed simulation, thus mitigating the need for ultra-reliable hardware.

Molecular dynamics has proven to be an effective method for studying the creep process in a silica-water system, even though the chosen force field gave unsatisfactory results for studying the transport properties of water. Atomistic resolution was necessary to observe the chemical effect of water at the interface of two asperities being pushed together, and molecular dynamics is currently the only simulation technique capable of achieving atomistic resolution in a system with hundreds of thousands of atoms at a timescale of tens or hundreds of nanoseconds. LAMMPS has been an invaluable tool in my simulations, with its large array of built-in features allowing me to focus on the physical system at hand rather than implementation details. I have especially benefited from its parallel performance as well as GPU capabilities, without which my simulations would have taken

approximately a century.

Like all other simulation methods, molecular dynamics is not without its flaws. The flow simulations and viscosity measurements clearly showed that the force field is not a perfect model for water. Consequently, one should never take molecular dynamics results at face value without checking the kind of experimental data to which the parameters have been fitted. Furthermore, some simplifications are always made in molecular dynamics simulations. For example, I initialised the spherical asperities as perfect β-cristobalite crystals with no defects or impurities. Crystal imperfections often strongly affect mechanical properties, so this is a shortcoming of my simulations.

8.2 Conclusion

Transport properties in water and creep in a silica-water system were studied with molecular dynamics using a new force field for systems consisting of silica and water. I found the viscosity of the water model to be $\eta = 0.16(1)$ mPa s at $T = 300$ K, and the viscosity was unchanged or slightly increased at 367 K. These observations are not consistent with experimental values (0.89 mPa at 300 K). When simulating flow of this water in molecular dynamics, the velocity profiles were shaped similarly to those of continuum water with the same viscosity, but the velocity magnitude was smaller in the molecular dynamics simulation than in continuum. In conclusion, the water force field does not accurately model transport properties and the fluid behaviour of water.

The creep process was studying in a system consisting of two opposing spherical asperities being pushed together by a constant normal stress. I derived a theoretical expression for the system height as a function of time from a recent paper on a new steady state friction law. The molecular dynamics results showed good agreement with the theoretical prediction, particularly when the spherical asperities were not equally sized. After close to a microsecond, the measured system height plateaued, in disagreement with the $\ln(1 + t/t_c)$ term in the analytical expression. Adding passivating hydroxy groups on the silica surface did not change the trend of the creep, but the total amount of creep was reduced as the spherical shape of the asperities remained more intact. Water in the initial void between the asperities led to substantially more creep and a creep rate approaching some non-zero constant. Eventually, the water started diffusing into the silica asperities at the point of contact, and the crystalline order of silica broke down in this region. In conclusion, the theoretical prediction based on thermally activated creep was accurate except when water was added, as the water had a strong chemical effect.

8.3 Outlook and future work

As this is a master's thesis, I have not had the time to do an exhausting study of the systems and processes discussed in this thesis. And as I am a master's student, I have done many things suboptimally. Furthermore, molecular dynamics simulations are computationally

demanding, hence I have been limited in both the size and variety of my systems, as well as the timescales at which they have been simulated. There are, therefore, many possible minor and major corrections and extensions to my work, as well as new directions for molecular dynamics.

The most obvious area for improvement and further study is the one in which my results did not agree with neither experiments nor intuition. Water's transport properties were not well reproduced in my simulations, due to the force field being fitted to structural and chemical properties rather than e.g. viscosity. Evidently, a different force field is required to perform simulations of water flow in silica geometries. The problem is, of course, that this force field was developed for a reason — there are no other classical force fields which accurately models the structure and chemistry in silica-water systems. As such, the best hope is currently to reparametrise the force field with the current functional form to also include transport properties, and hope that this form can reproduce both structural, chemical and transport properties.

If this is not possible, a new potential must be developed. In recent years, several groups have worked on automated development of molecular dynamics potentials via machine learning models which learn from quantum mechanical calculations such as density functional theory. Such a potential would be much more general than those we currently have. Therefore, they might be expected to work for my silica-water system as well, and the adaptive approach taken by some of the projects should also make the potential viable for extreme conditions such as a large normal stress. Furthermore, generally applicable potentials would open a world of new systems to be studied.

With a working force field capable of reproducing experimental transport properties, my flow simulation would suddenly become a lot more interesting. Assuming that molecular dynamics simulations with the new force field end up reproducing continuum results when the systems are sufficiently large, one could study smaller and smaller systems and determine the breakdown point of continuum fluid mechanics. It would be relatively simple to repeat this for different kinds of flow channel geometries.

My creep simulations showed more promising results than the flow simulations, which open up a wide range of questions about the different effects I observed. There are, however, also many smaller details in my simulation procedure which should be more thoroughly analysed. For example, the barostat damping time was seen in figure 7.5 on page 59 to have a significant effect on the shape of the creep results and the quality of the fit with the analytical prediction. The water simulations were especially heavy, and thus I only had time to try a single damping time for the xy barostat in this simulation. With sufficient computational resources, it would be trivial to repeat the simulation for a series of damping times, but I have not had access to unlimited computing time. The size of the two asperities are other quantities which should be studied systematically, as the asymmetric systems followed the theoretical expectation more closely than the symmetric system.

The most important extension of my work on creep would be a more thorough analysis of the results that I have already obtained. So far, I have only had the time to study the system height as a function of time, compare it with my analytical expression and then get

a vague idea of what is causing the differences by visually inspecting a rendering of the different systems. The next step would be to study how the silica structure evolves as a function of time in the different systems. Measures such as order parameters can be used to characterise local structure based on the surrounding of the atoms. Calculating these, one could study the creation and development of defects and dislocations, to get a proper understanding of the creep process and how it is affected by surface hydroxy groups or water.

Of course, understanding the creep process is only the first part of verifying Aharonov and Scholz's proposed model for steady state friction and understanding the underlying molecular process. The ultimate goal is to finally get a proper idea of how friction works, and in particular a model which works across a large range of slip rates. If this is achieved, the new theory of friction can be used to design materials and surfaces with specific frictional properties — perhaps even negative coefficients of friction, which has recently been observed both experimentally [42] and in molecular dynamics simulations [43].

Part III

Appendices

Appendix A

Code overview

All code for my simulations is available on my GitHub profile, github.com/anjohan. The repositories relevant for this thesis are called mdtools, lammps-velocityaverage, lammps-binary-dump-reader, friction, water-viscosity and water-flow. See the READMEs located in the repositories for an overview. In the case of the simulation repositories, the most instructive thing to do after cloning them may be to do a dryrun of the make command found in the job script used to run on Abel or our GPU machine.

For example, running `make -n ITERATIONS=3 data/restart.creep_water_3` in the friction/creep directory shows how the different scripts are run in sequence to perform a creep simulation with water.

```
mpirun lmp  -in in.setup
python addH_data.py data/data.setup data/data.setup_withH
mpirun lmp -in in.passivate
lmp -in in.write_packmol
packmol < data/add_water.inp
python xyz2data.py
mpirun lmp -var VARIANT 2 -in in.down
python3 find_z_pressure.py _water
mpirun lmp  -var VARIANT 2 -var z_pressure $(cat data/average_z_pressure_water.
    txt) -var ITERATION 1 -in in.creep
mpirun lmp  -var VARIANT 2 -var z_pressure $(cat data/average_z_pressure_water.
    txt) -var ITERATION 2 -in in.creep
mpirun lmp  -var VARIANT 2 -var z_pressure $(cat data/average_z_pressure_water.
    txt) -var ITERATION 3 -in in.creep
```

Appendix B

Bonus topics

B.1 Diffusion

Diffusion is the phenomenon of a net flux of particles as a result of random motion combined with a concentration gradient. Microscopically, diffusion can be modelled as random motion. The simplest model is a particle jumping with fixed rate and length in the positive or negative x, y or z-direction, a so-called Gaussian process, while Brownian motion is a more advanced model of a particle acted upon by random forces. Macroscopically, diffusion determines the time development of concentration, through the diffusion equation,

$$\frac{\partial C(\vec{r},t)}{\partial t} = D\nabla^2 C(\vec{r},t), \tag{B.1}$$

where $C(\vec{r},t)$ is the number density at a point $\vec{r}$ and a time t and D is the diffusion coefficient.

B.1.1 Estimating the diffusion coefficient

In molecular dynamics, the diffusion coefficient, which is a macroscopic quantity, has to be estimated through positions and velocities of atoms, which are microscopic quantities. The microscopic motion of particles can be connected to the macroscopic phenomenon of diffusion by modelling the motion of each atom as a random walker. This random walker starts at the origin at $t = 0$, and makes a jump at each time $t_n = n\Delta t$. Each jump is assumed to be on the form

$$\vec{r} \mapsto \vec{r} + d \cdot (\pm 1, \pm 1, \pm 1), \tag{B.2}$$

where each sign is chosen randomly. The mean squared displacement is then

$$\left\langle \|\vec{r}(t) - \vec{r}(0)\|^2 \right\rangle = 6Dt, \tag{B.3}$$

where D has been defined as $d^2/2\Delta t$.

In molecular dynamics, the diffusion constant of a material is easily determined by averaging the mean squared displacement of many atoms. As $t \to \infty$, the mean squared displacement will approach a linear function of time with slope $6D$. A major caveat in the implementation is to revert the periodic boundary conditions, as these clearly disturb the mean squared displacement. LAMMPS has the option of writing the "unwrapped" coordinates to file using the specifiers `xu yu zu`.

B.2 Orthogonal unit cell for α-quartz

As described in section 7.3.3 on page 61, β-cristobalite is not the stable polymorph of SiO_2 at the temperatures used in my simulations. The reason for using β-cristobalite is simply convenience, as it has a cubic unit cell and the same density as amorphous silica.

α-quartz is the most common silica polymorph, and it is also the one most commonly found in nature. For molecular dynamics simulations, it is more unpleasant to work with when shaping the system. The unit cell of α-quartz is trigonal, i.e. not even orthogonal. Fortunately, only $\gamma = 120°$ is different from 90°.

In preparation for doing simulations of α-quartz, I have created an orthogonal supercell. From the Crystallography Open Database [44] I found that α-quartz has space group 154 with $a = b = 4.9134$ Å, $c = 5.4052$ Å, $\alpha = \beta = 90°$ and $\gamma = 120°$. Additionally, it tells me that silicon is found in Wyckoff position 3a with coordinates $(0.4699, 0, 0)$ and that oxygen is in Wyckoff position 6c with coordinates $(0.4141, 0.2681, 0.1188)$.

I then went ahead and plugged these numbers into the Atomic Simulation Environment (ASE) [45] Python package. This gave the wrong results, as it turns out that the Crystallography Open Database follows a different Wyckoff position convention than ASE. ASE seems to follow the same convention as the Bilbao Crystallographic Server [46]. Comparing their definitions of the 3a position shows that their unit cells are shifted by $\frac{2}{3}c$ in the z-directions, as Bilbao defines the relevant 3a position as $(x, 0, \frac{2}{3})$.

With this issue sorted out, I wrote the Python script below for replicating the trigonal cell, redefining the cell and removing atoms outside the new cell. Since $\gamma = 120°$ and $\cos(\gamma) = \frac{1}{2}$, simple trigonometry gave the replications from which an orthogonal unit cell could be extracted. Figure B.1 on page 82 shows the resulting unit cells.

```
import numpy as np
import ase
from ase.visualize import view
from ase.spacegroup import crystal, Spacegroup

# Cell parameters
a = b = 4.9134
c = 5.4052
alpha = beta = 90
gamma = 120

# Atoms in Wyckoff positions
```

```
Si = ase.Atom("Si", [0.4699, 0, 2 / 3])
O = ase.Atom("O", [0.4141, 0.2681, 0.1188 + 2 / 3])

# Define unit cell
spacegroup = Spacegroup(154)
cell = crystal([Si, O], spacegroup=spacegroup, cellpar=[a, b, c, alpha, beta,
    gamma])

# Repeat unit cell
cell = cell.repeat([2, 2, 1])

# Define new unit cell
cell.set_cell([[a, 0, 0], [0, b * np.sqrt(3), 0], [0, 0, c]])

# Wrap atoms into new cell, remove duplicates
cell.wrap()
ase.geometry.get_duplicate_atoms(cell, delete=True)

positions = cell.get_positions()
atomic_numbers = cell.get_atomic_numbers()
basis_vectors = cell.get_cell()

# Write data file for LAMMPS
with open("orthogonal_alpha_quartz.data", "w") as outfile:
    outfile.write("# Orthogonal alpha quartz supercell\n\n")
    outfile.write(
        f"{len(positions)} atoms\n"
        "2 atom types\n"
        "\n"
        f"0.0 {basis_vectors[0,0]} xlo xhi\n"
        f"0.0 {basis_vectors[1,1]} ylo yhi\n"
        f"0.0 {basis_vectors[2,2]} zlo zhi\n"
        "\n"
        "Atoms\n"
        "\n"
    )
    for i, atomic_number in enumerate(atomic_numbers):
        atom_type = 1 if atomic_number == 14 else 2
        outfile.write(f"{i+1} {atom_type} " + " ".join(map(str, positions[i]))
            + "\n")
```

Resulting `data` file:

```
# Orthogonal alpha quartz supercell

18 atoms
2 atom types

0.0 4.9134 xlo xhi
0.0 8.510258437908922 ylo yhi
0.0 5.4052 zlo zhi

Atoms
```

```
1 1 2.30880666 0.0 3.6034666666666664
2 1 3.7589966700000006 1.999485219986701 1.801733333333333
3 1 1.3022966700000003 2.2556439989677597 0.0
4 2 1.375997670000001 1.140800143601691 4.245604426666667
5 2 3.237439260000002 0.6212488659673514 2.4438710933333327
6 2 2.756663070000002 2.4930802093854187 0.6421377600000001
7 2 0.2999630700000005 1.7620490095690424 1.1595955733333334
8 2 3.832697670000001 3.11432907535277 2.961328906666667
9 2 0.7807392600000005 3.6338803529871098 4.76306224
10 1 4.765506660000002 4.25512921895446 3.6034666666666664
11 1 1.3022966700000014 6.254614438941162 1.801733333333333
12 1 3.7589966700000015 6.510773217922221 0.0
13 2 3.8326976700000013 5.395929362556152 4.245604426666667
14 2 0.7807392600000016 4.876378084921813 2.4438710933333327
15 2 0.2999630700000016 6.74820942833988 0.6421377600000001
16 2 2.7566630700000014 6.017178228523504 1.1595955733333334
17 2 1.3759976700000016 7.369458294307231 2.961328906666667
18 2 3.237439260000002 7.889009571941571 4.76306224
```

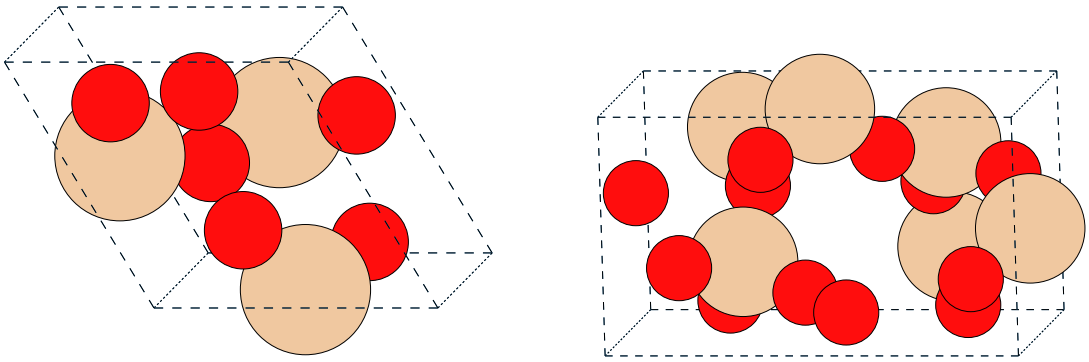


Figure B.1: Original, trigonal and new, orthorhombic unit cells of α-quartz as created by the Python script.

List of Figures

References

[1] R.L. Rudnick and S. Gao. "3.01 - Composition of the Continental Crust". In: *Treatise on Geochemistry*. Oxford: Pergamon, 2003, pp. 1–64. ISBN: 978-0-08-043751-4. DOI: https://doi.org/10.1016/B0-08-043751-6/03016-4.

[2] Christopher H. Scholz. "Earthquakes and friction laws". In: *Nature* 391.6662 (Jan. 1998), pp. 37–42. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/34097. URL: https://doi.org/10.1038/34097.

[3] Dongshuai Hou, Hongyan Ma, Zongjin Li and Zuquan Jin. "Molecular simulation of "hydrolytic weakening": A case study on silica". In: *Acta Materialia* 80 (Nov. 2014), pp. 264–277. ISSN: 1359-6454. DOI: 10.1016/j.actamat.2014.07.059. URL: https://doi.org/10.1016/j.actamat.2014.07.059.

[4] Einat Aharonov and Christopher H. Scholz. "A Physics-Based Rock Friction Constitutive Law: Steady State Friction. A physics-based friction law". In: *Journal of Geophysical Research: Solid Earth* 123.2 (Feb. 2018), pp. 1591–1614. ISSN: 2169-9313. DOI: 10.1002/2016jb013829. URL: https://doi.org/10.1002/2016jb013829.

[5] Aneesur Rahman. "Correlations in the Motion of Atoms in Liquid Argon". In: *Physical Review* 136.2A (Oct. 1964), A405–A411. DOI: 10.1103/physrev.136.a405. URL: https://doi.org/10.1103%2Fphysrev.136.a405.

[6] Frank H. Stillinger and Thomas A. Weber. "Computer simulation of local order in condensed phases of silicon". In: *Physical Review B* 31.8 (Apr. 1985), pp. 5262–5271. ISSN: 0163-1829. DOI: 10.1103/physrevb.31.5262. URL: https://doi.org/10.1103/physrevb.31.5262.

[7] P. Vashishta, Rajiv K. Kalia, José P. Rino and Ingvar Ebbsjö. "Interaction potential forSiO2: A molecular-dynamics study of structural correlations". In: *Physical Review B* 41.17 (June 1990), pp. 12197–12209. ISSN: 0163-1829, 1095-3795. DOI: 10.1103/physrevb.41.12197. URL: https://doi.org/10.1103/physrevb.41.12197.

[8] Priya Vashishta, Rajiv K. Kalia, Aiichiro Nakano and José Pedro Rino. "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 2007), p. 103515. ISSN: 0021-8979, 1089-7550. DOI: 10.1063/1.2724570. URL: https://doi.org/10.1063/1.2724570.

[9] Paulo Sergio Branicio, José Pedro Rino, Chee Kwan Gan and Hélio Tsuzuki. "Interaction potential for indium phosphide: A molecular dynamics and first-principles study of the elastic constants, generalized stacking fault and surface energies". In: *Journal of Physics: Condensed Matter* 21.9 (Jan. 2009), p. 095002. ISSN: 0953-8984, 1361-648X. DOI: 10.1088/0953-8984/21/9/095002. URL: https://doi.org/10.1088/0953-8984/21/9/095002.

[10] Aiichiro Nakano, Rajiv K. Kalia and Priya Vashishta. "First sharp diffraction peak and intermediate-range order in amorphous silica: Finite-size effects in molecular dynamics simulations". In: *Journal of Non-Crystalline Solids* 171.2 (Aug. 1994), pp. 157–163. ISSN: 0022-3093. DOI: 10.1016/0022-3093(94)90351-4. URL: https://doi.org/10.1016/0022-3093(94)90351-4.

[11] Weiqiang Wang, Aiichiro Nakano, Rajiv K. Kalia and Priya Vashishta. "Interatomic potentials for molecular dynamics simulations of hydrolysis and stress corrosion cracking of silica glass". Unpublished.

[12] Jérôme Delhommelle and Philippe Millié. "Inadequacy of the Lorentz-Berthelot combining rules for accurate predictions of equilibrium properties by molecular simulation". In: *Molecular Physics* 99.8 (Apr. 2001), pp. 619–625. ISSN: 0026-8976, 1362-3028. DOI: 10.1080/00268970010020041. URL: https://doi.org/10.1080/00268970010020041.

[13] Nikola Tchipev et al. "TweTriS: Twenty trillion-atom simulation". In: *The International Journal of High Performance Computing Applications* (Jan. 2019), p. 109434201881974. ISSN: 1094-3420, 1741-2846. DOI: 10.1177/1094342018819741. URL: https://doi.org/10.1177/1094342018819741.

[14] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola and J. R. Haak. "Molecular dynamics with coupling to an external bath". In: *The Journal of Chemical Physics* 81.8 (Oct. 1984), pp. 3684–3690. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.448118. URL: https://doi.org/10.1063/1.448118.

[15] Hans C. Andersen. "Molecular dynamics simulations at constant pressure and/or temperature". In: *The Journal of Chemical Physics* 72.4 (Feb. 1980), pp. 2384–2393. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.439486. URL: https://doi.org/10.1063/1.439486.

[16] Don S. Lemons and Anthony Gythiel. "Paul Langevin's 1908 paper "On the Theory of Brownian Motion" ["Sur la théorie du mouvement brownien," C. R. Acad. Sci. (Paris) 146, 530–533 (1908)]". In: *American Journal of Physics* 65.11 (Nov. 1997), pp. 1079–1081. ISSN: 0002-9505, 1943-2909. DOI: 10.1119/1.18725. URL: https://doi.org/10.1119/1.18725.

[17] Robert D. Groot and Patrick B. Warren. "Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation". In: *The Journal of Chemical Physics* 107.11 (Sept. 1997), pp. 4423–4435. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.474784. URL: https://doi.org/10.1063/1.474784.

[18] Shuichi Nosé. "A unified formulation of the constant temperature molecular dynamics methods". In: *The Journal of Chemical Physics* 81.1 (July 1984), pp. 511–519. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.447334. URL: https://doi.org/10.1063/1.447334.

[19] William G. Hoover. “Canonical dynamics: Equilibrium phase-space distributions”. In: *Physical Review A* 31.3 (Mar. 1985), pp. 1695–1697. ISSN: 0556-2791. DOI: 10.1103/physreva.31.1695. URL: https://doi.org/10.1103/physreva.31.1695.

[20] Mark E Tuckerman, José Alejandre, Roberto López-Rendón, Andrea L Jochim and Glenn J Martyna. “A Liouville-operator derived measure-preserving integrator for molecular dynamics simulations in the isothermal–isobaric ensemble”. In: *Journal of Physics A: Mathematical and General* 39.19 (Apr. 2006), pp. 5629–5651. ISSN: 0305-4470, 1361-6447. DOI: 10.1088/0305-4470/39/19/s18. URL: https://doi.org/10.1088/0305-4470/39/19/s18.

[21] Steve Plimpton. “Fast Parallel Algorithms for Short-Range Molecular Dynamics”. In: *Journal of Computational Physics* 117.1 (Mar. 1995), pp. 1–19. ISSN: 0021-9991. DOI: 10.1006/jcph.1995.1039. URL: https://doi.org/10.1006/jcph.1995.1039.

[22] H. Carter Edwards, Christian R. Trott and Daniel Sunderland. “Kokkos: Enabling manycore performance portability through polymorphic memory access patterns”. In: *Journal of Parallel and Distributed Computing* 74.12 (Dec. 2014), pp. 3202–3216. ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2014.07.003. URL: https://doi.org/10.1016/j.jpdc.2014.07.003.

[23] L. Martínez, R. Andrade, E. G. Birgin and J. M. Martínez. “PACKMOL: A package for building initial configurations for molecular dynamics simulations”. In: *Journal of Computational Chemistry* 30.13 (Oct. 2009), pp. 2157–2164. ISSN: 0192-8651, 1096-987X. DOI: 10.1002/jcc.21224. URL: https://doi.org/10.1002/jcc.21224.

[24] Q.H. Tang. “Effect of water on brittle fracture of SiO2 by molecular dynamics study”. In: *Computational Materials Science* 45.2 (Apr. 2009), pp. 429–433. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2008.11.002. URL: https://doi.org/10.1016/j.commatsci.2008.11.002.

[25] Eduardo R. Cruz-Chu, Aleksei Aksimentiev and Klaus Schulten. “WaterSilica Force Field for Simulating Nanodevices”. In: *The Journal of Physical Chemistry B* 110.43 (2006), pp. 21497–21508. ISSN: 1520-6106, 1520-5207. DOI: 10.1021/jp063896o. URL: https://doi.org/10.1021/jp063896o.

[26] Ali A. Hassanali and Sherwin J. Singer. “Model for the Water-Amorphous Silica Interface: The Undissociated Surface”. In: *The Journal of Physical Chemistry B* 111.38 (2007), pp. 11181–11193. ISSN: 1520-6106, 1520-5207. DOI: 10.1021/jp062971s. URL: https://doi.org/10.1021/jp062971s.

[27] T. S. Mahadevan and S. H. Garofalini. “Dissociative Water Potential for Molecular Dynamics Simulations”. In: *The Journal of Physical Chemistry B* 111.30 (Aug. 2007), pp. 8919–8927. ISSN: 1520-6106, 1520-5207. DOI: 10.1021/jp072530o. URL: https://doi.org/10.1021/jp072530o.

[28] T. S. Mahadevan and S. H. Garofalini. “Dissociative Chemisorption of Water onto Silica Surfaces and Formation of Hydronium Ions”. In: *The Journal of Physical Chemistry C* 112.5 (Feb. 2008), pp. 1507–1515. ISSN: 1932-7447, 1932-7455. DOI: 10.1021/jp076936c. URL: https://doi.org/10.1021/jp076936c.

[29] Adarsh Shekhar, Ken-ichi Nomura, Rajiv K. Kalia, Aiichiro Nakano and Priya Vashishta. "Nanobubble Collapse on a Silica Surface in Water: Billion-atom Reactive Molecular Dynamics Simulations". In: *Physical Review Letters* 111.18 (Oct. 2013). ISSN: 0031-9007, 1079-7114. DOI: 10.1103/physrevlett.111.184503. URL: https://doi.org/10.1103/physrevlett.111.184503.

[30] Adarsh Shekhar, Rajiv K. Kalia, Aiichiro Nakano, Priya Vashishta, Camilla K. Alm and Anders Malthe-Sørenssen. "Universal stretched exponential relaxation in nano-confined water". In: *Applied Physics Letters* 105.16 (Oct. 2014), p. 161907. ISSN: 0003-6951, 1077-3118. DOI: 10.1063/1.4899279. URL: https://doi.org/10.1063/1.4899279.

[31] Yong Zhang, Akihito Otani and Edward J. Maginn. "Reliable Viscosity Calculation from Equilibrium Molecular Dynamics Simulations: A Time Decomposition Method". In: *Journal of Chemical Theory and Computation* 11.8 (July 2015), pp. 3537–3546. ISSN: 1549-9618, 1549-9626. DOI: 10.1021/acs.jctc.5b00351. URL: https://doi.org/10.1021/acs.jctc.5b00351.

[32] Nikolai V. Priezjev, Anton A. Darhuber and Sandra M. Troian. "Slip behavior in liquid films on surfaces of patterned wettability: Comparison between continuum and molecular dynamics simulations". In: *Physical Review E* 71.4 (Apr. 2005). ISSN: 1539-3755, 1550-2376. DOI: 10.1103/physreve.71.041608. URL: https://doi.org/10.1103/physreve.71.041608.

[33] Anders Hafreager. "Flow of dilute gases in complex nanoporous media". Master's Thesis. University of Oslo, 2014.

[34] Binquan Luan and Mark O. Robbins. "Contact of single asperities with varying adhesion: Comparing continuum mechanics to atomistic simulations". In: *Physical Review E* 74.2 (Aug. 2006). ISSN: 1539-3755, 1550-2376. DOI: 10.1103/physreve.74.026111. URL: https://doi.org/10.1103/physreve.74.026111.

[35] Filip H. Larsen. "Molecular Dynamics modeling of single asperity contact". Master's Thesis. University of Oslo, 2017.

[36] Shuyin Jiao and Yashashree Kulkarni. "Molecular dynamics study of creep mechanisms in nanotwinned metals". In: *Computational Materials Science* 110 (Dec. 2015), pp. 254–260. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2015.08.017. URL: https://doi.org/10.1016/j.commatsci.2015.08.017.

[37] Ankit Rohatgi. *WebPlotDigitizer*. 2011. URL: https://automeris.io/WebPlotDigitizer.

[38] Kenneth R. Harris and Lawrence A. Woolf. "Temperature and Volume Dependence of the Viscosity of Water and Heavy Water at Low Temperatures". In: *Journal of Chemical & Engineering Data* 49.4 (July 2004), pp. 1064–1069. ISSN: 0021-9568, 1520-5134. DOI: 10.1021/je049918m. URL: https://doi.org/10.1021/je049918m.

[39] A. Einstein. "Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen". In: *Annalen der Physik* 322.8 (1905), pp. 549–560. ISSN: 0003-3804, 1521-3889. DOI: 10.1002/andp.19053220806. URL: https://doi.org/10.1002/andp.19053220806.

[40] Wikipedia contributors. *Bulk modulus — Wikipedia, The Free Encyclopedia*. 2019. URL: https://en.wikipedia.org/w/index.php?title=Bulk_modulus&oldid=890441600 (visited on 07/06/2019).

[41] Jessica M. Rimsza, Reese E. Jones and Louise J. Criscenti. "Chemical Effects on Subcritical Fracture in Silica From Molecular Dynamics Simulations". In: *Journal of Geophysical Research: Solid Earth* 123.11 (Nov. 2018), pp. 9341–9354. ISSN: 2169-9313. DOI: 10.1029/2018jb016120. URL: https://doi.org/10.1029/2018jb016120.

[42] Zhao Deng, Alex Smolyanitsky, Qunyang Li, Xi-Qiao Feng and Rachel J. Cannara. "Adhesion-dependent negative friction coefficient on chemically modified graphite at the nanoscale". In: *Nature Materials* 11.12 (Oct. 2012), pp. 1032–1037. ISSN: 1476-1122, 1476-4660. DOI: 10.1038/nmat3452. URL: https://doi.org/10.1038/nmat3452.

[43] Davide Mandelli, Wengen Ouyang, Oded Hod and Michael Urbakh. "Negative Friction Coefficients in Superlubric Graphite–Hexagonal Boron Nitride Heterojunctions". In: *Physical Review Letters* 122.7 (Feb. 2019). ISSN: 0031-9007, 1079-7114. DOI: 10.1103/physrevlett.122.076102. URL: https://doi.org/10.1103/physrevlett.122.076102.

[44] Saulius Gražulis, Daniel Chateigner, Robert T. Downs, A. F. T. Yokochi, Miguel Quirós, Luca Lutterotti, Elena Manakova, Justas Butkus, Peter Moeck and Armel Le Bail. "Crystallography Open Database – an open-access collection of crystal structures". In: *Journal of Applied Crystallography* 42.4 (May 2009), pp. 726–729. ISSN: 0021-8898. DOI: 10.1107/s0021889809016690. URL: https://doi.org/10.1107/s0021889809016690.

[45] Ask Hjorth Larsen et al. "The atomic simulation environment—a Python library for working with atoms". In: *Journal of Physics: Condensed Matter* 29.27 (June 2017), p. 273002. DOI: 10.1088/1361-648x/aa680e. URL: https://doi.org/10.1088%2F1361-648x%2Faa680e.

[46] Mois I. Aroyo, Asen Kirov, Cesar Capillas, J. M. Perez-Mato and Hans Wondratschek. "Bilbao Crystallographic Server. II. Representations of crystallographic point groups and space groups". In: *Acta Crystallographica Section A Foundations of Crystallography* 62.2 (Mar. 2006), pp. 115–128. ISSN: 0108-7673. DOI: 10.1107/s0108767305040286. URL: https://doi.org/10.1107/s0108767305040286.