SIMULATING THE MECHANICAL PROPERTIES OF SOLIDS WITH DIFFERENT MOLECULAR BONDS

by

SIGURD WENNER

THESIS

for the degree of

MASTER OF SCIENCE

(Master in Computational Physics)

Faculty of Mathematics and Natural Sciences
Department of Physics
University of Oslo

Sept. 2010

Det matematisk-naturvitenskapelige fakultet

Universitetet i Oslo

Contents

Chapter 1

Introduction

Chapter 2

Basics of molecular dynamics

Atoms are objects which have an extension in the order of one Ångström, 10^{-10} meters. They are composite of electrons and a nuclei which is also consists of more elementary particles. This is, even when classical mechanics is used to describe the motion of particles, a complicated system. And we know that what must be applied to systems of this small size is quantum mechanics. All elementary particles are quantum fields with an associated wavefunction and a quantized energy. In this project, I will study systems containing thousands of atoms. Ab initio calculations, containing only strictly controlled approximations, no parameters and a fully quantum mechanical description of the system, is no feasible approach.

The Born-Oppenheimer approximation says that the nucleus of an atom has a much greater mass than the electrons, and will stand perfectly still in a electronic time scale. The degrees of freedom of the electrons and the nucleus are separated. The energy of a collection of atoms can be calculated using only the electronic wavefunctions, with e.g. density functional theory (DFT). This enables calculation of forces on nuclei, and the atomic movements can be simulated. Such a DFT-MD coupling is very accurate, but also extremely time-consuming. Classical molecular dynamics (MD) goes one step further and considers not only nuclei, but entire atoms as point particles. Classical mechanics describe the motion of the atoms, but the potential between the atoms tries to take into account their electronic structure (more of this in section 3). The interatomic potentials can be improved to the point that the resulting dynamics is equal to what one would get using an ab initio approach.

2.1 Equations of motion

Lagrange's and Hamilton's formulations of classical mechanics are often used in molecular dynamics derivations. This is useful when simulating e.g. rigid molecules with constraints on the atoms' relative motion, using generalized coordinates in addition to cartesian ones. As I will not consider molecules in this project, but crystalline solids, I will stick to the simpler Newtonian formulation. In a collection of atoms i,

the equation responsible for the dynamics is Newton's second law,

$$m_i \frac{\mathrm{d}^2 \mathbf{r}_i}{\mathrm{d}t^2} = \mathbf{F}_i, \tag{2.1}$$

where m_i is the mass of atom i, $\mathbf{F}_i$ is the force vector exerted on atom i by other particles or external conditions, and $\mathbf{r}_i$ is the position of the atom. All forces are dependent on the positions of all the atoms in the system.

The force acting on a particle with index i is the negative gradient of the potential energy associated with that particle, U_i. In the usual case of interaction pairs, this equals the sum of the negative gradient of interaction potentials U_{ij} from other particles j.

$$\mathbf{F}_i = -\nabla_i U_i(\mathbf{r}_i) = -\sum_{j \neq i} \nabla_i U_{ij}(\mathbf{r}_{ij}) \tag{2.2}$$

The meaning of ∇_i is the gradient with respect to the position of particle i. $\mathbf{r}_{ij}$ is shorthand for $\mathbf{r}_i - \mathbf{r}_j$. By Newton's third law, $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$. By exploiting this, only half of the possible potential gradients need to be calculated when finding the forces on all particles. In a system of N particles, this means $\frac{1}{2}N(N-1)$ force terms must be calculated in a straightforward approach.

2.2 Time integration

In most cases, to integrate Newton's second law (2.1), molecular dynamics programs use symplectic integrators like the Verlet and Leapfrog methods [1]. All integrators conserve momentum exactly and energy approximately. Symplectic integrators are time reversible and also conserve phase space ($\{\mathbf{r}_i\}$-$\{\mathbf{p}_i\}$-space) volume between trajectories.

My program uses the velocity Verlet method, because of its numerical stability and ability to efficiently estimate the velocity and positions of particles at simultaneous times. Given initial positions, velocities and forces, these are the steps of the algorithm for each particle:

$$\mathbf{v}(t + \Delta t/2) = \mathbf{v}(t) + \frac{\mathbf{F}(t)}{2m}\Delta t \tag{2.3}$$

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t + \Delta t/2)\Delta t \tag{2.4}$$

$$\mathbf{F}(t + \Delta t) = -\nabla U(\mathbf{r}(t + \Delta t)) \tag{2.5}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t + \Delta t/2) + \frac{\mathbf{F}(t + \Delta t)}{2m}\Delta t \tag{2.6}$$

The time step should be set so that the thermal oscillation of atoms are well-described by the discrete dynamics. In my simulations, I put the time step to a few fs (10^{-15} s), and one oscillation goes over approximately 20 time steps. An atom with room temperature thermal velocity will use about 100 time steps to reach from its own place to an adjacent atom's position. This ensures that atoms do not get too close to each other, preventing unphysically large forces.

2.3 Thermodynamic ensemble

The most fundamental principle of classical mechanics is conservation of energy. This will be approximately satisfied by discrete time integrators. Therefore, particles simulated using only such a time integrator will constitute a microcanonical (NVE) ensemble.

In macroscopic experiments, it is generally easier to keep the temperature constant istead of the energy. This is the most prominent reason for using the canonical (NVT) ensemble. This ensemble can be simulated by using methods popularly referred to as thermostats.

According to the equipartition principle, every degree of freedom which is quadratic in position or speed will contribute with an average energy of

$$E_{\text{dof}} = \frac{1}{2} k_B T \tag{2.7}$$

to the total energy, where k_B is the Boltzmann constant and T is the temperature of the system. This applies to a canonical ensemble in thermodynamic equilibrium. This formula can be summed over all kinetic degrees of freedom, and inverted to give an estimate of the system temperature. As the total translational kinetic energy is $U_k = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 = \frac{3}{2} N k_B T$, we get

$$T = \frac{1}{3N k_B} \sum_{i=1}^{N} m_i v_i^2 \tag{2.8}$$

in three dimensions. This can be used to verify how a thermostat performs over time.

Regardless of the thermostat used, all velocities are initialized to random values obeying the Maxwell-Boltzmann distribution, with the standard deviation $\sqrt{\frac{T_{\text{target}}}{m_i}}$.

Equation 2.8 applies only to systems in equilibrium, where energy is evenly distributed between potential and kinetic degrees of freedom. However, only feasible ways of changing the temperature of the kinetic degrees of freedom exist, so this must be done continously until equilibrium is achieved.

2.3.1 Thermostats

One way of keeping the temperature constant is to use a simple rescaling factor. All the particle velocities are multiplied by a factor α so that $T \to \alpha^2 T = T_{\text{target}}$. This rescaling factor is easily found from equation 2.8:

$$\alpha = \sqrt{\frac{3N k_B T_{\text{target}}}{\sum_{i=1}^{N} m_i v_i^2}} \tag{2.9}$$

This rescaling will give the system the correct temperature, but it will not sample the Maxwell-Boltzmann-distribution correctly for all degrees of freedom, and does therefore not satisfy the ergodicity requirements of the canonical ensemble.

Many different thermostats have been proposed to fix this problem, but all have their weaknesses [1]. The Andersen thermostat replaces the velocities of random particles with velocities from the Maxwell-Boltzmann distribution. The Berendsen and Nosé-Hoover thermostats add a ficticious friction force which gradually decrease or increase the particles' velocities to get the correct temperature.

The Andersen thermostat is known to work well for equilibrating systems, although it should not be used when simulating dynamics. The algorithm is strinkingly simple. Define a collision time τ, significantly larger than the time step Δt, after which all particles in the system on average have exchanged energy with the heat bath. For each time step, the probability of replacing a particle's velocity by a Maxwell-Boltzmann distributed one is given by $\Delta t/\tau$. The energy absorbed from the heat bath is calculated by simply computing the kinetic energy before and after the thermostat has been used.

The thermostat I will use during dynamical processes is one recently developed by Bussi, Donadio and Parrinello [2] (I will refer to it as the BDP thermostat). It contains a friction term and a stochastic force, like in Langevin dynamics for e.g. solvent particles. In its original infinitesimal form, the change in kinetic energy due to the termostat is

$$\mathrm{d}U_k = \frac{U_k^{\text{target}} - U_k}{\tau}\mathrm{d}t + 2\sqrt{\frac{U_k U_k^{\text{target}}}{3N\tau}}\mathrm{d}W. \tag{2.10}$$

The first term is the one used in the Berendsen thermostat. The second term is the stochastic term, which can be shown to sample the canonical distribution for kinetic energy. $\mathrm{d}W$ is a infinitesimal Wiener process path element. The parameter τ defines a relaxation time for the thermostat, that is, how fast the system is being cooled or heated. The equation must be integrated using stochastic integration, and in the paper, the resulting rescaling factor is found:

$$\alpha^2 = D + \frac{T_{\text{target}}}{3NT}(1-D)\sum_{i=1}^{3N} R_i^2 + 2\sqrt{\frac{T_{\text{target}}}{3NT}D(1-D)}R_1, \tag{2.11}$$

where $D = e^{-\Delta t/\tau}$, $\{R_i\}$ are $3N$ Gaussian distributed random values, and R_1 is just one of these.

Velocities are rescaled by simply setting $\mathbf{v}_i \to \alpha\mathbf{v}_i$ for all simulated particles i. The result of using the BDP thermostat is ergodic sampling of the canonical distribution (when in equilibrium), undisturbed dynamics (which is important in my case) and tunability by changing the τ parameter. Getting the random values is a tedious task, so this thermostat will be more CPU-intensive than the others, but the workload will be negligble in comparision to the calculation of forces between particles.

For one velocity rescaling, the energy absorbed from the heat bath is $U_k(\alpha^2 - 1)$, where U_k is the total kinetic energy of the system. Assuming that the velocities are rescaled every time step, the power exerted on the system by the heat bath is

$$P = \frac{1}{\Delta t}U_k(\alpha^2 - 1). \tag{2.12}$$

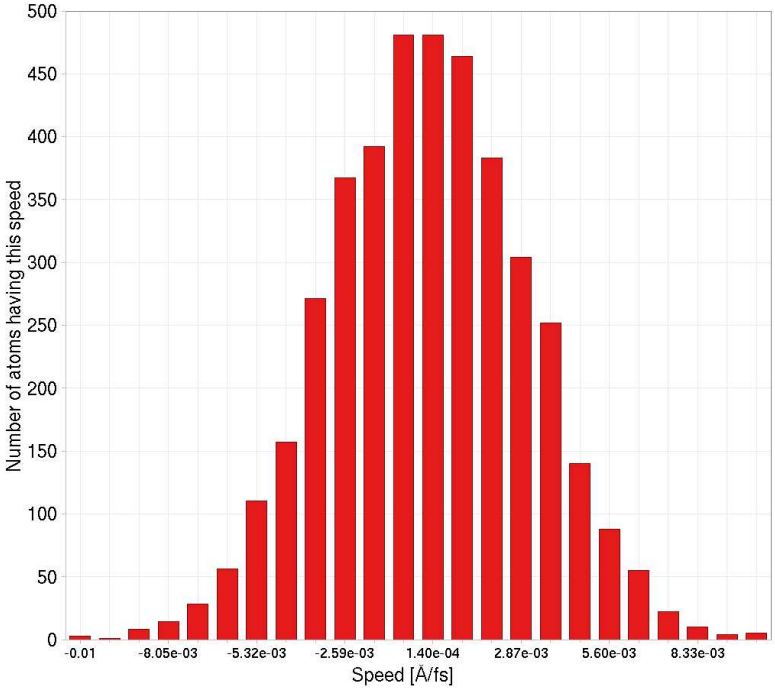


Figure 2.1: The distribution of one velocity component for 4096 particles in a simulation using the BDP thermostat.

2.4 Equilibration

It is crucial to equilibrate a molecular dynamics system before measurements can be made or dynamics can be simulated. The standard approach I use is to simulate the dynamics in a usual fashion with a heat bath coupling to all the particles using the BDP thermostat. However, some systems are harder to equilibrate than others and require special treatment.

In the sphere-surface interface experiments, it is hard to keep the sphere from bouncing, rolling and vibrating internally after being pushed down towards the surface. This type of computational experiments therefore requires an extensive equlilibration epoch. An example of a method to equlilibrate this system is the following phases:

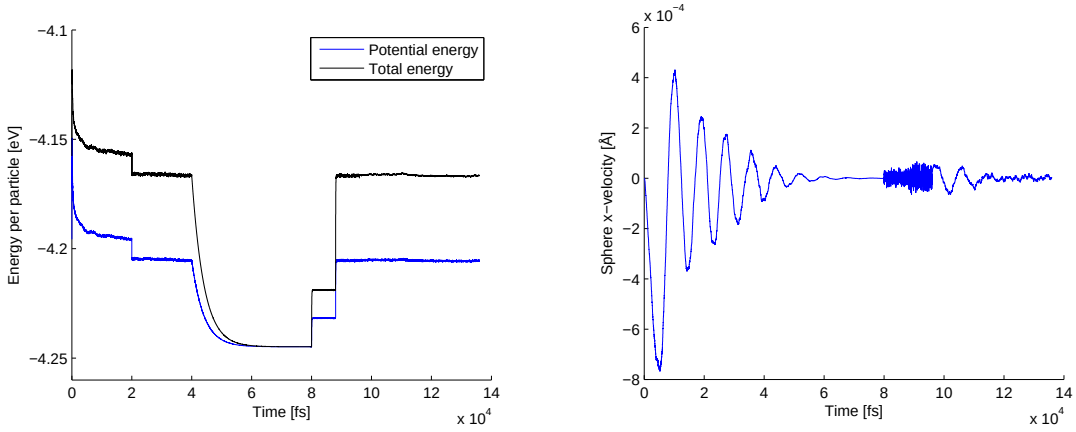
1. Initial dynamics, letting the sphere "fall" onto the surface so a contact is established. Using the BDP thermostat with global coupling. Ended when the sphere is slightly compressed and ready to start an vertically oscillating bounce motion.

2. Energy dissipation phase, adding a force term $-\gamma \mathbf{v}_i$ to all particles to simulate viscocity. The friction parameter is chosen so that $\gamma \Delta t / m_i \ll 1$. To avoid requiring a "hard" implementation of this in the equation of motion integration,

the velocity of the previous time step is used in the force, so the dynamics will have an error comparable to that of the Euler scheme for integrating the equations of motion. The temperature is decreased and the oscillations die out gradually, optimally making the system totally static. No thermostat is used in this phase.

3. As velocity rescaling would have restarted the oscillations, the Andersen thermostat is now used to make the atoms vibrate more independently. I often divide this phase into two phases where the heat bath temperature is increased abruptly from 50-100 K to e.g. 300 K.

4. The sphere should be in equilibrium with the heat bath of the desired temperature, so the data gathering phase can start. The BDP thermostat is used in a heat conduction simulation form (see section 5.2.4). Collective vibrations will reappear in the sphere, but these are naturally occuring phonons, and have an amplitude which is comparable to the thermal fluctuations of independent particle positions.



(a) (b)

Figure 2.2: Time development of energy and collective velocity of a big Si sphere pushed towards an Si surface during all simulation phases.

Figure 2.2 shows the energy and center of mass velocity in the x direction during a typical silicon adhesion experiment. The initial drops in potential energy are caused by introduction of a gravitational field of discontinually increasing strength. During the dissipative dynamics phase, both the kinetic and potential energy reach a minimum where the sphere is frozen (at zero temperature). Subsequently, the Andersen thermostat heats up the system in two phases by giving them independent

random kinetic energies. The transition to the data gathering phase where the BDP thermostat is used is seamless.

The center of mass velocity x component show that the sphere oscillates as it hits the fixed particle surface, as expected. These oscillations are killed by the dissipative dynamics, and there is no collective motion apart from fluctuations when the Andersen thermostat is applied. The rightmost section of the graph shows the combination of the mentioned phonons and fluctuations caused by particles "going against the flow" because of thermal excitations (the phonon oscillations do not die out at the end, even if it appears so in this particular case). This is how a crystal should behave in equilibrium, and good conditions for gathering e.g. mean stress data.

Chapter 3

Interaction potentials

Molecular dynamics is a classical theory, but tries to describe quantum mechanical phenomena with specially constructed interaction potentials. The electronic degrees of freedom are not taken into account explicitly in the dynamics, even though both the nucleus and the electrons effect how atoms will interact with each other. Using quantum mechanical energy calculations, researchers have tried to create approximate potential forms which will give the correct dynamics even though only classical mechanics is used. These potentials usually contain parameters which are determined by trial and error. When the parameters give the same macroscopic behaviour for a material as measured in experiments, they are accepted.

The interaction potential between two atoms will be determined by functions which have a continous dependence only on the distance between these atoms. The total potential energy is the sum of interaction potentials between all pairs of atoms. In the case of periodic boundary conditions (see section 5.2), this sum also goes over infinitely many copies of the atoms in the system.

$$U = \sum_{\mathbf{R}} \sum_{i=1}^{N} \sum_{j=1}^{i} U_{ij} \left(\| \mathbf{r}_{ij} + \mathbf{R} \| \right) \tag{3.1}$$

Here U_{ij} is the two-particle interaction potential and r_{ij} the distance between particles i and j. $\mathbf{R}$ represents all translation vectors to copies of the system ($\mathbf{R} = n_x L_x \mathbf{e}_x + n_y L_y \mathbf{e}_y + n_z L_z \mathbf{e}_z$). Particles don't interact with themselves, so the $i = j$ term is skipped if $\mathbf{R} = 0$.

The force exerted on particle i by particle j is

$$\mathbf{F}_{ij} = -\nabla_i U_{ij} = -\frac{\partial U_{ij}}{\partial r_{ij}} \frac{\mathbf{r}_{ij}}{r_{ij}}. \tag{3.2}$$

In covalent bonds, the potentials also have angular dependencies because the electron densities are not spherically symmetric around atomic nuclei. An easy way to model this is by including three-body forces. The Stillinger-Weber potential (section 3.3) is one example.

3.1 The Lennard-Jones potential

One of the simplest and most sucessful interatomic potentials constructed is the Lennard-Jones (LJ) potential. It contains a repulsive part caused by the Pauli exclusion principle, which must be taken into account when electron wavefunctions have a significant overlap. It also contains an attractive part, because electron densities are slightly higher between a pair of atoms than elsewhere, inducing dipole moments which interact.

With optimal parameters, the potential is very close to the actual potential between pairs of noble gas atoms. The first one who used a continuous potential in a MD simulation was Rahman [3], who studied liquid argon using the LJ potential. It has become a standard to use this potential to model generic short-range repulsion for all sorts of atoms.

The form of the potential is

$$U_{ij}(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{3.3}$$

where ϵ_{ij} and σ_{ij} are parameters specific to the the different atomic elements in a simulation.

The force is straightforward to calculate:

$$\mathbf{F}_{ij} = -24\frac{\epsilon_{ij}}{\sigma_{ij}^2}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{14} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{8}\right]\mathbf{r}_{ij} \tag{3.4}$$

Atom	ϵ_{ii} [eV]	σ_{ii} [Å]	Source
Ar	$1.0318 \cdot 10^{-2}$	3.405	[3]
Na^+	$5.6478 \cdot 10^{-3}$	2.35	[4]
Cl^-	$4.3501 \cdot 10^{-3}$	4.40	[4]

Table 3.1: LJ parameters for a few atoms and ions of interest.

Table 3.1 shows the two LJ parameters for interactions between pairs of equal atoms. The parameters for atoms of two different elements are calculated by two combination rules. A geometric and an arithmetic mean of the parameters for interactions between atoms of the same element are used [5]:

$$\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}} \qquad \sigma_{ij} = \frac{1}{2}\left(\sigma_{ii} + \sigma_{jj}\right) \tag{3.5}$$

3.1.1 Implementation details

The simplest form of real space cutoff possible with periodic boundary conditions (section 5.2.1) is the minimum image convention. Distances are in each direction

calculated as the smallest one of x_{ij}, $x_{ij} + L_x$ and $x_{ij} - L_x$. In words, the particles are only interacting with the nearest of their copies. This is used in my program, but together with a neighbour list algorithm.

Neighbour lists, or Verlet lists, are lists of all particles that are closer to each other than a specified cutoff length. In my implementation, all particles i have their own lists of particles $j < i$ that are close to them. The force evaluations happen only between neighbouring pairs of particles, and the lists are updated every 10th timestep or so. The cutoff length must be set so that short range potentials like the LJ potential are negligble beyond it.

Regardless of how big the cutoff length is, cutting off the potential will introduce a discontinuity. There are many possible modifications of the LJ potential that avoid a discontinuity, but I will not go into this matter. The systems I am investigating will be strongly bound, and only the very short-range part of the LJ potential will make an important contribution to the dynamics.

3.2 The Coloumb potential

Electrically charged atoms, ions, interact through the Coloumb potential in addition to short-range quantum mechanically derived potentials like the LJ potential. The Coloumb potential is exact in classical theory, if the quantization of electromagnetic radiation is neglected. However, a more critical approximation is that of describing the abundant or deficient electron as a point charge. This must be done to complete the calculations in a reasonable amount of time. It is, though, possible to use partial charges for the atoms to explain concentrations of electrons (see e.g. [6]).

In reduced units, the Coloumb potential between two particles with charge q_i and q_j is

$$U_{ij}(r_{ij}) = \frac{q_i q_j}{r_{ij}} \tag{3.6}$$

Regardless of how simple it may look, this very important potential causes problems with PBC because of its long ranged nature. To see this, it is useful to write down the total electrostatic energy due to the Coloumb interactions. $\mathbf{R}$ will play the role of a generic translation vector of the simulation cell.

$$U = \sum_{\mathbf{R}} \sum_{i=1}^{N} \sum_{j=1}^{i} \frac{q_i q_j}{\|\mathbf{r}_{ij} + \mathbf{R}\|} \tag{3.7}$$

Even for the simple case of $N = 1$ in one dimension, we encounter infinities. The total potential energy will then be $2q_1^2\zeta(1)$, where $\zeta(n)$ is the Riemann zeta function. $\zeta(1)$ is the harmonic series, which barely diverges. For the case of every other particle on a line being positively charged and the rest negatively charged, the interaction energy of one particle will be an approximation to $2q_1^2 \ln 2$. Even if this is finite, the convergence rate is so slow that the required precision for this kind of calculations will not be achieved in a reasonable amount of time. So in all cases with periodic boundary

conditions in one or more directions, the electrostatic energy of the particles can not be computed using a real space cutoff approach. Many methods have been used to solve this problem. In this project, I use the particle-particle Ewald summation technique, which is the simplest one.

3.2.1 3D Ewald summation

The Ewald summation technique is used to efficiently capture the long range interaction properties of the Coloumb interaction with a finite number of summation terms [1]. It can also be applied to other long range interactions like the van der Waals potential.

First, we need to express the pair potential as a Fourier transform:

$$U_{ij}(\|\mathbf{r}_{ij}+\mathbf{R}\|) = \frac{q_i q_j}{(2\pi)^3}\int \frac{4\pi}{k'^2} e^{i\mathbf{k}'\cdot(\mathbf{r}_{ij}+\mathbf{R})}\mathrm{d}^3\mathbf{k}'. \tag{3.8}$$

The integration goes over the entire reciprocal space. The following identity comes straight from the definition of the reciprocal lattice vectors $\mathbf{k}$:

$$\sum_{\mathbf{R}} e^{i\mathbf{k}'\cdot\mathbf{R}} = \frac{(2\pi)^3}{V}\sum_{\mathbf{k}}\delta^3(\mathbf{k}'-\mathbf{k}) \tag{3.9}$$

The $\mathbf{k}$-vectors are reciprocal to the translation vectors $\mathbf{R}$ for the entire system. V is the volume of the entire system. Using the identities 3.8 and 3.9 in equation 3.7, we will get rid of the $\mathbf{R}$-vector sum and replace it with a sum over $\mathbf{k}$-vectors. Due to the charge neutrality of the systems I am considering, the $\mathbf{k}=0$ term, which corresponds to infinite range interactions, will consist of infinite terms that cancel each other out.

$$U = \frac{1}{V}\sum_{\mathbf{k}\neq 0}\sum_{i=1}^{N}\sum_{j=1}^{i} q_i q_j \frac{e^{i\mathbf{k}\cdot\mathbf{r}_{ij}}}{k^2} \tag{3.10}$$

We now have a sum that is computable, but will converge very slowly. This is because a linear combination of plane waves $e^{i\mathbf{k}\cdot\mathbf{r}_{ij}}$ do a terrible job at describing the charge distributions, which for point particles are Dirac delta functions. The trick of the Ewald summation technique is to add and subtract the total energy of a Fourier transformed potential where the charge distribution is instead a Gaussian:

$$\rho_i(\mathbf{r}) = q_i\left(\frac{\alpha^2}{\pi}\right)^{3/2} e^{-\alpha^2|\mathbf{r}-\mathbf{r}_i|^2} \tag{3.11}$$

The standard deviation is $\frac{1}{\sqrt{2}\alpha}$, where α is an adjustable parameter. By doing this, the total energy will have three terms. The first one, which comes from summing up the Gaussian distribution energies, is long ranged. The second one, which is the correction term (dirac delta minus Gaussian), is very short ranged and can be summed

with neglecting other copies of the system. The third term removes reciprocal space interactions between a particle and itself, and is constant.

$$U = \frac{2\pi}{V} \sum_{\mathbf{k}\neq 0} \frac{1}{k^2} \left|S(\mathbf{k})\right|^2 e^{-\frac{k^2}{4\alpha^2}} + \sum_{i=1}^{N} \sum_{j=1}^{i-1} q_i q_j \frac{\mathrm{erfc}(\alpha r_{ij})}{r_{ij}} - \frac{\alpha}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2 \tag{3.12}$$

The structure factor $S(\mathbf{k})$ is defined as

$$S(\mathbf{k}) = \sum_{j=1} q_j e^{i\mathbf{k}\cdot\mathbf{r}_j} \tag{3.13}$$

and the complimentary error function is $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$, where

$$\mathrm{erf}(x) = \frac{2}{\pi} \int_0^x e^{-t^2} \mathrm{d}t. \tag{3.14}$$

The α parameter will decide which sum converges most quickly. If it is big, the second term converges with fewer terms, but the first one will require more terms, and vise versa for a small α [7].

The total electrostatic energy in equation 3.12 contains not only the distance between particles, but the actual position vector between them. The force arising from the first term must therefore be calculated by taking the gradient directly. This gives terms which can be written as sums of sines and cosines of $\mathbf{k} \cdot \mathbf{r}_{ij}$, but I choose to use complex numbers explicitly, both in derivations and the numerical implementation. The derivative of the error function is a Gaussian. We get the following expression for the force exerted on particle i:

$$\begin{aligned} \mathbf{F}_i = &- \frac{4\pi q_i q_j}{V} \sum_{\mathbf{k}\neq 0} \frac{1}{k^2} \mathrm{Im}\{e^{-i\mathbf{k}\cdot\mathbf{r}_i} S(\mathbf{k})\} e^{-\frac{k^2}{4\alpha^2}} \mathbf{k} \\ &+ q_i \sum_{i\neq j}^{N} q_j \left(\frac{2\alpha}{\sqrt{\pi}} \frac{e^{-\alpha^2 r_{ij}^2}}{r_{ij}^2} + \frac{\mathrm{erfc}(\alpha r_{ij})}{r_{ij}^2} \right) \mathbf{r}_{ij}, \end{aligned} \tag{3.15}$$

where $\mathrm{Im}\{z\}$ is the imaginary part of a complex number z. The particle index i is always used as an index and should not be confused with the imaginary unit i.

3.2.2 2D Ewald summation

In some cases, two-dimensional PBC are used even if the simulated system is in three dimensions. This actually make the Ewald summation more difficult and time-consuming. I have looked at two methods, Kawata and Mikamis (KM method) [8] and the PHL method, presented in the same paper.

For the reciprocal space term of $\mathbf{k} = 0$, which for the 2D case is non-zero, I use the PHL method. For the $\mathbf{k} \neq 0$ terms, I have implemented both methods, but use the KM method for the calculations in this project. This is because the terms are

computationally demanding, and the KM method of computing them has a favourable scaling.

The direction without PBC will in my case be along the x axis, and I refer to the lengths of the simulation box in the other directions as L_y and L_z. The short-range sum will be exactly the same as in the 3D case, so I only present the long range reciprocal space terms here.

The potential energy of the charged systems is the sum of three contributions. The first one is the quasi-2D analog of the reciprocal 3D sum. The $\mathbf{k} \neq 0$ term corrects for that the real-space potential sum has too short a reach in the x direction, where a regular Coloumb potential should have been used. The last term is the same self-interaction correction term as in the 3D case.

$$U^L_{\mathbf{k}\neq 0} = \frac{1}{L_y L_z} \sum_{\mathbf{k}\neq 0} \int_{-\infty}^{\infty} \frac{1}{k^2+h^2} e^{-\frac{1}{4\alpha^2}\left(k^2+h^2\right)} \left|S(\mathbf{k},h)\right|^2 \mathrm{d}h \tag{3.16}$$

$$U^L_{\mathbf{k}=0} = -\frac{1}{L_y L_z} \sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j \left[\frac{\sqrt{\pi}}{\alpha} e^{-\alpha^2 x_{ij}^2} + \pi x_{ij} \mathrm{erf}(\alpha x_{ij})\right] \tag{3.17}$$

$$U^L_{\mathrm{const}} = \frac{\alpha}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2 \tag{3.18}$$

Here, x_{ij} is the distance from particle j to particle i in the x direction. The integration variable h acts as a continous reciprocal lattice vector component in the x direction. The modified quasi-2D structure factor has this form:

$$S(\mathbf{k},h) = \sum_{j=1}^{N} q_j e^{-i(\mathbf{k}\cdot\mathbf{r}_j + h z_j)} \tag{3.19}$$

The force on particle i will have independent contributions in the y-z and x directions. For $\lambda = y, z$, with reciprocal vector components k_λ, the force components are:

$$\left(\mathbf{F}^L_{\mathbf{k}\neq 0,i}\right)_\lambda = -\frac{2}{L_y L_z} \sum_{\mathbf{k}\neq 0} k_\lambda \int_{-\infty}^{\infty} \frac{1}{k^2+h^2} e^{-\frac{1}{4\alpha^2}\left(k^2+h^2\right)} \mathrm{Im}\{e^{-i(\mathbf{k}\cdot\mathbf{r}_i + h z_i)} S(\mathbf{k},h)\} \mathrm{d}h \tag{3.20}$$

The force in the x direction get contributions from both $U^L_{\mathbf{k}\neq 0}$ and $U^L_{\mathbf{k}=0}$. The two contributions are:

$$\left(\mathbf{F}^L_{\mathbf{k}\neq 0,i}\right)_x = -\frac{2}{L_y L_z} \sum_{\mathbf{k}\neq 0} \int_{-\infty}^{\infty} \frac{h}{k^2+h^2} e^{-\frac{1}{4\alpha^2}\left(k^2+h^2\right)} \mathrm{Im}\{e^{-i(\mathbf{k}\cdot\mathbf{r}_i + h z_i)} S(\mathbf{k},h)\} \mathrm{d}h \tag{3.21}$$

$$\left(\mathbf{F}^L_{\mathbf{k}=0,i}\right)_x = \frac{2\pi q_i}{L_y L_z} \sum_{j=1}^{N} q_j \mathrm{erf}(\alpha z_{ij}) \tag{3.22}$$

3.2.3 Implementation details

The short ranged parts of the Coloumb interaction sums are carried out with the same neighbour lists as with the LJ potential. The cutoff length r_c from equation 6.7 is used.

The long ranged parts are summed over **k**-vectors of small length, prioritizing the contributions of longest range, which are the dominant ones. To keep the isotropic nature of the infinite crystal, a spherical cutoff is used, so that $k_x^2 + k_y^2 + k_z^2 \leq k_{\text{cutoff}}$ In the 2D case, k_x is simply put to zero. There will be thousands of vectors in the sum for the 3D case, and hundreds for the 2D case, to put it roughly.

In both the 3D energy and force sums, the symmetry operation $\mathbf{k} \rightarrow -\mathbf{k}$ does not change the inner expression. This is used to reduce the computational cost by a factor of 2. Three sums are being carried out, a triple sum with $k_x > 0$, a double sum with $k_x = 0$ and $k_y > 0$ and a single sum with $k_x = 0$, $k_y = 0$ and $k_z > 0$.

The integrals in the 2D $\mathbf{k} \neq 0$ sums are approximated by using Simpsons composite rule, although the more sophisticated Gaussian quadrature would work very well here.

In some cases, the simulation will contain both particles with and without PBC interactions. The particles which are not periodically repeated will interact amongst each other using a direct Coloumb term, $q_i q_j / r_{ij}$.

In the `cmath` header from the C++ standard library, the functions exp, sin, cos, erf and erfc are all defined and implemented in an optimized way, so these are what I use in the program.

3.3 The Stillinger-Weber potential

Modelling of atoms with an angular potential dependence can be achieved with a three-body potential. For that purpose, I use the Stillinger-Weber (SW) potential created especially for silicon [9]. This potential reproduces the cohesive energy and melting point of silicon matter. I will use a slightly different notation and organization than the standard one.

The total SW potential energy has three terms (system replica sums are dropped for convenience):

$$U = \sum_{i=1}^{N} U_i^{(1)} + \sum_{i=1}^{N} \sum_{j=1}^{i-1} U_{ij}^{(2)} + \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{j-1} U_{ijk}^{(3)} \tag{3.23}$$

In the last term, the sum does not include the terms where $j = i$ or $k = i$. The double counting of index j occures because particle i occupies a special position, the middle of the atom triplet. There is a symmetry between the atoms not in the middle position, $U_{ijk}^{(3)} = U_{ikj}^{(3)}$, but no symmetries between i and j or k.

The first term is constant, and included to get the correct cohesive energy of silicon's crystal structure. There are $\frac{1}{2}N(N-1)$ terms in the double sum, as before, and $\frac{1}{2}N(N-1)(N-2)$ in the triple sum. These are drastically reduced by the potential's natural cutoff, which encourages the use of neighbour lists.

The two-body part of the potential have this form:

$$U_{ij}^{(2)} = \begin{cases} A\left(\frac{B}{r_{ij}^4} - 1\right)\exp\left(\frac{\sigma}{r_{ij}-r_c}\right) & \text{for} \quad r_{ij} < r_c \\ 0 & \text{for} \quad r_{ij} \geq r_c \end{cases} \tag{3.24}$$

All derivatives of the potential are continuous at $r_{ij} = r_c$. The potential can act both attractively and repulsively, like the LJ potential.

The SW three-body potential looks like this:

$$U_{ijk}^{(3)} = \begin{cases} \lambda \exp\left(\frac{\gamma}{r_{ij}-r_c} + \frac{\gamma}{r_{ik}-r_c}\right)\left(C_{ijk} + \frac{1}{3}\right)^2 & \text{for} \quad r_{ij}, r_{ik} < r_c \\ 0 & \text{else} \end{cases} \tag{3.25}$$

There is a term similar to the exponential term in $U_{ij}^{(2)}$, but also a multiplicative term of angular dependence. This can force smaller or greater angles between triplets of atoms, and therefore act both attractively and repulsively in collections of several atoms. The three-body dot product term C_{ijk} is the cosine of the angle from $\mathbf{r}_{ij}$ to $\mathbf{r}_{ik}$:

$$C_{ijk} = \cos\theta_{jik} = \frac{\mathbf{r}_{ij} \cdot \mathbf{r}_{ik}}{r_{ij} r_{ik}} \tag{3.26}$$

The $\frac{1}{3}$ in equation 3.25 is there because solid state silicon atoms are arranged in the diamond structure (section 5.1.2). In this equilibrium structure, each atom has four bonds (atoms within cutoff distance) and the angles between atom pair vectors are such that $\cos\theta_{\text{eq}} = -\frac{1}{3}$ ($\theta_{\text{eq}} \simeq 70.53°$). Other structures can be produced by setting θ_{eq} to a different value.

The constants in the SW potential are adjusted to fit experimental data [9]. They are summarised in table 3.2 with the units I use externally.

r_c [Å]	A [eV]	B [Å^4]	σ [Å]	λ [eV]	γ [Å]
3.7712	12.840	11.603	2.0951	38.248	2.5141

Table 3.2: The six constants required in the SW potential (equations 3.24 and 3.25).

3.3.1 Forces

The two-body force between atoms i and j can be derived straightforwardly from equation 3.24:

$$\mathbf{F}_{ij}^{(2)} = \frac{A}{r_{ij}}\left[\frac{4B}{r_{ij}^5} + \left(\frac{B}{r_{ij}^4} - 1\right)\frac{\sigma}{(r_{ij}-r_c)^2}\right]\exp\left(\frac{\sigma}{r_{ij}-r_c}\right)\mathbf{r}_{ij} \tag{3.27}$$

In calculating the three-body force, one can use that the forces arising from a potential term must be zero if you sum over all three particles: $(\nabla_i + \nabla_j + \nabla_k)U_{ijk}^{(3)} = 0$

[10]. Therefore, the most complicated term, $\nabla_i U^{(3)}_{ijk}$, does not have to be calculated. These are the forces arising from $U^{(3)}_{ijk}$:

$$\begin{aligned}\mathbf{F}^{(3)}_i &= (\nabla_j + \nabla_k)U^{(3)}_{ijk} \\ \mathbf{F}^{(3)}_j &= -\nabla_j U^{(3)}_{ijk} \\ \mathbf{F}^{(3)}_k &= -\nabla_k U^{(3)}_{ijk}\end{aligned} \tag{3.28}$$

Also, the force on particle k can be gained by switching the indices j and k in the force on particle j. Only one term needs to be analytically differentiated.

In differenting the potential, one needs the derivative of the three-body dot product term.

$$\nabla_j C_{ijk} = \frac{\mathbf{r}_{ij}\cdot\mathbf{r}_{ik}}{r_{ik}}\nabla_j\frac{1}{r_{ij}} + \frac{\nabla_j(\mathbf{r}_{ij}\cdot\mathbf{r}_{ik})}{r_{ij}r_{ik}}$$

The first term is done in the same way as with two-body potential (equation 3.2). The second term is problematic, but can be calculated using this general index notation formula:

$$\nabla(\mathbf{A}\cdot\mathbf{B}) = \sum_{ij}\left(A_j\frac{\partial^2 B_j}{\partial x_i^2} + B_j\frac{\partial^2 A_j}{\partial x_i^2}\right)\mathbf{e}_i, \tag{3.29}$$

where $\mathbf{e}_i$ is the unit vector in the ith direction. Using this, I obtain $\nabla_j(\mathbf{r}_{ij}\cdot\mathbf{r}_{ik}) = \mathbf{r}_{ik}$, and the whole derivative becomes:

$$\nabla_j C_{ijk} = \frac{1}{r_{ij}}\left(\frac{C_{ijk}\mathbf{r}_{ij}}{r_{ij}} - \frac{\mathbf{r}_{ik}}{r_{ik}}\right) \tag{3.30}$$

With equation 3.30, the required potential derivative can be calculated.

$$\begin{aligned}\nabla_j U^{(3)}_{ijk} = \lambda\exp&\left(\frac{\gamma}{r_{ij}-r_c} + \frac{\gamma}{r_{ik}-r_c}\right)\left(C_{ijk}+\frac{1}{3}\right) \\ &\cdot\left[\left(C_{ijk}+\frac{1}{3}\right)\frac{\gamma}{(r_{ij}-r_c)^2}\frac{\mathbf{r}_{ij}}{r_{ij}} + \frac{2}{r_{ij}}\left(\frac{C_{ijk}\mathbf{r}_{ij}}{r_{ij}} - \frac{\mathbf{r}_{ik}}{r_{ik}}\right)\right]\end{aligned} \tag{3.31}$$

3.3.2 Implementation details

For storing the interacting neighbours, the same neighbour lists as in the LJ case are used. The cutoff length is set to r_c (table 3.2). Each particle now keeps track of the neighbours with both lower and higher indices than their own. The particle checking for neighbours is always the middle particle, the one with the first index in equations 3.25 and 3.28. The third index will always be smaller than the second one, as when calculating forces between pairs of particles. That way, all bonds between simulated particles are accounted for exactly once.

Sometimes, an open ends boundary condition with fixed particles on the sides are necessary (see section 5.2). Calculation of three-body forces between simulated and

fixed particles requires some extra thought. Four types of three body fixed particle terms (TBFPT) are possible, all illustrated in figure 3.1. By having fixed particles in the neighbour list, the first two terms are automatically included in the sums. TBFPT4 requires fixed particles to have their own neighbour lists, specifying which simulated particles that are close to them. TBFPT3 requires these lists to also include other fixed particles, but force terms are of course only calculated when at least one of the interacting particles are not fixed.

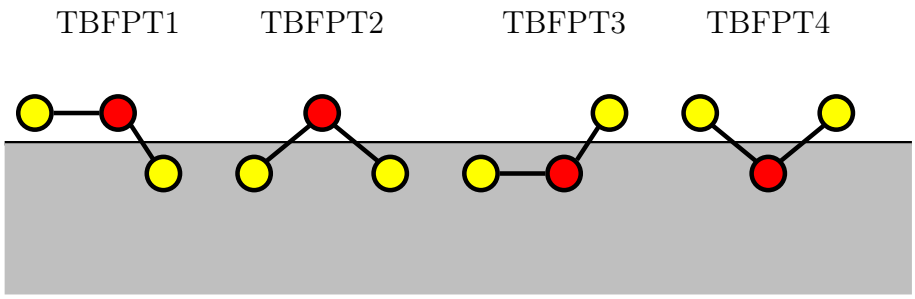


Figure 3.1: The four different three-body fixed particle terms in force and potential calculations. Particles in the gray area are fixed.

3.4 Comparision of the potentials

I have now presented all the interatomic potentials used in this project. Their mathematical forms are very different in nature. The LJ and Coloumb potentials are both spherically symmetric, in the sense that only the distance between the interacting pair of atoms vary the potential energy. The Columb potential, however, have different signs when equally and oppositely charged particles interact. The range of the Columb potential is very long, while the LJ potential has a stronger divergence when atoms are close. The SW potential has one part much like the LJ potential, but also a part which is not spherically symmetric around an atom.

Figure 3.2 shows radial plots of the various two-body potentials used in this project. Sodium parameters are used in the LJ potential, and silicon parameters in the SW potential. As the potentials behave quite differently, plotting them in the same diagram is not very informative. The bottom of the SW potential well is far below what's seen (-1.819 eV) while the LJ well is too small to be seen at all ($-\epsilon = -5.6478 \cdot 10^{-6}$ eV).

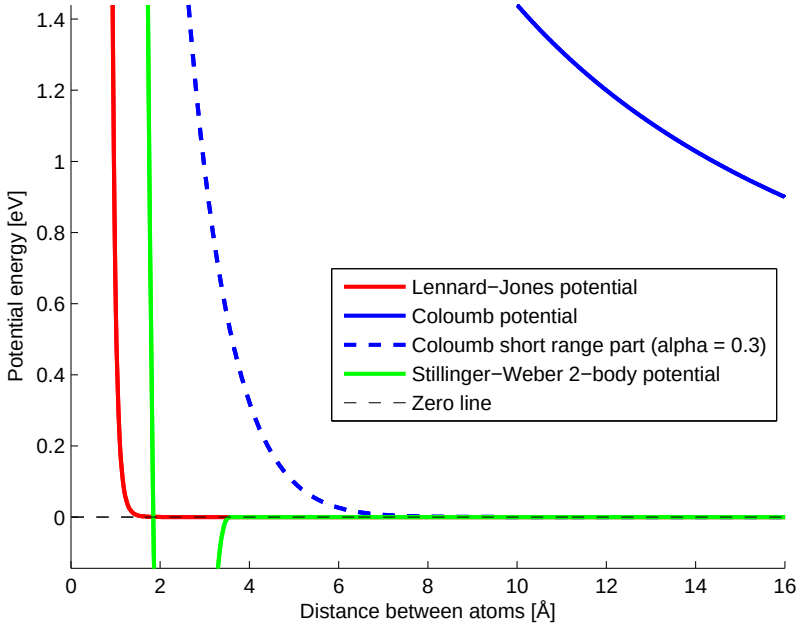


Figure 3.2: Comparision of interaction potentials as a function of the distance between two sodium/silicon atoms.

Chapter 4

Dynamical processes

4.1 Fracture mechanics

It is a well-known fact that the usual reason for fracture in a solid state material is its defects. As discovered by A. A. Griffith in 1921, the only significant quantity that influences the breaking stress and is not material-inherent is the linear size of the biggest flaw. Originally, this meant the length of a crack inside the material, with direction normal to the direction of inflicted stress. As I in my simulations apply periodic boundary conditions in such directions, the flaw will be a void shaped as a sphere or cylindrical disk. The position of these flaws will be of no importance, so they are always placed in the middle of the simulation cell for reasons having to do with visualization. The flaw size refers to the radii of these voids.

Griffiths relation relates the tensile stress at which a fracture occurs σ_f to the flaw size a:

$$\sigma_f \propto \frac{1}{\sqrt{a}}. \tag{4.1}$$

The expression for the proportionality constant was found by Griffith and improved by G. R. Irwin:

$$\sigma_f \sqrt{a} = \sqrt{\frac{(2\gamma + G_p)E}{\pi}}. \tag{4.2}$$

Here, γ is the surface energy density in the material. When a surface is created inside the material, some potential energy corresponding to γ is lost. G_p is the energy dissipation to elastic waves, heat, etc. The value of the parenthesis is the approximate total energy which is converted between different forms in the fracture process. Finally, E is Young's linear elasticity modulus of the material.

The strain ϵ and tensile stress σ can be measured in a molecular dynamics simulation in order to test that it reproduces equation 4.1. I work with an extension in the x direction, and call the equilibrium length L_x and the deviation from this length ΔL_x. The strain is the easiest to calculate:

$$\epsilon = \frac{\Delta L_x}{L_x}. \tag{4.3}$$

Before calculating the stress, the energy balance between to time steps t and $t+\Delta t$ must be examined. The velocity Verlet algorithm is supposed to conserve the system energy, so the only processes that change the energy is temperature adjustment ΔU_T and applied stress ΔU_σ. With U denoting the sum of potential and kinetic energy of the particles, the energy balance equation reads

$$U(t+\Delta t) - U(t) = \Delta U_T - \Delta U_\sigma. \tag{4.4}$$

Thus, by storing the energy of the previous time step and calculating the energy absorbed from the heat bath, the difference in energy caused by stress can be found. The exerted force from the deformation is calculated with the discretized derviative $F_\sigma \approx \frac{\Delta U_\sigma}{\Delta L_x}$, where the minus sign is discarded because the force from the simulated particles act against the elongation. The one-dimensional stress is nothing but force per area, so the average stress (over the system volume) in the x direction is

$$\sigma = \frac{\Delta U_\sigma}{L_y L_z \Delta L_x}. \tag{4.5}$$

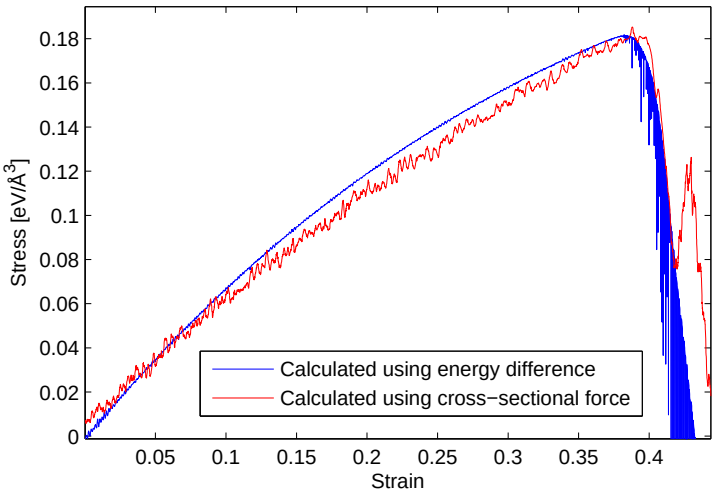


Figure 4.1: Comparision between the two methods of stress calculation, showing a good correspondence.

I have also tested a method of calculating the stress on a specific y-z-cross-section in the middle of the volume. For particle pairs interacting across this cross-section, the force working on the particle on the right side is added up. For particle triplets, the three-body force working on a middle atom is added up if it is on the right side of the cross-section and either one of the other two atoms is on the left side. The cross-sectional stress is then

$$\sigma_{\mathrm{cs}} = \frac{F_{\mathrm{cs}}}{L_y L_z}. \tag{4.6}$$

The stresses σ and σ_{cs} are compared for a 4096-Si-atom system undergoing a fracture in figure 4.1. The energy-calculated stress σ is more averaged (also over the x direction), and has much smaller fluctuations from the general trend than the cross-sectional stress σ_{cs}. Both methods give the same maximal stress, though the slope is slightly different over the whole curve. The results after the fracture are irrelevant.

In the rest of the thesis, I will only mention the x-direction-averaged stress σ. The choice of a cross-section is arbitrary, and the cross-sectional breaking stress will give too low results when the material has defects. These can cause the area around the cross-section to have an atomic deficiency in comparision with ordinary bulk matter. The breaking stress is estimated as the highest achieved stress, which for the solid starts to show signs of failiure. This can be pinpointed by examining when a few particles start to move rapidly in the area of the fracture. The recent motion quantity (section 6.4) is ideal for this.

According to Hooke's law for elastic materials, there is a linear relationship between applied stress and resulting strain, when these quantities are sufficiently low. The proportionality constant is Young's modulus E:

$$\sigma = E\epsilon. \tag{4.7}$$

using this relation, it is an easy task to estimate E. Although it is not included it in this project, estimation of γ and G_p is also possible, so that equation 4.2 may be verified.

4.2 Friction

Friction between materials is a phenomenon which have been observed and measured for hundreds of years, but which have not yet been understood from basic principles. The problem is that it is a quite complex process and there are different questions that have to be asked and answered on different length scales. The only well-known law of friction that approximately describes macroscopic materials is the one formulated by both DaVinci, Amontons and Coloumb. It states that the force of friction is neither dependent on the sliding velocity v or object area A, but proportional to the normal force (or load) L.

$$F = \mu L \tag{4.8}$$

The proportionality constant μ is called the friction coefficient. It is material dependent and most often takes values between 0.2 and 0.8.

Even if one sticks to the macroscopic scale, researchers have discovered several complications by the use of experiments and computer models. One needs for instance to consider the coupling between the mechanical process and heating of both surfaces in contact. Important mechanical phenomenons that appears when sliding two rough materials against each other are fractures and slips near the material interface [11].

A general observation done in the course of the last decades is that the friction force indeed varies with the contact area between the materials. This is, however, the

real area of contact A_C, which is much smaller than the geometrical area A of the material slabs. As surfaces have a roughness on all scales, only a very few asperities make contact to counteract the load L. Ringlein and Robbins [12] provide a refined friction formula based on their research on friction:

$$F = \mu L + cA_C \tag{4.9}$$

They also point out that reason that equation 4.8 often work well for macroscopic materials is that the real area of contact A_C often is proportional to the load L, giving apparently only a load dependece.

Digging deeper into the origins of friction requires understanding of the friction between asperities, which I attempt to model in this thesis. This have been investigated by measuring the stress between a sphere and a plane or between two spheres of some type of atoms [13, 14].

4.2.1 Contact area

Defining the area of contact between materials becomes problematic on the atomic scale. How big is the contact area between an atom and a surface of atoms? This could be solved by a geometric approach of finding an area enclosing all atoms close to the surface, but such a method will require too many parameters and will not be reliable enough for my purpose. It is a better idea to assign a cross-sectional area to an atom based on how far it is from the surface atoms [14]. The method I use is based on this idea.

The area of contact A_C is measured in my simulations by calculating the potential energy between particles belonging to the sphere and the surface, U. This energy is then divided by the surface energy per area of bulk material, σ_0:

$$A_C = \frac{U}{\sigma_0} \tag{4.10}$$

The constant σ_0 is material dependent and must be measured in a computational experiment. In order to to this, I simultated bulk crystals with PBC in the y and z directions and walls of fixed particles in the x direction. The simulation was performed at a temperature of 10 K for the potential energies of the atoms not to have too great deviations from the minima. I then defined σ_0 as the calculated potential energy between simulated particles and left side fixed particles, divided by the area $L_y L_z$.

My results are $\sigma_0 = -0.2471$ eV/Å^2 for silicon and $\sigma_0 = -0.02522$ eV/Å^2 for sodium chloride. The NaCl calculations were done with four atomic layers of fixed particles on the sides of the simulation cell. The numbers show no deviation when changing the system size up or down from 4096 particles.

When pushing a sphere and a surface of equal materials together, the A_C vs t graph is continous and seems to describe well what is going on. The system does not reach the same equilibrium state for slightly different initial conditions (because of a different random generator seed). The atoms in the interface will often be disordered

and able to fall into many local energy minima. In such types of experiments, a contact area measurement will have a low precision and is only used to extract qualitative information about trends for different cases.

Chapter 5

Systems

The systems I consider are atoms initialized to an energetically preferrable crystal structure. The velocities of these atoms are generated according to the Maxwell-Boltzmann speed distribution for a given initial temperature. The total linear momentum of the system is set to zero before a simulation starts.

5.1 Materials

5.1.1 Sodium chloride

Sodium chloride is a solid composed of Na^+ and Cl^- ions forming an ionic bond lattice. The mean mass of sodium atoms and chlorine atoms are 22.9898 amu and 35.453 amu, respectively. Sodium chloride has a density of 2165 kg/m^3, corresponding to a mixed particle density of 0.045 Å^{-3} [15].

The most stable crystal structure of sodium chloride is the double face centered cubic structure. Each of the ions are placed in a Bravais FCC lattice, displaced with a vector $\mathbf{u} = \frac{a}{2}\mathbf{e}_x + \frac{a}{2}\mathbf{e}_y + \frac{a}{2}\mathbf{e}_z$, where a is the lattice constant. The basis used in my program is arranged so that surfaces parallell to the cartesian axes will be charge neutral when copying a unit cell (visible in figure 5.7). a has the value 5.63 Å. The conventional unit cell has four atoms of each type. One filled unit cell of a NaCl crystal is shown in figure 5.1.

For the interactions amongst sodium and chlorine ions, I use the LJ and Coloumb potentials. These will imitate the very short-range nuclear repulsion and the ionic bonding, respectively. The short-range Coloumb interaction is strong enough that the attractive force of the LJ interaction is almost negligble in the dynamics. The most important role of the LJ potential is stopping equally charged particles from crashing into each other. The LJ parameters can be found in section 3.1.

5.1.2 Silicon

Silicon is a semi-metal which forms covalent bonds between the atoms. The mean mass of a silicon atom is 28.0855 amu. Its solid phase density is 2329 kg/m^3, corres-

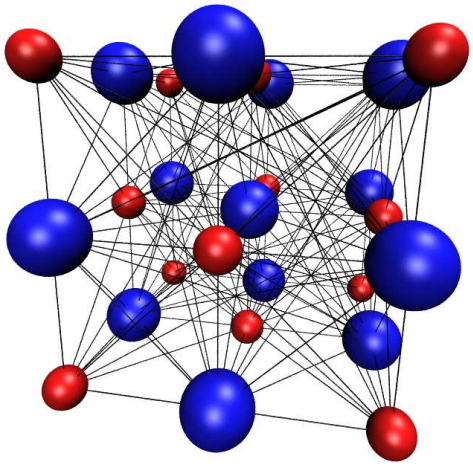


Figure 5.1: Conventional double face centered cubic unit cell with Na^+ and Cl^- ions.

ponding to a particle density of 0.050 $Å^{-3}$ [15].

The most stable crystal structure of silicon atoms is the diamond structure. The atoms are placed in two Bravais FCC lattices, displaced with a vector $\mathbf{u} = \frac{a}{4}\mathbf{e}_x + \frac{a}{4}\mathbf{e}_y + \frac{a}{4}\mathbf{e}_z$. The conventional unit cell has eight atoms and a lattice constant 5.43 Å. Figure 5.1 shows one filled unit cell of a diamond crystal consisting of silicon atoms.

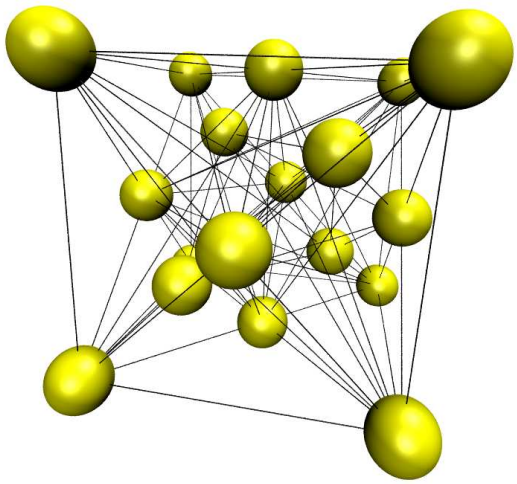


Figure 5.2: Conventional diamond unit cell with Si atoms.

The silicon atoms interact with the specialized SW potential, which should give a highly accurate description of the covalent bonds. The angular dependence in this potential is the reason for the equilibrium diamond structure. Silicon has a remarkable

feature which it shares with water. The liquid phase has a higher density than the solid (crystalline) phase. Thus, a silicon crystal will melt when compressed to a sufficiently low density.

5.2 Boundary conditions

In my molecular dynamics simulations, all the particles are either free or in some way confined to a cuboid, a rectangular prism with right angles. This box, which I call the simulation cell, has side lengths L_x, L_y and L_z in the three cartesian directions.

5.2.1 Periodic boundary conditions (PBC)

In materials science MD simulations, the most used kind of boundary condition is the periodic one. Call the length of a particular PBC direction of the simulation box L. If a particle escapes the box on the left or right side, its position will be shifted in that direction with $+L$ or $-L$, respectively. Particles will also interact with an infinite number of copies of themselves and other particles in the system. If all three directions have PBC, the vectors to these system copies are $\mathbf{R} = nL_x\mathbf{e}_x + lL_y\mathbf{e}_y + mL_z\mathbf{e}_z$, for integers n, l and m. There will not be any edges, only an infinite amount of bulk particles.

The reason for using PBC is the interest of what is happening inside materials. A great amount of research connected to the properties of surfaces are taking place, but PBC is still used in this case in the directions parallell to the surface, to have only one e.g. material-air-interface. A box with PBC in all directions will enable the possibility of bulk particle simulation. The simulated atoms will behave like atoms in the middle of a material of macroscopic size.

In the program, all particles have flags which tell in which directions they experience PBC. This is useful in adhesion and friction experiments, where a group of particles with no PBC is in contact with a surface. The surface particles, either if fixed or simulated (semi-fixed), will always have PBC in the directions tangential to the surface. When a pair of particles interact, the interaction itself will have PBC in a certain direction if one of the particles experience PBC in that direction. As an example, sliding particles will not fall of the edge of a surface, but continue to move as if the surface continued to infinity in its PBC-directions.

5.2.2 Fixed particles

The simplest kind of boundary condition is open ends, that is, nothing is done to prevent particles from escaping. In fracture experiments, I use this boundary condition in one direction in tandem with fixed particle forces. A perfect lattice of fixed particles are placed at the sides in the direction without PBC. This will not strictly contain the particles in the box, but prevent them from dispersing towards infinity.

The thickness of the fixed particle lattice layer will always be greater than the cutoff length for forces and potentials. If the simulated particles are in a crystalline state, the difference between the dynamics with PBC and open ends with fixed particles in one direction is almost negligble.

An application of this boundary condition is to study a crack propagating in the y-z-plane in an otherwise perfect crystal. Two-dimensional y-z-PBC with open ends and fixed particles in the x direction will limit the number of crack copies in the x direction to one. Figure 5.3 shows the two-dimensional analog of the setup schematically.

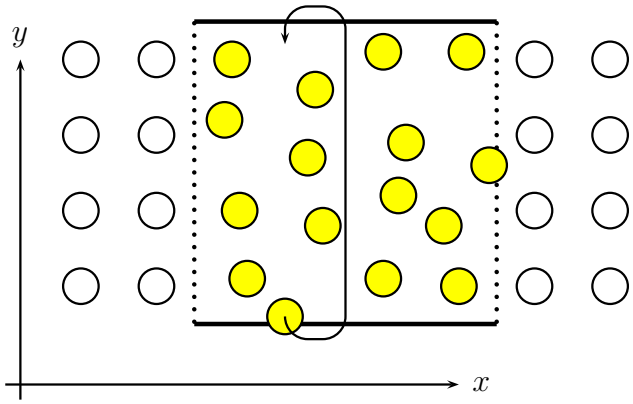


Figure 5.3: The boundary conditions for a system with fixed particles on the x-sides. The thermal motion is greatly exaggerated.

5.2.3 Friction BC

In adhesion and friction experiments, I am interested in a solid with a surface at $x = 0$ and an infinity of unit cells in the $-x$ direction. The electrostatic interactions are in this case hard to compute with the Ewald summation method. There has been attempts, e.g. [16]. However, it is stated in the results section of this paper that there is only a 1% difference in the interaction energy between a single ion and my hypothetical half-infinite NaCl solid and between a single ion and a two-dimensional monolayer of NaCl. The reason for this is the obvious cancellation of forces, making the effective range of interaction for a neutral layer of NaCl approximately one lattice constant a. For this reason, I model the surface which the sphere of atoms interact with as a slab of thickness equal to the cutoff range of the potentials summed in real space, for both the Si and NaCl systems. In practice, this means about 4 atomic layers.

With simulation tests, the lattice energy of such a system was measured to be -4.014 eV per particle, a difference of 0.55% from the bulk system with PBC in all

directions (see section 7). For this test, layers of fixed particles were put on both x-sides, and PBC was applied in the y-z-plane. When a sphere of particles is used, the positive x direction layers of fixed particles are removed and replaced by vacuum.

5.2.4 Heat conduction

When simulating the dynamics of systems far from equilibrium, one can not assume that the heat bath has an equal thermal coupling to all the simulated particles. In most cases, I am interested in thermal coupling to the perfect crystals on the sides of the simulation cell in one direction, as explained in the previous section. This can be accomplished by performing a temperature rescaling only on the atoms near the cell sides. For this purpose, I use the BDP thermostat introduced in section 2.3.1.

To avoid an unreasonably sudden cutoff, the relaxation time τ_i of the thermostat is choosen differently for each particle. An atom directly on the side wall will have a relaxation time τ_0 read in as a parameter. Diverging at the cutoff point, the relaxation time decays from this point as $1/x$. Atoms which are further inside the cell than the cutoff point have no thermal coupling to the outside. A $1/x$ form is chosen because the relaxation time enters the energy difference equation 2.10 as $1/\tau$. So the energy coupling to the outside heat bath decreases linearly inwards from the cell sides.

Since the relaxation time varies with the distance from the sides, how temperature is measured before a rescaling must be changed. Fouriers (macrocopic) law of heat conduction says that the rate at which heat enters a system is proportional to the difference between the temperatures of the system and the surroundings. Equation 2.10 takes this directly into account. As the measured temperature affects the thermostat, the temperature must be measured only near the sides of the cell. If not, the edges would be cooled down too much when the temperature of the middle of the cell rises.

I choose to weight the energies of the atoms by a linearly decreasing factor w_i in the temperature measurement. These weights are normalized so that the measured side temperature is

$$T = \frac{1}{3k_{\mathrm{B}}} \sum_{i=1}^{N_{\mathrm{incl}}} w_i m_i (v_i - v_{\mathrm{coll}})^2 \quad \text{with} \quad \sum_{i=1}^{N_{\mathrm{incl}}} w_i = 1 \tag{5.1}$$

where N_{incl} is the number of atoms included in the rescaling. v_{coll} is the collective (average) velocity of the particles on one of the sides. Removing this component is necessary when all the atoms are moving in one direction on one of the sides, e.g. when shear stress is applied.

This method has no solid physical foundation, but serves as a tunable way to consistently conduct heat in and out of the system. The amount of heat conducted can be seen by plotting the power absorbed from the heat bath, as in the case of uniform atom to heat bath coupling.

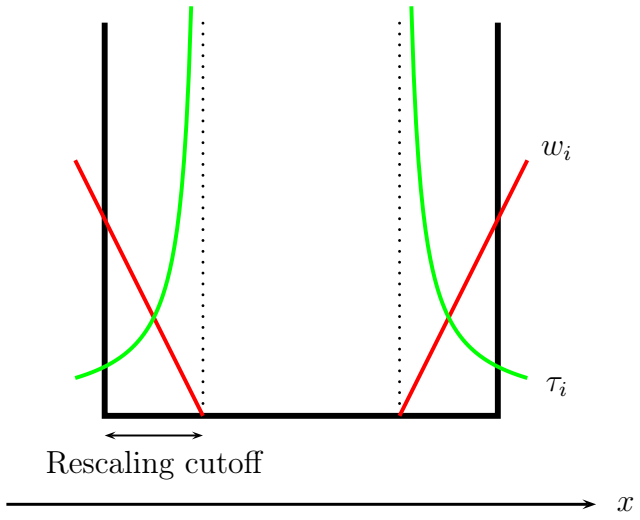


Figure 5.4: The variation of the temperature measurement weight w_i and thermostat relaxation time τ_i for a particle i.

5.3 Applied deformations

5.3.1 Tensile stress

To see how much stretching a material can take before it fractures, I apply a steadily increasing strain to the system. The typical speed at which the system is elongated is 5-10% of the speed of sound in the material.

Open ends with fixed particles is the x direction boundary condition. The distance between the fixed particles will not be stretched, as they exist only to keep the simulated particles at the edges in place. The distance between simulated particles will be uniformly stretched to correspond to the stretching of the simulation box and moving of the fixed particles. If ξ denotes the distance from the middle to a particle, the displacement rate caused by tensile strain will be $\dot{\xi} = \frac{\xi v}{L}$, where L is the length of the box, and $v/2$ is the expansion rate of the box in one direction.

The displacements by the tensile strain function will not give any contribution to the velocity of particles. This would disturb the dynamics and give higher temperature measurements.

5.3.2 Shear stress

A shear deformation exerts inhomogenous stress on different parts of a material. With fixed particles outside the x direction edges, I move these particles in opposite directions in the y direction. This alone will move the simulated particles correspondingly at each side. The interesting thing will be what happens in the middle of the

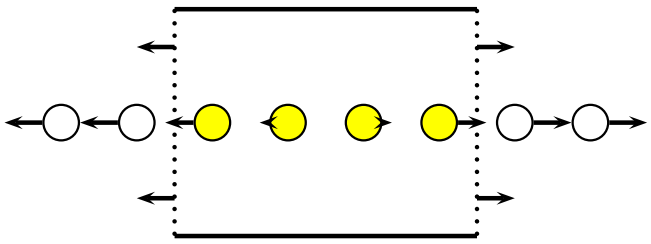


Figure 5.5: Displacements caused by tensile strain.

simulation cell. Again, the speed of the fixed particle displacements are constant.

The fixed particles themselves experience PBC in the y direction, so that they appear at the opposite y side if they are displaced out of the cell.

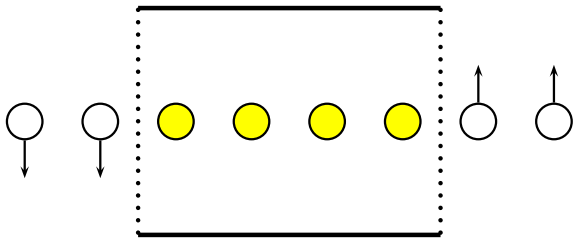


Figure 5.6: Displacements caused by shear strain.

5.3.3 Adhesion and friction

In numerical experiments regarding adhesion and friction, I look at the interaction between a sphere or sphere segment of particles and a plane surface of the same types of particles. The attraction between such interfaces with the same atomic species is called cohesion, but I will stick to the more general word adhesion. The sphere form is meant to model an asperity on a rough material surface. It corresponds to the stepped crystal surface in a similar model by Luan and Robbins [13]. The surface consists of some atomic layers of fixed particles, as mentioned in section 5.2.3. Additional layers of simulated particles can be added for the surface not to be completely rigid. The surface particles have PBC in the directions parallell to the surface, while the sphere particles have no PBC.

The sphere is pushed down towards the surface by a constant force on the upper half of the particles. This force corresponds to a homogenous gravitational field,

although it is several thousands times stronger. An alternative would be a force on all particles in the sphere, but this is unphysical and could cause the lower half of the sphere to disintegrate in a unrealistic fashion. Another alternative, which I also use, is pushing only a lower sphere segment downwards using e.g. another surface of the same material.

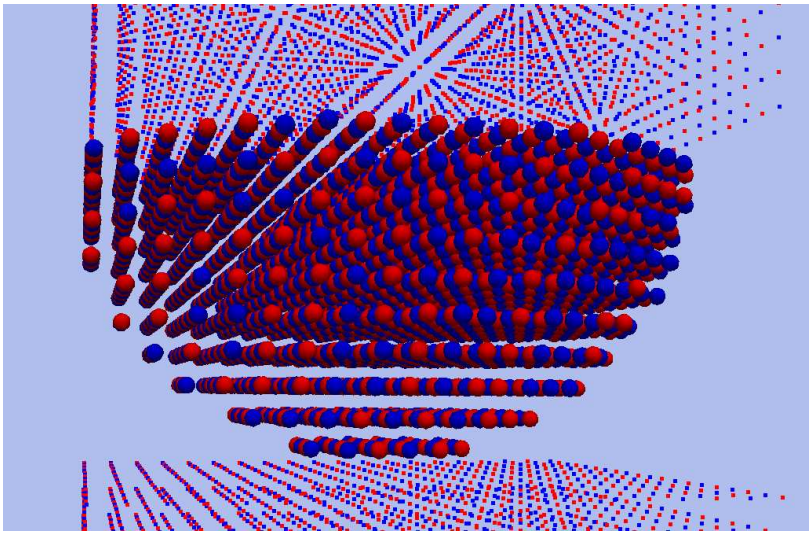


Figure 5.7: A microscopic sphere segment and surface of NaCl. The initial positions in a simulation.

The sphere method is used to simulate the reaction from pushing the sphere with forces of different magnitude. Potential energy from the constant force in the negative x-direction is summed up and added to the total and single particle energies. However, the number of particles this force affects is not constant. Only particles positioned higher than the mass center of the sphere are pushed. This is checked for each time step. If the sphere begins to roll, as it may do in friction experiments, the upper half of the sphere is still the only part that is forced downwards, giving a greater stability. The total force, which I for simplicity call $\mathbf{F}$, is distributed over the atoms above the sphere's center of mass in the following way:

$$\mathbf{F}_i = \frac{m_i}{\sum_j m_j}\mathbf{F}, \tag{5.2}$$

where m_i is the mass of an atom i and j runs over all the inflicted particles. This will cause the force to be distributed non-uniformly across the cross-section of the sphere, because there are more particles above the middle than at the edges. The effect of this unphysical force distribution on the lowermost part of the sphere is small, since inner tension in the sphere will even the forces out.

The sphere segment method will simulate a smaller number of dynamical particles and demand less computing time for the same sphere curvature. The input force distribution will also be uniform. In the simulations, The upper fixed particle layers

will be moved downwards very slowly with a constant velocity and then stop for measurements to be performed. The applied force is difficult to control when this approach is used. Though, the force can be measured, and the interface stress graphs give results very similar to those obtained with the sphere method. A possible method which I have not tried in practice is basing the movement of the upper fixed particles on the force between the simulated particles and the lower fixed particles. In this way, the entire surface could be moved until the forces reached a specific value.

The contact area between the sphere and the surface A_C is measured as explained in section 4.2.1. The stress between the sphere and the surface σ is measured in a similar way, but using microscopic forces instead of potential energy. The surface is divided into rectangular bins, about 20 in each direction. For each time step, the sphere particles are distributed among these bins according to their positions, and the forces from the surface particles are calculated. For each bin b, all forces on sphere particles are summed up and divided with the area of the bin to find the stress,

$$\sigma_b = \frac{N_b}{L_y L_z} \sum_{i \in b} \sum_{j \in \text{surface}} F_{ij}, \tag{5.3}$$

where N_b is the number of bins and $L_y L_z$ is the area of the surface (inside the PBC cell). This gives an approximation to the stress normal to the surface. I have also attempted to simplify this to stress in donut-shaped bins for specific radii $\sigma(r)$, with less stable results. Long range Ewald forces between simulated and lower fixed particles are neglected when the interface stress is calculated. Early measurements show that these are many orders of magnitude (10^{-16}) smaller than the other force contributions.

Once the sphere is pushed towards the surface and the system is near equilibrium, the simulation of the dynamics can take place. Friction force is most often measured in the case that two interfaces have a constant relative velocity. This condition is applied by adding a small velocity to all particles so that the collective velocity of the sphere $\mathbf{v}_{\text{coll}}$ is constant. The only large scale forces that are at work tangential to the surface are the driving force that keeps the constant velocity condition fulfilled and the friction force. By Newton's first law, these forces are equal. For each time step, the friction force is therefore calculated in this way:

$$F_{\text{friction}} = \frac{\Delta U_{\text{drive}}}{\Delta r_{\text{COM}}} = \frac{\Delta U_{\text{drive}}}{v_{\text{coll}} \Delta t} \tag{5.4}$$

This is a discretized derivative with no minus sign, so the force is positive when it works in the same direction as $\mathbf{v}_{\text{coll}}$. U_{drive} is simply the change in kinetic energy when driving the particles to keep a constant velocity.

Chapter 6

Program details

6.1 Physical units

The physical units used in input and output (external units) are chosen so that the order of the values will be close to, or mostly larger than, 1. These units must be converted to and from an additional set of units used for calculations internally in the program. Variables in internal units $\bar{A}$ are related to the corresponding variables in external units A in this way: $A = A_0\bar{A}$.

Since position and velocity are the most frequently output variables, I use the same units externally and internally for length and time. I also decided to put the only important constant, the Coloumb force constant, to 1.The charge of particles will be measured in elementary charges, $e = 1.602176487 \cdot 10^{-19}$C. The Coloumb force law will give rise to a constraint in this way:

$$F = k_e \frac{q_1 q_2}{r^2} \quad \Rightarrow \quad F_0 \bar{F} = k_e \frac{e^2 \bar{q}_1 \bar{q}_2}{r_0^2 \bar{r}^2} \tag{6.1}$$

$$\bar{F} = \frac{\bar{q}_1 \bar{q}_2}{\bar{r}^2} \quad \text{and} \quad \frac{k_e e^2}{r_0^2 F_0} = \frac{k_e e^2 t_0^2}{m_0 r_0^3} = 1 \tag{6.2}$$

This determines the mass conversion factor, m_0. Other conversion factors are combinations of r_0, t_0, e and m_0. The detailed list of used units are in table 6.1.

6.2 Random numbers

Amongst other things, getting velocity values from the canonical distribution requires randomly distributed numbers. For numerical simulations, deterministic pseudo-random number generators are sufficient. My program uses a complex long period generator developed by L'Ecuyer and Bays-Durham. The implementation from *Numerical Recipes* [17], which I have copied, contains some added safeguards. The generator gives me uniformly distributed values in the open interval $(0, 1)$.

Normally distributed random numbers are obtained by performing a Box-Muller transform on the uniformly distributed numbers. Let u and v be uniform numbers in

	External	Internal
Length	Å	$r_0 = 1$Å
Time	fs	$t_0 = 1$fs
Charge	e	$q_0 = e$
Mass	amu	$m_0 = k_e t_0^2 e^2 / r_0^3 = 0.138935459$ amu
Force	eV/Å	$F_0 = m_0 r_0 / t_0^2 = 14.399645$ eV/Å
Energy	eV	$E_0 = m_0 r_0^2 / t_0^2 = 14.399645$ eV
Temperature	K	$T_0 = m_0 r_0^2 / k_B t_0^2 = 1.6710075 \cdot 10^5$K

Table 6.1: Physical units for I/O and calculations in the program.

the interval $(-1, 1)$. These numbers will only be accepted for the transformation if $s = u^2 + v^2$ is in the interval $(0, 1)$. In that case, we obtain two normally distributed numbers n_1 and n_2 by multiplying u and v with a constant,

$$n_1 = Su, \qquad n_2 = Sv, \qquad S = \sqrt{\frac{-\ln s}{s}}. \tag{6.3}$$

n_1 and n_2 will have standard deviations of 1, but multiplying all generated numbers with a constant will give a distribution with that constant as the standard deviation.

6.3 Parallellization

As my program requires a considerable amount of CPU time for the systems I am simulating, parallell computations with several processors are required. I therefore use MPI to pass data between the processor cores (called nodes) involved in the calculations.

For the most CPU-intensive tasks, calculations are split into work lists for each node. This happens to quantities that are sums of many terms. Each node performs its own sum and returns the result to a master node. The master node, which also does calculations, sums up the partial sums and broadcasts total results to all nodes.

The most important section of the program that is parallellized is the computation of forces. Especially with the very intensive Ewald summation, this has a high degree of parallel efficiency. The forces that must be computed makes an upper triagonal matrix with the index of particles in an interaction i and j. The rows of this matrix are put into work lists so that each node gets an equal amount of force terms F_{ij} to calculate. The exception is the last node, which gets "what's left". Figure 6.1 shows this splitting of work.

The same procedure is used in calculating the total potential energy of the system. The BDP thermostat requires the sum of many squared random numbers. This is also done distributively, with different seeds for the random generators. The random numbers for the initial velocities are calculated in serial, and drawn from random generators with the same seed, producing the same numbers for all nodes.

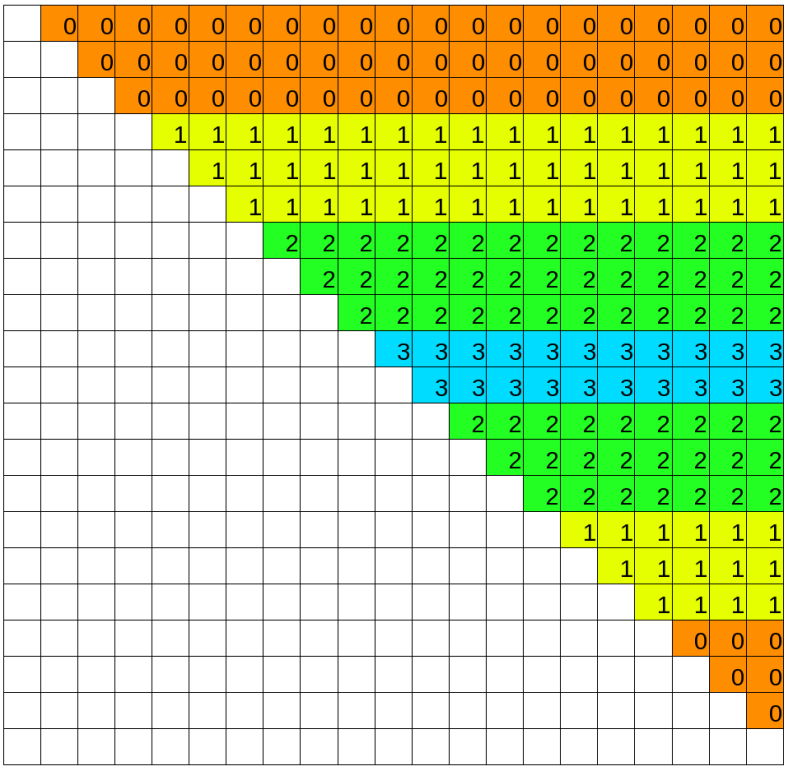


Figure 6.1: Splitting the calculation of the forces between 20 particles amongst four nodes.

Everything that is not parallellized, like performing the Verlet time integration of positions and velocities, is done simultaneously at all nodes. This reduces the amount of communication between different memory registers. On heterogenous clusters, this could cause deviations between the nodes induced by unequal round-off errors. Output is done only from the master node.

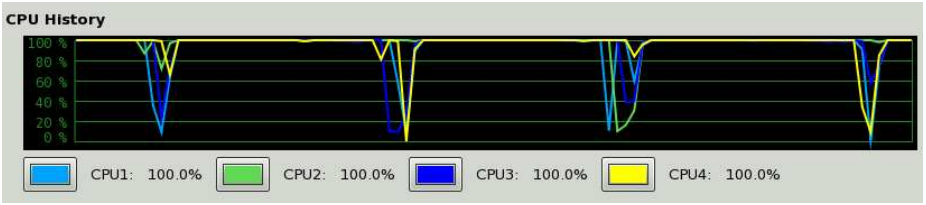


Figure 6.2: CPU usage in a big silicon system.

For small systems, parallellization will require too much time for communication between nodes in comparision to the actual computations. The force calculations must therefore be quite extensive for parallellization to give an increased efficiency. Figure 6.2 shows the CPU usage on a quad-core computer running a 15581-Si-atom

simulation. The calculations involved in one time step takes about 25 seconds. 20 seconds of the time, all the cores are busy with force calculations, and the last 5 seconds are apparently spent at communication, output and waiting for other cores. This suggests a rough estimate of the efficiency. A single core could have done the calculations for one time step in 80 seconds, giving a parallellization efficiency of $80/(4 \cdot 25) = 80\%$ for 4 nodes. Simulations with the more demanding NaCl systems on a various number of cores give similar numbers when the time usage is compared.

The neighbour list update process is another program segment which could gain a speed-up from parallellization. This however involve big difficulties and require much communication between nodes. Therefore, it continues to be a $O(N^2)$ bottleneck for very big systems.

The reason why I'm not using parallellization in Si experiments is that I make these physically big (like the one mentioned), which require a significant amount of memory. This memory amount is still small enough so that the most frequently accessed data can be stored in the CPU's cache, which enables the program to work about ten times faster than it would otherwise. The cores of the CPU have their individual memory, so a four-core parallell run would require four times as much memory, and the most frequently used variables and arrays will not fit in the cache, slowing the program down. This problem does not not occur for the smaller NaCl systems.

6.4 Visualization

For visualization of atom trajectories, I use the program ParaView. It is a general purpose visualization program built on the C++ code VTK (Visualization toolkit). My output class writes to files of the legacy `.vtk` format.

Plotting of atom positions is performed using polydata with simple vertices. In this report, I usually show the positions with a parallell prosjection. Scalar values for each atom includes electric charge, potential energy and recent motion. They also have velocity and force vectors associated with them, which can be easily visualized in ParaView. For the images included in this report, I use both orthographic (parallell) and perspective projections.

Recent motion is an ad hoc quantity for seeing where things happen to which particles. For one particle, I define $E_{\text{over}} = E_k - n_\sigma E_{\text{thermal}}$, where $E_{\text{thermal}} = 3k_{\text{B}}T_{\text{bath}}$. If E_{over} is negative, I put it to zero. This way, it gives a measure of how much the kinetic energy of a particle exceeds a certain number of standard deviations in the Maxwell-Boltzmann distribution for the heat bath temperature. Good values are $n_\sigma = 2$-3. Recent motion is then defined as

$$\texttt{recmot} = \sum_{i=0}^{\infty} a^i E_{\text{over}}(t - i\Delta t), \tag{6.4}$$

where a is a decay parameter, set to e.g. 0.98. The quantity will be mostly zero when the particles are in equilibrium, and significant if particles have moved recently, as

in dynamical processes like fracture. When visualizing, this gives a better picture of what is happening than the magnitude of the instantaneous velocity.

Field values are stored as structured points. The types of fields I have been working with is potential energy, density and temperature. The volume of the system is cut into cells where these quantities are approximately evaluated. Fields can be viewed as three-dimensional scalar functions or two-dimensional slices.

In some cases, atom positions can also be output as files of the simpler `.xyz` format for visualization in VMD (Visual Molecular Dynamics). Macroscopic quantities for the whole system are treated in a more basic way. Energies, temperature, etc. are output to a file for each time step for later plotting with Matlab.

6.5 Structure overview

My simulation program is written in C++, contains 9 classes and a total of about 4400 source code lines. (CHECK THIS!) The `main`() function reads parameters from a file, acts as a stopwatch, manages and backs up the files in the output folder and outputs most of the cosole messages. It contains the equilibration and dynamics time loops, and administers what is happening and what is being measured through the public methods of a `dynamics` object. Some error checks are performed, and great increases in temperature (implying a critical event) are reported.

Molecular dynamics classes

- `dynamics`: Master class which contains the essential calculations and depends on most of the other classes.
  - `set_parameters`(): Transfers some parameters from the input file to this class and determines Ewald summation parameters.
  - `initialize`(): Declares all required objects, arrays and output files. Also sets the initial positions and velocities of the particles.
  - `velocity_verlet_1`(): Calculates half-step velocities and new positions. See section 2.2.
  - `velocity_verlet_2`(): Calculates new velocities.
  - `position_pbc`(): Applies periodic boundary conditions to the particle positions. Called at each time step.
  - `define_equilibrium`(): Defines equilibrium values for some quantities.
  - `calc_properties`(): Calculates some macroscopic quantities, e.g. stress and strain, and the recent motion of particles.
  - `forces_potentials`(): Calculates forces and potential energy between all pairs and triplets of atoms, also between fixed and non-fixed particles.

- `update_neighbours`(): Updates the neighbour lists of all particles. Called e.g. every 10 time steps.
- `rescale_temperature`(): Simple temperature rescaling. Usually not in use.
- `andersen_thermostat`(): Adjusts the temperature using the Andersen thermostat. Used for equilibration in specific cases.
- `bdp_thermostat`(): Adjusts the temperature using the BDP thermostat (section 2.3.1).
- `bdp_thermostat_sides`(): Adjusts the temperature of the simulation box sides (section 5.2.4).
- `dissipative_force`(): Adds a friction force term to all particles for damping oscillations.
- `expand_volume`(): Expands the simulation cell in the x direction with a constant length. Used every time step in tensile stress simulations.
- `displace_lattice`(): Displaces the fixed particles in the perfect crystals on the sides of the simulation cell. Used every time step in shear stress simulations.
- `insert_gap`(): Inserts a gap between crystal layers in the x direction.
- `add_force`(): Adds a constant force on specific particles and adds up the corresponding potential energy.
- `adjust_vcoll`(): Forces the collective velocity to a certain vector by adding a correction term to certain particles.
- `kin_energy`(): Calculates the kinetic energy of the particles.
- `temperature`(): Calculates the average temperature of atoms belonging to a specific type, subtracting the collective velocity.
- `output_pos`(): Outputs the positions of all atoms, including the vector and scalar values mentioned in section 6.4.
- `output_field`(): Outputs the temperature and density fields.
- `output_xyz`(): Outputs the positions of the simulated particles to a simple `.xyz` file for visualization with VMD.
- `fpot_short`(): Adds up short range forces and potentials, including LJ and short range Ewald terms.
- `fpot_coloumb`(): Adds up LJ and direct Coloumb forces and potentials, for systems with no PBC.
- `fpot_long`(): Adds up 3D long range Ewald forces and potentials.
- `fpot_long_2d_1`(): Adds up $\mathbf{k} \neq 0$ 2D long range Ewald forces and potentials.

  - `fpot_long_2d_2`(): Adds up $\mathbf{k} = 0$ 2D long range Ewald forces and potentials.
  - `potential_constant`(): Calculates the constant correction term in the Ewald summation. Only called in the `initialize`() function.
  - `fpot_sw_2b`(): Adds up SW potential two-body forces and potentials.
  - `fpot_sw_3b`(): Adds up SW potential three-body forces and potentials.

- `phase`: The `main`() function reads phase-specific parameters like the number of time steps, thermostat type and temperature, output level and parameters concerning dynamic processes. These parameters are stored in objects of this class. Also contains timer objects for timing the execution of each phase.

- `atom_type`: Contains information about chemical elements, like mass, charge and LJ parameters.

- `atom`: Contains the position, velocity, and force on an individual atom.
  - `dist2`(): Calculates the squared distance between two atoms, taking PBC into account.
  - `dist_vector`(): Calculates the distance vector between two atoms, especially useful in three-body force calculations.
  - `dotted_dists`(): Calculates the scalar product $\mathbf{r}_{ij} \cdot \mathbf{r}_{ik}$ for particles i, j and k, for use in three-body force calculations.
  - `update_recmot`(): Updates the recent motion quantity of an atom.
  - `flush_neighbours`(): Clears the neighbour list of an atom.
  - `add_neighbour_si`(): Adds a neighbour which has a lower index than this atom.
  - `add_neighbour_li`(): Adds a neighbour which has a higher index than this atom. Required only for three-body force calculations.

- `lattice`: Generates a lattice with the chosen geometry and elements.
  - `generate_positions`(): Fills a two-dimensional array with position values for all atoms in a lattice for later use.
  - `spherical_void`(): Removes particles inside a sphere with a specific position and radius.
  - `cylindrical_void`(): Removes particles inside a cylinder with a specific position, radius and height.
  - `sphere`(): The "opposite" of `spherical_void`(). Cuts the crystal to a sphere, can leave atomic layers of simulated particles on top of the fixed ones. Preserves charge neutrality.

  - `remove_particle`(): Used by the particle removal functions to remove pointers to a particle and fix the particle indices.

- `ewald`: Generates the **k**-vectors and some functions of them which are needed in Ewald summations. Has two subclasses, ewald3d and ewald2d.

  - `find_vectors`(): Finds the required **k**-vectors by using a spherical or circular cutoff in reciprocal space. Tabulates some quantities depending on only **k**-vectors.
  - `calculate_h`(): Finds the required values of the integration variable h and tabulates some values depending on only **k** and h. Only for ewald2d objects.
  - `find_structure_factor`(): Calculates the structure factor $S(\mathbf{k})$ or $S(\mathbf{k}, h)$ of the entire simulation cell with the current configuration.

Utility classes (methods not shown)

- `array`: Easy allocation and deallocation of multidimensional arrays. Not actually a class.
- `complex`: Complex number arithmetic.
- `parameters`: Reading parameters from a file.
- `qots`: Displaying physics quotes while the program is running. Not actually a class.
- `random_generator`: Generation of random numbers.
- `vtkwriter`: Writing positions, field values, etc. to `.vtk` files.

6.6 Parameters

A parameters file defines a setup for a numerical experiment, and is taken into the program as the first of two input arguments (the second one is the output folder). Parameters come in two types, global and phase-specific ones. The global parameters contain both initial values and parameters used in calculation during the whole simulation. Some of the parameters will be irrelevant to certain types of simulations, but they are all always included in the file. Here is a brief list of what the parameters file contains:

Global: System ID specifying material, the number of conventional crystal unit cells in each direction, a PBC flag, a flag turning on/off fixed particles on each x-side, initial density, initial temperature, the thermostat parameter τ, time step Δt, number of time steps per visialized frame, temperature rescaling and neighbour list update, cutoff for the LJ potential, constants C and ϵ from eq. 6.7, gap position and width

(not used much in the project), size of spherical flaw, crystal sphere flag, the number of simulation phases.

Space specific: Number of time steps, heat bath temperature, thermostat selection flag, dissipation coeficcient, volume expand rate in each direction, fixed particle displacement rate for each of the x-sides, constant force vector, collective velocity adjustment vector, output level flag.

In the Ewald summation, the required parameters are calculated internally in the same way as in the Protomol MD package [7]:

$$\alpha = C\left(\frac{N}{V^2}\right)^{1/6} \tag{6.5}$$

$$r_c = \frac{\sqrt{-\ln\epsilon}}{\alpha} \tag{6.6}$$

$$k_c = 2\alpha\sqrt{-\ln\epsilon} \tag{6.7}$$

In this way we change from three to two input parameters and gain a $O(N^{3/2})$ scaling. C is a constant adjusting the ratio between the time usage of the real-space and reciprocal-space sums, and is mostly set to 2.0. ϵ is the relative accuracy of the calculations. I put this to 10^{-5} most of the time, as this is more than enough for studying the phenomena the program is made for.

To prove the scaling hypothesis, I note that $V \propto N$, such that $\alpha \propto N^{-1/6}$. A particle has a number of neighbours proportional to $r_c^3 \propto \alpha^{-3} \propto N^{1/2}$. This must be computed for all N particles, giving a $N^{3/2}$ scaling for the real-space sum. The number of $\mathbf{k}$-vectors inlcuded in the reciprocal-space sum is proportional to $k_c^3 V \propto N^{1/2}$, because the density of $\mathbf{k}$-vectors is $\frac{V}{2\pi}$. This sum also happens for all particles, giving it a $N^{3/2}$ scaling.

Chapter 7

Fracture results

The energy required to break a perfect crystal is very high, and more comparable to the energies of the atomic bonds than the energy required to break a crystal with defects. ...

I will state some observations from the blablabla.

The BDP thermostat generally does a good job cooling down edges of the simulation cell when a fracture is occuring. As a lot of energy is dissipated in the process, it takes time to conduct all the energy out of the system. Most of the kinetic energy arising is in the form of elastic waves which slowly loses energy to heating up the crystal. Such collective motions make temperature measurements inaccurate, which they in any case will be in all non-equilibrium situations.

As a test of the Ewald summation technique implementation, I tried to reproduce the lattice energy of NaCl. The energy required to sublimate a NaCl crystal into a gas of Na^+ ions and Cl^- ions is -787 kJ/mol $= -4.079$ eV per particle [15]. This is, by a good approximation, the total potential energy of the lattice. With 4096 ions, the potential energy of the perfect lattice measures -4.076 eV per particle. This number is very similar for smaller system sizes. At a temperature of 300 K, the potential energy fluctuates around -4.036 eV per particle with a standard deviation of 0.001 eV per particle. More specifically, the energies from the Lennard-Jones interaction, the short ranged Coloumb interaction and the long range Coloumb interaction including self-energy correction is:

$$U^{LJ} = +0.432 \text{ eV} \qquad U^{SC} = -3.024 \text{ eV} \qquad U^{LC} = -1.445 \text{ eV} \tag{7.1}$$

All the numbers are stated per atom.

The experimentally measured cohesive energy of the silicon lattice is -4.638 eV [18]. With a temperature of 300 K, my simulation using the Stillinger-Weber potential gives the slightly higher number -4.2967. The energy is distributed among the force terms in this way:

$$U^{(1)} = -0.6938 \text{ eV} \qquad U^{(2)} = -3.6238 \text{ eV} \qquad U^{(3)} = 0.0209 \text{ eV} \tag{7.2}$$

Again, these numbers denote potential energy per atom. In the diamond structure,

all atoms are able to have an angle of 70.53° between their neighbours simultaneously, giving a low energy contribution from the three-body interaction term.

7.1 Tensile stress

...

Figure 7.1 shows a simulation of a silicon crystal under tensile stress and at a starting temperature of 260 K. Two fracture spots are visible. The first fracture occures at about 43% strain, which is extremely high. After the fractures, elastic waves of great energy hit the sides of the simulation cell and are reflected back. This is physically unrealistic, as the fixed atoms should be in just the same situation as the simulated ones. The reason I have not gone into energy transfer by such collective phenomena is that the dynamics I am interested in here happens before and during a fracture.

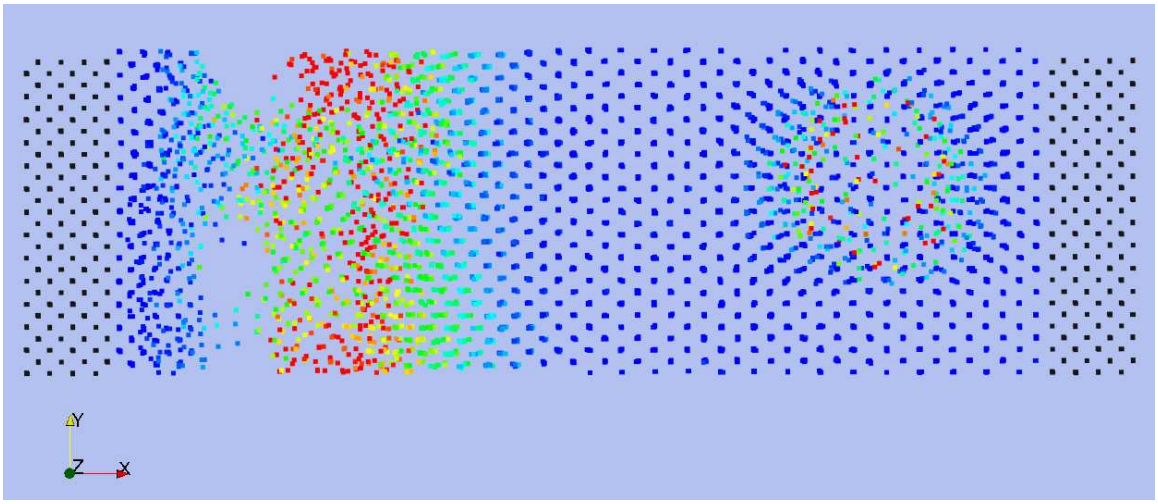
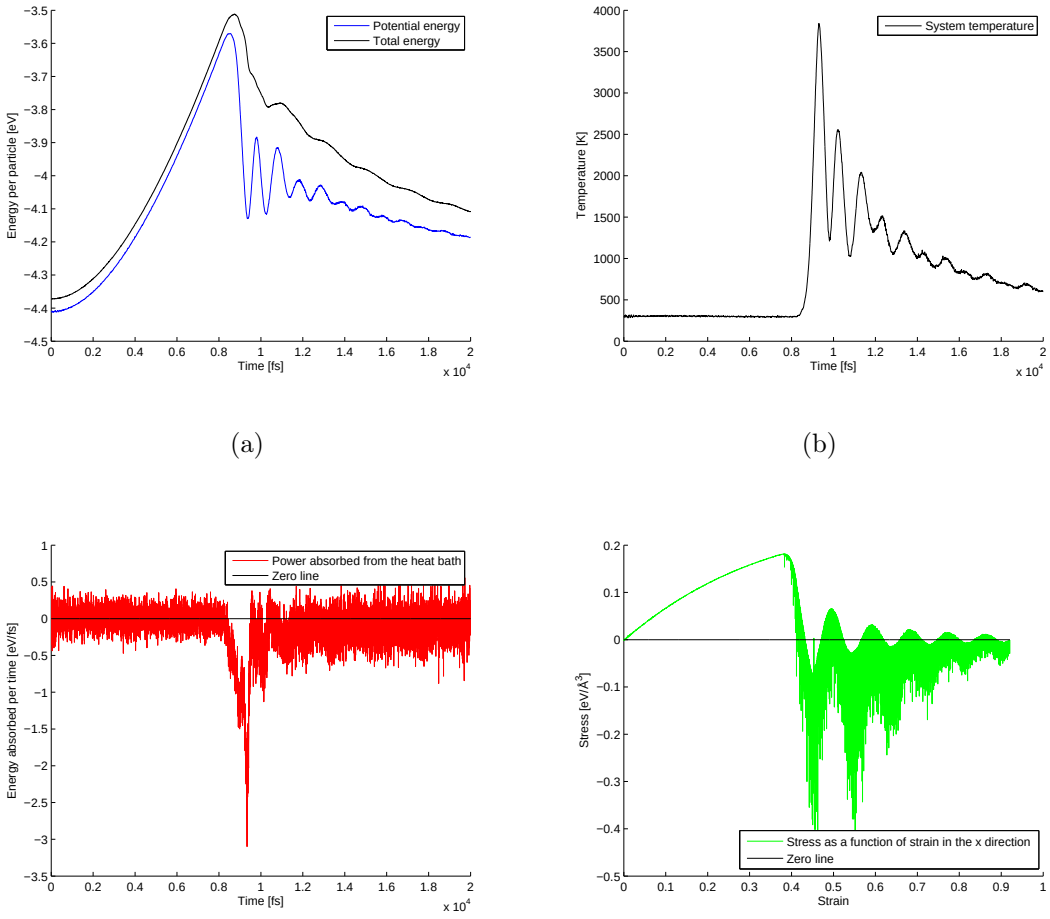


Figure 7.1: Tensile stress fracture of a crystal with 5488 Si atoms at 260 K. The colouring shows the values of the recent motion quantity. The fixed particles are shown in black.

If you neglect the thermal movement, we are dealing with a perfect crystal of atoms being streched. In my simulations of this system, fractures always start out by the breaking of single bonds. Normally, an atom in a diamond structure will have four neighbours. This means four two-body terms for each particle. At the moment this number decreases to three for an atom, nearby atoms start to show rapid motion. What happens is that the thermal motion kicks two atoms further apart than the SW interaction range. A sphere of vacuum with its center where the bond breakage occured appears inside the material. This is how a fracture starts in a silicon crystal in all the cases I have observed. An example of this is visible at the right side of the crystal in figure 7.1.



(a) (b)

(c) (d)

Figure 7.2: How macroscopic quantities behave when a silicon crystal fractures. The temperature is estimated with $T = 2U_k/3k_\mathrm{B}$.

Figure 7.2 shows the set of output macroscopic quantities from a typical silicon tensile stress simulation with 4096 particles. The oscillations in the potential energy are caused by the elastic waves of compression and expansion occuring in the two crystal parts after fracture. The same goes for the kinetic energy and thereby temperature. However, the thermostat acting on the edges of the simulation box provides a damping of these oscillations due to outgoing heat. As the collective motion is subtracted from the measured temperature when the it is adjusted in this way, the energy has to transform to actual heat before leaving the system. This is seen to happen in great amounts rigth after the fracture.

The fracture starts right when the applied stress is at its greatest, which I estimate as the breaking stress. Long before this point, it is clear that Hooke's law $\sigma = E\epsilon$ fails. For small strain, as the law is intended, we see a linear stress-strain relationship, and Young's modulus E is measurable. From figure 7.2(d), I read this value:

$$E = (0.670 \pm 0.004)\ \mathrm{eV/\AA}^3 \tag{7.3}$$

EKSPERIMENTELL VERDI ER $185GPa = 1.156eV/angstrom^3$!!
The vertical lines beneath the stress graph are an artifact of the simulation. Neighbour lists have a certain cutoff range in space and is updated after a specified number of time steps. When a critical event like a fracture occurs, atoms are moving very fast and the forming of new atomic bonds are not discovered in time because of the limited neighbour list update frequency. This creates a small shift in potential energy. The reason this is not fixed is that CPU-time usage is very sensitive to these two parameters, and they are at sufficient values to simulate what happens until the fracture occurs.

The importance of defects was well illustrated by a bug in my program for the initial silicon simulations. One two-body force term for interaction with a fixed particle was not taken into account. This resulted in a significantly lower breaking stress, and that the fracture always started where this missing bond was located.

When defects of particle deficiency grow larger, the consequenses do not change as greatly as from zero to one missing particle. Figure 7.3 shows measurement of the constant in Griffith's relation (4.1). The initial number of particles (before introduction of the defect) is 4096. The relation is well reproduced, with a mean of 0.2657 and a standard deviation 0.0087.

When sodium chloride is stretched, it does not build up potential energy in the same violent fashion as the silicon crystal. As there are both strong attractive and repulsive forces which cancel each other, it does not take much energy to break a sodium chloride crystal. Regardless, I observed a fracture at about 63% strain in a simulation. Before a fracture occurs, the atoms float about lazily in a substance of low density. There is, however, evidence of crystal structure under the whole simulation.

The NaCl crystal breaks as a very brittle material, often cut straight into two pieces separated by a y-z-plane. The two pieces stay charge neutral, which is energetically favourable. The velocities of the particles make a less considerable jump then in the silicon crystal case when the pieces drift apart. In many cases, the fracture occurs

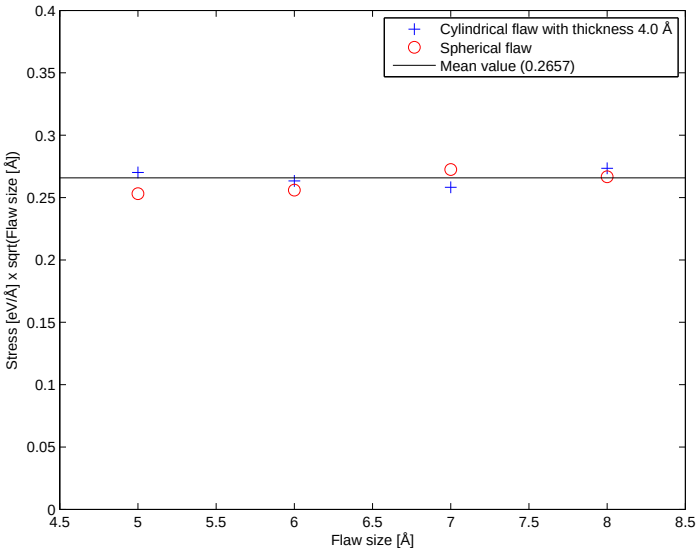


Figure 7.3: $\sigma_f\sqrt{a}$ as a function of a for silicon. The "flaw sizes" a are the radii of the spherical and cylindrical voids.

close to the edge of the simulation box, probably because of the constant distance between the fixed atoms, making this a special place. In figure 7.4, a spherical void of radius 7 Å is inserted in the middle of the cell. This causes the fracture to occur at this point, although this is not always the case with NaCl crystals containing defects. Threads of ions are often seen connecting the two pieces right after a fracture, but these disintegrate easier than corresponding silicon atom strings.

The small build-up of potential energy seems to happen because of the long-range part of the Coloumb interaction. In a simulation without the Ewald summation over **k**-vectors, the crystal breaks in the same fashion, but at a lower strain. The potential energy has a deviation of roughly 2.5% during the course of the simulation. The defect size does not seem to influence the breaking stress of the NaCl crystal. The dynamics stays the same for different spherical void sizes, but a greater defect has a bigger probability of influencing where the fracture will occur.

KOMMENTER NEGATIV STRESS!

7.2 Shear stress

When shear stress is applied to a silicon crystal, the atomic layers will be skewed and the potential energy will build up. At some critical point, this energy will be converted into a great amount of kinetic energy, causing a phase transition at some points in the simulation cell.

In figure 7.6, these areas emerge at the side of the cell, making the energy distri-

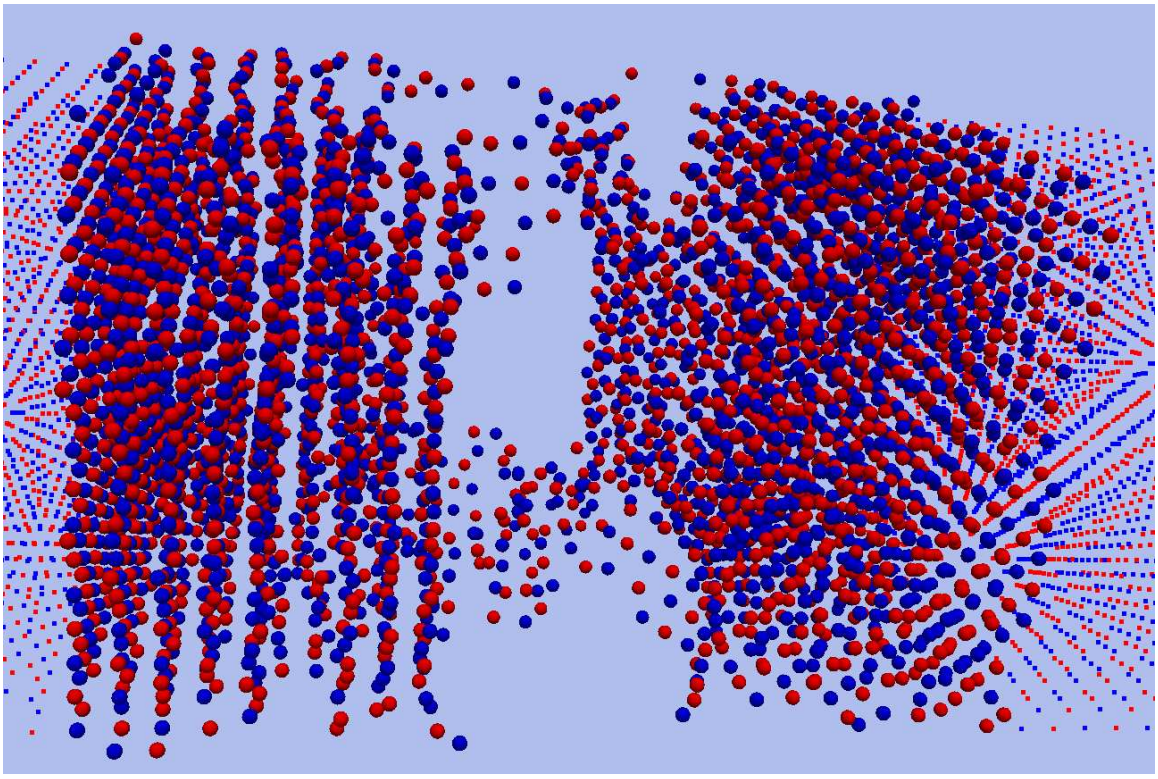
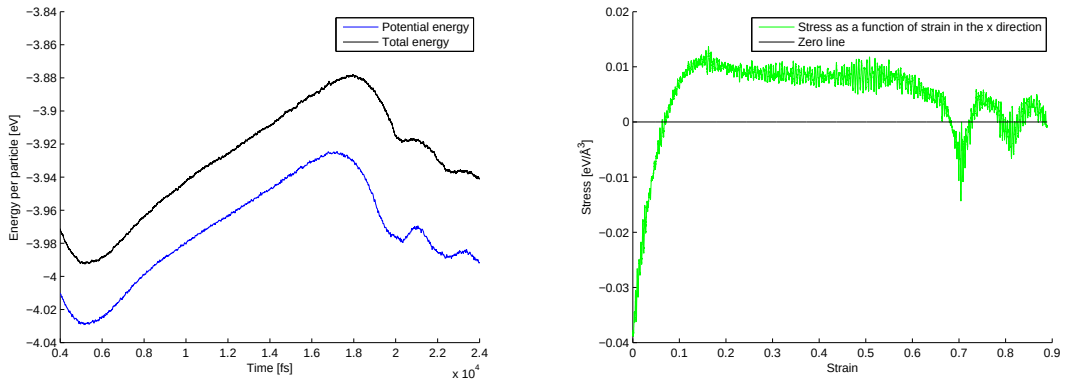


Figure 7.4: Tensile stress fracture of a NaCl crystal with 4096 atoms at 300 K.

bution symmetric along the x axis. Sometimes, the phase transition happens in the middle of the cell. The areas that do not become liquid keep their crystal structure. Silicon compresses as it melts, so liquid atoms have more neighbours. Therefore, their potential energy is higher, which is visible in figure 7.6(a).

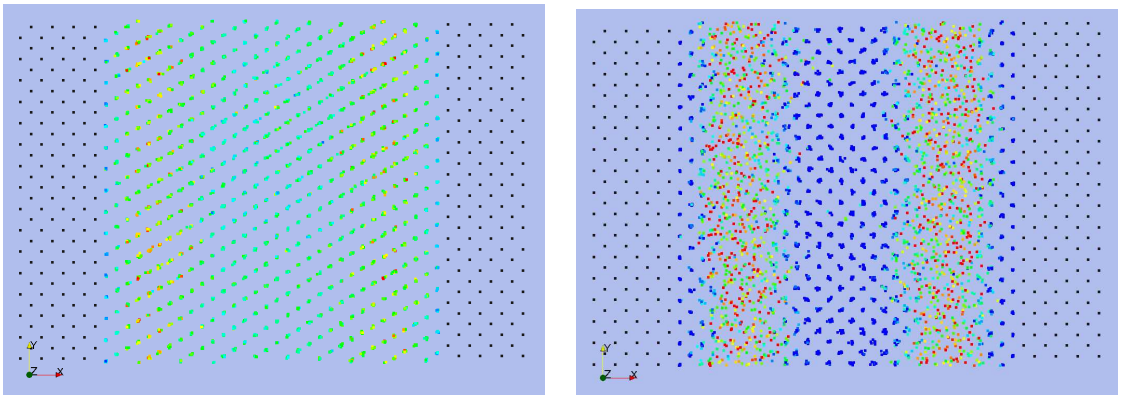
Sodium chloride shows a quite different behaviour when deformed with shear stress. The atomic layers get more and more lopsided before the layers slip vertically at an y-z-plane in the middle of the cell. The system then looks almost like a crystal in equilibrium, but some heat is generated. The cycle of tilting and atomic layer slipping repeats itself as long as the fixed particles are in motion. There are big temperature fluctuations, but a mean temperature of about 650 K is established as all the heat generated from the deformation is in average conducted out. This applies to a fixed particle speed of 0.002 Å/fs.

There is little difference in the dynamics for a simulation without long-range Coloumb interactions. The potential and total energy graphs look rugged in both cases. There is a slight spring-like effect when the atomic planes slip with full interactions turned on. This is a process which happens simultaneously everywhere in the middle y-z-planes. The long-range forces are expected to contribute significantly in the case of a large-scale displacement from equilibrium. In this case, a counteracting force arises.



(a) (b)

Figure 7.5: How the energy and stress varies when a NaCl crystal fractures.



(a) Right before fracture (b) Right after fracture

Figure 7.6: Shear stress on a crystal with 4096 Si atoms at 260 K. The colouring shows the potential energy per particle. The fixed particles are shown in black.

Chapter 8

Results for adhesion and such

8.1 Sphere-surface contact

I have performed computational contact experiments for the systems described in section 5.3.3. I will first present the results from the sphere-surface systems, which I have used the most time on. For the calculations to be executed in a reasonable time, I have limited the system size to 4232 simulated and 1600 fixed atoms for silicon, and 2106 simulated and 2048 fixed atoms for sodium chloride. The pseudo-gravitational force has been varied logarithmically in the interval from 0.01 eV/Å to 30.0 eV/Å.

Results for the measured contact area using equation 4.10 are presented in figure **??**. The spheres do not seem to compress much because of the force affecting their top halves. In a large force interval, They are pushed towards the surface and the bottom atoms rearrange slightly, but there are no further deformations. When the force reaches a certain threshold, the sphere collapses under its own "weight". The two materials act similar in and this respect, but the microscopic processes are different.

In the case of silicon, the lowermost atoms display a liquid-like behaviour, as expected from the property of higher liquid density than solid density. The sodium chloride sphere compacts itself vertically and extends horizontally, not mainly by compression. The slip mechanism also occuring in the shear strain experiment is responsible for reducing the number of atomic layers in the vertical direction. The configurations preceding and following the slip motion are shown in figure 8.3.

When measuring the stress between a ball and a surface of similar materials, the system size has to be big to get a clear image of what the stress distribution looks like. This is especially the case for silicon, where the particles close to the surface arrange quite arbitrarily. Therefore, I have not only averaged the stress measurements over time, but over 8 different initial configurations (random seeds). This makes the structure of the stress graph (figure 8.4) more apparent. The graph is similar to the result for the bent crystal in [13], where a harmonic potential is used, but it also contains a great deal of noise, like it should, seeing that the bottom sphere particles arrange randomly. The outer moat of attraction is caused by adhesive forces between atoms not pushed close enough together to repel. Around 20000 dynamic particles

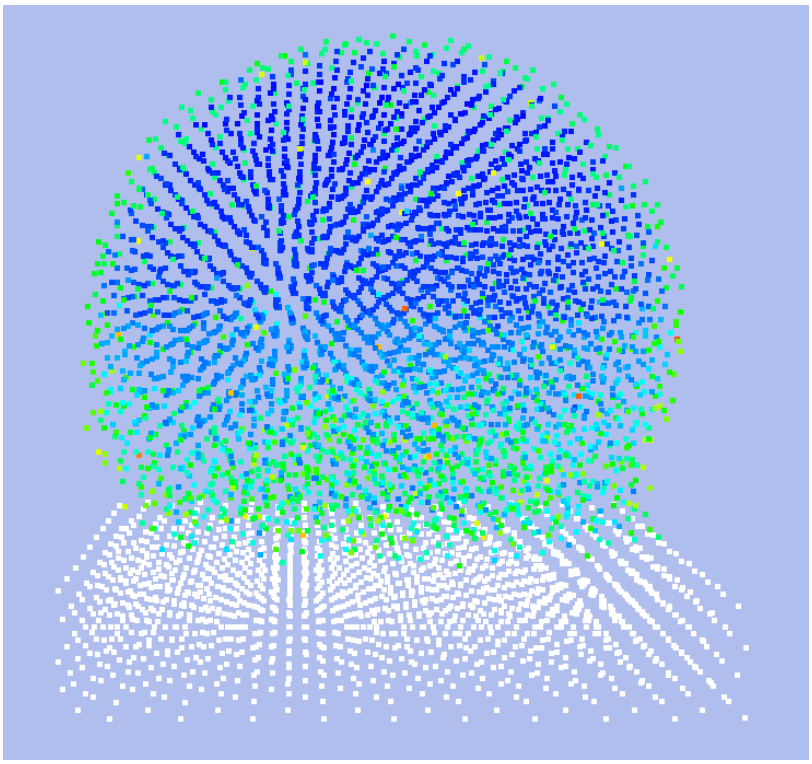
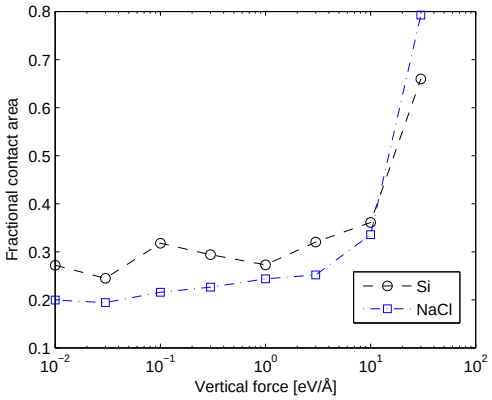


Figure 8.1: A ball of silicon pushed against a silicon surface with a force of 30.0 eV/Å. The colours show potential energy. The discontinuity in this colouring is caused by the fact that only the upper half of the particles are subject to the ficticious gravitational force. Perspective projection.

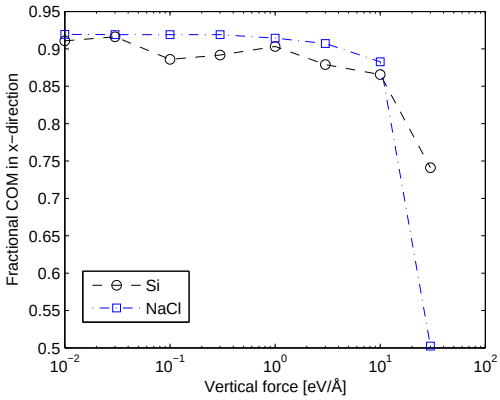
were used in the simulations.

The stress graph for the NaCl system is very different. The signs of the stresses have the opposite character: The middle particles attract each other slightly while the outer particles repel each other (figure 8.5). The graph looks the same for different initial conditions, with the exception that the ball may be displaced one atomic distance in some direction on the surface. The sphere particles arrange very regularly. Small, regular zigzag patterns in the stress graph are caused by oppositely directed forces between the ions. Around 7000 dynamical particles were used in the simulation. The reason for the odd surface stress graph is investigated in the next section.

In both the Si and NaCl cases, the sum of forces between the sphere and surface particles lie within 1% of the applied pseudo-gravitational force. This validates the results by assuring that Newtons second law is fulfilled.

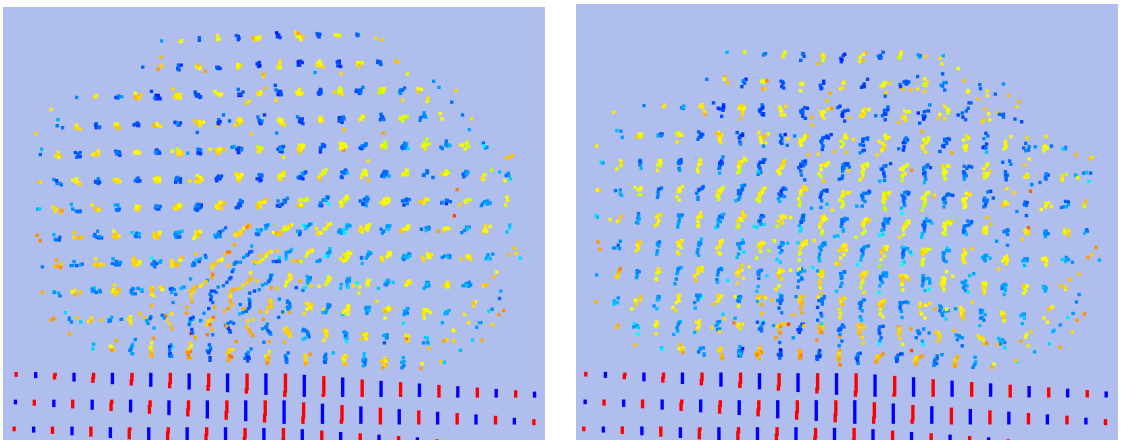


(a) Contact area in units of one sphere cross-section. Average over two runs per point.



(b) Center of mass in the x direction in units of one sphere radius.

Figure 8.2: How quantities vary when an increasing vertical force is applied to a sphere on a surface.



(a) (b)

Figure 8.3: The deformed NaCl sphere before and after a slip motion. A height difference is evident. The colouring shows the atoms' potential energy, excluding long range (recprocal sum) electrostatic energy.

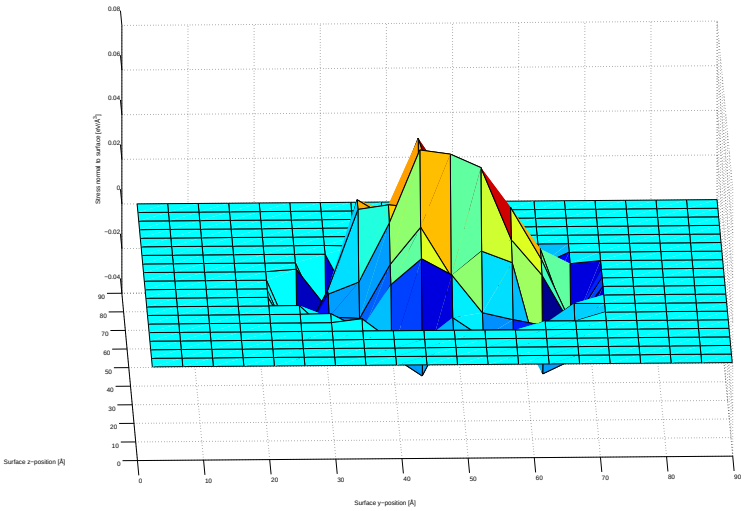


Figure 8.4: Interface stress between an Si ball and an Si surface at a downward force of 3.0 eV/Å.

8.2 Sphere segment-surface contact

When the forces behind the NaCl stress graph are examined more carefully, the Lennard-Jones potential appears to be the culprit behind the great repulsion at the sides of the contact area. The ionic interactions actually lessen the effect of this and cause the attraction in the middle of the contact area. The short-range part of the Coloumb forces do all the work, while the long ranged part do not contribute significantly to this process. The stress graph looks very different when the sphere segment system is used. The most important difference is that fixed particles keep the simulated particles strongly in place at the positive x side of the simulation box. Therefore, global oscillations are avoided. Natural oscillations are nearly certainly the reason behind the stress spikes. According to the particle trajectory animations, the sphere tip very slightly over to the left and right side of the contact point, which cause the contact particles at one side to come closer to the surface and experience a vastly bigger LJ force. This creates a peak which is significantly taller than the mean stress across the contact area. When averaged over time, this effect is the prominent feature of the stress graph all around the edges.

Without the oscillations, when the sphere segment system is slightly compressed between two surfaces, a more uniform stress is measured. Figure 8.6 shows such a stress graph, with a total force between the dynamical particles and the lower fixed particles of 131 eV/Å. This number can seem very high, but the force is applied in a more gradual manner than before, and the sphere is still not collapsed. With a little more applied compressive strain, the sphere segment will deform in much the same way as before, creating a bigger contact area and lowering the total force. NaCl has a much stiffer crystal structure than Si, which in this case makes it more durable in

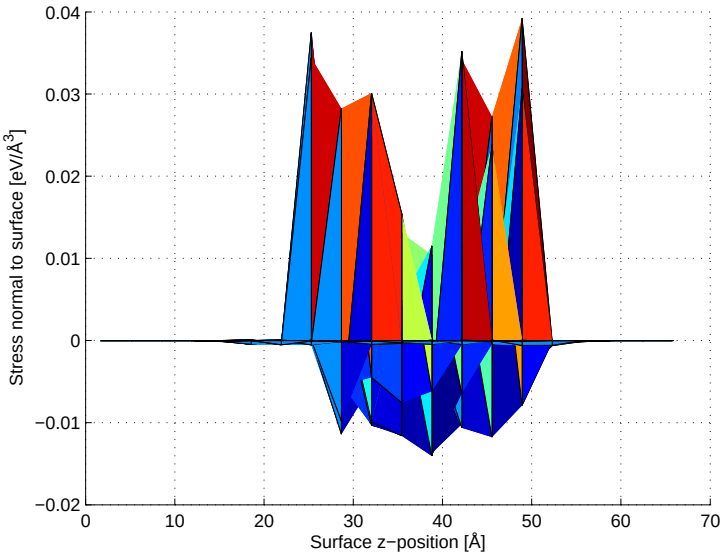


Figure 8.5: Interface stress between a NaCl ball and a NaCl surface at a downward force of 3.0 eV/Å.

the absence of a more sudden impact. The intact stepped crystal structure makes a uniform stress across the contact area seem likely.

In another attempt to explain the strange sphere-surface stress graph, I observed that the lowermost atomic layer creates a deformation in the rest of the sphere for large applied forces. However, this deformation is too small to create such a big effect, and it occurs also in the sphere segment system. I refer to the deformation, which is clearly visible in figure 8.7, as the bow phenomenon. The figure is from the same simulation run as figure 8.6. The effect of the bow phenomenon can be seen as a small ring outside the stress from the lowermost atomic layer. It is a peculiar phenomenon which most probably happens in real systems, but the effect is not significant.

The stress graph of the silicon system still has stability issues, but the results from sphere-surface interactions are reproduced.

8.3 Friction experiments

Measuring friction parameters from atomic simulations is a very ambitious task. I have only attempted to simulate friction using the sphere-surface interface system, and the sphere, which unevitably begins to roll, may do a poor job at describing a rough macroscopic surface asperity in a time-dependent situation. While the spheres rolled/slipped across the infinite layers of fixed surface atoms, more sphere particles became attached to the surface, tearing off more and more of the sphere. Data for the coupling between contact area and frictional force is therefore only reliable in the

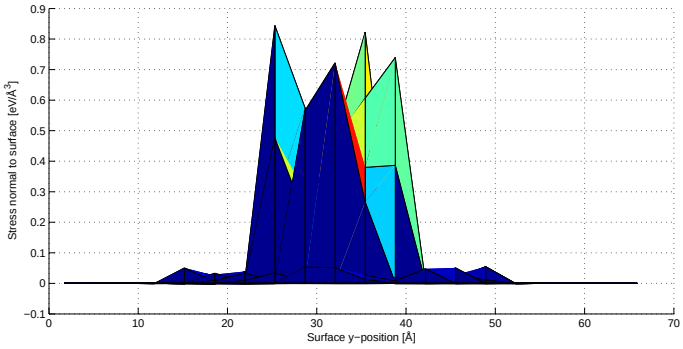


Figure 8.6: Interface stress between a NaCl sphere segment and a NaCl surface. The applied compressive strain is 4.1%.

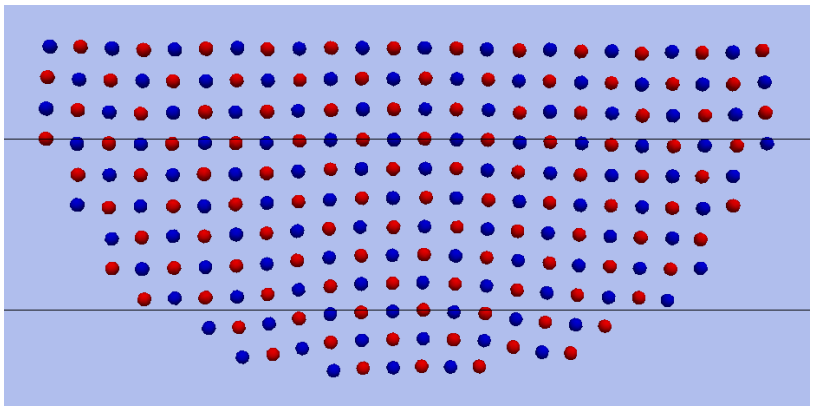


Figure 8.7: Arrangement of one x-y-layer of NaCl sphere segment ions. The applied compressive strain is 4.1%. The straight lines are just eye guides.

start of the simulation.

The data contains fluctuations, but large-scale shapes of the A_c-F_{friction}-distribution are visible. In the NaCl case, the frictional force seems to be distributed equally around zero at all times, which I have made no effort to explain. Interestingly, the silicon data seem to indicate a linear relationship between A_c and F_{friction}, though the fluctuations are of the same size as the increased force from small to large contact area. For different values of the driving velocity and the pseudo-gravitational force, I see no general trend of change in the frictional force. This may or may not be accurate, but is either way only valid for a single asperity. The larger-scale mechanisms of friction, which cause F_{friction} to vary linearly with the normal force, are explained in [12].

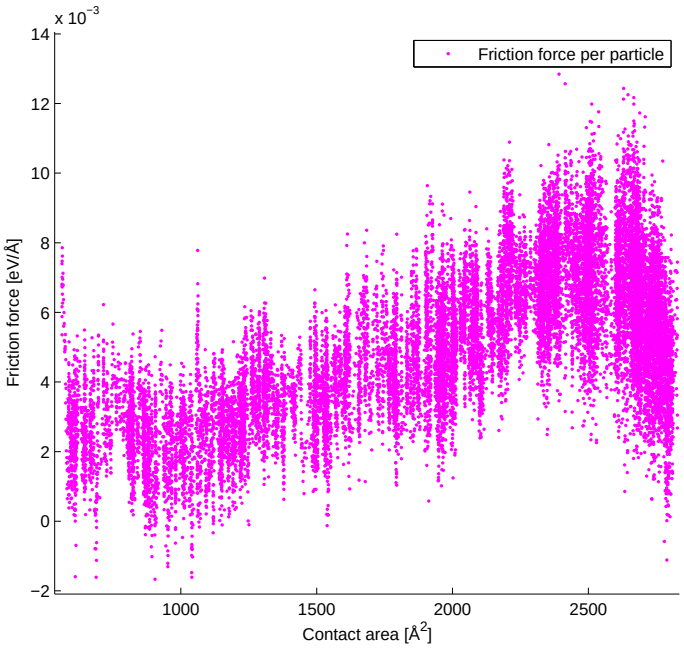


Figure 8.8: F_{friction} vs A_c for a silicon friction experiment with about 4000 particles. One data point per time step.

Chapter 9

Concluding remarks

Bibliography

[1] J. M. Thijssen, *Computational physics*. New York, NY, USA: Cambridge University Press, 1999.

[2] G. Bussi, D. Donadio, and M. Parrinello, "Canonical sampling through velocity rescaling," *The Journal of Chemical Physics*, vol. 126, no. 1, p. 014101, 2007.

[3] A. Rahman, "Correlations in the motion of atoms in liquid argon," *Phys. Rev.*, vol. 136, pp. A405–A411, Oct 1964.

[4] D. E. Smith and L. X. Dang, "Computer simulations of nacl association in polarizable water," *The Journal of Chemical Physics*, vol. 100, no. 5, pp. 3757–3766, 1994.

[5] J.-P. Hansen and I. R. Mcdonald, *Theory of Simple Liquids*. Academic Press, 3 ed., April 2006.

[6] P. Vashishta, R. K. Kalia, J. P. Rino, and I. Ebbsjö, "Interaction potential for *sio*2: A molecular-dynamics study of structural correlations," *Phys. Rev. B*, vol. 41, pp. 12197–12209, Jun 1990.

[7] T. Matthey, "Plain Ewald and PME." May 2005.

[8] M. Kawata and M. Mikami, "Rapid calculation of two-dimensional ewald summation," *Chemical Physics Letters*, vol. 340, no. 1-2, pp. 157 – 164, 2001.

[9] F. H. Stillinger and T. A. Weber, "Computer simulation of local order in condensed phases of silicon," *Phys. Rev. B*, vol. 31, pp. 5262–5271, Apr 1985.

[10] D. C. Rapaport, *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2 ed., April 2004.

[11] E. Gerde and M. Marder, "Friction and fracture," *Nature*, vol. 413, pp. 285–288, Sep 2001.

[12] J. Ringlein and M. O. Robbins, "Understanding and illustrating the atomic origins of friction," *American Journal of Physics*, vol. 72, no. 7, pp. 884–891, 2004.

[13] B. Luan and M. O. Robbins, "The breakdown of continuum models for mechanical contacts," *Nature*, vol. 435, 2005.

[14] F. Gilabert, A. Krivtsov, and A. Castellanos, "A molecular dynamics model for single adhesive contact," *Meccanica*, vol. 41, pp. 341–349, Jun 2006.

[15] "Wikipedia, the free encyclopedia." http://www.wikipedia.org.

[16] V. K. W. Cheng and E. R. Smith, "The electrostatic potential energy at a plane surface of a point ionic crystal. ii. numerical results for an ion near the (100) kcl surface," *Journal of Physics A: Mathematical and General*, vol. 20, no. 10, pp. 2733–2742, 1987.

[17] W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, February 2002.

[18] X.-P. Li, D. M. Ceperley, and R. M. Martin, "Cohesive energy of silicon by the green's-function monte carlo method," *Phys. Rev. B*, vol. 44, pp. 10929–10932, Nov 1991.