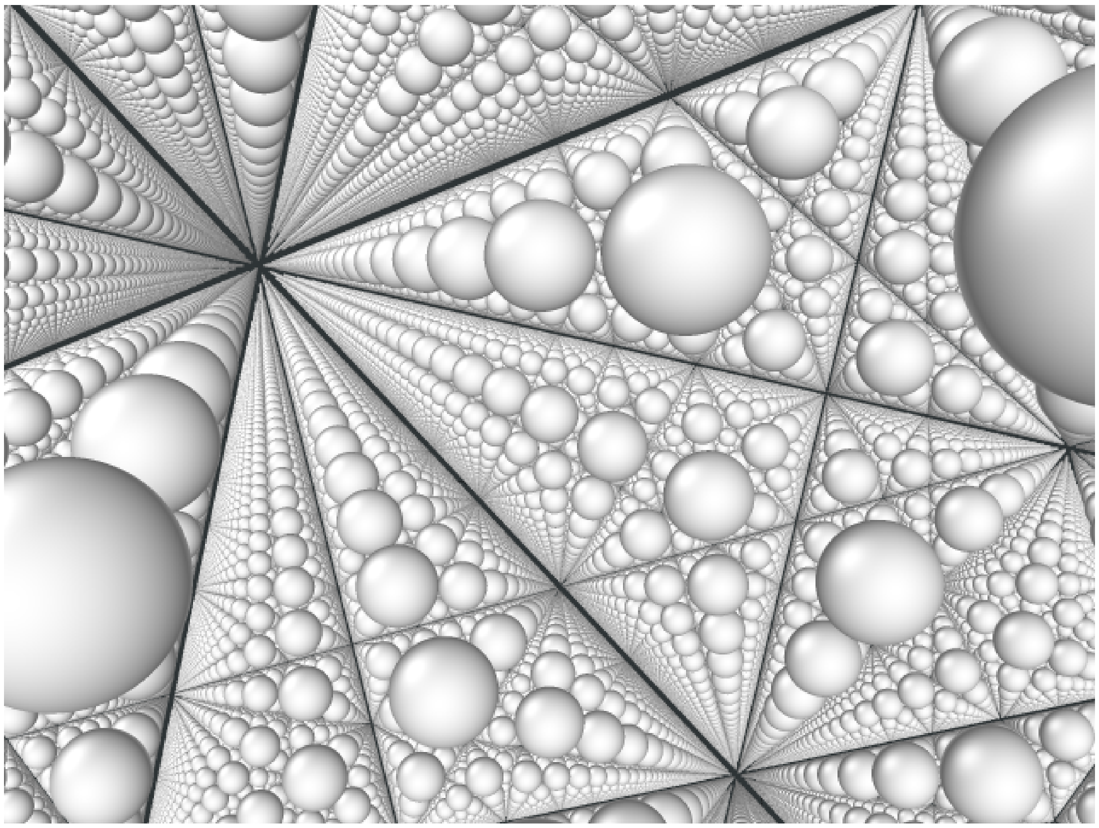
Molecular Dynamics
Project 5
FYS3150

Ragnar Bruvoll Halvard Sutterud

Desember 2017



Abstract

In this report we develop a Molecular Dynamics simulator to calculate the macroscopic quantity of melting temperature from a microscopic system, using periodic boundary conditions. A method is developed to place unit cells of an argon crystal in a lattice, iterate the system forward with the velocity verlet method and Lennard-Jones potential and gather statistics. The model is implemented in the programming language `c++`, based on the code from Anders Hafreager [4], with visualization made with `ovito`. The velocity verlet method is shown to conserve energy, as expected from theoretical predictions [5]. We also manage to extract a melting temperature for argon of 300K, which is far from the expected value of 84K [1]. This is however likely to fall on the experimentors and not the method, even though we measure our positions in units of length.

Contents

1 Introduction

Molecular Dynamics is useful in bridging the gap between the microscopic world and the macroscopic quantities measured in a lab. It is a common term for computer simulation methods with the intent of studying the properties of molecules and atoms at an individual level as well as their collective movement when they are a part of a system.

Molecular dynamics can be applied to several fields of science. In biology and chemistry, it is used to examine the properties and make three dimentional representations of proteinstructures, DNA and other other macromolecules. Material science and nanotechnology also reaps obvious benefits from the conclusions we extract from MD as we can learn how different types of materials behave under pressure, how they conduct electricity and other material properties essential in physical engineering.

The methods simulate the evolution of the system by numerically calculating the interactions caused by the molecular force fields and interatomic potentials between the particles. Under simple assumptions about how molecules interact, one then produce results that are in a sense exact; for a specific system, the accuracy of the predictions are only restricted by the assumptions made and the accessible computer power.

By assuming that all states are equally probable over time, we can use the evolution of one MD system to determine the macroscopic thermodynamic properties of the system, by sampling the relevant microscopic quantities. This is similar to the Monte Carlo based method of Project 4, but this time we let the dynamics of the molecules determine the evolution of the system state/configuration.

In the theory part, we will scratch the surface of the basic theoretical footwork needed to establish the interactions and calculate expectation values. In the next section we go through the various methods and details necessary in the setup of the program, before visualizing the consequences of the methods in section 4. In the same section we also simulate a number of different temperatures to obtain a melting temperature, calculated from the mean square displacement. We then discuss the results gained from the simulations.

2 Theory

2.1 Maxwell-Boltzmann distrubution

The Maxwell-Boltzmann distrubution is a probability distrubution descibing the velocity of the particles in an ideal gas moving freely and independently inside a container. The probability of getting a velocity in a small interval $\mathrm{dv_i}$ is [6]

$$P(v_i)\mathrm{d}\mathrm{v}_i = \left(\frac{m}{2\pi k_B T}\right)^{1/2} \exp\left(-\frac{m v_i^2}{2k_B T}\right) d\mathrm{v}_i, \qquad (1)$$

where v_i is the i'th component of the velocity vector of an atom, m is the mass of the particle, T is the temperature of the gas and k_B is Boltzmans constant. As seen the velocity of the particle depends on the temperature and mass. This probability distrubution is a simplified version and does not take factors such as van der Waals and quantum exchange interactions.

2.2 Energies

Many of the physical properties can be calculated by extracting the energy E, namely the kinetic and potential. The kinetic energy is the classical one

$$E_k = \sum_{i=1}^{N_{atoms}} \frac{1}{2} m_i v_i^2 \tag{2}$$

and the potential energy is given by

$$E_v = \sum_{i>j}^{N_{atoms}} U(r_{ij}) \tag{3}$$

where $U(r_{ij})$ is said the **Lennard-jones potential** between particles i and j given as

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right]. \tag{4}$$

ϵ is the "depth" of the potential well and σ is the distance where makes the potential between particles is zero. $r_{ij} = |\mathbf{r_i} - \mathbf{r_j}|$ is the distance between the two particles in question. The first term is the *Pauli repulsion* which represents the repulsion caused by overlapping electron orbitals, while the second term represents the attractive *van der Waals force.*

2.2.1 Forces

All the particles interact with each other through forces, derived from the **Lennard-jones potential**. The force, F is the gradient of the potential,

$$\mathbf{F}(r_{ij}) = -\nabla U(r_{ij}), \tag{5}$$

The force between the particles will either attract or repulse depending on the distance between them, with a sign change at $r = \sigma$. Theoretically, the force can be zero if all the particles are perfectly aligned and the Pauli repulsion and van der Waals force are equal. The force in the x dimension is

$$F_x(r_{ij}) = -\frac{\partial U}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial x_{ij}} \tag{6}$$

$$= -4\epsilon \left[-12 \left(\frac{\sigma}{r_{ij}}\right)^{12} \frac{1}{r_{ij}} + 6 \left(\frac{\sigma}{r_{ij}}\right)^{6} \frac{1}{r_{ij}} \right] \cdot \frac{x_{ij}}{r_{ij}} \tag{7}$$

with x_{ij} as the x-component of r_{ij}. As the y and z components are on the same form, we can then write the force vector as

$$\mathbf{F}(r_{ij}) = 24\epsilon \frac{\mathbf{r_{ij}}}{r_{ij}^2} \left[2 \left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right]. \tag{8}$$

2.2.2 Equipartition theorem and ideal gas

The equipartition theorem [7, p. 238] states that for a system with energies in the form of quadratic degrees of freedom, the average energy per degree is just

$$\langle E \rangle = \frac{1}{2} k_B T \tag{9}$$

For N particles each with three degrees of freedom, the energy of an ideal monatomic gas is

$$\langle E \rangle = \frac{3}{2} N k_B T \tag{10}$$

2.3 Diffusion constant

The diffusion constant is given by the Einstein relation [6] as

$$\langle r^2(t) \rangle = 6Dt, \tag{11}$$

where D is the diffusion constant, t is time and $\langle r^2(t) \rangle$ is the mean square displacement calculated as

$$\langle r^2(t) \rangle = |\mathbf{r}_i(t) - \mathbf{r}_i(0)|^2. \tag{12}$$

Solving for D gives

$$D = \frac{\langle r^2(t) \rangle}{6t} \tag{13}$$

2.4 Units

As our system is at a microscopic size, we scale our units to a set of more manageable ones to be able to represent the numbers properly on a computer.

$$\begin{aligned}
\text{1 unit of mass } m_0 &\equiv 1.661 \times 10^{-27}\text{kg (a.m.u.)}, \\
\text{1 unit of length } L_0 &\equiv 1.0 \times 10^{-10}\text{m (Angstrom)}, \\
\text{1 unit of energy } E_0 &\equiv 1.651 \times 19^{-21}\text{J}, \\
\text{1 unit of temperature } T_0 &\equiv 119.735\text{K},
\end{aligned}$$

where a.m.u. means atomic mass unit which is approximately the mass of a nucleon. Angstrom is a unit of length commonly used for expressing lengths at an atomic and molecular level. Other units can then be derived from these, giving one unit of time as

$$1 \text{ unit of time} = \text{length} \times \sqrt{\frac{\text{mass}}{\text{energy}}} = 1.002\,24 \times 10^{-13}\,\text{s}. \tag{14}$$

In these units, the diffusion constant is then

$$[D] = \mathrm{L}_0^2/\mathrm{t}_0 = 0.9978 \times 10^{-7}\,\mathrm{m}^2/\mathrm{s} \tag{15}$$

3 Methods

This section will explain how we applied the aforementioned theory on our project.

3.1 Setting up the system

3.1.1 Random particles

We start by simulating a gas of randomly distributed particles, before we advance to the more complex lattice. The particles will have velocity given by the aforementioned distrubution, and will therefore expand the boundaries of the system with a rate depending on their speed.

3.2 Particles not in a box

To get a more controllable system, we implement periodic boundary conditions. Instead carrying on into infinity, we set a fixed system size that the particles are contained in. When crossing the boundary, the particles will continue their trajectory from the opposite side of the systems boundaries. This will prevent the particles from expanding the system by travelling with constant velocity in each their direction, and lets us keep the particle density constant. We also make sure to always let the atoms interact with the closest image of another particles. We also count the number of "jumps" a particle does, for use in calculating displacement later. The algorithm for testing if particles are properly bounded is represented by pseudo code as

```
for i :=1 to 3
    if x[i] > bounds[i]
        x[i] -= bounds[i]
        count[i] -= 1
    else if x[i] < 0
        x[i] += bounds[i]
        count[i] += 1.
```

3.3 Removing momentum

As all the particles in the gas start off with a random velocity, the box will have a net movement in a random direction. In order to acchieve a static box of particles, we remove the net momentum. This is done through the *removeMomentum* function in the *System* class, and guarantees that the particles won't get any unneccecary drift.

3.4 Face-centered cubic lattice

In order to extract any useful information about how the particles interact, we need to structurize them. We do this by placing them in a grid formed crystal, one that we know that solid argon is known to exhibit. We will see that the potential energy of these atoms will keep them stable, confirming that this is a stable structure. The stable crystal structure for the system is the Face Centered Cubic [2]. We use an algorithm to determine the particle positions, by dividing the crystal into unit cells consisting of four atoms. These unit cells are then regularly spaced in three dimensions to form the lattice.

3.5 Making periodic boundary conditions and zero total momentum

The `removeMomentum` function looks as follows

```
for all atoms:
    totalMomentum += atom.velocity * atom.mass
    totalMass += atom.mass
for all atoms:
    atom.velocity -= totalMomentum/totalMass;
```

3.6 Face-centered Cubic lattice

To study the crystalline structure of argon, we implement an algorithm to place the atoms in a face-centered cubic (FCC) lattice. Consider a unit cell with four atoms whose local position is given by

$$\begin{aligned}
\mathbf{r}_1 &= 0\hat{\imath} + 0\hat{\jmath} + 0\hat{k} \\
\mathbf{r}_2 &= \frac{b}{2}\hat{\imath} + \frac{b}{2}\hat{\jmath} + 0\hat{k} \\
\mathbf{r}_3 &= 0\hat{\imath} + \frac{b}{2}\hat{\jmath} + \frac{b}{2}\hat{k} \\
\mathbf{r}_4 &= \frac{b}{2}\hat{\imath} + 0\hat{\jmath} + \frac{b}{2}\hat{k}
\end{aligned}$$

within the cell, where b is the lattice constant that defines the unit cell size. The origin of each unit cell $\{i, j, k\}$ (in three dimensions) is

$$R_{i,j,k} = bi\hat{\imath} + bj\hat{\jmath} + bk\hat{k}. \tag{16}$$

4 Results

In this section we will cover all the results and figures achieved through our methods.

4.1 Random Particles

Our initial system is simulated through **Ovito** without boundary conditions and with random positions and velocities. Initially the particles are placed as seen in fig. 1, but over time this is evolved into what one can see in fig. 2. While nice to look at, this doesn't give much information about the melting of argon.

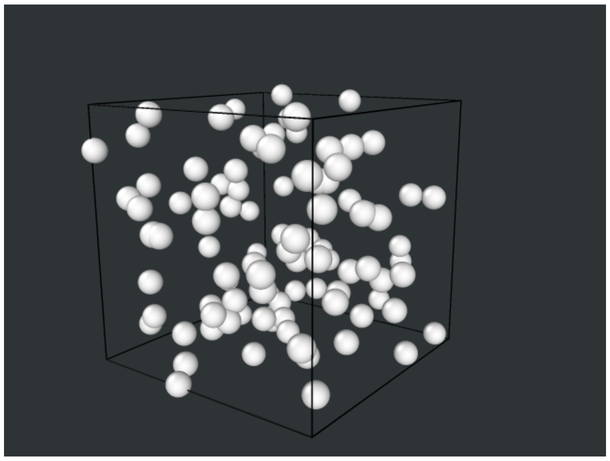


Figure 1: The initial positions of 100 particles simulated without boundary conditions with Maxwell Boltzmann distribution of velocities.

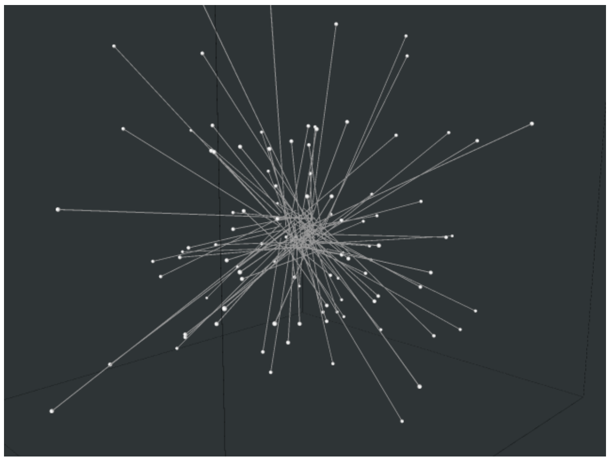


Figure 2: The final positions with trajectories of 100 particles simulated for 10×10^{-11} s without boundary conditions.

4.2 Periodic Boundaries

A simulation of 100 particles at $T = 300K$ for 10×10^{-11} s with periodic boundaries was then run. The results was visualized with Ovito and are shown in fig. 3. When a particle leaves one boundary, it is moved to the other side of the box, as one can see from the straight lines.

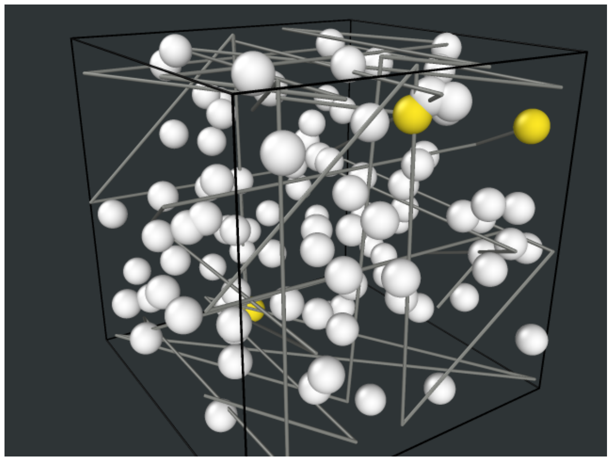


Figure 3: The final positions of 100 atoms simulated with boundary conditions over 10×10^{-11} s, with trajectories for the three particles marked yellow.

4.3 FCC Lattice

The FCC lattice created by the function `createFCCLattice` can be seen in fig. 4.

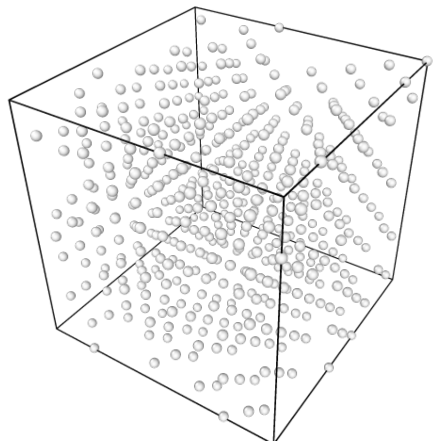


Figure 4: The initial structure of the face centered cubic lattice. The space is packed tightly, with each atom in the same distance to the same number of neighbors.

4.4 Remove Momentum

Before implementing the remove momentum function, the system drifts slowly as the positions are all randomly initiated. This can be seen in fig. 5, where all particles have a net drift in the same direction.

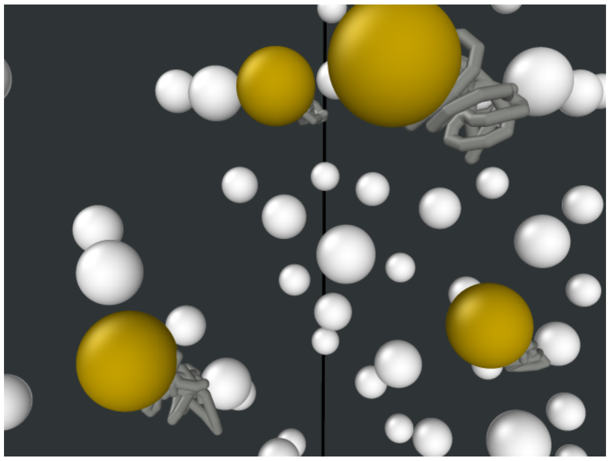


Figure 5: Drifting over 10×10^{-11} s of the lattice when the `removeMomentum` function is not present.

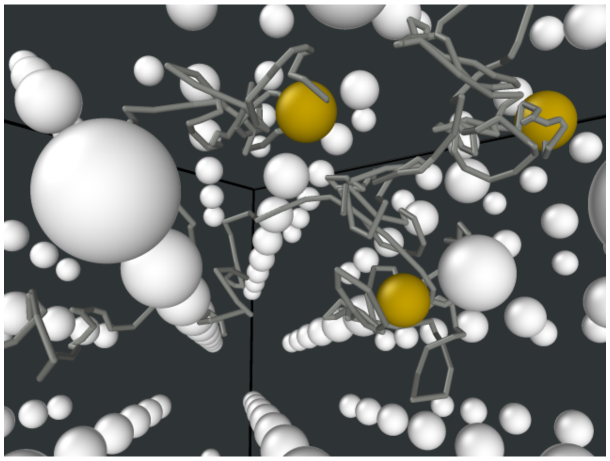


Figure 7: The trajectories of three atoms in a 5x5x5 unit cell lattice, run with T = 800K as initial temperature for 10×10^{-11} s.

4.5 Diffusion and melting

Refering to eq. (13), for a series of runs between $50K$ and $1350K$ we extract the diffusion coefficient at the end of each run and plot it against the temperature at that time in fig. 6. Figure 7 show the trajectories of the atoms for T = 800K. For low temperatures, the trajectories are not visible outside our atom plotting-radius.

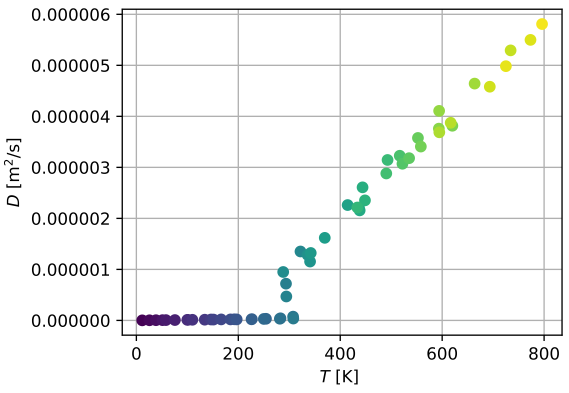


Figure 6: The diffusion constant as a function of final temperature, for several runs at inital temperatures between 50 and 1400 Kelvin for 10×10^{-11} s.

4.6 Final temperature

Figure 8 shows the approximated temperature of the first part of a simulation, for a selection of initial temperatures. The temperature, and with it the kinetic energy, stabilizes around 50% of the initial temperature. The ratio of inital temperature with deviations, averaged over all runs, can be seen in fig. 9.

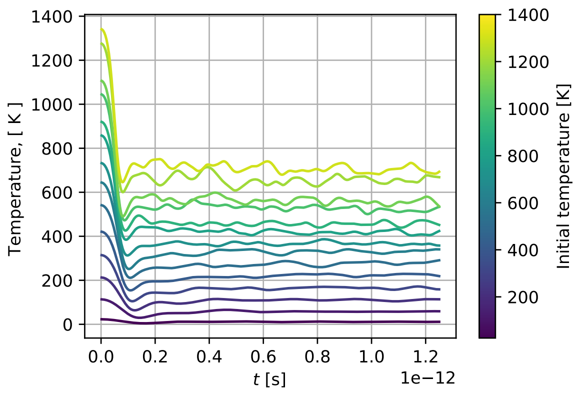


Figure 8: The average temperature the atoms in a simulation, for several values of the initial temperature between T = 50 and T = 1400.

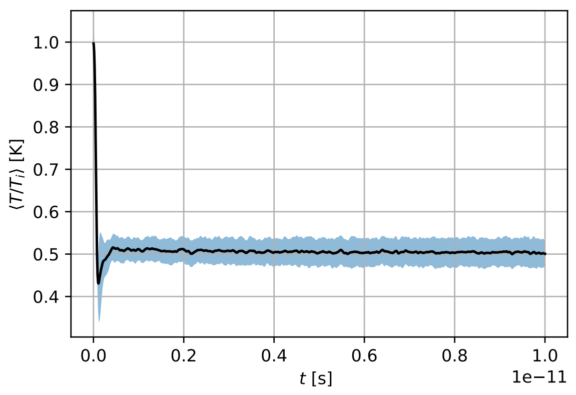


Figure 9: Mean ratio T/T_i as a function of time, averaged over all runs with temperatures ranging from 50K to 1400K, with the error margin plotted as a band.

4.7 Energy conservation

In fig. 10, the conservation of energy of the velocity verlet method and the total energy as a function of temperature is plotted.

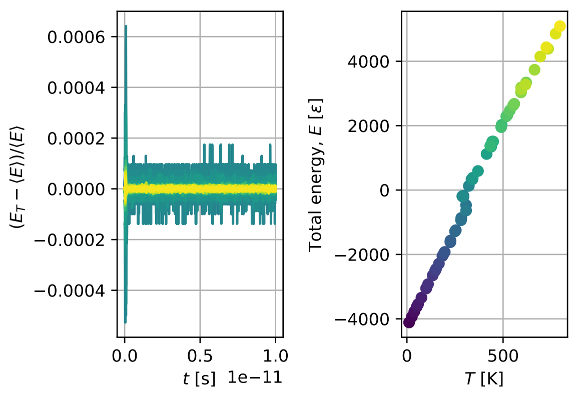


Figure 10: **Left:** Relative total energy averaged over the atoms in each simulation(colored by temperature). **Right:** Total energy as a function of final temperature after 10×10^{-11} s.

5 Discussion

5.1 General comments on the project

By iterating through the different parts of the code given by Hafreager [4], we gained new insights into how one can structure a program. The main lesson was probably that one should not be afraid to produce many small classes, each dedicated to a task. Separating parts prone to different kinds of error from each other, makes debugging (potentially) a lot easier, and helps build modular and more general code.

5.2 Code

5.2.1 Periodic boundary conditions

While simulating a gas, periodic boundary conditions lets us approximate an infinite gas or solid. Because the Lennard Jones potential is limited in range, using the nearest image convention [3] is a good approximation. As long as the system is of a decent size, one can then use the average values of the ensemble at equilibrium to calculate expectation values, similarly to what we did in project 4.

5.2.2 Remove Total Momentum

Removing total momentum is important. A net drift of all the atoms, as seen in fig. 5, would cause a too high estimate of the temperature. A net drift won't change the properties that are normally temperature dependant, but it will directly cause a higher temperature.

5.2.3 FCC Lattice

The FCC lattice is shown to be a stable configuration of the Lennard Jones potential. This is a good thing.

5.3 Diffusion

For low temperatures the diffusion constant is almost zero, while near a critical temperature it gets a much higher value and starts increasing linearly (fig. 6). The discontinuity is around $300K$, which neatly corresponds to the temperature where the total internal energy is non-zero (fig. 10). When kinetic energy of an atom is equal to the potential it is in, the atom is in an unbound state, free to escape its neighbors. The crystal structure is then broken, the lattice melted, and the system goes into a liquid phase.

When melted, the system behaves as we expected. Figure 7 shows that the atoms are free to move away from their initial positions. This is despite the high density of the system; when we increase the temperature the atoms are not as stuck in the energy minima of their neighbors as they would be at lower temperatures, and are free to move between each other and even travel quite far.

5.4 Energy

Combining eq. (2) and eq. (9), one would expect a temperature proportional to the kinetic energy. However we saw in fig. 8 and fig. 9 that the temperature dropped to exactly half its value after initialization. One possible explanation may be that the `FCCLatice` function with a spacing $b = 5.26\text{Å}$ is used*, giving a distance between atoms of bit more than $\sigma^\dagger$. All the atoms are then placed in the energy minima of the potential to its nearest neighbors. Consider that near a minima the leading order taylor term of a potential is the quadratic term (except a constant term giving no force) [7, p. 241]. As total energy is conserved, each atom then receives three quadratic degrees of freedom which will receive equal amounts of energy, by the equipartition theorem. The kinetic energy of the equilibrated system is then half of the initial kinetic energy, which would explain the values seen in fig. 9. Even though dense van der Waal gas is usually not subject to the equipartition theorem, for a dense liquid we propose that one can still use it to explain the temperature drops seen in fig. 8 and fig. 9.

5.4.1 Conservation

As the forces are conservative, we expect the total energy to be conserved. This is indeed the case as seen in fig. 10. This is a property of the velocity verlet algorithm; for N-body systems under the influence of conservative potentials, it can be shown ([5], using Noethers theorem) that the method conserves energy and angular momentum. This makes it a good choice for performing calculations on a molecular dynamics system.

6 Conclusion

In this project we have implemented the necessary functions to run a MD simulation and applied the tools on an argon crystal. Molecular Dynamics is an easy tool to use, at least for simple systems such as this one, but for larger simulations one would want to limit the range of the potential as well as parallelize the heavy part of the code, namely the calculation of forces.

We tested the output with Ovito, as seen in fig. 1 to fig. 5, and found a lot of functionality. Further, in fig. 6 and fig. 10 we showed that our argon crystal melted around $300K$, which is a lot higher than the expected value of 84K. The reason for this discrepency between experiment and expectations is unknown; heavy duty debugging has been performed, and this was the closest we got to the proper value (at least it is closer than 20000K...).

In addition to this, we theoretized that the equipartition theorem is applicable to certain systems not expected to exhibit only quadratic degrees of freedom, as it would explain fig. 9. We also showed in fig. 10 that the energy of our implementation was conserved up to at least order 1×10^{-4}, with a deviation over all runs and all time steps as $\pm 1.56 \times 10^{-5}$ relative to the total energy of that run.

*This value is given without further explanation in [6].

$^\dagger b$ corresponds to the spacing between unit cells, which gives a minimum distance between two atoms of $b/\sqrt{2}$

Appendix

See `https://github.com/halvarsu/FYS3150/projects/Project5` for code. table 1 shows the different programs developed, with brief explanations. A flow chart showing the code forked from Anders Hafreager [4] is shown in fig. 11, with an interactive version found at `http://folk.uio.no/halvarsu/files/FYS3150/Project5/graph/main_8cpp.html`

Table 1: Description of programs developed in this project. This project is less python-dependent compared to earlier projects.

c++ (in folder `cpp`)
`main.cpp` Accepts inputs, creates a system of particles, and simulates for a given time, outputting particle positions and statistics every 100 time step
`atom.cpp` Contains information about a single atom
`system.cpp` Contains atoms and a potential, and methods for removing center of mass drift, applying boundary conditions and creating lattices.
`lennardjones.cpp` Calculates forces and energies for the atoms in a system.
`velocityverlet.cpp` Steps a system forward in time, using Newtons law.
`statisticssampler.cpp` Gets and updates the statistics of a system, and saves it to file.
`io.cpp` Outputs positions.
`unitconverter.cpp` Contains the values of the units used in simulation relative to SI-units, and converts between the two.

python (in folder `analyse`)
`analyse.py` Extracts statistics from runs (either all or a hard coded range) and generates all analysis plots.

bash (in folder `shell`)
`readrun_temps.sh` Simple script for interacting with files containing temperatures to be simulated. Reads line by line, simulates if they haven't been done yet and marks as done afterwards. Output is put in out/T_{temperature}.

7 References

References

[1] Argon periodic table. `http://www.rsc.org/periodic-table/element/18/argon`. Accessed: 2017-12-12.

[2] Wikipedia contributors. Cubic crystal system — wikipedia, the free encyclopedia, 2017. [Online; accessed 14-December-2017].

[3] Wikipedia contributors. Periodic boundary conditions — wikipedia, the free encyclopedia, 2017. [Online; accessed 15-December-2017].

[4] A. Hafreager. Molecular dynamics fys3150. `https://github.com/andeplane/molecular-dynamics-fys3150`, 2014.

[5] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric numerical integration illustrated by the störmer/verlet method. *Acta Numerica*, 12:399–450, 2003.

[6] M.H.Jensen. Fys3150 lecture notes. *CP. UiO*, fall 2015.

[7] D.V. Schroeder. *An Introduction to Thermal Physics*. Addison Wesley, 1999.

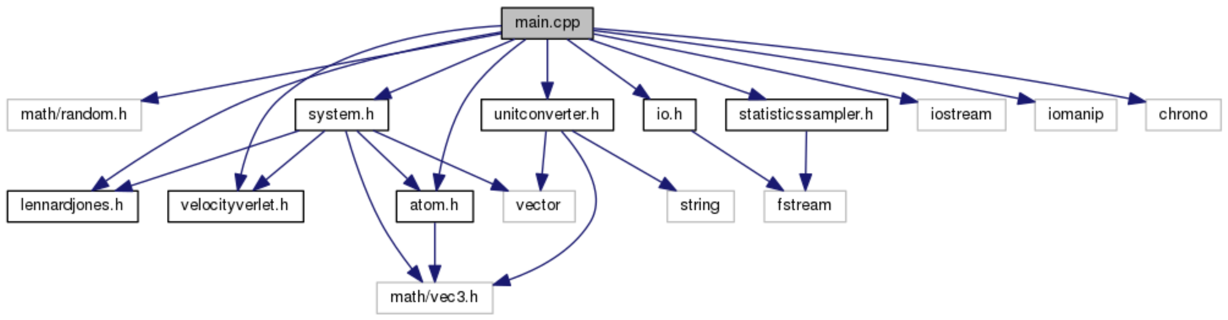


Figure 11: Flow chart showing an overview of the code structure, with dependency arrows between programs.