Molecular dynamics simulation of Argon

Python

AP3082 Computational Physics

by

Abi Kanagaratnam & Jim Koning

Team of Applied Physics (MSc)

at the Delft University of Technology.

Group Members: Kanagaratnam, Abi (4481968)
Koning, Jim (4481674)

Assignment: March 21st, 2021

TU Delft

Abstract

The goal of this computational assignment is to find the diffusion coefficient using a working molecular dynamics simulation of Argon using the Verlet algorithm. The diffusion coefficient was determined using the mean-square displacement of Argon atoms in a FCC lattice. This diffusion coefficient was determined in the configuration in 3 different states of matter; solid, gas and liquid. These different states are obtained by varying simulation parameters determined by the literature[2]. For the gas, liquid and solid the following diffusion coefficients were found: 2.5e-6, 1.0e-6 and 0.5e-6 in dimensionless units, respectively. However, these constants are not correct. A non-zero total momentum was present in the simulation, resulting in a artificial inflation of the kinetic energy making the results invalid. The observed states in this paper are likely all solid.

Contents

Introduction

In this part of the report certain elements will be explained in order to understand the rest of the report. We will for instance explain the choice of particles first. We will also discuss about the literature that is already available. And finally, we will shortly discuss about the future of this research field.

Choice of particles

Our aim is to choose particles that are easily and simply defined for classical mechanics simulations. For this assignment, the choice has been to let our particles be atoms. Molecules would have been an option as well. However, molecules would be subject to complexities such as asymmetric shape and molecular-level bonds (such as hydrogen bonds). To avoid such complexities, we want to avoid using molecules and atoms that could form into molecules. Atoms that are unlikely to form into molecules are noble gases. Therefore, our choice of which particle to choose is now limited to an atom which is a noble gas as well. Historically, one of the most studied noble gases is Argon. This would help us to compare whether our simulation meets the expectations from prior studies. For verification and simplicity reasons, our simulation will be using Argon atoms.

Literature

Due to Argon being one of the most studied particles for microscopic-level particle simulations. We would be able to compare our values to literature values.

Future research

We believe that computational physics would be a good approach to study atoms and interactions on a microscopic scale. We believe that there is still a lot to discover in this area. Due to the technology advancing, we believe that more complex simulations should be possible as well. It is important to study atoms and interactions on a microscopic scale, so we will have a better understanding of physics on the smallest scales.

Chapter 1

Methodology

Theory

In this section the theory for running our molecular dynamics simulation of Argon atoms will be explained. Also, the theory for the mean-square displacement and the diffusion will be given.

1.1 Interaction of Argon atoms

Argon atoms are neutral in charge, therefore there is no Coulomb interaction present. However, Argon atoms do have a dipole-moment caused by the electrons and the nucleus. This gives them an attractive force. Though, Argon atoms also have a repulsive force which prevents Argon atoms to get too close to each other and consequently prevents them from forming molecules. The attractive potential has a proportionality to the distance of two atoms given as:

$$U(r) \propto \frac{1}{r^6}, \tag{1.1}$$

where r is the distance between the two atoms.

1.1.1 Lennard-Jones potential

Combining the attractive force and the repulsive force of Argon atoms will result in the following potential.

$$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{1.2}$$

where $\epsilon/k_B = 119.8$K and $\sigma = 3.405$ Angstrom, with $k_B = 1.38064852 \cdot 10^{-23}$. More on these values can be found in the chapter **nummersss**
This potential is also known as the Lennard-Jones potential

1.1.2 Infinite space system

In our simulation we would want to be able to simulate an infinite amount of space where our Argon atoms could be in. However, infinite is an unknown quantity to computer. Therefore, we simplify our simulations by implementing periodicity. With the periodic boundary condition in place, the Argon atoms in our main simulation box will also be affected by the Argon atoms in the surrounding simulation boxes (due to periodicity). In the end, it will appear as if there is copies of our main simulation box around the main simulation box.

1.1.3 Initial locations of atoms

The initial locations of the atoms can be quite important. If the atoms start with a location very close to an other atom, the velocities could increase to extremely high values in the first few timesteps. Due

to the step iterative method of calculating locations and velocities, a particle with a high velocity can 'teleport' to a location far away from its previous location. The simulation is only valid, if the potential change the particles 'feel' is gradual to conserve energy. More on the step iterative method in the chapter **implementation algorithms.**

For starting locations, a FCC lattice is chosen. This is a physical representation of Argon, and approximately, depending on the lattice constant, resembles the locations where the atoms are in a equilibrium state.**source needed**

1.2 Mechanics

We will be using Newton's laws to simulate the classical motion of Argon particles. This will be explained next.

1.2.1 Newton's equation

The motion of the particles is given as follows:

$$m\frac{d^2\mathbf{x}_i}{dt^2} = \mathbf{F}(\mathbf{x}_i) = -\nabla U(\mathbf{x}_i), \tag{1.3}$$

where i is the index of a particle and $\mathbf{F}$ is the sum of the forces acting on particle i.
We see that the potential is dependant on the distance r. In a three dimensional situation we can see that $r = \sqrt{x^2 + y^2 + z^2}$ so the following relation holds:

$$\nabla U(r) = \frac{dU}{dr}\frac{\mathbf{x}}{r}, \tag{1.4}$$

1.2.2 Implementation algorithms

To implement the previously mentioned Newton's equation into our simulation we need to use an algorithm. This will be explained next.

1.2.2.1 Euler's method

The Euler's algorithm is given using the following equation:

$$t_{n+1} = t_n + h, \tag{1.5}$$

which results in the following two equations:

$$\mathbf{x}_i(t_{n+1}) = \mathbf{x}_i(t_n) + \mathbf{v}_i(t_n)h \tag{1.6}$$

and

$$\mathbf{v}_i(t_{n+1}) = \mathbf{v}_i(t_n) + \frac{1}{m}\mathbf{F}(\mathbf{x}_i(t_n))h, \tag{1.7}$$

where h is the time between time-steps and n is the index of the time-step. $\mathbf{x}_i$ and $\mathbf{v}_i$ are the position vector and the velocity vector, respectively.

1.2.2.2 Velocity-Verlet algorithm

To improve the accuracy of the Euler's method, the Verlet algorithm will be used [4]. The differences in accuracy are minor in comparison. However, the Verlet algorithm uses so-called symplectic integrators that works well with time-evolving an Hamiltonian system. The classical mechanics we will use in our simulation is an example of such Hamiltonian system. When slightly modifying the Verlet algorithm, we will obtain the velocity-Verlet algorithm. This is given as follows:

$$\mathbf{x}_i(t_{n+1}) = \mathbf{x}_i(t_n) + h\mathbf{v}_i(t_n) + \frac{h^2}{2}\mathbf{F}(\mathbf{x}_i(t_n)) \tag{1.8}$$

and

$$\mathbf{v}_i(t_{n+1}) = \mathbf{v}_i(t_n) + \frac{h}{2}(\mathbf{F}(\mathbf{x}_i(t_{n+1})) + \mathbf{F}(\mathbf{x}_i(t_n))) \tag{1.9}$$

Note that these equations are given in dimensionless units. Dimensionless units will be explained next.

1.3 Dimensionless units

Since our simulations run with an atomic scale system, the numbers that are used for our calculates will also have very different orders of magnitudes. Computers do not seem to work well with numbers that have higher orders of magnitudes; it could lead to floating-point errors or round-off errors. Also working on a program with numbers that have higher orders of magnitudes is more prone to human errors as well. Therefore, we will be using ”reduced” units, also called: dimensionless units.
We will be expressing time in $(\frac{m\sigma^2}{\epsilon})^{1/2}$, length in σ and energy in ϵ, these will have the following values:

$$t_{\text{DIMLESS}} = (\frac{m\sigma^2}{\epsilon})^{1/2} = 2.15 \cdot 10^{-12}\text{s}, \tag{1.10}$$

$$E_{\text{DIMLESS}} = \epsilon = 1.65 \cdot 10^{-21}\frac{\text{J}}{\text{K}}, \tag{1.11}$$

and

$$l_{\text{DIMLESS}} = \sigma = 3.405 \cdot 10^{-10}\text{m}. \tag{1.12}$$

From these the dimensionless unit for velocity and diffusion can be determined as following:

$$v_{\text{DIMLESS}} = \frac{l_{\text{DIMLESS}}}{t_{\text{DIMLESS}}} = \frac{\sigma}{(\frac{m\sigma^2}{\epsilon})^{1/2}} = \sqrt{\frac{\epsilon}{m}}. \tag{1.13}$$

$$D_{\text{DIMLESS}} = (\frac{\epsilon\sigma^2}{m})^{1/2} = 2.08 \cdot 10^{5}\text{m}^2/s, \tag{1.14}$$

1.4 Simulation constants

The objective of this paper was to reproduce some observables from the book *Computational Physics* published by *Jos Thijssen*. Thus, based on this, some of the following simulation constants were introduced.

1.5 Temperature rescaling

In order to allow our simulation to have phase transition for the Argon particles, it should be possible to allow the system to reach a desired target kinetic energy using a target temperature. Since the kinetic energy is a function of the temperature but also a function of velocity, it will mean that a change in temperature will result in a change in kinetic energy and consequently a change in velocity as well. The kinetic energy is given as:

$$E_{\text{kin}} = (N-1)\frac{3}{2}k_B T, \tag{1.15}$$

where N is the amount of particles. The kinetic energy is also given as:

$$E_{\text{kin}} = \sum_i \frac{1}{2}mv_i^2, \tag{1.16}$$

where v_i is the velocity of the individual particles.

Since the kinetic energy is dependent on the temperature, we can define a target temperature using a self-defined temperature. To make the velocities scale towards the target kinetic energy, we need to define a scaling factor as follows:

$$\lambda = \sqrt{\frac{(N-1)3k_B T}{\sum_i mv_i^2}}, \tag{1.17}$$

where λ is the rescaling factor to be applied to the velocities. Note that the previous 2 equations are not in dimensionless units.

1.6 Mean-squared displacement

For our simulation, we chose to have one observable which is the mean-squared displacement (MSD). One can calculate the mean-squared displacement using the following function:

$$\left\langle \Delta^2 \mathbf{x}(t) \right\rangle = \left\langle [\mathbf{x}(t) - \mathbf{x}(0)]^2 \right\rangle, \tag{1.18}$$

where $\mathbf{x}(0)$ is a suitable starting point.

We are not assuming we are having an ideal gas. This means the above equation cannot be simplified using the assumption that the particles move in straight lines.

When the mean-squared displacement is plotted against time, we could see three different behaviors:
- The mean-squared displacement grows quadratically with time, which means the particles are in a gas phase.
- The mean-squared displacement grows linearly with time, which means the particles are in a liquid phase.
- The mean-squared displacement is close to constant or shows oscillatory behaviour, which means the particles are in a solid phase.

It is important to track the mean-squared displacement for a long enough time because for very short times $\left\langle \Delta^2 \mathbf{x}(t) \right\rangle \propto t^2$ for gas, liquid and solid phase.

1.6.1 Diffusion

Using the mean-squared displacement, we can find the diffusion coefficient using the following Einstein relation:

$$D = \lim_{t \to \infty} \frac{1}{6t} \left\langle \Delta^2 \mathbf{x}(t) \right\rangle, \tag{1.19}$$

where D is the diffusion coefficient.

1.7 Errors

For the expectation of a physical observable, we are able to use the following equation:

$$\langle A \rangle = \frac{1}{N} \sum_{n>0}^{N} A_n, \tag{1.20}$$

where A is the physical observable and N is the total simulation time. In our case this observable would be the mean-squared displacement. We will be needing this to determine the errors in our observable data, this will be explained next.

1.7.1 Auto-correlation function

Since our simulation will produce time-evolved data, we cannot calculate the standard deviation through regular means. This is because time-evolved data is not uncorrelated/independent random data. Therefore, the auto-correlation function will be needed in order to determine the standard deviation. The normalized auto-correlation function (also known as the Pearson correlation coefficient) is given as:

$$\chi_A(t) = \frac{1}{\sigma_A^2} \sum_n (A_n - \langle A \rangle)(A_{n+t} - \langle A \rangle). \tag{1.21}$$

The auto-correlation function usually shows an exponential decay which is given by the following function:

$$\chi_A(t) = e^{-t/\tau}, \tag{1.22}$$

where τ is the correlation time (referring back to index n). When the value of the correlation time τ is known, we can calculate the error of our simulation observable as follows:

$$\sigma_A = \sqrt{\frac{2\tau}{N}(\langle A^2\rangle - \langle A\rangle^2)}. \tag{1.23}$$

Unfortunately, the auto-correlation function mentioned above is only valid for infinitely long time-evolved data. Since in our simulation, we are restricted to finitely long time-evolved data, the auto-correlation function becomes:

$$\sigma_A = \frac{(N-t)\sum_n A_n A_{n+t} - \sum_n A_n \cdot \sum_n A_{n+t}}{\sqrt{(N-t)\sum_n A_n^2 - (\sum_n A_n)^2}\sqrt{(N-t)\sum_n A_{n+t}^2 - (\sum_n A_{n+t})^2}}. \tag{1.24}$$

1.7.2 Data blocking

The idea of data blocking is to take averages from data blocks. A data block is a chunk of the complete data. When a block length is larger than the correlation time, the block averages become statistically independent random variables. This will solve the before mentioned problem of correlated data, since the averages of the data blocks are now uncorrelated. Therefore, we can now compute the error (standard deviation) through regular means as follows:

$$\sigma_A(b) = \sqrt{\frac{1}{N_b - 1}(\langle a^2\rangle - \langle a\rangle^2)}, \tag{1.25}$$

where b is data block size, N is the total data size and $N_b = N/b$. The averages a_i, which are needed for the previous equation, are calculated as follows:

$$a_i = \frac{1}{b}\sum_{(i-1)*b+1}^{i*b} A_i. \tag{1.26}$$

1.8 Reproducibility results

Reproducibility of results is self-evident in nearly all disciplines of science. While it is virtually impossible to reproduce the exact initial state of a simulation, the observables of a simulation should be reproducible. Finding (near)optimal, or even usable starting constants is complex, and not the scope of this paper. The box dimensions (L), target temperature (T) and lattice constant L_l are all based the chosen phase of Argon, and related to the the following constants used, based on the book *Computational Physics* [2].

$$\epsilon/k_B = 119.8\text{K}$$

$$\sigma = 3.405 Angstrom$$

$$L_l = 2/L$$

Here, it is important to note that the amount of particles in the simulation is fixed by 32. However, with minimal effort the simulation can also run for $4M^3$ particles, where M is an integer. The parameters for the supposed phases are found in the table down below. It must be noted, that these values are in dimensionless units and designed for 32 particles.

Solid	Liquid	Gas
$T = 0.5$ $L = 3.4196$	$T = 1$ $L = 3.3130$	$T = 3$ $L = 4.7425$

In the simulation, the particles start at a FCC lattice configuration, with the lattice constant being twice the lattice parameter. This is because it is a (approximate) stable configuration, where each particle is spaced evenly, while still adhering the molecular dynamics of Argon. As an initial velocity (also, energy or temperature) a Boltzmann distribution has been used, with a fixed temperature of 120K.

1.9 Verification of the results

Phase changes are often related to the following properties: Pressure, temperature, orientation, Verification of the results is vastly complex. For example, pressure is a macroscopic property, and can not be ”measured” from a 32 particle system, but has to be calculated [2]. Thus, the results consist of the mean squared displacement. However, the simulation can be extended to calculate more observables.

Chapter 2

Results

2.1 Results

First of all, we will show that the simulation works properly by verifying the simulation. Secondly, we will show how we will show how we pre-process our simulation data which we can later use for our observable. Third, we will be showing the mean-squared displacement of the Argon atoms since the mean-squared displacement is our observable. Fourthly, we will be showing how we derived the diffusion coefficient from the mean-squared displacement. And finally, we will show the errors for the values we have found for our observable. For the results, the following dimensionless constants have been used in addition to the values supplied at **section**.

$$\text{simulation steps} = 10000 \text{ or } 30000$$

$$\text{timestep} = 0.004$$

$$\text{rescaling} = \text{True}$$

2.1.1 Verification of simulation

To verify our simulation, we will first compare the results for different implementations of the classical mechanics. Next, we will present the results for a simple simulation. And finally, we will verify the beforementioned results to verify our simulation.

2.1.1.1 Simple simulation results

Here we can find the simple simulation results for 2 Argon particles.

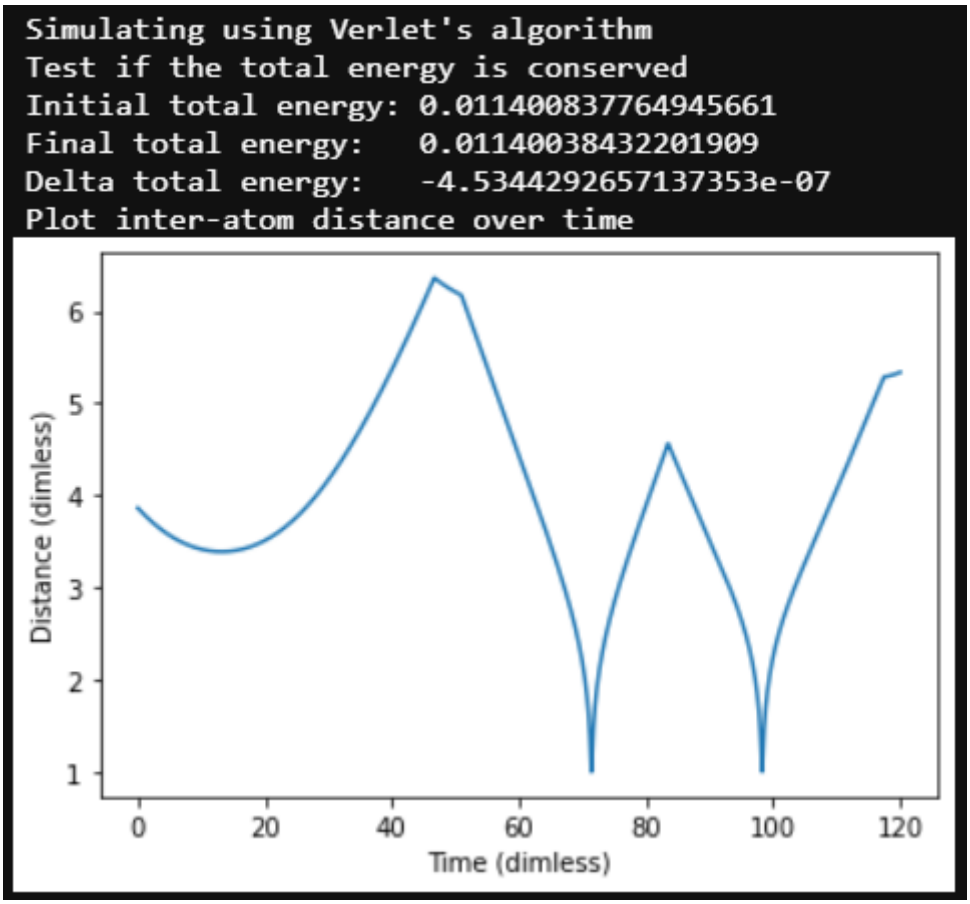

Figure 2.1: Verlet simulation data and inter-atomic distance plot.

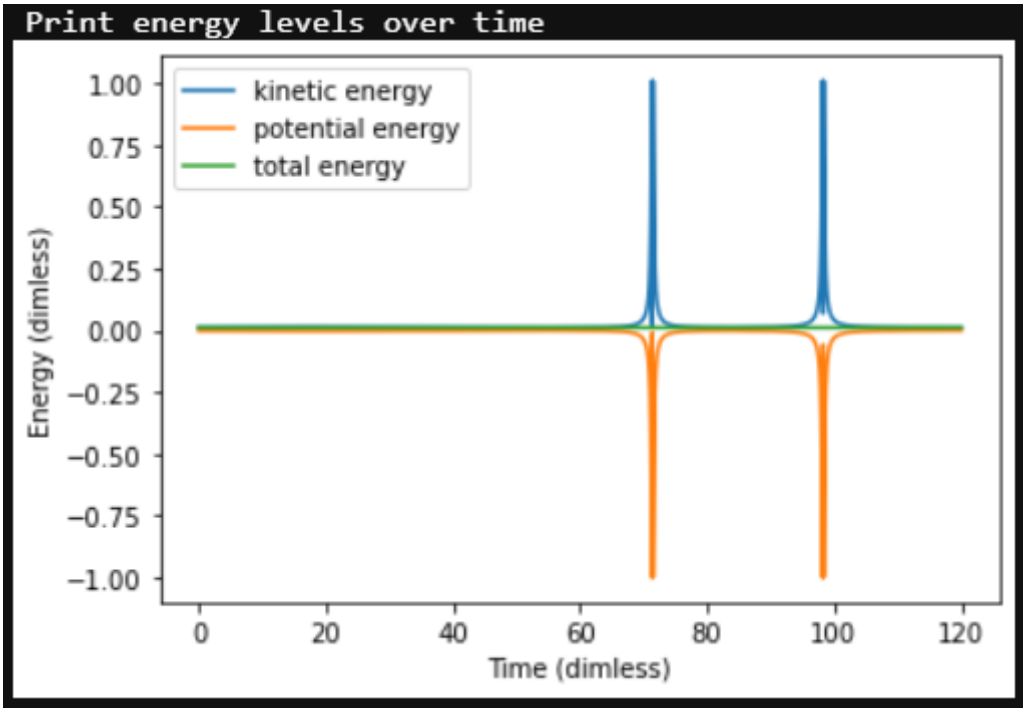

Figure 2.2: Verlet simulation energy plot.

Here, it is shown that in a simple configuration 2 particles will never 'touch', keeping a minimum distance of around 1. This is in line with the Lennard-Jones potential due to the powers. Energy is also conserved; from initial to end $4.5 * 10^{-}7$ has been lost, with a with a value of

2.1.1.2 Verlet vs. Euler results

We are comparing the different algorithms to show which algorithm performs better for our simulations and which we will be using to obtain simulation data later.

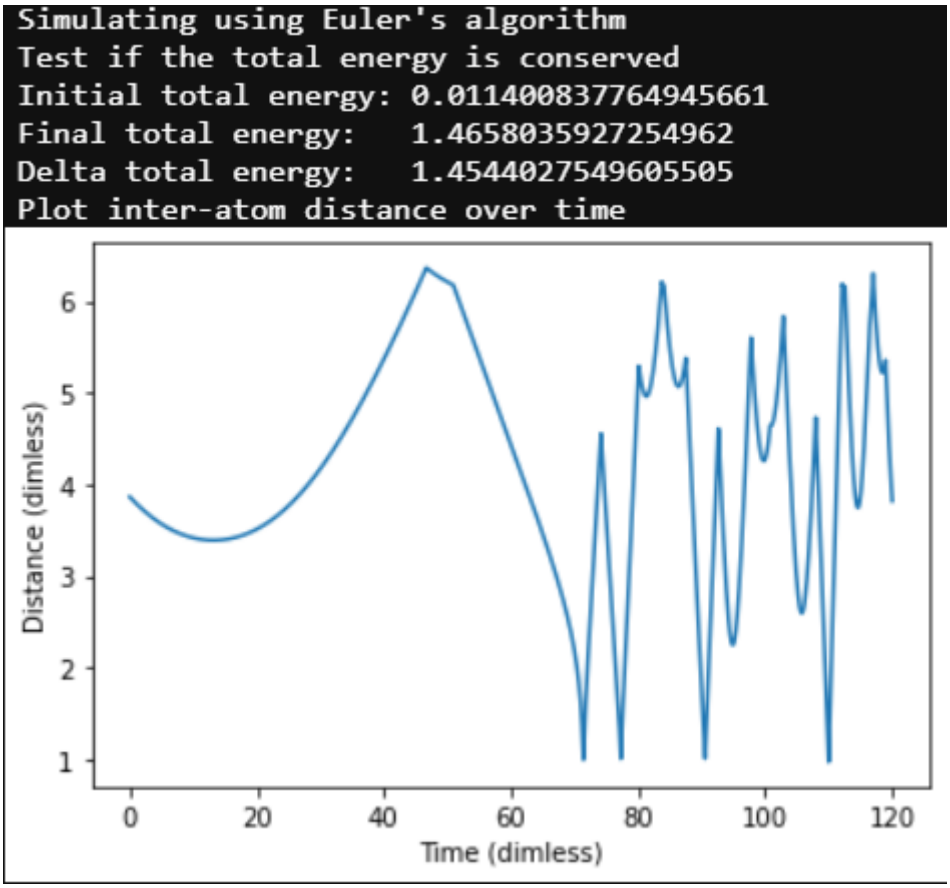

Figure 2.3: Euler simulation data and inter-atomic distance plot.

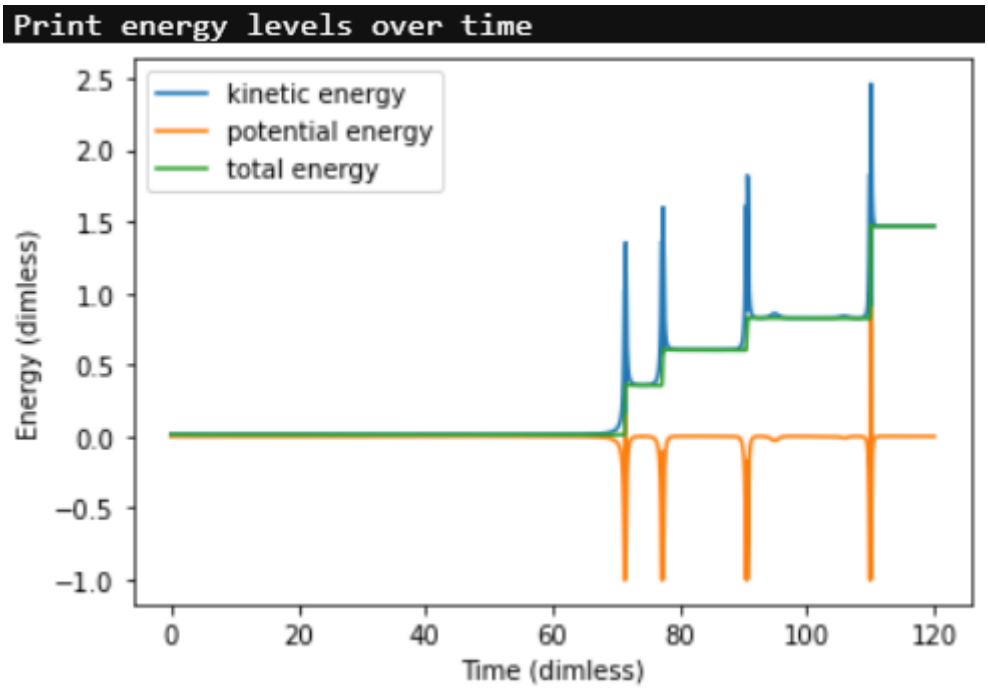

Figure 2.4: Euler simulation energy plot.

It is clear that the Euler method does not conserve energy. For simulations with periodic boundary conditions, this is a problem. The simulation should only change the total energy in the system, if rescaling is applied. The boundary conditions make sure that momentum is conserved; Particles that leave the system are introduced again on the other side of the box, using the nearest-neighbor method, thus every force vector pointing outside the box, has an other vector pointing in the box.[3]
From this moment on, the Euler method shall be disregarded during simulations. It has proven to be an unreliable method to do these types of simulations.

2.1.1.3 Verification

Here we will verify our simulation. There are multiple things that can be tested to verify a simulation:
- Conservation of energy
- Center of mass stays the same
- Manual approximation of the potential and comparing that to the simulation
- Kinetic energy does not converge after the last rescale

We were unable to test all of these requirements, it can be noted that total energy can be observed.
Reflecting on the results, the center of mass does not stay We will be showing that the conversation of energy holds for our simulation, this can be seen from the total energy in the Verlet simulation, which remains constant (neglecting the minor error).
We were unable to test all of these requirements, but regarding the conversation of energy, it is visible in the plots that this has been conserved. The discrete energy shift of the total energy are due to rescaling, and to be expected. Although the total energy does not seem to converge to an energy, this is to be expected since the rescaling is based on a average, kinetic energy. The final calculated temperature is 0.5069, which is close to the energy set by the rescaling (0.5). The potential energy does not change over time much, however this is due to the chosen box size.

2.1.2 Pre-process simulation data

It is important our velocities are in compliance with the Maxwell-Boltzmann distribution. Since our velocities are chosen to be distributed using a normal/Gaussian distribution, we will receive the following distribution of random velocities:

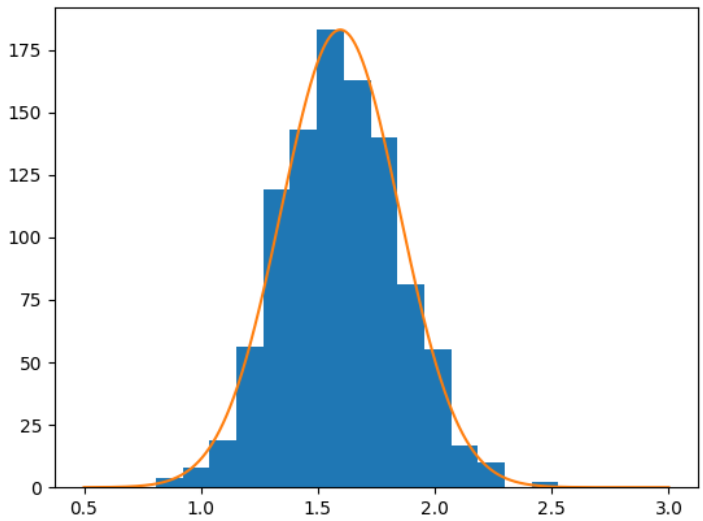

Figure 2.5: Histogram and Gaussian plot for the particle velocities.

As mentioned in the theory, our simulation will use periodic boundary conditions. This will affect the positions from our simulation as they will be stored in a way that it might seem that the movement of the particles over time is not in a continuous motion. To correct this, the periodic particle positions need

to be converted back to non-periodic particle positions. This will be required for the mean-squared displacement, which will be explained next subsection. Below we can find how the non-periodic particle positions over time.

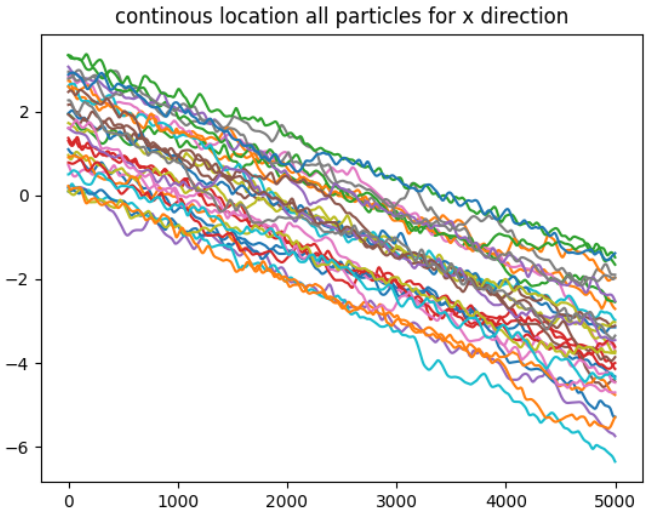

Figure 2.6: Non-periodic particle positions over time for x coordinates, respectively. (solid)

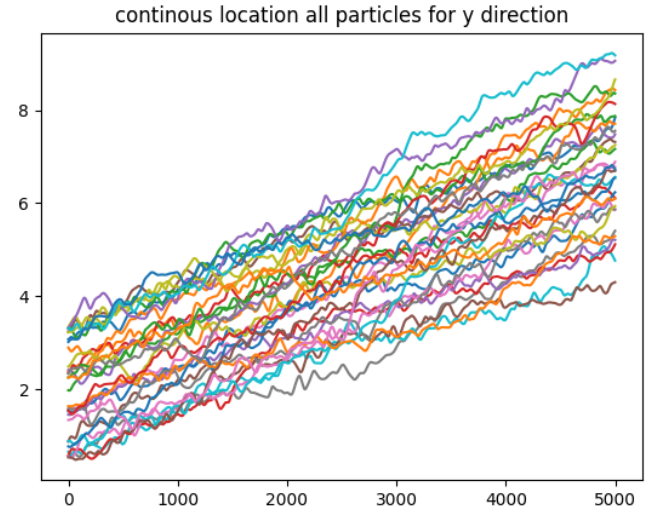

Figure 2.7: Non-periodic particle positions over time for y coordinates, respectively. (solid)

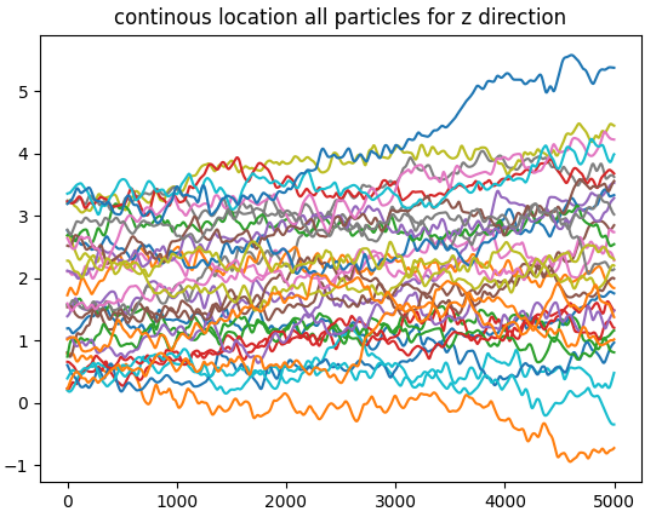

Figure 2.8: Non-periodic particle positions over time for z coordinates, respectively. (solid)

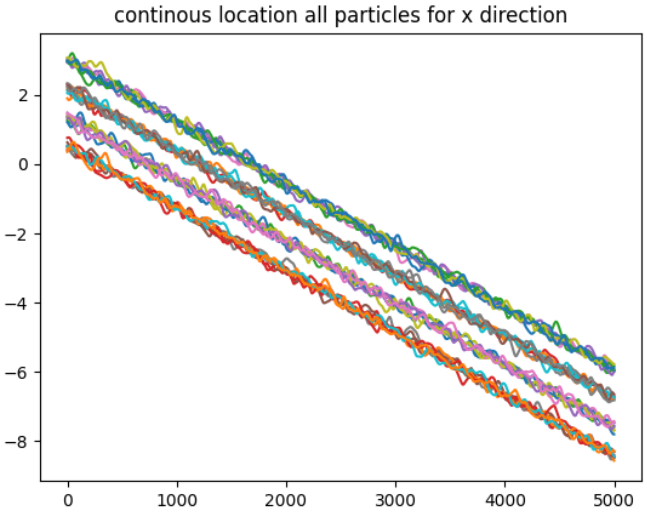

Figure 2.9: Non-periodic particle positions over time for x coordinates, respectively. (liquid)

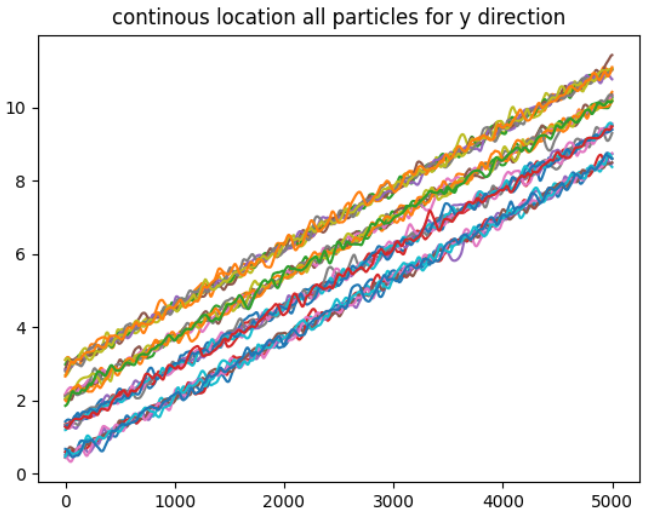

Figure 2.10: Non-periodic particle positions over time for y coordinates, respectively. (liquid)

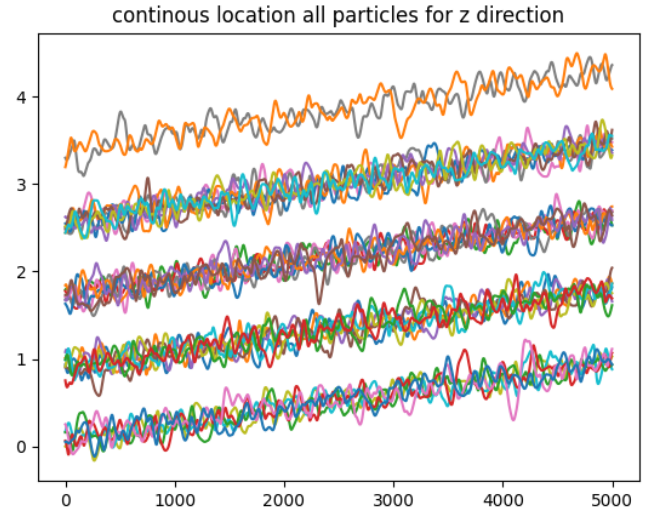

Figure 2.11: Non-periodic particle positions over time for z coordinates, respectively. (liquid)

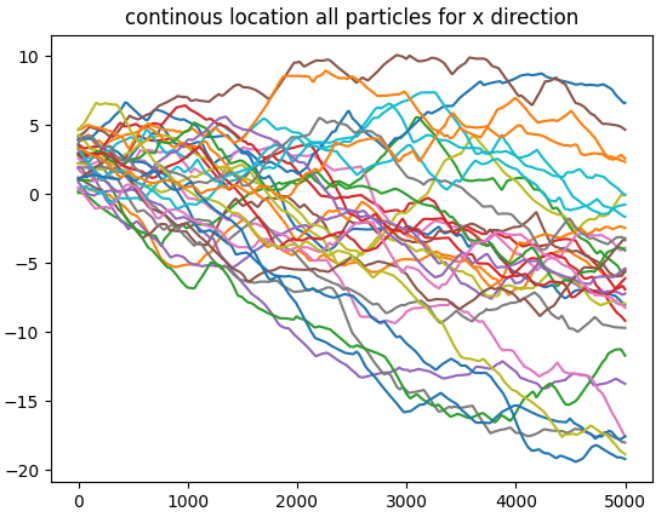

Figure 2.12: Non-periodic particle positions over time for x coordinates, respectively. (gas)

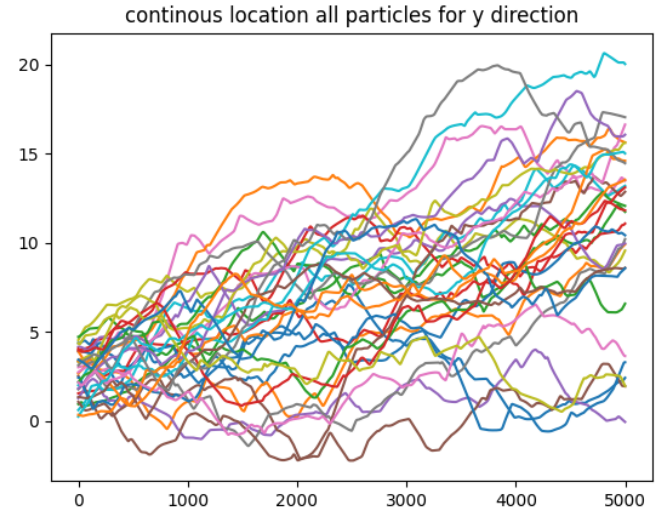

Figure 2.13: Non-periodic particle positions over time for y coordinates, respectively. (gas)

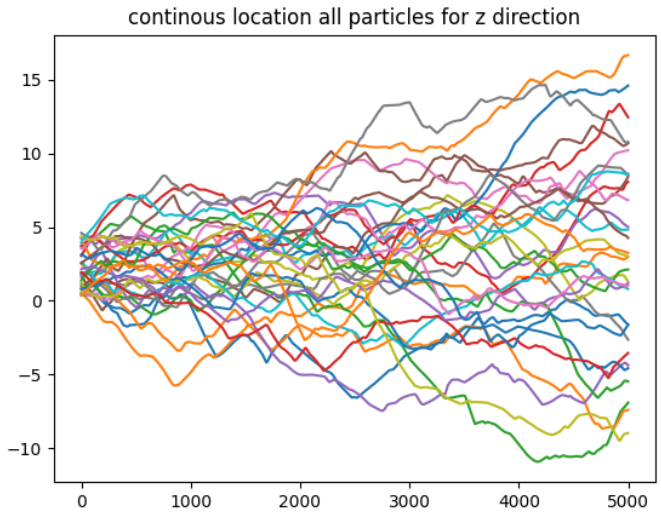

Figure 2.14: Non-periodic particle positions over time for z coordinates, respectively. (gas)

In 2.6 to 2.14 The locations are made non-periodic again.
Below we can find the a plot of the Verlet simulation over 10000 timesteps for a target dimensionless temperature of 0.5 (solid).

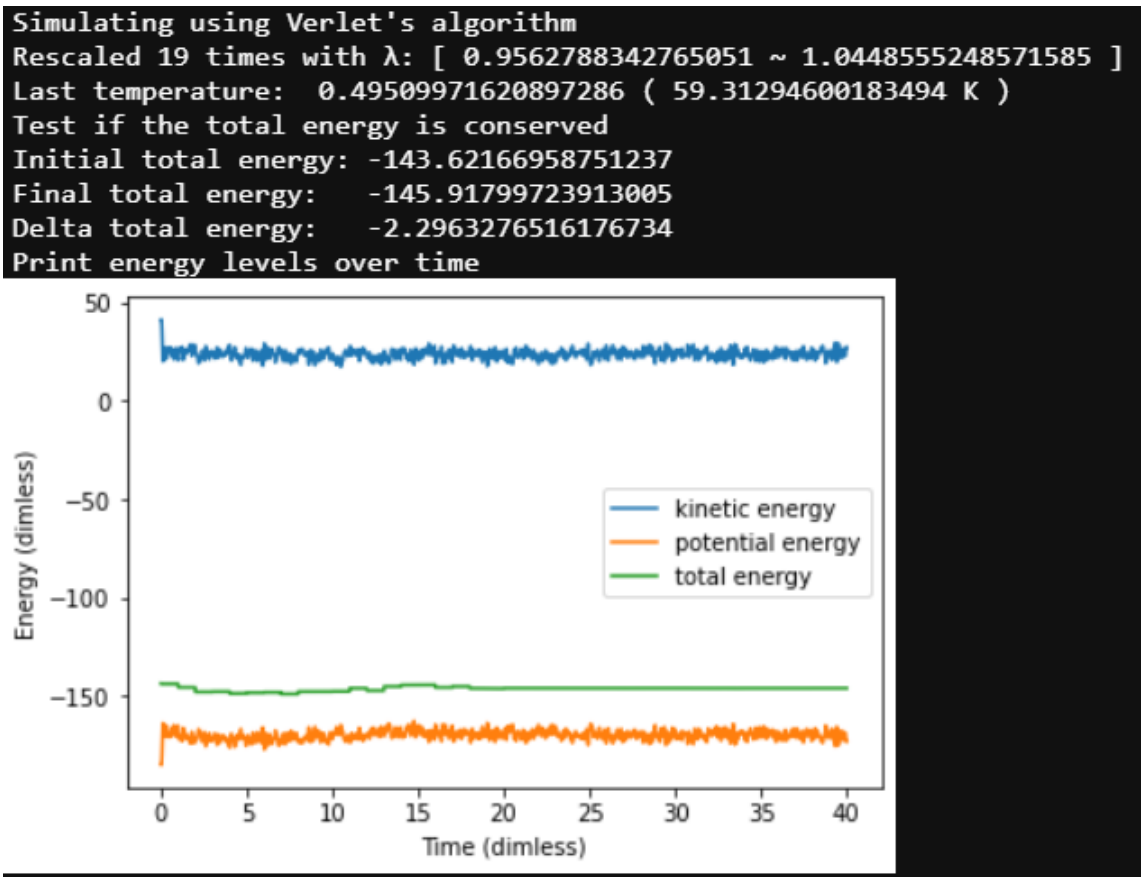

Figure 2.15: Verlet simulation data and energy plot. (solid)

Below we can find the a plot of the Verlet simulation over 10000 timesteps for a target dimensionless temperature of 1.0 (liquid).

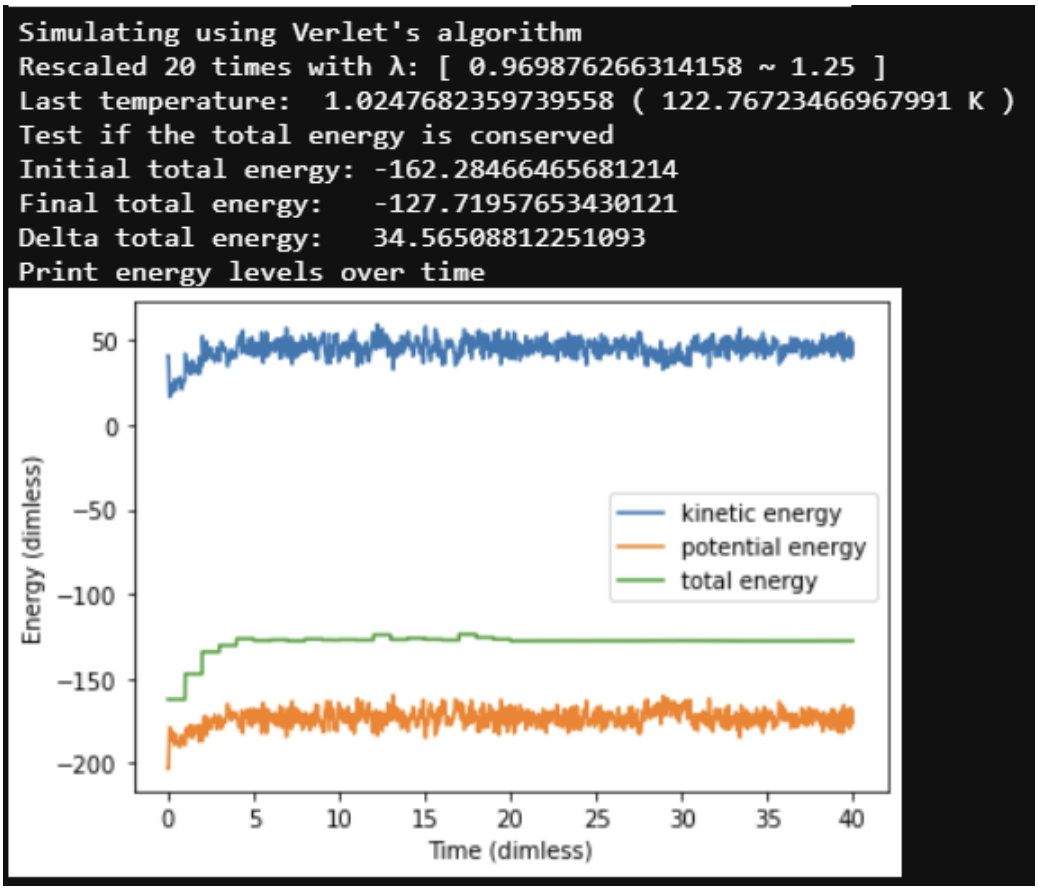

Figure 2.16: Verlet simulation data and energy plot. (liquid)

Below we can find the a plot of the Verlet simulation over 10000 timesteps for a target dimensionless temperature of 3.0 (gas).

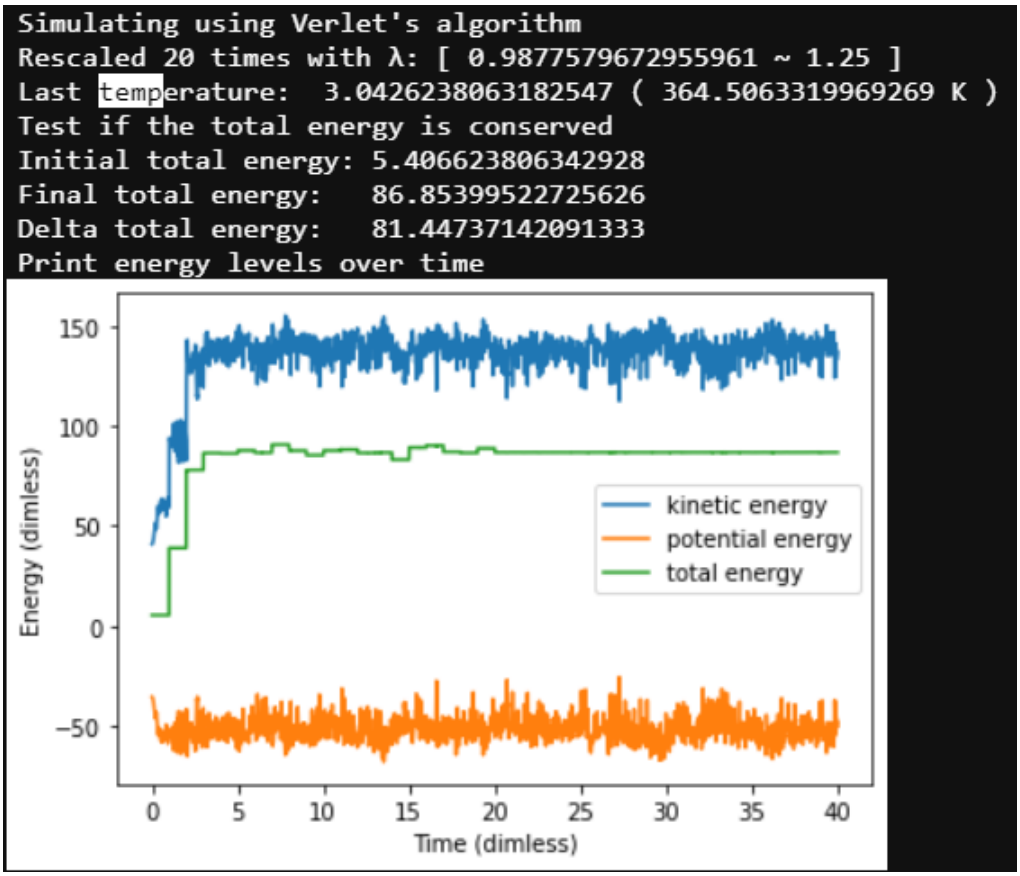

Figure 2.17: Verlet simulation data and energy plot. (gas)

In 2.17 the velocities has been rescaled

2.1.3 Mean-squared displacement

Using the non-periodic particle positions from the simulation data, we are able to calculate the mean-squared displacement. In the following plot, the mean squared displacement of the particles is shown.

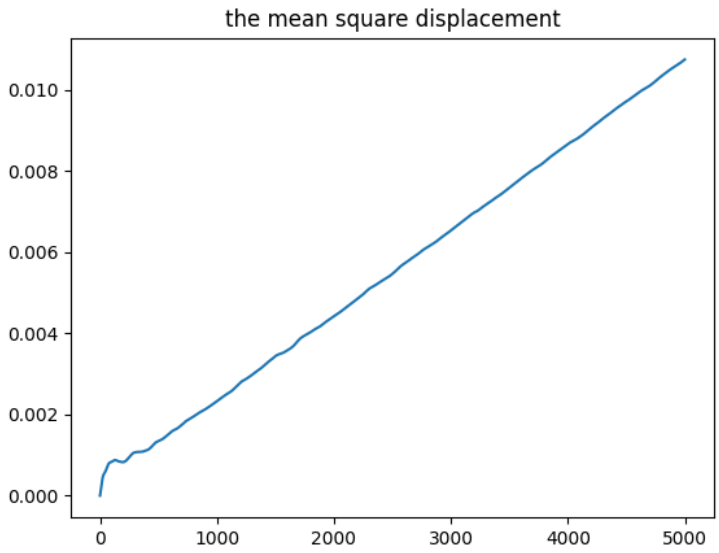

Figure 2.18: Mean-squared displacement of particles over time. (solid)

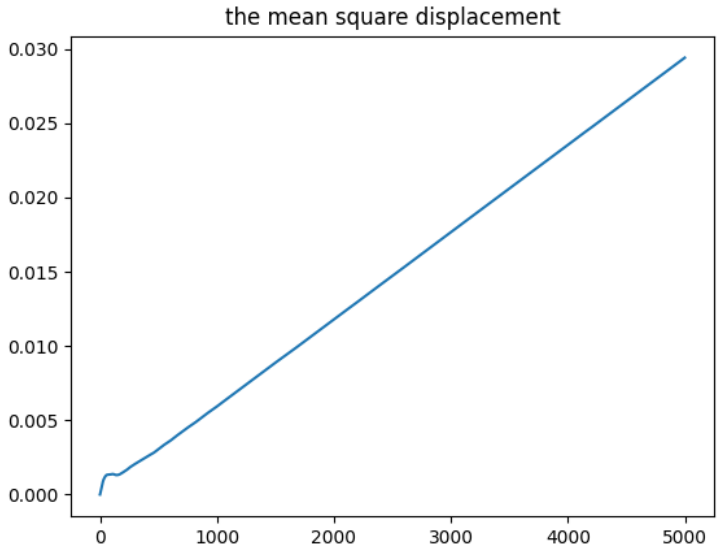

Figure 2.19: Mean-squared displacement of particles over time. (liquid)

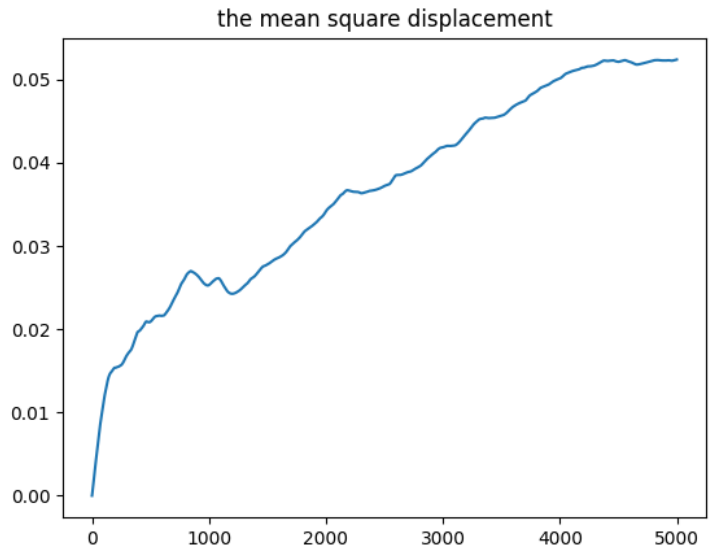

Figure 2.20: Mean-squared displacement of particles over time. (gas)

2.1.4 Auto-correlation function

The auto-correlation function is shown in the figure below.

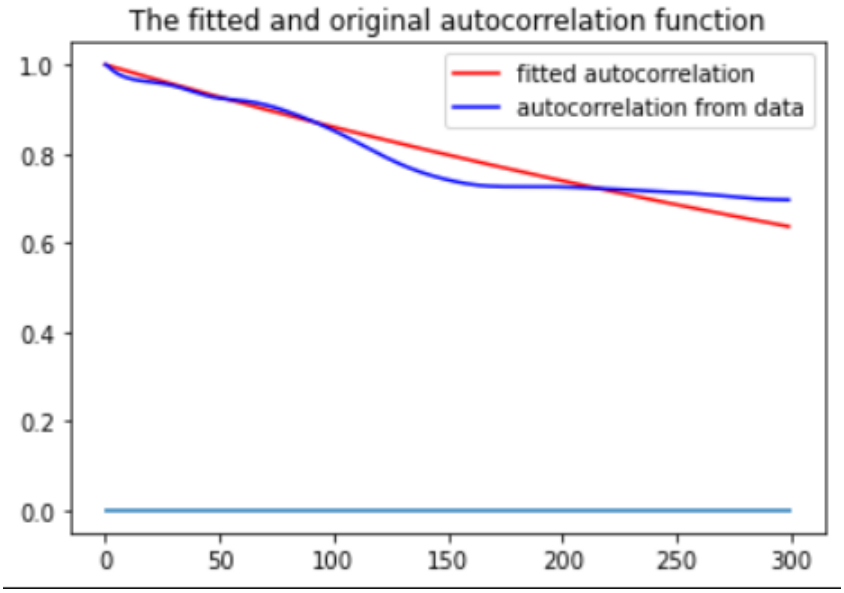

Figure 2.21: Auto-correlation function. (solid)

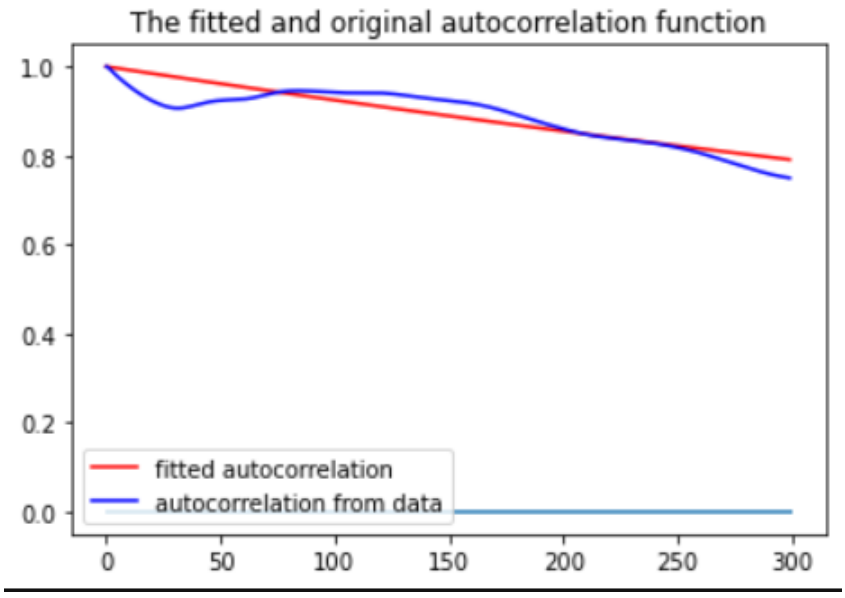

Figure 2.22: Auto-correlation function. (liquid)

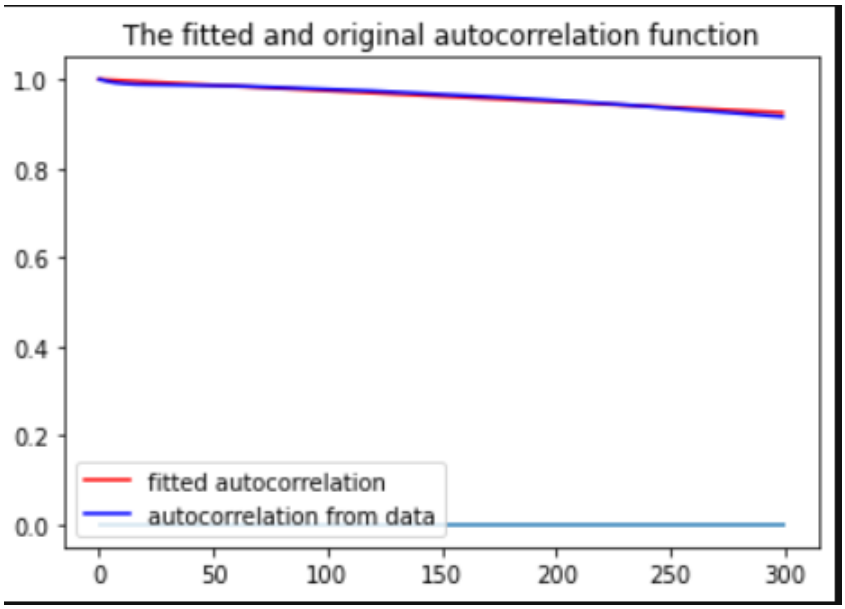

Figure 2.23: Auto-correlation function. (gas)

The auto-correlation does not seem to go to 0.

The dimensionless units for τ are for solid, liquid and gas as following: 665, 1280, 3880.

2.1.5 Diffusion coefficient

From the mean-squared displacement, we are able to estimate the diffusion coefficient. This is shown below. The diffusion coefficient approaches an constant value, indicating a solid.[2]
However, our simulation has a total momentum that is not zero. This will result in the diffusion coefficient being not usable.

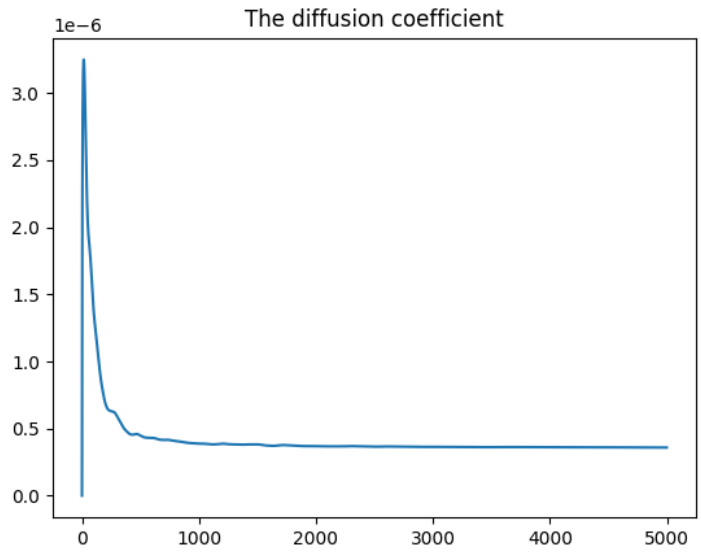

Figure 2.24: Diffusion coefficient. (solid)

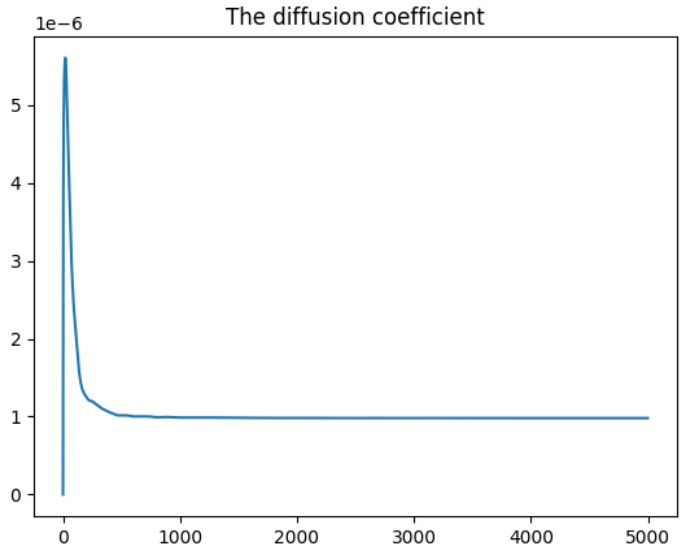

Figure 2.25: Diffusion coefficient. (liquid)

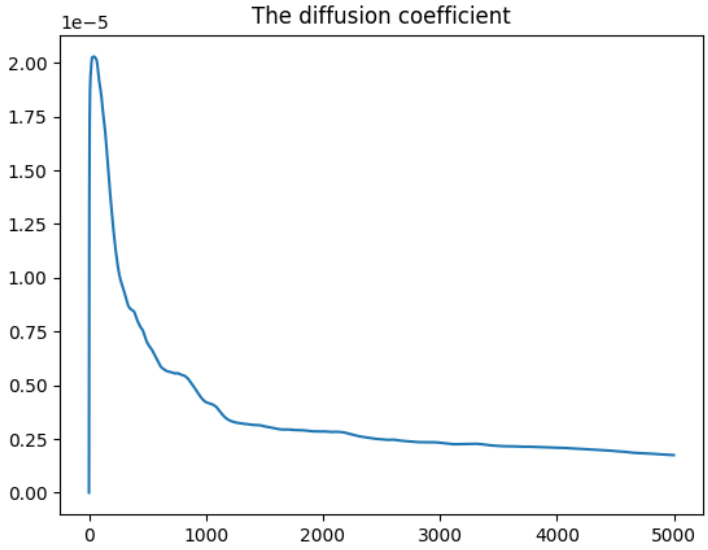

Figure 2.26: Diffusion coefficient. (gas)

For the gas, liquid and solid the following diffusion coefficients were found: 2.5e-6, 1.0e-6 and 0.5e-6 in dimensionless units, respectively.

2.1.6 Observable errors

Now that the diffusion coefficient has been found from the mean-squared displacement, we should also account for errors in the found values.

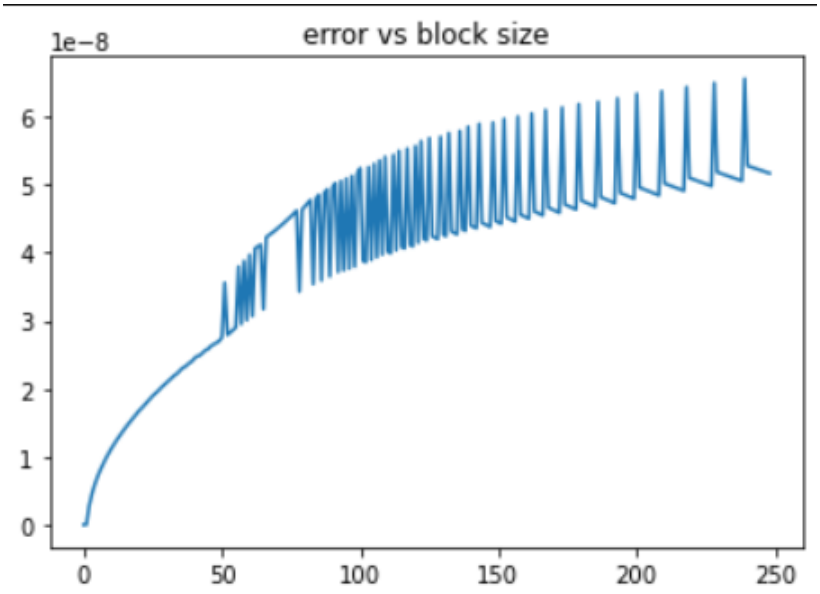

Figure 2.27: Data-blocking error. (solid)

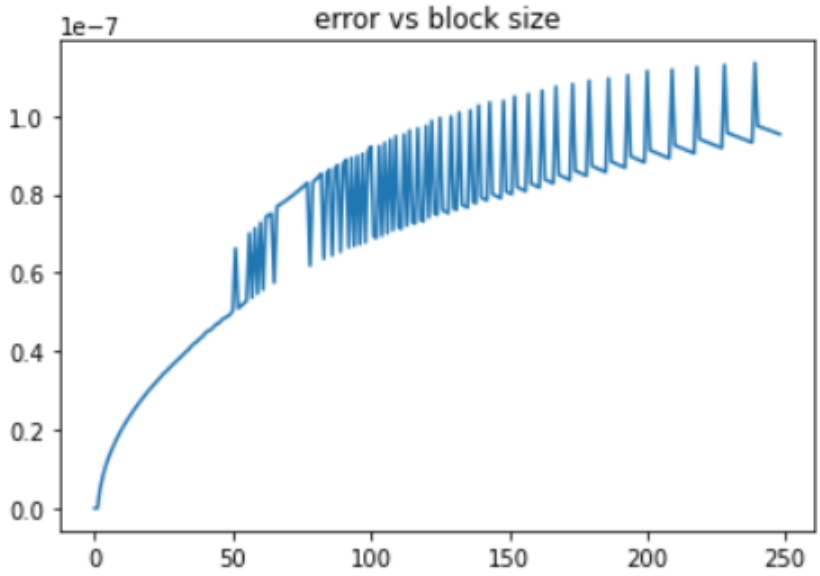

Figure 2.28: Data-blocking error. (liquid)

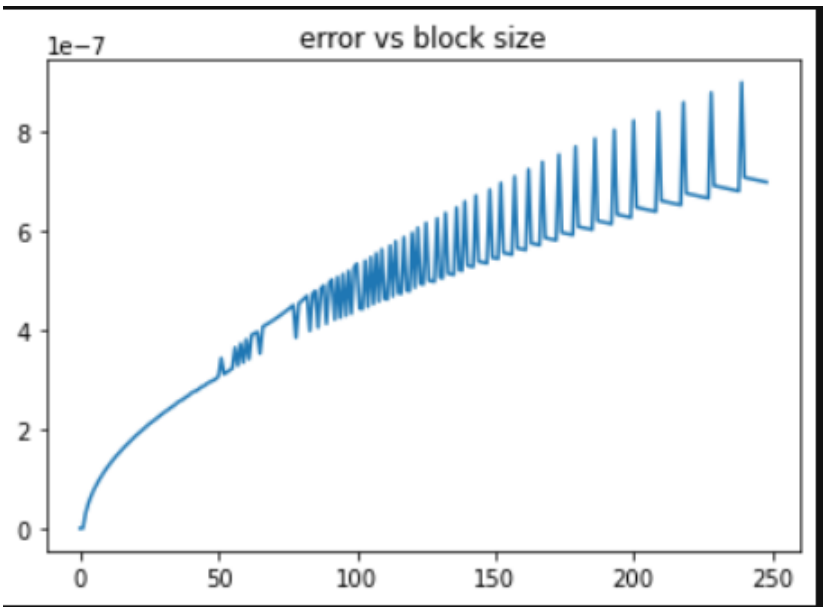

Figure 2.29: Data-blocking error. (gas)

The first note should be that in the higher range of the blocksizes, this is probably due to the error being small, resulting in either rounding errors or float overflows. However. the values can still be read.

	solid	liquid	gas
σ	$1.4 * 10^{7}$	$3.6 * 10^{-7}$	$3.8 * 10^{-6}$
σ_A	$6 * 10^{-8}$	$1 * 10^{-7}$	$8 * 10^{-7}$

Calculating the average seems is not useful, due to the momentum flow. However, the errors seem to be in the same order of magnitude.

Chapter 3

Discussion

There were problems with our simulation, these are most important and will be discussed first. Afterwards the code shall be discussed, and comparison to the literature will be made regarding the observed diffusion coefficient.

3.1 Discussion results

3.1.1 Velocities

In 2.6 to 2.14 the location is shown, starting from time 0. Here, a drift is present from timestep 0 (calculated from the last rescale) on. This should ideally not be present, this means that our system is moving with a velocity trough the periodic space, resulting in our energy being higher than supposed to. A drift means energy is 'leaking' out of the system in the form of momentum. Thus, our results are not valid. This invalidates our results. The temperature is now random, with the effect being that our temperatures are overestimated. The result is also that our mean square distance is mainly based on the net velocity of the system, resulting in the autocorrelation function being incorrect as well.

3.1.2 Diffusion literature

Compared to the literature for liquid Diffusion coefficients, the values are off by multiple magnitudes. [1] We believe this is due to a programming error.

3.2 Project discussion

The overall project went well in our opinion. The communication was good, we were in touch regularly on WhatsApp and on Discord. The division of work-load was fair for the both of us as well. The work was evenly split among us, which would be about 50/50. The GitLab repository was kept tidy at all times and we were concise with our weekly progress report. We used the original 'skeleton.py' file for the coding. We have tried to maintain a similar coding style. This also includes comments and documentation. We also believe we have written quite efficiently running code since our simulations run fast, although when putting the code together as 1 simulation following PEP8 as tightly as possible, mayor slowdowns were present. Beforehand, no knowledge of the most optimal way to achieve this was known. We have also actively tried to improve the performance of our code by, for instance replacing lists by NumPy arrays. A lot of new insights have been obtained, especially for the final steps of a project since there is a big difference between: running separate functions in sequence creating a figure and creating 1 function that does it all in a optimal way.

Things that could have gone better are for instance time-management. We usually had trouble to meet the weekly milestones due to various reasons, while our Python skills were improving we were having problems with the way we wrote previous code, which is inefficient. however overall our effort to create the simulation . Also finishing the project with acceptable results could have gone better.

3.2.1 Code efficiency

The efficiency of the program used in this paper is sub-optimal. Some mayor points will be discussed next.

3.2.1.1 The number of loops present

In the script, the potential energy is calculated after a simulation is completed, using the previously calculated values for the locations. Since the calculation of the potential energy requires the inter-atomic distances, and this is a relative time consuming function, a method should have been implemented to remove the need for this calculation (especially since the amount of computations scales quadratic with the amount of particles). More importantly, the inter-atomic distances were already calculated, however not stored. Due to the these values not being necessary at first, we never implemented it into our simulate function.

3.2.1.2 Compiling

Python has a significant decrease in efficiency when code is not compiled beforehand. Some major improvements can be used by using array functions, using the NumPy module, and then implementing the time-step iteration in a separate function. The usage of vectorized functions itself does probably not decrease efficiency, however it makes the code much easier to debug and/or verify. Which creates the oppurtunity to rewrite the time-step iteration function for calculating positions and velocities for example such that this function can be compiled beforehand. Since Python is interpreter-based, it might be better to use high-level programming languages such as C, C++ or Java. It would be possible to gain an performance boost to up to 400 times.

3.3 Points of improvement

Keeping the previous evaluation in mind, here is our list of things we believe could have improved:
- Better commenting to improve code-readability
- Using more optimal NumPy functions
- Vectorize functions
- Use a high-level programming language
- Automate optimal cutoffs for auto-correlation function
- Improve estimations of τ
- Improve initial velocities (dependency of temperature)
- Improve initial velocities (Maxwell-Boltzmann distribution)

Chapter 4

Conclusion

It can be concluded that it is possible to determine the diffusion coefficient from the mean-squared displacement. This should also be possible for various temperatures. This would indicate that it should be able to read off what phase the Argon particles are in, by observing the mean-squared displacement over time, plotted for a stabilized simulation towards a desired temperature. However, this simulation described in this paper has a problem with the velocities, resulting in a non-zero net momentum. This increases the kinetic energy in a non-physical, way results in this simulations being not useful for determining the diffusion coefficients.

Bibliography

[1] R.A. Fisher and R.O. Watts. *CALCULATED SELF-DIFFUSION COEFFICIENTS FOR LIQUID ARGON*. https://www.publish.csiro.au/ph/pdf/ph720529, 1972.

[2] J.M. Thijssen. *Computational Physics*. Cambridge University Press, 2007.

[3] Filion L. Valeriani C. van Meel, J.A. and D. Frenkel. *A parameter-free, solid-angle based, nearest-neighbor algorithm*. J. Chem. Phys. 136, 234107; https://doi.org/10.1063/1.4729313, 2012.

[4] L. Verlet. *Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules*. American Physical Society, https://doi.org/10.1103/PhysRev.159.98, 1967.