Computational Physics AP3082 - Project 1 - Molecular Dynamics Simulation

David Bakker,[1] Klara Chmiel[2] and Nico Kerkhoven[3]

[1] Msc Applied Physics, TU Delft (4675932, D.L.Bakker-1@student.tudelft.nl)
[2] Msc Applied Physics, TU Delft (5344247, K.M.Chmiel@student.tudelft.nl)
[3] Msc Applied Physics, TU Delft (4606841, N.C.Kerkhoven@student.tudelft.nl)

Abstract. In this report we present a molecular dynamics simulation of Argon gas. The mono-atomic gas molecules will be simulated in a closed container of fixed dimensions and at a fixed temperature. Minimal image convention periodic boundary conditions will be applied. The molecules will be placed on a 3D FCC lattice and evolved in time via Verlet method. Observables such as diffusion coefficient and specific heat capacity of the simulated gas will be determined and compared to the values from the literature. We will show that molecular dynamics simulations are a powerful tool that allows us to simulate real physical systems and extract meaningful physical quantities.
The structure and contents of this report are based on the lecture notes of AP3082 Computational Physics (2020/21 Q3), TU Delft.[1]

1 Introduction

Molecular dynamics (MD) is a widely used method for studying classical many-particle systems. It is for example one of the principal tools in the theoretical study of biological molecules. It's basis is integration of the equations of motion of the system numerically. It can therefore be viewed as a simulation of the system as it develops over a period of time. The equations of motion determine how the system 'moves' in phase space along its physical trajectory. The great advantage of the MD method is that it not only provides a way to evaluate expectation values of static physical quantities;dynamical phenomena, such as transport of heat or charge, or relaxation of systems far from equilibrium.

Molecular dynamics simulations generate information at the microscopic level, including atomic positions and velocities. The conversion of this microscopic information to macroscopic observables such as pressure, energy, heat capacities, etc., requires statistical mechanics.

The molecular dynamics method was first introduced by Alder and Wainwright in the late 1950's (Alder and Wainwright, 1957,1959)[2][3] to study the interactions of hard spheres. Their studies provided many important insights concerning the behavior of simple liquids. The next major advance was in 1964, when Rahman carried out the first simulation using a realistic potential for liquid Argon (Rahman, 1964)[4][5] which used a Lennard-Jones potential; calculations of system properties, such as the coefficient of self-diffusion, compared well with experimental data.[6]

Molecular dynamics simulations are an extremely powerful tool that allows scientists to study properties of molecules and molecule ensembles under chosen conditions and predict their behaviours without the need of experimental laboratory work. They provide insights into physical behaviours of substances in an efficient, low cost way and helped in making rapid advancements in fields such as materials science, biochemistry and biophysics.

In our investigation we have exploited the Verlet method of molecular dynamics simulation of Argon gas and extract physical quantities (observables) of the system. We present methods and results in this report.

1.1 Forces on the Atoms

Argon atoms are neutral so they do not interact via Coulomb interaction. However, small displacements between the nucleus and the electron cloud gives rise to a small dipole moment. Overall, the interaction between dipoles gives an attractive interaction. On the other hand, Argon atoms do not come so close to each other that they overlap (they do not form molecules), hence there must be a repulsive force at small distances. Both the attractive and repulsive interaction are parametrized in the Lennard-Jones potential, which takes the form:

$$U(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{1}$$

With fitting parameters for Argon $\epsilon/k_B = 119.8K$ and $\sigma = 3.405$ Angstrom.

1.2 Time Evolution and Governing Equations

The molecular dynamics simulation method is based on Newton's second law, the equation of motion, $\vec{F} = m\vec{a}$, where $\vec{F}$ is the force exerted on the particle, m is its mass and $\vec{a}$ is its acceleration. It is possible to determine the acceleration of each atom in the system once we know the force acting on it. Integration of the equations of motion then gives a trajectory that describes the positions, velocities and accelerations of the particles as they vary with time. From this trajectory, the average values of properties of the system can be determined. The method is deterministic; once the positions and velocities of each atom are known, the state of the system can be predicted at any time in the future or the past.[5]

Newton's equation of motion is given by

$$m\frac{d^2x_i}{dt^2} = F(x_i) = -\nabla U(x_i) \tag{2}$$

Where x_i is the position of particle i, t is time and U is the potential.

These equations cannot be solved exactly on a computer, but need to be numerically solved. Instead of continuous time, we thus evolve the system with finite time steps h.

A very simple and intuitive way of doing this is to use the Euler method. If $x_i(t_n)$ and $v_i(t_n)$ are are the position and velocity of particle i at time t_n, the position and velocity at the next time $t_{n+1} = t_n + h$ are given by:

$$x_i(t_{n+1}) = x_i(t_n) + v_i(t_n)h \tag{3}$$

$$v_i(t_{n+1}) = v_i(t_n) + \frac{1}{m}F(x_i(t_n))h \tag{4}$$

The normal Euler method for integrating ordinary differential equations (ODEs) has a major downside for the molecular dynamics simulations we are doing: they do not conserve the total energy of the system very well. It is a first-order method, which means that the local error (error per step) is proportional to the square of the step size, and the global error (error at a given time) is proportional to the step size. We therefore move on to a better algorithm - the Verlet algorithm.

The Verlet algorithm is the most popular algorithm for molecular dynamics because it has a huge advantage of belonging to a class of so-called symplectic integrators that are particularly suited for the time-evolution of Hamiltonian systems because they possess an additional symmetry, called symplecticity - they preserve the geometrical structure of the continuum solution in phase space. A striking property of the energy determined from the Verlet method is that it does not show any drift in the total energy (in exact arithmetic). This stability follows directly from the fact that the Verlet algorithm is time-reversible, which excludes steady increase or decrease of the energy for periodic motion[7]. The error per integration step is also improved compared to Euler method and is of the order h^4. The global error in position and velocity is of the order of h^2[8]. The Verlet equations are given by:

$$x_i(t_{n+1}) = x_i(t_n) + v_i(t_n)h + \frac{h^2}{2}F(x_i(t_n)) \tag{5}$$

$$v_i(t_{n+1}) = v_i(t_n) + \frac{h}{2}[F(x_i(t_{n+1})) + F(x_i(t_n))] \tag{6}$$

1.3 Dimensionless Equations

To simplify the code and generalize it, we rescale the units of the physical quantities to be dimensionless. This way, we don't need to use material specific constants in the code, which will (slightly) speed up the code and make it more readable. It is also less cumbersome to code this way and you are less prone to make mistakes. You don't need to add constants everywhere, or forget to do this. After running the code we can calculate the values of the observables in the correct units, in our case specific to Argon.

Let's take the potential energy as an example, see Eq. 1. You can see two different material specific constants in this equation, ϵ and σ. The natural thing to do is to choose $\tilde{r} = r/\sigma$ and $\tilde{U} = U/\epsilon$. In this case we get the following conversion,

$$U(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \rightarrow \tilde{U}(\tilde{r}) = 4\left[\left(\frac{1}{\tilde{r}}\right)^{12} - \left(\frac{1}{\tilde{r}}\right)^{6}\right] \tag{7}$$

If the model gives for the potential energy an output $U = 10$ this is in units of ϵ, meaning you have $U = 10\epsilon \approx 1.65 \times 10^{-21}$J. In a similar way, we can transform the equations of motion,

$$m\frac{d^2x_i}{dt^2} = -\nabla U(x_i) \rightarrow \frac{d^2\tilde{x}_i}{d\tilde{t}^2} = -\tilde{\nabla}\tilde{U}(\tilde{x_i}) \tag{8}$$

where we measure position in units of σ, potential energy in units of ϵ and time in units of $\sqrt{m\sigma^2/\epsilon}$. In our code we made all our units dimensionless, but we won't show every conversion here as this would be too much.

1.4 Periodic Boundary Conditions - Minimal Image Convention

We want to simulate an infinite system of Argon atoms - however, a computer cannot simulate infinite space. We thus use a finite box of length L, with periodic boundary conditions (to emulate an infinite system). A periodic boundary condition for a wave function means: $\Psi(x + L) = \Psi(x)$, where L is the size of the one-dimensional system. For a molecular dynamics simulation, we choose a periodic boundary condition to ensure that an atom will never reflect of a hard boundary. This has some subtle effects on the molecular dynamics simulation:

1. If a particle leaves the box, it re-enters at the opposite side
2. The evaluation of the interaction between two atoms is also influenced by the periodic boundary condition: the shortest distance between atoms may now include crossing the system boundary and re-entering.

It's important to realise that the forces calculated on each particle are dependent on the distances between the particles and therefore application of periodic boundary conditions will really affect our simulation.

To implement the periodic boundary condition we place our main simulation box in the centre of other 8 simulation squares that are its copies. This is modeled by introducing copies of the particles in volumes bordering the actual system. A problem occurs when we want to simulate a large number of particles. Calculating interactions between all particles in all 8 copies of the system and the original system will be computationally heavy and thus time consuming therefore we need a simplified way of applying the periodic boundary conditions. The method of applying periodic boundary conditions that will be of aid to achieve this goal is the called the Minimal Image Convention.

In the Minimal Image Convention only the force due to the nearest image of other particles and the original particles in the box is taken into account. Note that this does not mean that an atom is only influenced by its nearest neighbouring atom. Instead, we just consider the closest of all 'image partners' (plus the original) of every atom for calculating interaction. This condition is easily evaluated with Python numpy arrays and can be implemented as:

$$(x_i - x_j + L/2)\ \%\ L - L/2 \tag{9}$$

where x_i and x_j refer to positions of particles i and j.

1.5 Face-centered cubic lattice

A face-centered cubic (fcc) lattice is essentially a lattice given by the stacking of equally sized spheres. In a cubic unit cell atoms are positioned at the corners of the cube and at the centres of the sides. This is the most efficient way to pack spheres which means that in a fcc lattice the fraction of the volume occupied by spheres is the highest, the actual value is $\frac{\pi}{3\sqrt{2}}$. The initial positions of the particles are given by a fcc lattice such that the particles don't start too close to each other and because Argon atoms have a fcc lattice in solid form. A fcc lattice is fully described by the lattice constant which gives the size of a unit cell and three primitive basis vectors positioned at:

$$v_1 = \left[0, \frac{1}{2}, \frac{1}{2}\right]^T$$
$$v_2 = \left[\frac{1}{2}, \frac{1}{2}, 0\right]^T$$
$$v_3 = \left[\frac{1}{2}, 0, \frac{1}{2}\right]^T$$

1.6 Initial velocities and temperature

The velocities should follow a Maxwell-Boltzmann distribution, which depends on the temperature. The initial velocities are randomly picked from this distributions. In doing this you shouldn't forget that the average velocity should be zero, to prevent the system from drifting. It would seem like it is easy to choose the temperature of the gas, by just choosing the right temperature of the Maxwell-Boltzmann distribution, but this is not the case. The system will come to equilibrium, but the temperature of this system is hard to predict. Therefore we need to force the system to have the correct temperature. The temperature depends on the kinetic energy, following the equipartition theorem,

$$E_{kin} = (N-1)\frac{3}{2}k_B T \tag{10}$$

where N is the number of particles, k_B the Boltzmann constant and T the temperature. The $N-1$ is there because of some subtleties about the degrees of freedom in the system, which we will not discuss here. If we want to force the system to have the correct temperature, we need to force the system to have this kinetic energy. The plan is to calculate the average kinetic energy of our system, compare it to the kinetic energy above (Eq. 10), and rescale the velocities to get closer to the correct kinetic energy. Repeat this after a certain amount of time steps, until you have reached the desired temperature within a certain tolerance. In mathematical terms, the velocity should be rescaled as,

$$\vec{v}_i \rightarrow \sqrt{\frac{(N-1)3k_B T}{\sum_i m v_i^2}}\,\vec{v}_i \tag{11}$$

where we do this calculation until we converged to the correct temperature. From this point on wards we can use the results from our model to calculate observables.

1.7 Autocorrelation function

In a simulation we look at the time-evolution of data, every time-step calculations are made and the information is updated based on the previous data. As a consequence measurements in the simulation are highly correlated which means that there are only small variations between two time-steps. One way of quantifying the correlation is by using the autocorrelation function, it measures how long it takes such that the data is no longer correlated, given by the correlation time. The expression for the finite-time autocorrelation function is:

$$\chi(t) = \frac{(N-t)\sum_n A_n A_{n+t} - \sum_n A_n \times \sum_n A_{n+t}}{\sqrt{(N-t)\sum_n A_n^2 - (\sum_n A_n)^2}\sqrt{(N-t)\sum_n A_{n+t}^2 - (\sum_n A_{n+t})^2}} \tag{12}$$

Here we sum from 1 to $N-t$ with N being the total simulation time. For correlated data this expression can be approximated by $e^{\frac{-t}{\tau}}$ with τ the correlation time. In our simulation we have added a function which calculates the autocorrelation and it works fine on random correlated data. One should note that the autocorrelation function will not be used for the observables calculated in this report. The observables that we chose are not suitable for this as the data for the mean squared displacement and the heat capacity is too correlated, the autocorrelation function basically remains constant at 1 for these observables.

2 The Simulation

The molecular dynamic simulation was written in Python programming language. The code is attached in the Appendix. We will first show that our code actually works, by first looking at conservation of energy and then go over several observables we calculated using our code and compare them with experimentally measured values.

2.1 Conservation of energy

We want our simulation to represent an actual physically possible process. This means that the fundamental law of conservation of energy should also be obeyed by our simulation. The total energy is the sum of the potential and kinetic energy. They are exchanged between each other but their sum should be constant in time. The kinetic energy and potential energy are calculated every time step in the simulation and returned, so we can plot them over time and see if their sum is constant. In the plots below you can see that the energy in our simulation is indeed conserved. The left Figure is with 16 particles and the right Figure is with 256 particles. So even for a large number of particles the energy is conserved.

2.2 Observables

Specific heat (C_V)
It was shown by Lebowitz in his paper[9] that the specific heat of the system of N atoms can be calculated from fluctuations in the kinetic energy of the system as:

$$\frac{\langle \delta K^2 \rangle}{\langle K \rangle^2} = \frac{2}{3N}\left(1 - \frac{3N}{2C_V}\right) \tag{13}$$

where K is the kinetic energy, and $\langle \delta K^2 \rangle = \langle K^2 \rangle - \langle K \rangle^2$ the fluctuations of the kinetic energy and C_V is the total specific heat capacity for all N atoms.

The simulation was run 100 times with 2000 evolution steps of step size 0.004 in our dimensionless units of time. The number of particles was 256. The value of the specific heat capacity was obtained by finding a mean of the output values and the error was computed by finding their standard deviation.

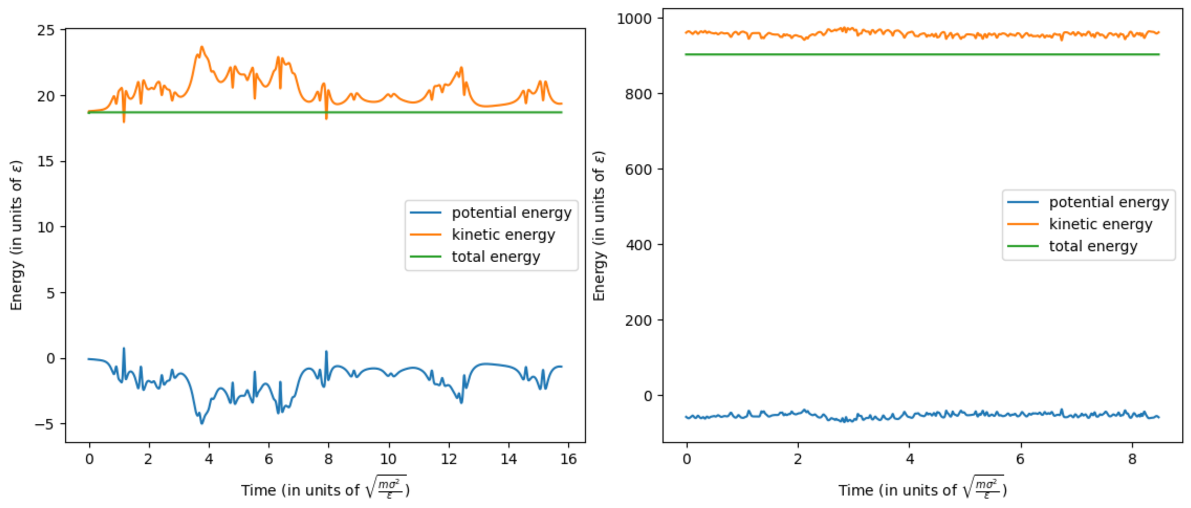

Fig. 1: These plots show the energies in our simulation over time. As you can see the total energy is conserved in both cases (left: 16 particles, right: 256 particles)

The resulting value of the specific heat per atom in dimensionless units was found to be $c_V = 1.50034 \pm 0.000366$ under the temperature of 100 K and in a box of side-size of 100 units.

The expected value of specific heat capacity under the constant volume of a mono-atomic gas in the high temperature limit is $c_V = 1.5$. This value falls within the uncertainty interval quoted above thus proving that the simulation gives reliable, physical outcomes.[10]

Mean Squared Displacement

The mean squared displacement (MSD) is a measure which looks at the average squared displacement of individual particles. The movement of the particles can be described by very short or longer timescales which can be split up into two regimes, the ballistic and diffusive. First we shall look into the ballistic regime which means that interactions are not (yet) relevant. This can be achieved either by looking at timescales which are small enough compared to the timescales of the force or by turning off the interactions completely. In the ballistic case the MSD grows quadratically with time as the particles does not "feel" any other particles (yet) which is described by the following formula:

$$\langle \Delta^2 x(t) \rangle = \langle v(0)^2 \rangle t^2 = 3v_0^2 t^2 \tag{14}$$

From this the constant factor $3v_0^2$ can be calculated in two different ways: as the mean initial velocity squared which in the case of the Maxwell-Boltzmann distribution is just the standard deviation squared and by fitting a second order polynomial to the curve for the mean squared displacement. To confirm this 256 particles were simulated without interaction and the MSD was calculated which can be seen in figure 2. For the constant factor we find: $\sigma^2 = 1.252$ from the Maxwell-Boltzmann distribution and from the fitted polynomial $v_o^2/3 = 1.250$. These values are less than 0.12 % apart which shows us that the simulation does the job.

To really confirm that this was not a lucky the simulation was run 15 times at different temperatures while all other parameters remain constant and for each temperature the mean initial velocity squared was calculated in two different ways and plotted in figure 3. In this figure one can very nicely see the linear dependence of the initial velocity squared with the temperature. The lines are almost perfectly on top of each other, the difference between the two methods was on average 0.1110% with a standard deviation of 0.005788%.

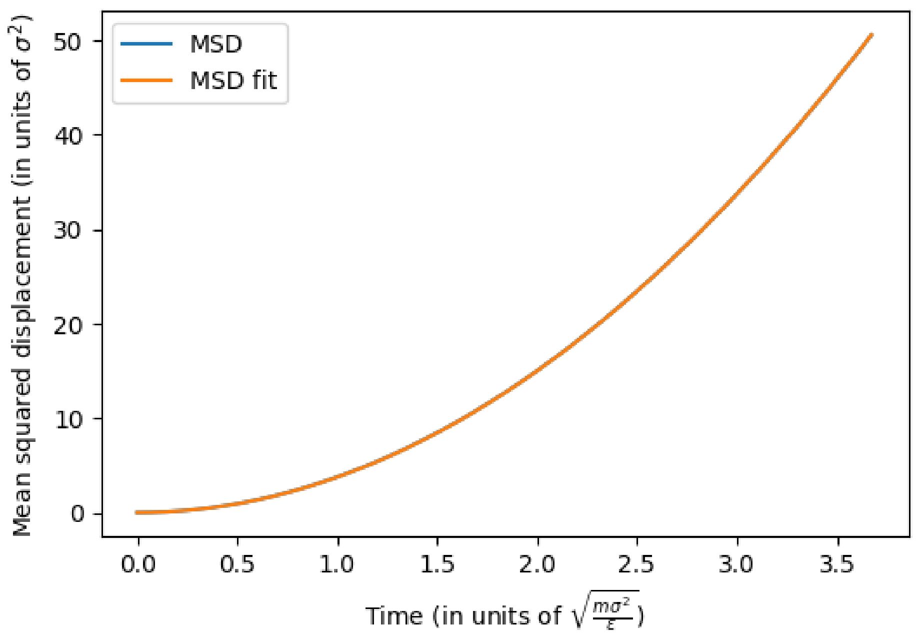

Fig. 2: Parabolic behaviour

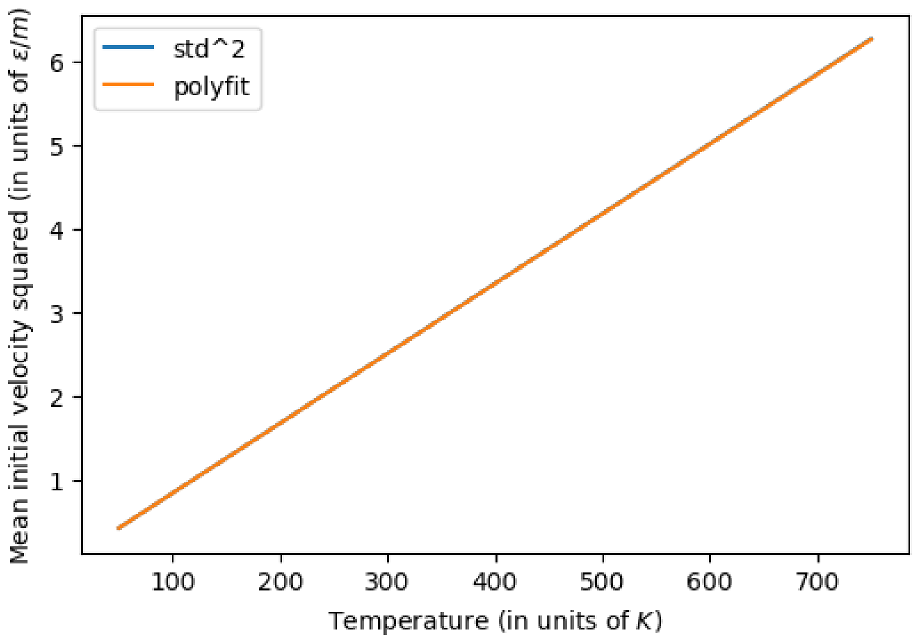

Fig. 3: Temperature dependence

Next the diffusive regime of simulated argon atoms was considered and compared to that of actual argon atoms. For timescales that are large enough, or densities that are high enough the interactions will have a large effect on the movement of the particles and the MSD will become linear.

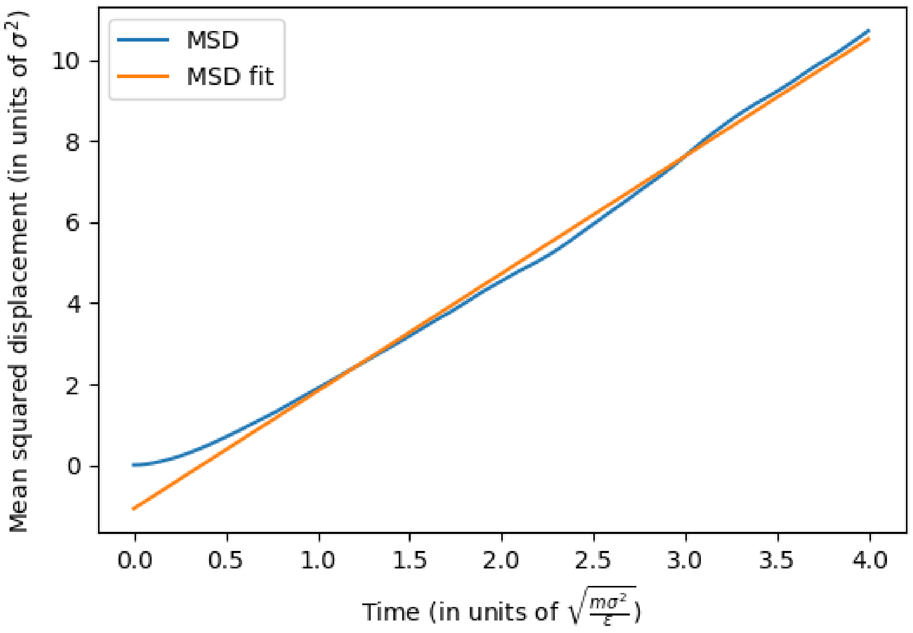

Fig. 4: Linear behaviour

From reference [11] we find that the diffusion constant for argon is $25.8 \times 10^{-9} m^2/s$ at 159.1 K and with a molar volume of 70.71 cm^3/mol. Converting these values to our non-dimensional units we find that this corresponds to 340 atoms in a box of 10x10x10 σ^3. For this example the MSD was calculated and can be seen in figure 4, the linear behaviour is clearly visible. To find the diffusion constant of our simulation a linear curve was fitted to the diffusive regime of the plot by ignoring the first 100 time-steps and using the Einstein relation (equation 15) we find the diffusion constant.

$$D = \lim_{t \to \infty} \frac{1}{6t} \langle \Delta^2 x(t) \rangle \tag{15}$$

The linear fit corresponding to figure 4 is very accurate as it has an $R^2 = 0.996696$, the diffusion constant we find is $D = 0.4838$. This value however is in units of σ, ϵ and m so we convert it back to SI units with the relation for the diffusion coefficient: $D_{si} = D_{nd} \times \sigma \sqrt{\frac{\epsilon}{m}}$ to find $D = 26.079 \times 10^{-9} m^2/s$, remember that we simulated argon atoms with $D = 25.8 \times 10^{-9} m^2/s$ which is spot on. Again we confirm that the code can reproduce physical observables, in this case the diffusion constant, from basic principals.

2.3 Code Efficiency

The simulation was run 20 times for a different number of particles and time-steps. The time it has taken to perform the simulations was measured and the results are summarised in the table below.

The numpy package was used to enhance the efficiency of the computations. Numpy has a feature which allows us to compute all elements of an array at once, instead of going one by one. It also make uses high efficiency C-code to perform the calculations.

It's expected that the most time consuming parts of the code will thus be any 'for loops' where we iterate the simulated particles. In our code we had one 'for loop' and another 'for loop' nested inside the first one. First loop iterates the number of simulation time-steps and the nested loop iterates the number of simulation dimensions (e.g. 3 iterations for 3D). It's therefore expected that the run-time of the code will be proportional to the number of time-steps multiplied by the number of simulation dimensions for a fixed number of molecules. Thus if we use 20 time-steps in 3D, a perfect simulation should be 100 times faster compared to a simulation with 2000 time-steps.

In the table below we see that for N=16 the simulation was 50 times faster for 20 time-steps and for N=256 it was 80 times faster than when 2000 steps were used. These are slower than the expected 100 times speed-up. This happens because there are parts of the code which also linearly depend on the number of dimensions, particles and time-steps. Each function carried out by the code will add to the time and will scale according to the value that it iterates. The exact time of going through a numpy array and how it scales with the number of array elements (number of molecules) isn't known and its analysis is beyond the material contained in this report, however we do show that the run-time scales linearly with the number of molecules simulated.

Knowing that the simulation of 16 particles in 3D over 20 time-steps takes 0.02 s we can approximate how long is should take to simulate 256 particles in 3D over 2000 time-steps. If we assume that the simulation of 2000 time-steps is 100 times slower than 20 time-steps and the run-time scales linearly with the number of particles. We find that simulation of 256 particles over 2000 time-steps should take approx. 32 seconds. This prediction is of the same order of magnitude as the true run-time of 42 s. Additional 10 seconds comes from other functions that do not contain 'for loop', their exact time-scaling with number of time-steps or molecules is difficult to predict.

number of particles	number of timesteps	**run-time (s)**	std (s)
16	20	**0.02**	0.003
16	2000	**1.1**	0.04
256	20	**0.5**	0.01
256	2000	**42.2**	0.9

3 Conclusion

By simulating a mono-atomic argon gas we have shown that molecular dynamics simulations are a powerful tool that allows us to simulate real physical systems and extract meaningful physical quantities.The Minimal image convention periodic boundary conditions were applied in order to make this system behave like a real physical system. The gas molecules were placed on an FCC lattice in a 3D box of fixed dimensions and temperature. They were then evolved in time via a Verlet method.

We have derived observables of the system. The diffusion coefficient was found to be $D = 26.079 \times 10^{-9} m^2/s$. The percentage error on this value compared to the value reported in the literature is only 1%. The specific heat capacity of the Argon atoms in the high temperature limit was determined to be $C_V = 1.50034 \pm 0.000366$. The upper bound of the percentage error on this value is only 0.1%. The total energy of the system was shown to be conserved therefore we conclude that the simulation obeys the physical laws and can be used to extract meaningful physical quantities of such system.

The efficiency of the code was investigated. It was shown that the most computationally demanding parts were the 'for loops' which iterate through arrays of values. The run-time of the code scales with the length of these arrays.

Our simulated system makes a great base point for more complicated simulations and can be easily extended to other physical systems, for example diatomic gas etc.

References

1. https://compphys.quantumtinkerer.tudelft.nl/proj1-moldyn-week1/
2. Alder, B. J. and Wainwright, Thomas K.,'Molecular Motions', Scientific American, vol.201, no.4, 1959, pp.113–126
3. Alder, B. J. and Wainwright, T. E., 'Velocity Autocorrelations for Hard Spheres',Physical Review Letters, vol.18, no.23, 1967, pp. 988–990
4. Rahman, A.,'Correlations in the Motion of Atoms in Liquid Argon',Physical Review, vol.136, no.2A, 1964
5. https://embnet.vital-it.ch/MD_tutorial/pages/MD.Part1.html
6. https://en.wikipedia.org/wiki/Molecular_dynamics
7. Thijssen, Johannes M. M. H, 'Computational Physics', Cambridge University Press, 2007, pp.214
8. ref https://en.wikipedia.org/wiki/Verlet_integration#Error_terms
9. Lebowitz, J. L. and Percus, J. K. and Verlet, L., 'Ensemble Dependence of Fluctuations with Application to Machine Computations',Physical Review, vol. 153, no.1, 1967, pp 250–254
10. https://www.concepts-of-physics.com/thermodynamics/specific-heats-cv-and-cp-for-monatomic-and-diatomic-gases.php
11. Fisher, R. A. and Watts, R. O.: Calculated Self-Diffusion coefficients for liquid argon, Aust. J. Phys., vol. 25 pp. 529-38 (1972)