Observing Thermodynamic Properties of Simulated Argon

A parameter study using Verlet's integration method

Francke, A. (4713672)
Glorie, M. (4279492)
Sangers, J. (4645197)

Applied Physics - Computational Physics
TU Delft

Abstract

Through molecular dynamics simulations of a mono-atomic Argon system, several system quantities such as the pair correlation function as well as the mean-squared displacement of the system were studied in this paper. The interactions between particles were modelled according to the Lennard-Jones potential and the time evolution of the system was modelled through the Velocity-Verlet numerical integration method. The resulting simulations show conservation of energy and analysis of the observables shows very good alignment with the expected results from literature. The results of the pair correlation and function and MSD allow for identification of the phase of our simulation through comparison to characteristic behaviour described in literature. The calculated diffusion coefficients for Argon in its liquid or gaseous phase do show large standard deviations, caused by the deviations in system temperature between different simulations.

Contents

Table of Contents **ii**

1 Introduction **1**

2 Methods **2**
- 2.1 General theory . . . 2
  - 2.1.1 Molecular dynamics simulation . . . 2
  - 2.1.2 Non-dimensionalisation . . . 2
- 2.2 Simulation implementation . . . 3
  - 2.2.1 Initial conditions . . . 3
  - 2.2.2 Finding system equilibrium . . . 4
- 2.3 Pair correlation function . . . 4
- 2.4 Diffusion coefficient . . . 5

3 Results **7**
- 3.1 Energy . . . 7
- 3.2 Pair correlation function . . . 9
- 3.3 Diffusion Coefficient . . . 10
  - 3.3.1 Liquid Argon . . . 10
  - 3.3.2 Gaseous Argon . . . 11
  - 3.3.3 Solid Argon . . . 12
- 3.4 Simulation Performance . . . 14

4 Conclusions **17**

References **18**

1 Introduction

Molecular dynamics simulations are a useful method to study the behaviour of various molecular systems. These simulations can become very complicated if we consider every different type of interaction between different types of molecules. This report aims to study a molecule which is limited in its complexity, namely a simulation of mono-atomic Argon in its liquid, gaseous and solid phases. This system is relatively easy to study because of the limited types of interactions Argon atoms have with each other and because the behaviour of Argon has been studied and documented well.

Chapter 2 describes the relevant theory underlying these interactions and explains the method of simulation used in our study. For a complete derivation of the theory we refer to [1] and [2]. We also introduce the quantities we wish to observe through our simulations, the pair correlation function and the diffusion coefficient, and go over the behaviour of these quantities as described in literature. Chapter 3 gives the results of our simulations and discusses the accuracy of these results by comparison to the expected behaviour and also discusses the performance of the used Python code. Finally, chapter 4 concludes the report by discussing points of improvement and giving recommendations for future research.

This project was performed as part of the Computational Physics course at the Technical University of Delft.

2 Methods

2.1 General theory

2.1.1 Molecular dynamics simulation

In order to analyse the behaviour of Argon atoms in different phases we simulate the interaction between two Argon atoms according to the Lennard-Jones potential. This potential is defined as,

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

and describes the the repulsive interaction between the molecules, which scales as r^{-12}, and the the attractive dipole interaction which scales as r^{-6}, where r is the distance between particles. In the case of Argon, $\epsilon/k_b = 119.8K$ and $\sigma = 3.405$ Ångstrom [3].

From Newton's second law of motion we can derive the movement of the particles which is described in as follows,

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

Solving this equation numerically requires use of a finite integration algorithm, for this purpose the Velocity-Verlet algorithm was chosen since simpler integration methods such as Euler methods do not conserve the total energy of the system. The Velocity-Verlet finite step algorithm is defined as follows[1],

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\mathbf{v}(t) + \frac{h^2}{2}\mathbf{F}(\mathbf{x}(t)), \tag{3}$$

$$\mathbf{v}(t+h) = \mathbf{v}(t) + \frac{h}{2}(\mathbf{F}(\mathbf{x}(t+h)) + \mathbf{F}(\mathbf{x}(t))). \tag{4}$$

Where h is our finite step size, and $\mathbf{F}(\mathbf{x}(t))$ the force on the particle as calculated from the Lennard-Jones potential.

The particles are simulated in a three-dimensional square box with periodic boundary conditions. Because of the effect of the periodic boundary conditions on the force, we calculate the forces on the particles based on the minimal image convention. This means we "extend" our box in all three dimensions with 26 additional 'image' boxes each containing the same number of particles at the exact same position, and we only consider the interaction between the atom and the closest of all 'image partners' of every other atom for calculating the force.

2.1.2 Non-dimensionalisation

To obtain numbers from our simulations which are easier to work with and less prone to round-off errors, we perform a non-dimensionalisation of our system. To this end we define characteristic scales for our position r as σ and for the energy we define the characteristic scale as ϵ. As such we we can write equation (1) as follows.

$$\tilde{U}(\tilde{r}) = 4(\tilde{r}^{-12} - \tilde{r}^{-6}) \tag{5}$$

Where $\tilde{r} = r/\sigma$ and $\tilde{U} = U/\epsilon$. Defining the characteristic time scale as $\sqrt{\frac{m\sigma^2}{\epsilon}}$ we can rewrite equation (2) as follows.

$$\frac{d^2\tilde{\mathbf{x}}}{d\tilde{t}^2} == -\tilde{\nabla}\tilde{U}(\tilde{\mathbf{x}}) \tag{6}$$

From our definition of the characteristic time scale it follows that the particles in our simulation will move a distance of order σ in a time step of order 1. In our simulations we therefore chose our step size to be of order 10^{-2} to 10^{-3} as to avoid events where the particles end up unrealistically close to one another.

2.2 Simulation implementation

2.2.1 Initial conditions

Initial position
The particles are simulated in a square box with sides of length L.
For the initial positions of the particles, the particles are arranged in an fcc lattice. The locations of the lattice points are described as,

$$\mathbf{R}_{n_1,n_2,n_3} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3, \tag{7}$$

where $|\mathbf{a}_1| = |\mathbf{a}_2| = |\mathbf{a}_3| = a$ and $\mathbf{a}_i = a\,\mathbf{e}_i$ for $i = 1, 2, 3$. All the positions of the particles in the fcc lattice are described as,

$$\begin{aligned}
\mathbf{R}^{\text{corner}}_{n_1,n_2,n_3} &= \mathbf{R}_{n_1,n_2,n_3},\\
\mathbf{R}^{\text{face−xy}}_{n_1,n_2,n_3} &= \mathbf{R}_{n_1,n_2,n_3} + \frac{a}{2}[1,1,0],\\
\mathbf{R}^{\text{face−xz}}_{n_1,n_2,n_3} &= \mathbf{R}_{n_1,n_2,n_3} + \frac{a}{2}[1,0,1],\\
\mathbf{R}^{\text{face−yz}}_{n_1,n_2,n_3} &= \mathbf{R}_{n_1,n_2,n_3} + \frac{a}{2}[0,1,1].
\end{aligned} \tag{8}$$

The particles are simulated in a square box with sides of length L, where the number of unit cells along one side is $N_{\text{cell}} = \frac{L}{a}$. Because each unit cell contains four particles, the total number of particles simulated is $N = 4\,N^3_{\text{cell}}$.

Initial velocity
The initial velocities of the particles are chosen such that they have a Maxwell-Boltzmann distribution (P_{MB}). The initial velocities v_x, v_y and v_z will therefore be chosen such that they obey a Gaussian distribution (P_{G}), shown in equation (9),

$$P_{\text{G}}(v_i)dv_i = \sqrt{\frac{m}{2\pi k_{\text{B}}T}}e^{-\frac{1}{2}\frac{m}{k_{\text{B}}T}v_i^2}dv_i = \frac{1}{\sqrt{2\pi\tilde{T}}}e^{-\frac{1}{2}\frac{\tilde{v}_i^2}{\tilde{T}}}d\tilde{v}_i = P_{\text{G}}(\tilde{v}_i)d\tilde{v}_i, \tag{9}$$

with the subscript i being $\{x, y, z\}$. The mean and the standard deviation of the Gaussian distribution are therefore 0 and $\sqrt{\tilde{T}}$ respectively. The Maxwell-Boltzmann distribution will therefore become,

$$P_{\text{MB}}(\tilde{v})d\tilde{v} = 4\pi\left(\frac{1}{2\pi\tilde{T}}\right)^{\frac{3}{2}}\tilde{v}^2 e^{\frac{\tilde{v}^2}{2\tilde{T}}}d\tilde{v} \tag{10}$$

Finally, to insure there is no particle drift, the total momentum of the system is set to zero, i.e. $\sum_{i=1}^{N} \mathbf{v}_i = 0$.

2.2.2 Finding system equilibrium

The equipartition theorem states that per momentum degree of freedom the kinetic energy of a particle at temperature T, has an energy of $\frac{1}{2}k_\mathrm{B}T$. Therefore, the total kinetic energy, E_kinetic, of a system with N particles and γ momentum degrees of freedom is,

$$E_\mathrm{kinetic} = \frac{\gamma}{2} N k_\mathrm{B} T. \tag{11}$$

The equipartition theorem will therefore allow us to determine the temperature of the system from the total kinetic energy of the system. However, as the total moment of the system is conserved by setting the total momentum of the system to zero to prevent particle drift, there are γ degrees of freedom less. Equation (11) then becomes,

$$E_\mathrm{kinetic} = \frac{\gamma}{2}(N-1) k_\mathrm{B} T. \tag{12}$$

When the simulation starts the system has to find its equilibrium, which tends to be at a different temperature from the temperature set at the initial conditions. The velocity of the particles therefore have to be rescaled in order for the system to have the intended temperature. As $E_\mathrm{kinetic} = \sum_i \frac{m}{2} |\mathbf{v}_i|^2 = \sum_i \frac{m}{2} v_i^2$, the scaling factor λ for the velocities is,

$$\lambda = \sqrt{\frac{(N-1)\gamma k_\mathrm{B} T}{\sum_\mathrm{i} m v_\mathrm{i}^2}} = \sqrt{\frac{(N-1)\gamma \tilde{T}}{\sum_\mathrm{i} \tilde{v}^2}}. \tag{13}$$

The dimensionless temperature $\tilde{T}$ is defined as $T\frac{k_\mathrm{B}}{\epsilon}$.
After the velocities are changed, the simulation is started again to find the new equilibrium. If the equilibrium temperature is within a specified range of the intended temperature the velocities are not rescaled anymore.

2.3 Pair correlation function

The pair correlation function, or sometimes also called the radial distribution function, is a function that expresses the probability of finding a particle at a certain distance from another particle. The function is usually measured trough scattering experiments [2], in this report the pair correlation function of argon will be determined by means of a simulation. Since the simulation only consists of Argon particles it is homogeneous. For a homogeneous system the pair correlation is defined as:

$$\begin{aligned} g(\Delta \underset{\rightarrow}{r}) &= \frac{V}{N(N-1)} \left\langle \int \mathrm{d}^3 r' \sum_{i \neq j} \delta(\underset{\rightarrow}{r}' - \underset{\rightarrow}{r}_i)\delta(\underset{\rightarrow}{r}' + \Delta \underset{\rightarrow}{r} - \underset{\rightarrow}{r}_j) \right\rangle, \\ g(r) &= \frac{2V}{N(N-1)} \frac{\langle n(r) \rangle}{4\pi r^2 \Delta r}, \end{aligned} \tag{14}$$

with $\langle n(r) \rangle$ a histogram of particles within a distance range $[r, r+\Delta r]$.
To calculate the pair correlation function, only the distances between particles are needed. These

can be computed from the global positions at every time step. The positions are recorded after the system has been initialised in a state and has been equalised to a specified temperature. Since the pair correlation function is a statistical average a lot of particles or time steps are needed. The simulation of gaseous Argon will therefore need to run for a significant amount of time to gathered the required time steps since only a few particles will be simulated. For solid Argon it is expected to see well-defined peaks in the pair correlation function since all atoms should be on a face-centred cubic lattice with well defined distances between particles. Liquid Argon however will not show such peaks since the particles will be free to move around.

2.4 Diffusion coefficient

Another interesting quantity to observe is movement of the atoms inside our square box. A method to do this is by computing what is known as the mean-squared displacement (MSD). The MSD is defined as,

$$\langle \Delta^2 \mathbf{x}(t) \rangle = \langle [\mathbf{x}(t) - \mathbf{x}(0)]^2 \rangle \tag{15}$$

In which $\mathbf{x}(t)$ denotes the particle position at time t, and $\mathbf{x}(0)$ the position of the particle at a chosen reference point. All equations in this section are in dimensionless form, the tilde are omitted for the sake of simplicity. Since we are interested in the evolution of the MSD over time we do not average over time for this observable, but instead we average over the number of particles in our simulation.

In our simulation we store the particle positions at each time step from which we can subtract the particles position at the reference time $t = 0$ to obtain the displacement at time t. The problem we encounter when calculating the displacement this way is due to the periodic boundary conditions used in our simulations. Namely when a particle crosses one of the boundaries and appears on the other side of the box there will be a discontinuity in its displacement with respect to the reference point at that time step, and the magnitude of the displacement will always be limited by our box size. What we instead do to calculate the MSD is make use of the stored forces and velocities at each time step to directly calculate the displacement between two subsequent time steps in accordance to the Velocity-Verlet algorithm we use to integrate the ODE of the positions. From equation (3) we derive that the displacement between two subsequent time steps is given by,

$$\mathbf{x}(t+h) - \mathbf{x}(t) = h\mathbf{v}(t) + \frac{h^2}{2}\mathbf{F}(\mathbf{x}(t)) \tag{16}$$

We can then find the displacement of the particle at time t with respect to the reference point $\mathbf{x}(0)$ by taking the sum of all the subsequent displacements from $t = 0$ up to the time t. We take the square of the displacement and average over all the particles to finally calculate the MSD.

In all three phases, liquid, gas, and solid, we expect the MSD to grow as t^2 for timescales short when compared to the typical collision time [2]. Because of the collisions slowing down the particles, after a longer period of time the growth of the MSD will start to slow down going from quadratic growth with t towards linear growth instead. This will happen more quickly in the liquid phase when compared to the gas phase because of the shorter collision time. In this linear regime we can use the Einstein relation to estimate the diffusion coefficient to the slope of the MSD [2],

$$\hat{D}(t) = \frac{1}{6t}\langle \Delta^2 \mathbf{x}(t) \rangle \tag{17}$$

Where taking the limit $t \to \infty$ will result in $\hat{D} \to D$.

In the case of solid Argon the MSD will instead plateau resulting in an effective slope of 0, indicating minimal diffusion when compared to the liquid and gas phases.

3 Results

3.1 Energy

One of the qualitative arguments used to determine whether our simulation is correct is to determine if energy is conserved at different conditions. In figure 1 a), b) and c) the total energy per particle is shown during the simulation. Although it fluctuates, the fluctuations are less than 1% of the total and the total energy does not diverge from the mean total energy. Therefore, we conclude that the energy is conserved.

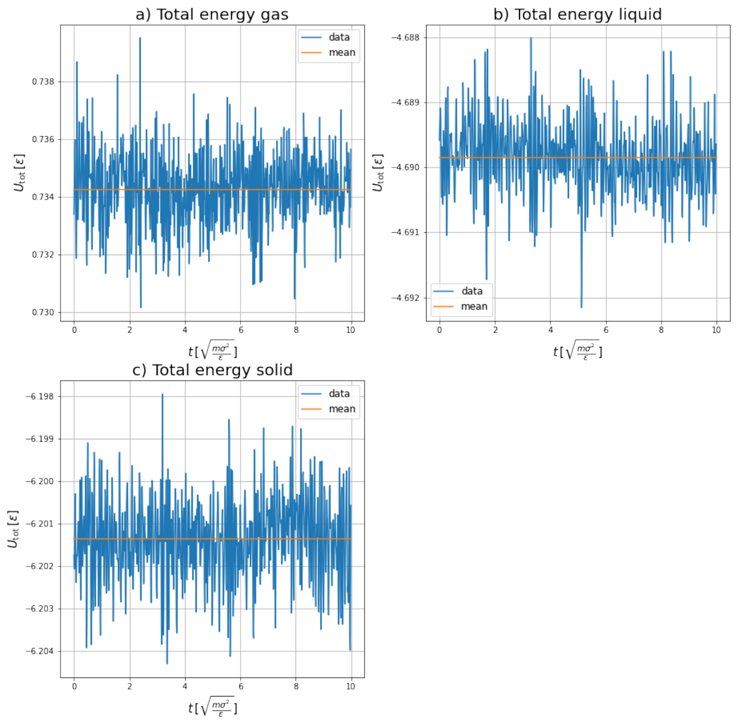


Figure 1: The average total energy of an Argon particle (for 108 simulated particles), through time (blue lines), and the mean total energy (red lines) of three simulations. a) In the gas phase at $T = 2.992 \pm 0.012 \frac{\epsilon}{k_{\mathrm{B}}}$ and $\rho = 0.3\sigma^{-3}$. b) In the liquid phase at $T = 1.057 \pm 0.0045 \frac{\epsilon}{k_{\mathrm{B}}}$ and $\rho = 0.8\sigma^{-3}$. c) In the solid phase at $T = 0.817 \pm 0.0029 \frac{\epsilon}{k_{\mathrm{B}}}$ and $\rho = 1.2\sigma^{-3}$. The uncertainty of the temperature has been determined using the autocorrelation method.

Another qualitative method to analyse the correctness of our simulation is to look at the velocity distribution of the particles at the end of the simulation. As mentioned in section 2.2.1 the velocity distribution of the particles should obey the Maxwell-Boltzmann distribution (equation (10)). The measured velocity distribution for the gas-, liquid-, and solid phase are shown in figure 2 a), b) and

c) respectively. They show that at the end of the simulation the particles still obey the Maxwell-Boltzmann distribution.

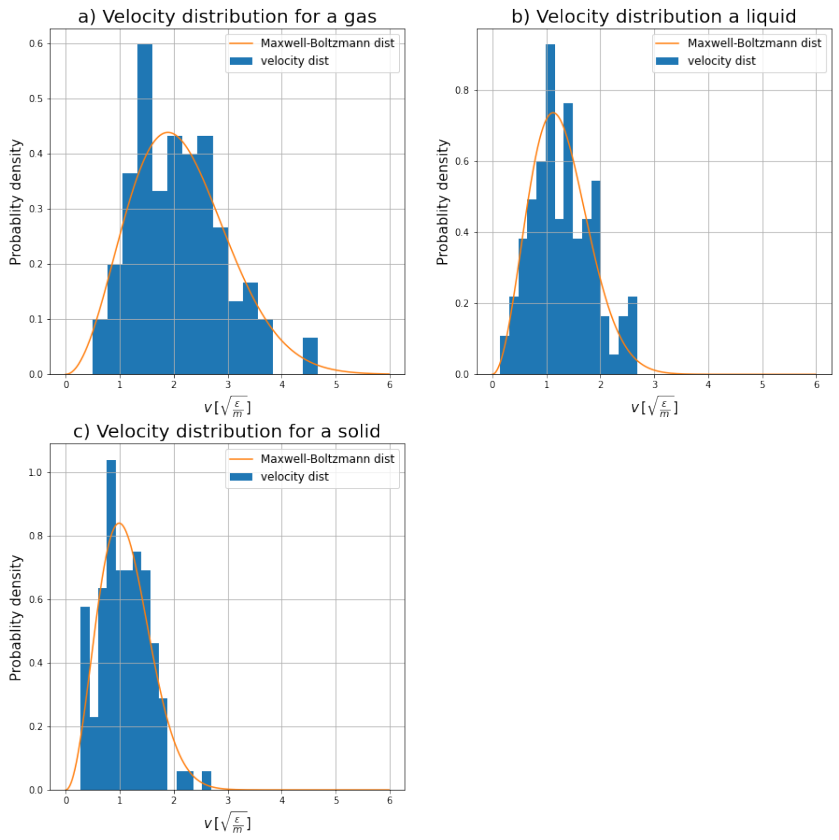


Figure 2: The velocity distribution at the end of the simulation for a 108 particle system, and the Maxwell-Boltzmann distribution (equation (10)) at the calculated determined temperature. a) For Argon gas at $T = 2.992 \pm 0.012\frac{\epsilon}{k_B}$ and $\rho = 0.3\sigma^{-3}$. b) For liquid Argon at $T = 1.057 \pm 0.0045\frac{\epsilon}{k_B}$ and $\rho = 0.8\sigma^{-3}$. c) For solid Argon at $T = 0.817 \pm 0.0029\frac{\epsilon}{k_B}$ and $\rho = 1.2\sigma^{-3}$.

Finally the potential energy for liquid Argon at different temperatures has been determined and is shown in figure 3. In table 8.1 in [2] the thermodynamic quantities of a Lennard-Jones liquid are provided. At $\rho = 0.80\sigma^{-3}$, $T = 1.010\frac{\epsilon}{k_B}$ and a pressure of $\frac{\beta P}{\rho} = 1.31$, the potential energy of a particle was determined to be -5.271ϵ. This deviates from the potential energy we have determined. However, as the pressure of our system is unknown we couldn't determine if these values are accurate.

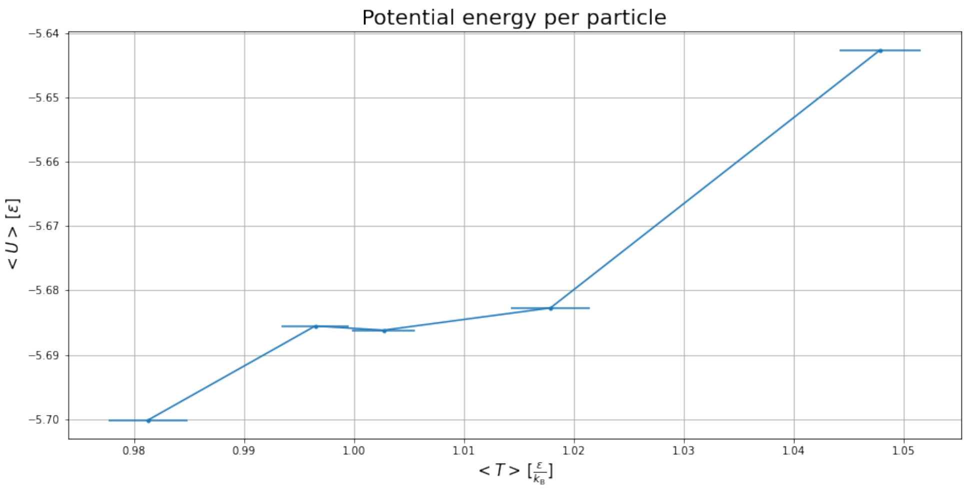


Figure 3: The potential energy per particle (for 108 simulated particles) for liquid Argon at different temperatures at a density of $\rho = 0.8\sigma^{-3}$.

3.2 Pair correlation function

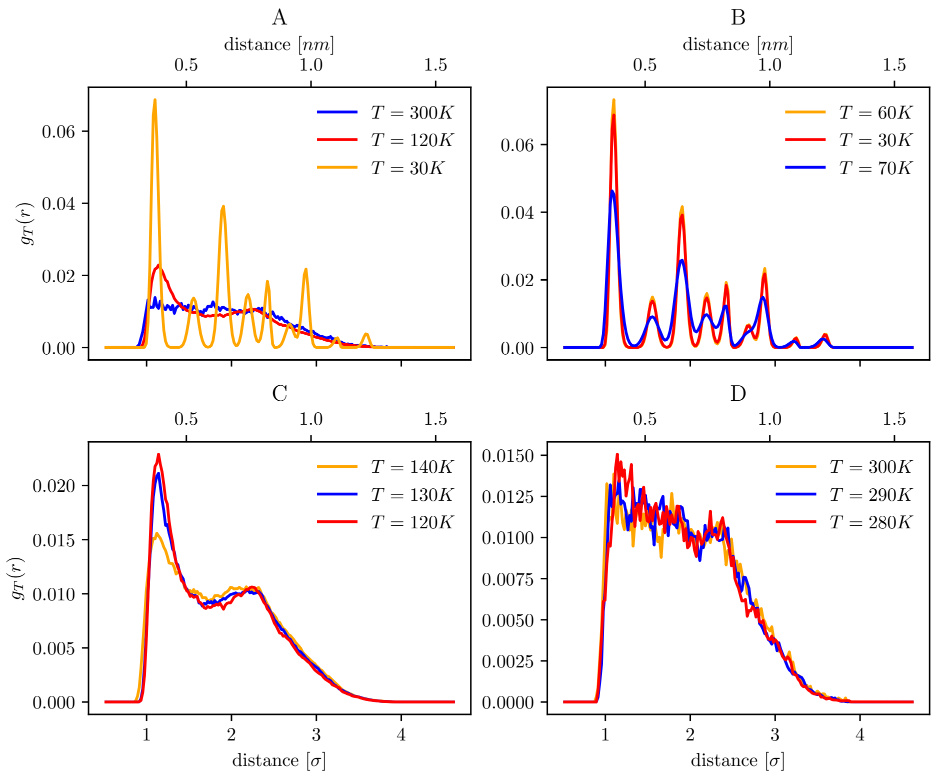


Figure 4: Pair correlation functions of Argon at different phases. (A) The pair correlation function plotted for Argon in a gaseous, liquid and solid form denoted by blue, red and orange solid lines respectively. (B) Shows the pair correlation function of solid Argon for 108 atoms in a box with volume $V = (3\tilde{\alpha})^3$. (C) Shows the pair correlation function of liquid Argon at several temperatures simulated with 30 atoms in box with volume $V = (3\tilde{\alpha})^3$. (D) Show the Argon pair correlation function for gaseous Argon at several temperatures simulated with 5 atoms in a box with volume $V = (3\tilde{\alpha})^3$. α is the literature reference lattice spacing, here taken to be dimensionless $\tilde{\alpha} = \alpha/\sigma$.

The results for the plots in Figure 4 were gathered differently for the gaseous and liquid state and the solid state. For the simulation of solid Argon, the system started with all its atoms perfectly spaced in a face-centred-cubic lattice structure and starting velocities in all directions drawn from a Gaussian distribution to randomise the initial velocity vectors. After initialisation, the system would be equalised to roughly the required temperature by means of a velocity scaling function, this function would stop equalising if the average temperature of the system over the last 10 time steps was within $1K$ of the requested temperature. The liquid and gaseous systems were prepared similarly except for the initial positions which were drawn from a uniform distribution.

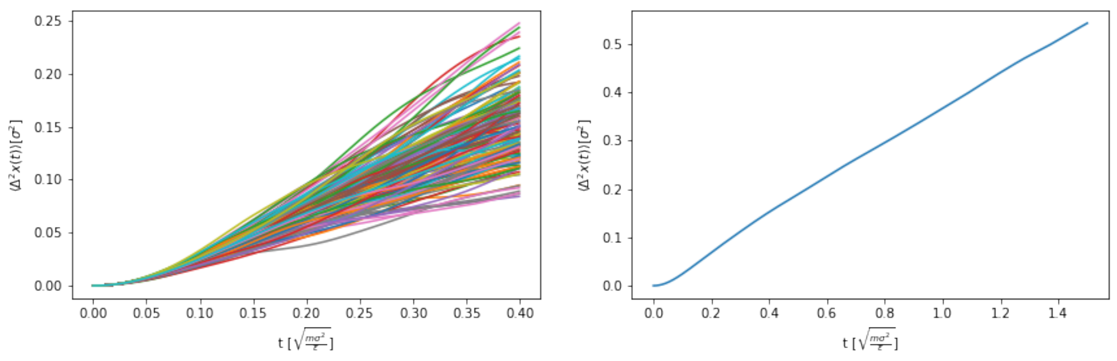
Comparing the pair correlation function of the different states of Argon in Figure 4A a few differences can be seen. The nearly discrete peaks of the solid Argon are due to the constant lattice spacing between atoms whereas both the gaseous and liquid states have a smoother distribution. All states however share the similarity that the pair correlation function is zero in the region of high repulsion at $r \leq \sigma$. For solid Argon the width of the peaks in the pair correlation function grows for greater temperature, as can be seen in Figure 4B. This is to be expected since for higher temperatures the individual atoms carry more kinetic energy and have therefore more ability to deviate from their equilibrium position. Thus widening the range where the atom can be found and decreasing the peak since the particle spends less time in the equilibrium position. Still looking at Figure 4B at the $70K$ trace a noticeable overlap between to peaks starts occurring at $\sigma \approx 3$. Figure 4C shows the pair correlation function for liquid Argon. It still retains the large peak at $\sigma = 1$ indicating that the atoms are still bouncing off of each other regularly. The highest temperature trace in the liquid state at $140K$ shows a plateauing between the two primary peaks, this feature seems to indicate the liquid starts acting more as a gas as is evident by its pair correlation function starting to look more like that of the gases in Figure 4D. In this figure the pair correlation functions for any temperature tend to be more flat as the simulated volume is relatively empty and not a lot of interactions occur. The spikiness of the graph is also due to the small sample of particles used to run the simulation. Both the results gathered for Argon in the solid state and that of Argon in the liquid state (Figure 4B and 4C respectively) align well with those previously reported in literature [4] [5].

3.3 Diffusion Coefficient

In order to simulate Argon in the three different phases, the simulations were initialised with the following parameters: $\tilde{T}$=1 and $\tilde{\rho}$=0.8 for liquid Argon, $\tilde{T}$=3 and $\tilde{\rho}$=0.3 for gaseous Argon, and $\tilde{T}$=0.5 and $\tilde{\rho}$=1.2 for solid Argon [2]. The this was done by choosing the volume of our box in such a way that $\frac{N}{V} = \tilde{\rho}$, where N is the number of particles we wish to simulate, in the case of liquid and gaseous Argon the number of particles was set to 30, while solid Argon was simulated using 32 particles in order to initialise them in a FCC lattice. The system was equalised to roughly the desired temperature, after which the particle movements were simulated for 1500 time steps of size 10^{-3} in the case of liquid and solid Argon, and 4000 time steps of size 10^{-2} in the case of gaseous Argon. This was done because the linear behaviour of the MSD for gaseous Argon is expected to set in at a later time. Increasing the time step size will unfortunately result in a decrease in the accuracy of our results.

3.3.1 Liquid Argon

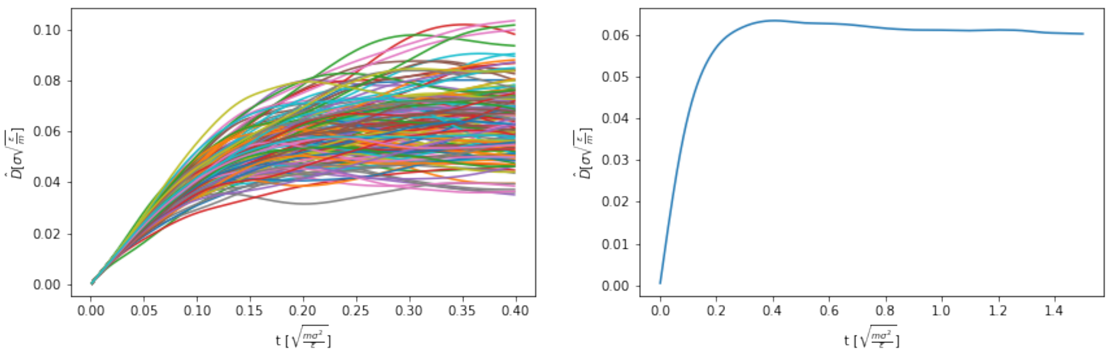
The dimensionless MSD of the liquid Argon simulations were calculated and are plotted in the figure below as a function of time.



(a) **(b)**

Figure 5: **(a)** 150 time evolutions of the dimensionless MSD of liquid Argon. **(b)** Averaged time evolution of the dimensionless MSD of liquid Argon over 150 simulations.

Figure 5(a) was plotted for $\tilde{t} \in [0, 0.4]$ to better visualise the behaviour of the MSD for small $\tilde{t}$. From this figure we can indeed see that for short times the MSD appears to grow quadratically with $\tilde{t}$, while after a longer period of time the linear behaviour becomes apparent. This behaviour is more clearly visualised in the figure below.



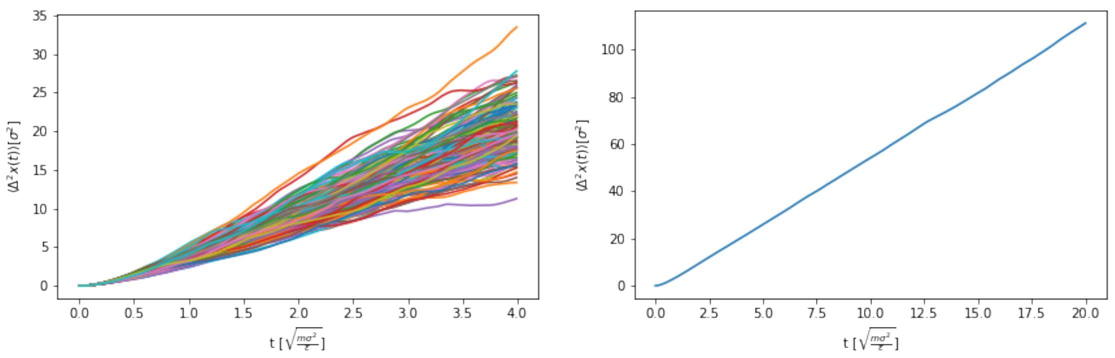
(a) **(b)**

Figure 6: **(a)** 150 time evolutions of the dimensionless estimate of D of liquid Argon. **(b)** Averaged time evolution of the dimensionless estimate of D of liquid Argon over 150 simulations.

We clearly see that after $\tilde{t}$=0.2 the estimate for $\tilde{D}$ starts to plateau which indicates linear growth of the MSD. We estimate the value of $\tilde{D}$ by taking the average of $\hat{D}(\tilde{t})$ at the end time of our simulation $\tilde{t}$=1.5. Since we have 150 independent realisations of $\tilde{D}$ we can determine the error by the square root of the variance in $\hat{D}(\tilde{t})$ at $\tilde{t}$=1.5. This resulted in a value of $\tilde{D} = 0.06 \pm 0.02\sigma\sqrt{\frac{\epsilon}{m}}$.

3.3.2 Gaseous Argon

The dimensionless MSD of the gaseous Argon simulations were calculated and are plotted in the figure below as a function of time.

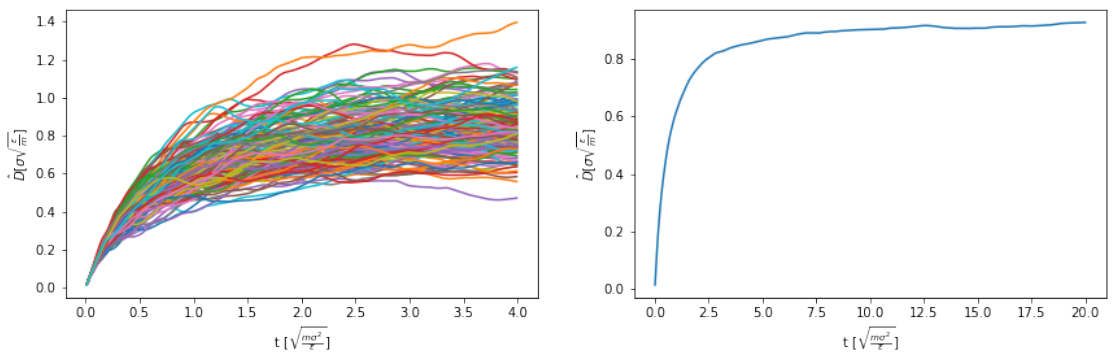


(a) **(b)**

Figure 7: **(a)** 150 time evolutions of the dimensionless MSD of gaseous Argon. **(b)** Averaged time evolution of the dimensionless MSD of gaseous Argon over 150 simulations.

We see quite a difference in the behaviour of the MSD when comparing the liquid and gas phases; not only does the MSD of gaseous Argon grow a lot quicker, which makes sense since the velocities of the particles are higher, we also see that the MSD stays quadratically dependent on $\tilde{t}$ far longer.

We again visualise this by plotting $\hat{D}(\tilde{t})$.



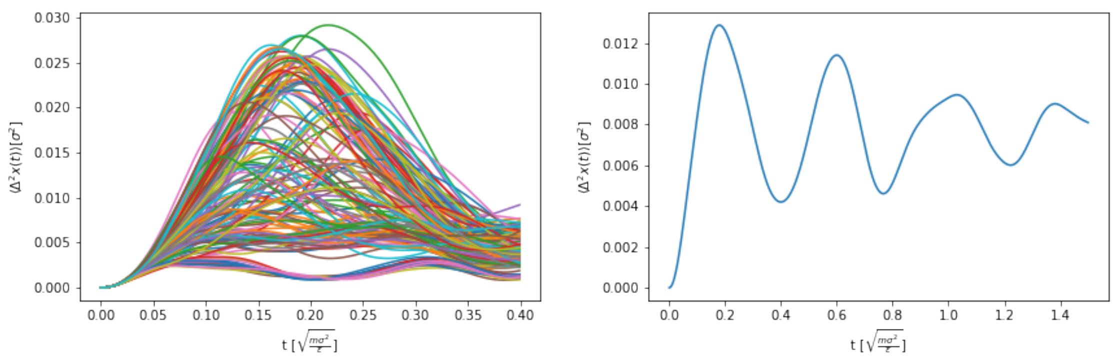
(a) **(b)**

Figure 8: **(a)** 150 time evolutions of the dimensionless estimate of D of gaseous Argon. **(b)** Averaged time evolution of the dimensionless estimate of D of gaseous Argon over 150 simulations.

We can see $\hat{D}(\tilde{t})$ converging to a constant value as $\tilde{t}$ increase. We again estimate the diffusion coefficient by the average value if $\hat{D}(\tilde{t})$ at $\tilde{t}$=4, which was calculated to be $0.9 \pm 0.1\sigma\sqrt{\frac{\epsilon}{m}}$.

3.3.3 Solid Argon

The dimensionless MSD of the solid Argon simulations were calculated and are plotted in the figure below as a function of time.



(a) **(b)**

Figure 9: **(a)** 150 time evolutions of the dimensionless MSD of solid Argon. **(b)**Averaged time evolution of the dimensionless MSD of solid Argon over 150 simulations.

What is noteworthy is the huge fluctuations in where the MSD ends up plateauing. Plotting the average MSD of each simulation against average system temperature of each simulation reveals why this is.

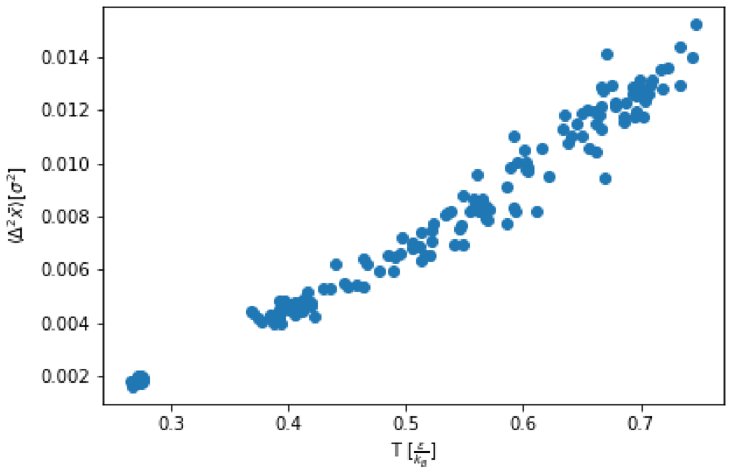


Figure 10: Average MSD of each simulation against average system temperature of each simulation.

Not only do we that the average system temperature deviating from the desired temperature of $\tilde{T} = 0.5$, we also see that there is a strong correlation average system temperature and the average MSD. This makes sense of course since a higher temperature means have a higher kinetic energy and in this case larger fluctuations. It seems in many of the simulations the system did not equalise at the desired temperature. What we can see however is that in each case the MSD quickly starts to oscillate around a plateau value, which is the characteristic behaviour we expect from solid Argon.

For all three phases we see quite large fluctuations in the MSD values between the 150 simulations.

As discussed before this is due to the large differences in the temperatures at which the simulations equalised. Calculating the average system temperature over 150 simulations reveals that the average system temperature is close to the desired system temperature. We therefore expect the averaged diffusion coefficients to be accurate however we do see they are not very precise with large standard deviations.

3.4 Simulation Performance

For analysing the performance of our Python code the cProfile python module was used. A test case was set-up were the profiler would run a simulation of $N = 108$ atoms structured in a FCC lattice for $h = 200$ steps, the results were sorted by total time spent running the function and have been listed below. It took almost 24 seconds to run the simulation with the most time being spent in calculating the forces between two atoms in the `compute_force_potential` function. This is however not the only culprit, the `image_forces_potentials` function which is called every time step and calculates the force on every particle by looping over them and calling `compute_force_potential` spends 3.8 seconds for only 200 calls while not computing anything itself. Below the profiler printout the `image_forces_potentials` function has been printed. Looking at the code the the problem becomes clear, the double `for`-loop. Having to loop over every particle combination makes this implementation for N particles run at $\mathcal{O}(N^2)$ time. The double `for`-loop is necessary however to loop over all particle pairs and is already sped up by realising the force on one particle is opposite of that on the other, imaged particle or not.
Turning our attention back to the `compute_force_potential` function, which is printed below the previous function, it is clear that nothing can be done in pure Python to further optimise the function. Since the two "slow" functions only use simple datatypes and NumPy functions which are both supported by Numba's just-in-time compiler it is possible to encapsulate the functions in Numba's `@njit` decorator to compile the code to a lower level language to speed it up. Looking at the last profiler output it is clear that for the same simulation parameters there is a $\approx 94\%$ reduction in computation time.

```
         20827606 function calls (20827006 primitive calls) in 23.887 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
  1155600    9.444    0.000   19.391    0.000 skeleton.py:491(compute_force_potential)
      200    3.818    0.019   23.210    0.116 skeleton.py:524(image_forces_potentials)
  2311800    3.194    0.000    3.194    0.000 {method 'reduce' of 'numpy.ufunc' objects}
  2311800    1.910    0.000    5.920    0.000 fromnumeric.py:70(_wrapreduction)
2313600/2313000    1.168    0.000    8.854    0.000 {built-in method numpy.core._multiarray_umath.im
  1155600    0.923    0.000    4.025    0.000 fromnumeric.py:2111(sum)
  1155600    0.669    0.000    3.378    0.000 fromnumeric.py:2256(any)
  2311800    0.602    0.000    0.602    0.000 fromnumeric.py:71(<dictcomp>)
```

```python
def image_forces_potentials(lookup_arr: np.ndarray) -> np.ndarray:
    """
    Function to calculate the net force on each particle

    Parameters
    ----------
    lookup_arr : np.ndarray
        array of location of closest particles
        intepretation:
            Element (i,j,k) represents location of particle k closest to particle j in dimension i

    Return
    ------
    array_forces : np.ndarray
        net force on each particle
        intepratation:
            Element (i,j) represents the net force on particle i of dimension j
    """
    (dims, part_amnt, tmp) = lookup_arr.shape

    array_forces = np.zeros((part_amnt, dims))
    array_potentials = np.zeros(part_amnt)

    for parti in range(part_amnt):
        icoords = lookup_arr[:,parti,parti]
        for partj in range(parti+1, part_amnt):
            jcoords = lookup_arr[:,parti,partj]
            force, pot = compute_force_potential(icoords, jcoords)
            array_forces[parti] += force
            array_forces[partj] -= force
            array_potentials[parti] += pot

    return array_forces, array_potentials
```

```python
def compute_force_potential(part_one_pos: np.ndarray,
                  part_two_pos: np.ndarray) -> Tuple[np.ndarray,np.ndarray]:
    """Function that computes the force between two particles abiding
    a Lennart-Jones potential and returns its magnitude in the x- and y-direction

    Parameters
    ----------
    part_one_pos : np.ndarray
        numpy array containing the coordinates of the first particle
    part_two_pos : np.ndarray
        numpy array containing the coordinates of the second particle
    Returns
    -------
    Tuple[np.ndarray, np.ndarray]
        numpy array of the force on particle two due to particle one in the x-
        and y-direction
    """
    #distances are calculated from particle one to particle two:
    distance_vector = part_two_pos - part_one_pos
    dist_magn    = np.sqrt(np.sum(distance_vector**2))

    r       = dist_magn
    prefctr = 24/(r**2)*((1/r)**6 -2*(1/r)**12)
    force  = 1*distance_vector*prefctr
    E_pot = 4*((r) ** -12 - (r) ** -6)

    if np.any(np.isnan(force)):
        raise RuntimeError("force between particles is nan.")


    return force, E_pot
```

```
      26406 function calls (25806 primitive calls) in 1.293 seconds

Ordered by: internal time

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   200    0.588    0.003    0.588    0.003 skeleton.py:528(image_forces_potentials)
   600    0.239    0.000    0.239    0.000 {method 'reduce' of 'numpy.ufunc' objects}
   200    0.215    0.001    0.529    0.003 skeleton.py:329(closest_images)
     1    0.176    0.176    1.293    1.293 skeleton.py:278(Verlet_integrate_images)
```

4 Conclusions

Qualitative assessment of our molecular dynamics simulation was done by calculation of the total energy of our system. Conservation of this quantity as was shown gives credibility to the correctness of our simulations. In addition we observe that the velocity distribution of our particles at the end of the simulation obeu the Maxwell-Boltzmann as expected.
Both the observables, the pair correlation function as well as the mean-squared displacement, that we studied in this report are useful quantities for assessing which phase, liquid, gas or solid, our molecular system is in. The results for both observables align well with the expected behaviour that is reported in literature. With the pair correlation function forming distinct peaks in the case of solid Argon with the peaks widening and starting to overlap as the temperature, and thus the kinetic energy of the particles, increases.

For the MSD we also observed qualitative behaviour as described in literature, even though our separate simulations were not all equalised correctly at the desired temperature, leading to large deviations between different simulations. On average we did observe quadratic growth of the MSD with time for short times in the case of liquid and solid Argon, and longer times in the case of gaseous Argon. After which the MSD for both liquid and gaseous Argon settled into a regime of linear growth with time, and plateaued for solid Argon. Estimating the diffusion coefficient from this data as the slope to the MSD for our last time step we found a value of $\tilde{D} = 0.06 \pm 0.02\sigma\sqrt{\frac{\epsilon}{m}}$ for liquid Argon, and $0.9 \pm 0.1\sigma\sqrt{\frac{\epsilon}{m}}$ for gaseous Argon, for solid Argon we found an effective slope of 0 as expected. The the measurement of these coefficients are not very precise due the large deviations in the system temperature. What these deviations did show us however is the strong linear dependence of the plateau value of the solid Argon MSD on the system temperature.

Through encapsulation of the simulation functions in Numba's @njit decorator we were able to drastically cut down on the computation times of our simulations. This means simulations with a higher number of particles or simulations over a longer period of time, making sure the system equalises to the desired temperature, would be possible. This would add greatly to the precision of our results however, in view of time these simulations were not repeated since our results already showed a very good alignment with expectations and can certainly be used to identify which phases our Argon simulations are in.

References

[1] D. Levesque, L. Verlet, and J. Krkijarvi, “Computer "’experiments"’ on classical fluids. iv. transport properties and time-correlation functions of the lennard-jones liquid near its triple point,” *pra*, vol. 7, pp. 1690–, 01 1973.

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

[3] M. Wimmer and A. Akhmerov, “Computational Physics ap3082,” 2022.

[4] S. Franchetti, “Radial distribution functions in solid and liquid argon,” *Il Nuovo Cimento B Series 11*, vol. 26, pp. 507–521, Apr. 1975.

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