# School of Physics and Astronomy


# Phase Diagram of Argon

## Computer Modelling Project Report

Larisa Dorman-Gajic
Project Partner: Margot van Laar March 2019

**Abstract**

This project aimed to simulate atoms of argon in three states; gas; fluid and solid using the simulator VMD. The code describes the N-body system adhering to the Lennard-Jones potential. It outputs graphs for the mean square displacement (MSD), the radial distribution function (RDF) and the energies of the particles simulated at different phases. The particles were simulated according to periodic boundary conditions (PBC) and minimum image conventions (MIC). Here we will discuss the results of the simulation and compare it them to another code with the same task. which introduce a box containing a finite number of particles that can be thought of as having infinite iterations of itself in all directions. PBC is set up such that when a particle moves out of the box an image of it will re-enter in the opposite side, thought as being from the equivalent particle in the next door box. MIC uses this idea, such that when finding the separation of two particles, an image of one may be closer than the original particle at play.

# Contents

# 1 Descriptions of the Programme

The programme consists of a class called `particle3D.py`, the main module `N-body.py`, a pre-made module `MDUtilities.py` and three separate modules, `forces_energies.py`, `observables.py`, and `PBC.py`, containing functions to calculate features of the particles we wish to simulate. The code also inputs one of three text files that include the conditions for argon in each of the three states as well as the number of particles one wishes to simulate, the cut off radius from where forces would be no longer calculated, the time step used for the time integration algorithms, and the number of these time steps one wishes to run the algorithms over. This was different to the Design Document where we said that there would only be the module `N-body.py` and the class `Particle3D.py`, we decided to split up the code into separate files to make it clearer and easier to alter or debug.

We did not set the initial positions and velocities within the class but called the functions from MDUtilities into the main. The periodic boundaries were in there own separate module opposed to being in particle3D as the design document suggests, also the energy functions stated to be in the `Particle3D` class were put into a separate module with the forces, `forces_energies`. There is a static method in our class that was not described within the design document, this is `create_particle` to put the particles in the right form.

For the particles we have used reduced units for distances, energies, mass, temperature and time, shown respectively as:

$$r^* = \frac{r}{\sigma} \qquad E^* = \frac{E}{\epsilon} \qquad m^* = 1 \tag{1}$$

$$T^* + \frac{\epsilon}{k_B} \qquad t^* = \sigma\sqrt{\frac{m}{\epsilon}}$$

These keeps the calculations and numerical outputs more manageable.

The interactions of the particles were calculated with accordance to the Lennard Jones potential; this being the most common approximation of interactions of pairs of non-polar atoms such as argon.

$$U_{\mathbf{r}_1\mathbf{r}_2} = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{2}$$

$$U_{\mathbf{r}_1\mathbf{r}_2} = 4\left[\frac{1}{r^{12}} - \frac{1}{r^6}\right] \tag{3}$$

Where equation (2) is the energy of two particles at $\mathbf{r}_1$ and $\mathbf{r}_2$ and the equation (3) is the same equation but using the reduced units stated in (1). The pairwise forces were calculated as:

$$\mathbf{F}_1 = -\Delta U_{\mathbf{r}_1\mathbf{r}_2} \tag{4}$$

$$\mathbf{F}_1 = 48\left[\frac{1}{r^{14}} - \frac{1}{2r^8}\right](\mathbf{r}_1 - \mathbf{r}_2) \tag{5}$$

$$\mathbf{F}_2 = -\mathbf{F}_1 \tag{6}$$

Where, as shown in equation (6), the force on one of the pairwise particles will be equal but opposite that on its partner. A force matrix, $F_{ij}$, was created containing the forces of each particle, $i$ , due to each other particle, $j$. $F_{ij}$ is a diagonal matrix with all diagonal components being zero, as there is no force from a particle and itself. to reduce the number of calculations having to be made, the equal and opposite force was put into the matrix instead of calculating the same thing again, as $F_{ij} = -F_{ji}$. The force was only calculated up to a cut-off radius, $r_c$, where at separation greater than it the forces exerted between two particles is zero. This reduces the computational effort, and can be done due to the forces at larger distances being negligible.

The particles positions and velocities were updated based on a velocity Verlet algorithm, this consisted of updating each particle's velocity every time step according to:

$$v_{t+dt} = v_t + \frac{f_t dt}{m} \tag{7}$$

and position, to second order time step:

$$r_{t+dt} = r_t + v_t dt + \frac{f_t dt^2}{2m} \tag{8}$$

Velocity Verlet was used due to its ability to accurately determine trajectories with large numbers of timesteps. Although not as accurate at high orders as other methods, it allows for calculations to be done quickly.

PBC introduces the idea of the particles being within a box, it corrects particle's positions such that a particle that moves over a boundary of the box will re-enter from the opposite side as shown by the blue particles in figure 1. This is done by finding the remainder of a particle's position with the dimensions of the box, ensuring that if a particle has moved outside the box it will be brought back in to where it would theoretically enter the adjacent box. MIC is the convention that we compute interactions of the closest image of a particle, this image not necessarily being within the same box as the original particle. How this is used here is finding the images of all particles over many boxes, not just the ones closest, up to a inputted cut of radius, $r_c$ . As shown in figure 1 if all the green particles were within $r_c$ from the central red particle, their force contributions to the red particle would be calculated.

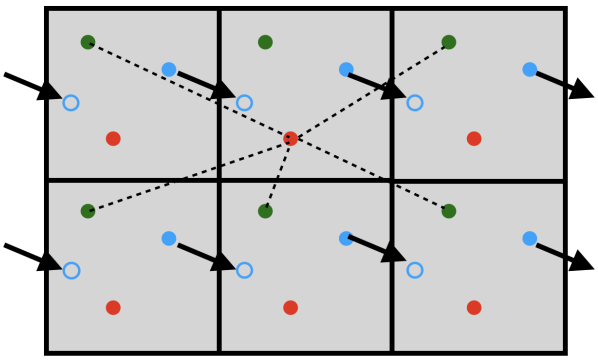

Figure 1: Diagram illustrating how PBC is implemented, blue particles, and what MIC is, dashed lines, the idea of infinitely many theoretical boxes that are all identical. The black arrows show the trajectory of the blue particle to its new position. The dashed lines show the central red dots distance from all the images of the green dot not in its box.

How a particle deviates over time from a reference point,$r_{i0}$, is the MSD. In our code $r_{i0}$ is the initial positions of the particles, so we calculated the deviation from a particles initial position:

$$MSD(t) \equiv \left\langle |\mathbf{r}_t - \mathbf{r}_0|^2 \right\rangle \tag{9}$$
$$= \frac{1}{N} \sum_i |\mathbf{r}_{it} - \mathbf{r}_{i0}|^2$$

The code averaged the MSD over all particles.

The RDF measures the probability of finding a particle at a given distance, indicating how ordered the simulated argon atoms are within the system. The definition is:

$$RDF(r) = \frac{1}{N\rho} \left\langle \sum_{i,j} \delta(r_{ij} - r) \right\rangle \tag{10}$$

The code graphs the RDF as a histogram by finding the separation of particle pairs and separating them into bins. This is done for each timestep and averaged.

| State of Argon | Temp $\frac{\epsilon}{k_B}$ | Reduced Particle Density, $\rho$ | Cut off Radius, $r_c$ | no. particles, $N$ | no. timesteps | timestep |
|---|---|---|---|---|---|---|
| Gas | 1.583 | 0.02 | 3.5 | 80 | 10000 | 0.0001 |
| Liquid | 1.125 | 0.594 | 2.5 | 80 | 10000 | 0.001 |
| Solid | 0.625 | 1.18 | 2.5 | 108 | 10000 | 0.001 |

Table 1: A table showing all the input parameters for argon in the three states; gas, liquid, and solid. The temperature and densities were found from argon's phase diagram in Figure 2.

# 2 Results and Discussion

The code successfully simulated argon atoms interacting via the Lennard-Jones potential in VMD by outputting a file `trajectory.xyz`, which when inputted into VMD simulated the particles interactions within a box adhering to PBC and MIC discussed above. The run time of the code with a timestep of 10000 takes around 40 minutes on a MacBook Pro, this shows that the code is not at optimal efficiency.

The temperature and densities used for the particles in each state were decided by looking at argons phase diagram, as shown in Figure 2. These were then calculated in reduced unit as seen from (1), all the parameters for each simulation are shown in Table 1. The code was run for these three instances of argon and graphs were outputted for the MSD; the total, kinetic, and potential energies; and the RDF.

Using the MSD graphs, shown in Figure 3, the self-diffusion coefficient was found via:

$$MSD(t) = 6Dt \tag{11}$$

For finding the diffusion constant PBC was taken out of the simulation so to find the true diffusion of the particles, without the particles being put back into the box. For a solid no diffusion is meant to occur, with the atoms vibrating around a fixed position. This can be seen as the MSD being constant over time. As shown in Figure 3 the MSD fluctuates around a value of $3.69(8)x10^{-22}m^2$. This is what is to be expected with a small uncertainty. For liquid, the MSD should be proportional to time, which is shown to be true in this simulation by the second graph in Figure 3.

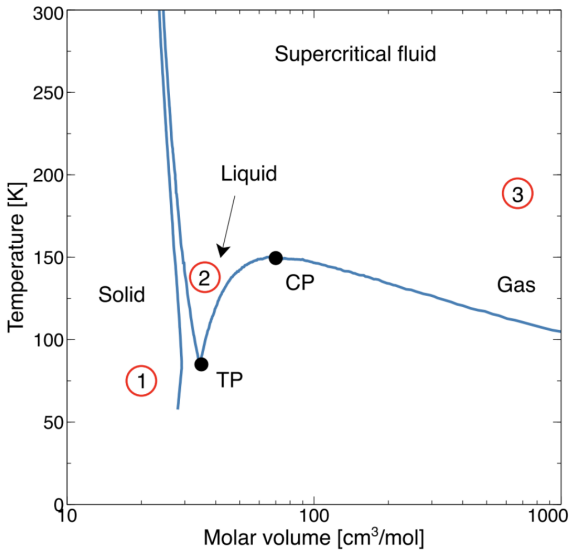


Figure 2: Phase diagram of argon. Points 1, 2 and 3 the conditions of simulation for solid, liquid and gas respectively. For point 1, temperature is 75K (in reduced units this is 0.625) and molar volume $20cm^3mol^{-1}$ (corresponding to a reduced particle density of 1.18). Point 2, temperature is 135K (1.125 in reduced units) and molar volume $40cm^3mol^{-1}$ (0.594 reduced particle density). Point 3 has temperature 190K (1.583 in reduced units) and molar volume of around $1000cm^3mol^{-1}$ (0.02 reduced particle density).

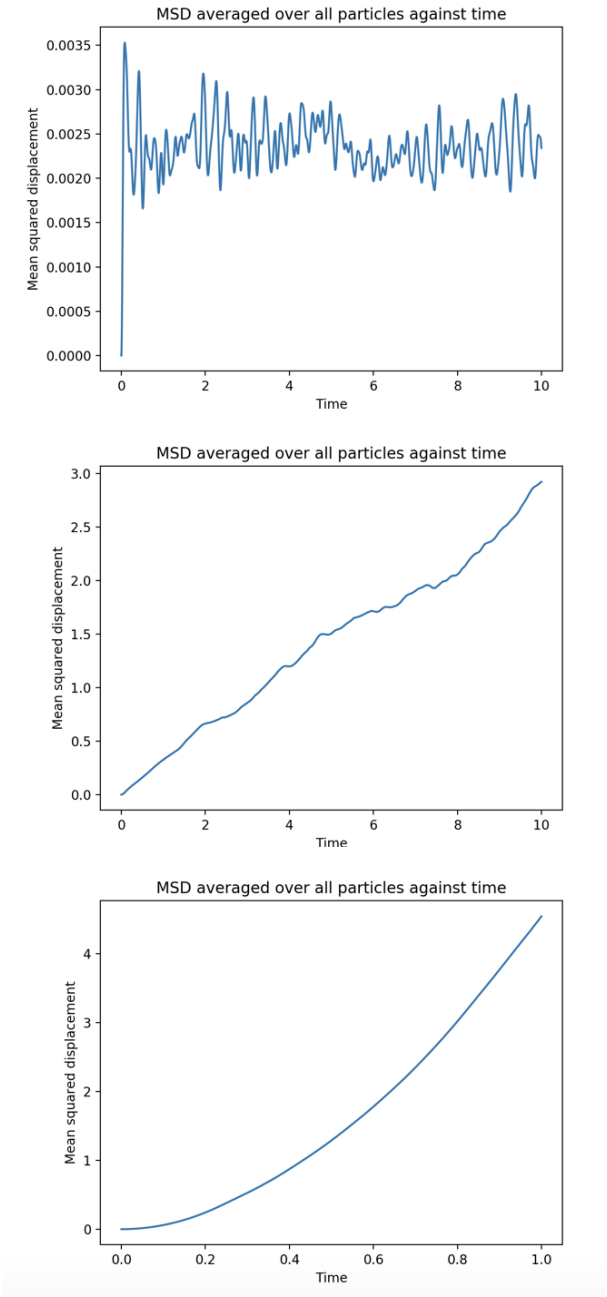


Figure 3: MSD against time graph for solid (top), liquid (middle) and gas (bottom) with parameters given in Table 1. For solid, fluctuation about average value of 0.0024(7) which is equal to $3.69(8)x10^{-22}\mathrm{m}^2$. For liquid D=0.292393 which is $1.57x10^{-8}m^2s^{-1}$. For gas, D=0.75593 which is $4.05x10^{-8}m^2s^{-1}$.

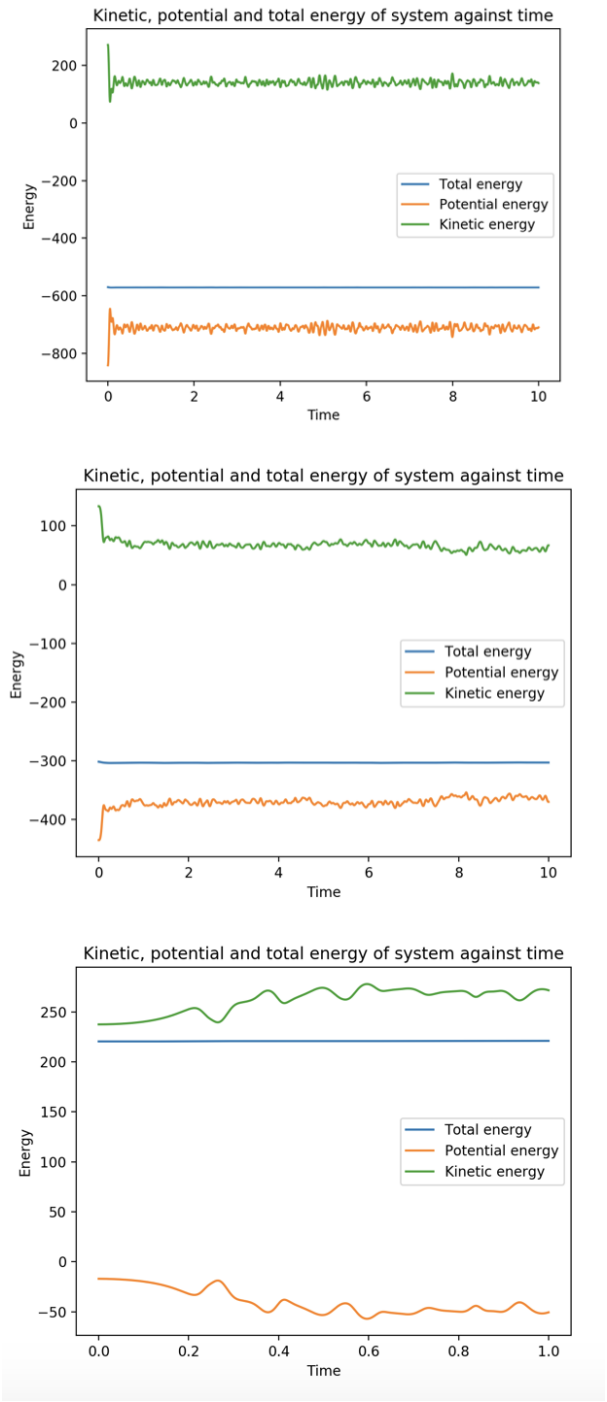


Figure 4: Graph showing the total, potential and kinetic energies of the system over time for solid, liquid and gas.

The diffusion constant for argon here was found to be $1.57x10^{-8}m^2s^{-1}$. For gas the too meant to be linear with tailing off as you get further away. The diffusion constant should also be higher than it is for

liquid due to the particles being further apart so smaller force containing the particles. The diffusion constant was found to be $4.05x10^{-8}m^2s^{-1}$.

The energy of the outputted graphs for argon in its three states can be seen in Figure 4. Due to the system being closed, having no external inputs or outputs, the energy should remain the same throughout the simulation.

The graphs in Figure 4 show the potential and kinetic energies fluctuating while total energy remains constant. However, this simulation is not perfect, and zooming in on the seemingly straight line shows fluctuations around a constant average. For solid this is found to be -740.8(1), with the uncertainty being that of the size of the fluctuation. For liquid it is -303.2(5), and for solid 219.5(4). There is an inherent equilibration time for the total energy, this is due to the initial energy of the system being perceived as being greater, then stabilising. The equilibration time for solid is 46 timesteps, for liquid; 123 timesteps and for gas; 202. These are all low compared to the 10000 timesteps being used in each simulation. This shows that the code quickly corrects for these initial errors.

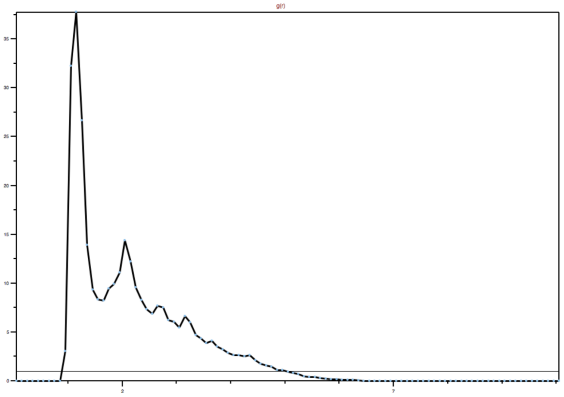

Figure 5: Graph outputted from VMD for the RDF of liquid argon set to the parameters in Table 1. The y-axis is RDF and the x-axis is the distance.

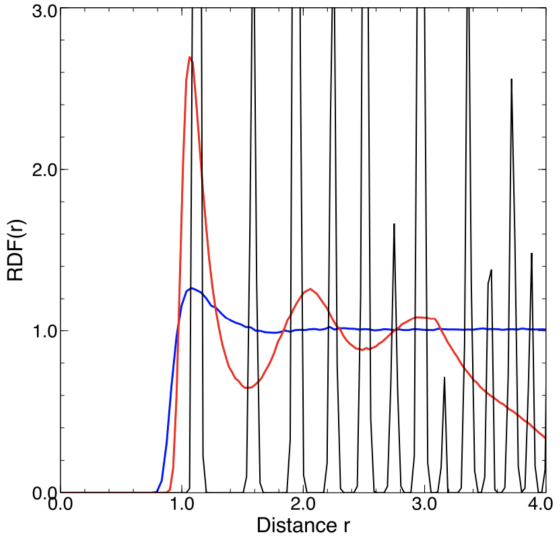


Figure 6: Graph of the expected RDF graphs for solid (black), liquid (red) and gas (blue).

Our RDF graphs have not been normalised so have a factor of left as well as the number of timesteps and particles in the system. This means that our graphs, as seen in Figure 4, aren't accurate representations for the RDF of the system. The graphs outputted by VMD reflect the RDF of the system more closely, as seen in Figure 5. Figure 6 shows that predicted trends of RDF, this matches the VMD graph much more closely. Our graph for solid does, however, show the expected shape, with the characteristic peaks being that of the atoms vibrating in position.

The simulation is far more accurate at smaller timesteps, however very small timesteps are unfeasible due the need to have more of them which causes the code to run slower. Any greater than a certain timestep causes the simulation to break down. To determine the maximum timestep possible to use for each phase I looked at the energy graph, when the simulation breaks down, so does the consistency of the total energy. The maximum timestep for solid, liquid and gas respectively are 0.033, 0.031 and 0.029 although even at these timesteps

the accuracy was diminished.

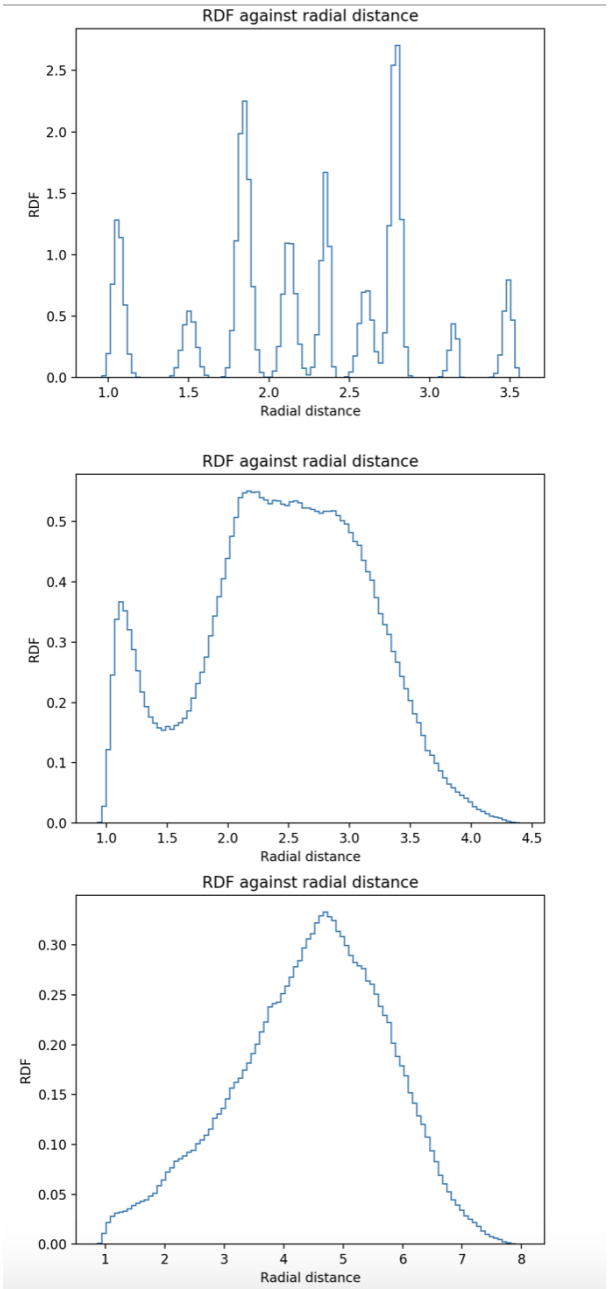


Figure 7: RDF histogram for solid, liquid and gas with parameters given in Table 1. Top graph representing solid shows characteristic peaks.

# 3 Conclusions

Our programme successfully simulated argon atoms in its solid, liquid and gas phases. The data collected from the simulation seemed to correlate with what was expected. The code runs very slowly and if done again I would explore less computationally expensive methods of calculation. Also the RDF not being normalised made that data not very useful, if done again I would make sure to do this. The team worked well together, both members inputting useful work into the programme.