Atomic modeling of Argon

Andreas V. Solbrå
University of Oslo
Department of Computational Physics

February 25, 2013

Abstract

The goal of this project is to gain experience with the methods used in simulations of molecular dynamics. In the project we will use Argon atoms as a prototype, and measure the different physical properties of our model.

1 Setting up a system

We start by placing the particles in a face centered cube formation, as shown in figure 1. We can describe this by one of the corners of the cube, and the relative position of the other particles in the grid. The positions of the cells in a $N \times N \times N$ lattice are therefore

$$\mathbf{R}_{i,j,k} = i\hat{\mathbf{u}}_1 + j\hat{\mathbf{u}}_2 + k\hat{\mathbf{u}}_3 \tag{1}$$

where the basic unit vectors are the cartesian vectors multiplied by the length b of the unit cell,and the relative positions of the atoms in the cell are

$$\begin{aligned}
\mathbf{r} &= 0\hat{i} + 0\hat{j} + 0\hat{k} && (2)\\
&= \frac{b}{2}\hat{i} + \frac{b}{2}\hat{j} + 0\hat{k} && (3)\\
&= 0\hat{i} + \frac{b}{2}\hat{j} + \frac{b}{2}\hat{k} && (4)\\
&= \frac{b}{2}\hat{i} + 0\hat{j} + \frac{b}{2}\hat{k} && (5)
\end{aligned}$$

for solid Argon, the lattice constant is $b = 5.260$ Å. The code for this can be found in the function `setPosFFC()` in the file `system.cpp` in the source code located at `www.github.com/andreavs/fys4460`

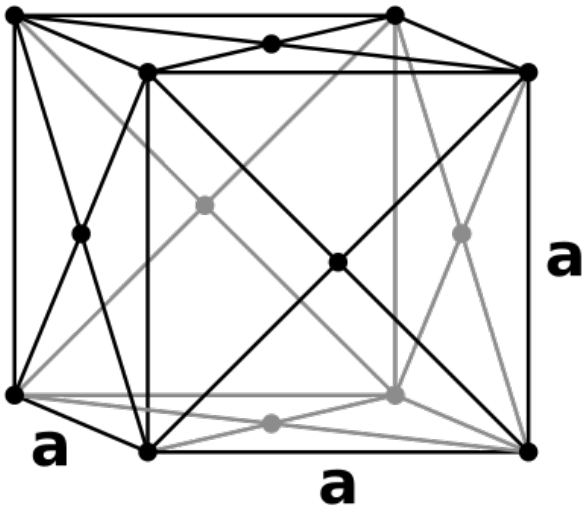


Figure 1: The figure shows a face centered cube

1.1 setting initial velocities

From statistical physics we know that that at equilibrium temperature T, the velocities are given by the Boltzmann distribution, which mean that the individual components are normally distributed. The standard deviation is $\sqrt{k_B T/m}$ where m is the mass of the atom, T is the temperature of the system and k_B is the Boltzmann constant. To avoid any drift in the system, we can calculate the the total drift in the system, and then subtract the average drift from each particle. The code for this can be found in the function `setVelNormal()` in `system.cpp`.

2 Time evolution of the system

We use the symplectic and numerically stable velocity Verlet algorithm to integrate particle motion. Even though this integrator is simple, it is time reversible and provides excellent energy conservator.

For each particle i, the steps are as follows:

$$\mathbf{v}_i(t + \Delta t/2) = \mathbf{v}_i(t) + \frac{\mathbf{F}_i(t)}{2m}\Delta t \tag{6}$$

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t\Delta t/2)\Delta t \tag{7}$$

$$\mathbf{F}_i(t + \Delta t) = -\nabla_i U_i(\mathbf{r}(t + \Delta t)) \tag{8}$$

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t + \Delta t/2) + \frac{\mathbf{F}_i(t + \Delta t)}{2m}\Delta t \tag{9}$$

This is implemented in the function `timeEvolve()` in `system.cpp`. If we implement this as

it stand using $U_i = 0$ as a first test, what we will see is that the particles spread out into the empty space with their initial velocity. We do not want this behaviour. As we want to look at the statistics of our system, we can rather impose periodic boundary conditions.

2.1 Periodic boundary conditions

We want to keep our particle density constant. One way to achieve this is to say that if a particle travels beyond the edge of one of the walls in the system, is simply enters the system on the oposite side. A good way to implement this is to impose this requirement right after the position step in the integrator. This requirement is implemented in the method `setPeriodicBoundaries()` in `system.cpp`.

Implementing this let's us see the particles move within their assigned box, but they still move in a straight line with their initial velocities as there are no forces acting on them. In the next subsection we implement a simple force model.

2.2 Forces

We use a Lennard-Jones potential for the interatomic interactions. The Lennard Jones potential has the following form:

$$U(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{10}$$

where r is the distance between the atoms. For Argon optimal values of the parameters are:

$$\frac{\epsilon}{k_B} = 119.7\text{K}, \quad \sigma = 3.405\text{Å}. \tag{11}$$

From this we find the force between particle i and j to be

$$\mathbf{F}(\mathbf{r}_{ij}) = -\nabla U(r_{ij}) \tag{12}$$

$$= -\frac{\partial U}{\partial r_{ij}} \hat{\mathbf{r}}_{ij} \tag{13}$$

$$= \frac{24\epsilon}{r} \left[2\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right] \hat{\mathbf{r}}_{ij} \tag{14}$$

where $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, $r_{ij} = |\mathbf{r}_{ij}|$ and $\hat{\mathbf{r}}_{ij} = \mathbf{r}_{ij}/r_{ij}$. We may note that the equilibrium distance is $r = 2^{1/6}\sigma \approx 1.122\sigma$. This is a good time to choose appropriate units for our simulation. Introducing the dimensionless variable $\tilde{r} = r/\sigma$, and meassuring energy in units of ϵ, we do not have to pay attention to these in our program.

We want our forces to adhere to our periodic boundaries, so that we calculate the distance to the other atom using the minimum image convention, as illustrated in figure 2. To a point,

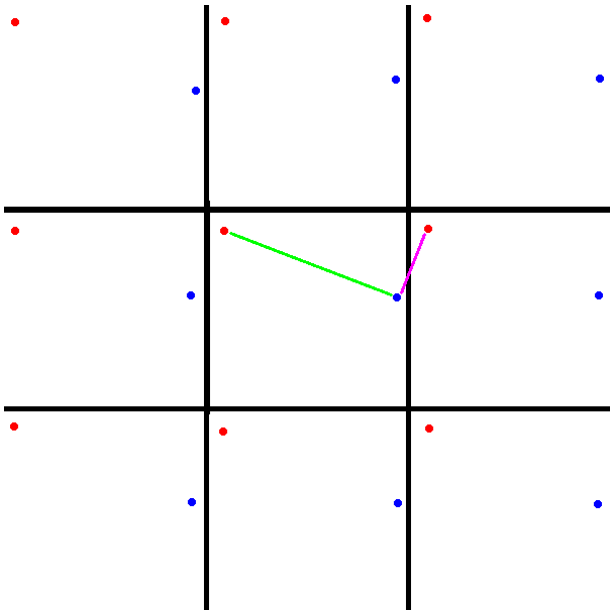


Figure 2: The figure illustrates the minimal image convention. The middle box is our system, and the outlying boxes are the periodic extensions. Withour the minimal image convention, the green line would be used as the distance. However, the purple line is a better choice, so this is used instead.

we let the distance in the x direction become $\min\{|x_i - x_j - L|, |x_i - x_j|, |x_i - x_j + L|\}$ and so on.

This force model is implemented in the function `singlePairForces()`, which is called by the function `calculateForcesLJMIC()`, whose job it is to run over all the pairs of particles in the system. This is done using Newton's third law so there are no superflous calculations. This operation, however, still scales as N^2 where N is the number of particles, which quickly becomes far to time consuming for large systems. In the next section we will look into how we can adress this. However, we know have a working model for simulating Argon, and it is worthwhile to simulate this using software such as VMD or Ovito. An example of the latter is shown in figure 3, where the an $8 \times 8 \times 8$ grid is simulated using a starting temperature of 100 Kelvin. We see that after some initial scattering due to the initial velocities, the system relaxes into a solid grid.

3 Speed up: Computation cells

The force calculations are very time consuming. We want to improve this by negleting force terms for particles that are very far apart. The Lennard-Jones potential can be neglected for distances longer than $r_{\text{cut}} = 3\sigma$. A fairly straight forward way to this is to divide our system into several cells, as illustrated in figure 4. We then let each cell calculate forces within itself

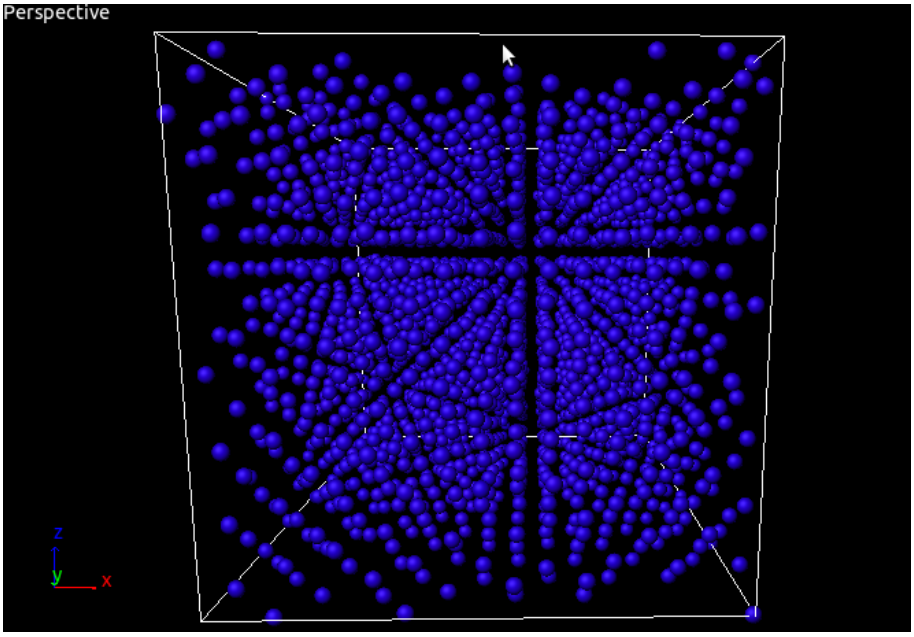


Figure 3: The figure shows a simple animation of our Argon system in Ovito.

and its 26 neighbouring cells.

We may want to implement Newton's third law in this system, and one way to do this is to only let one cell point to 13 of its neighbours, and let the 13 remaining neighbours point to the cell, so there are no redundant calculations. This is illustrated in figure 5.

The atoms can be stored in the cells using linked lists. The implementation of computation cells can be found in the functions `placeAtomsInCells()` and `calculateForcesCellsLJMIC()`, as well as the class `Cell`. The neighbour lists are made in the constructor of the system.

4 Macroscopic Observables

We can test the quality of our model by meassuring different physical properties of the system, and compare the results with the predictions from statisical mechanics. The rest of the report will be dedicated to this.

4.1 Maxwell-Boltzmann distribution at equilibrium

According to the 2nd law of thermodynamics and entropy considerations, the velocity distribution will evolve into a Maxwell-Boltzmann distribution independent of the initial conditions. We can test this by letting the initial velocity components of the particles be uniformly distributed, and making histograms of the velocities at different times. This is shown if figure 6. We see that the distribution tend to a Maxwell-Boltzmann distribution.

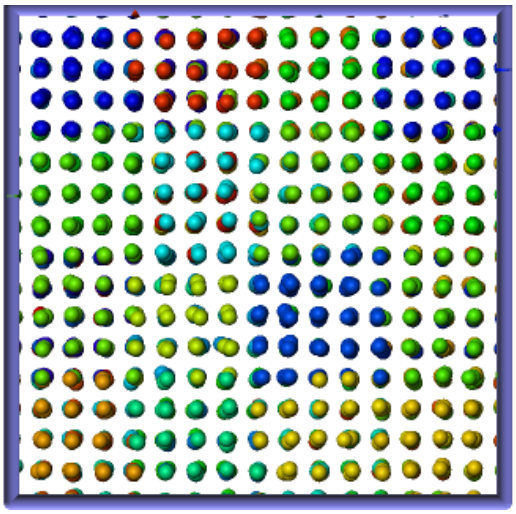


Figure 4: In this figure each computation cell is assigned a random color, and and the particles are colored by their cell.

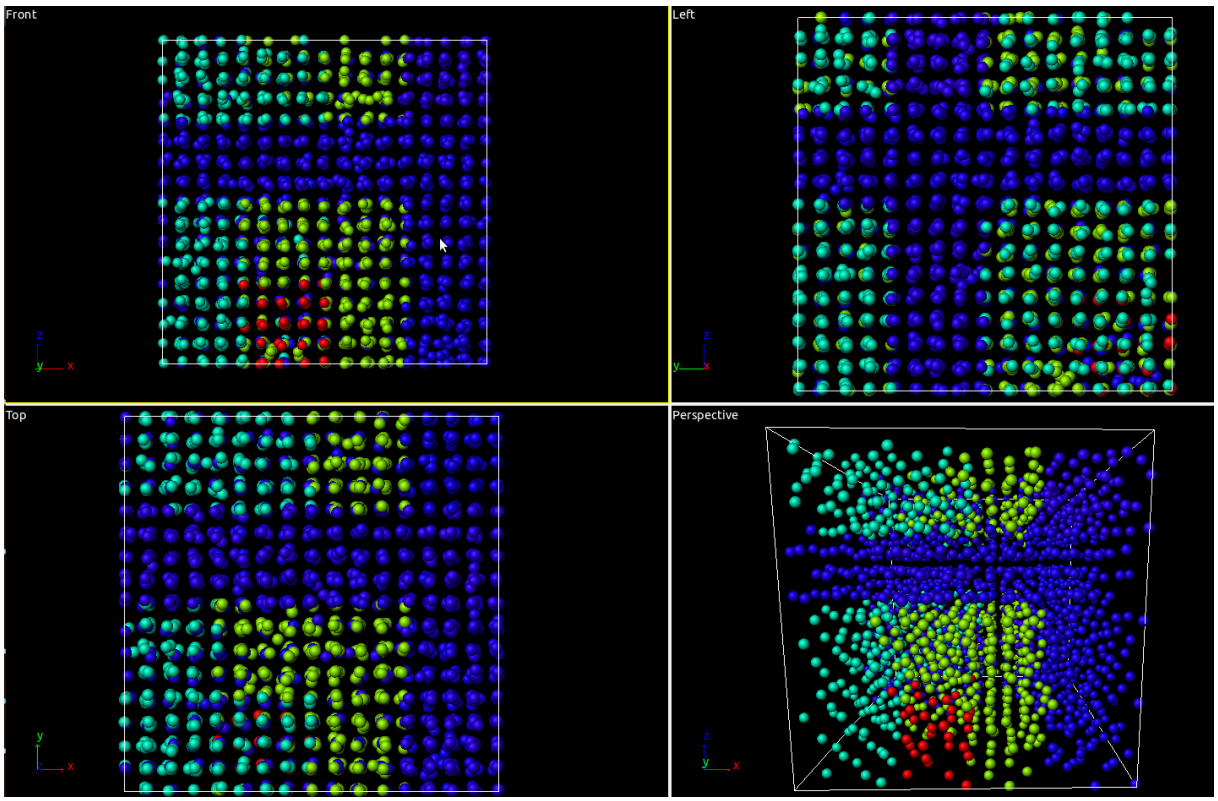
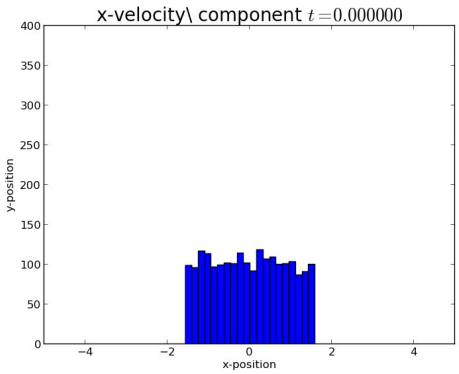
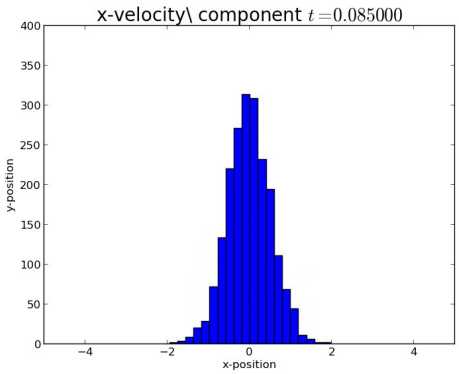


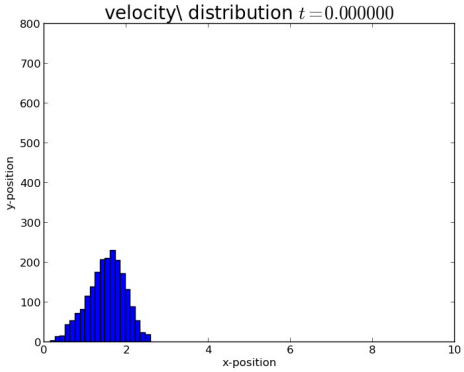
Figure 5: The figure illustrates the one-way neighbours. The red cell is the main character, the green cells are the neighbours it points to, and the cyan cells point to the main cell. The different pictures are different orientations of the same system.



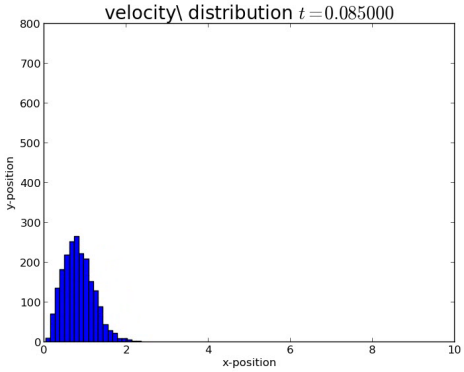
(a) initial x-component of the velocity



(b) x-component of the velocity at a later time



(c) Initial velocity distribution



(d) Velocity distribution at a later time

Figure 6: The plots show the distribution of the x-component and the size of the velocities a initially and after the system has settled.

4.2 Energy, temperature and pressure of the system

From the velocities and the potential, we can find the kinetic, potential and total energy of the system. From this, we can use

$$\langle E_k \rangle = \frac{3}{2} N k_B T \tag{15}$$

to estimate the temperature of the system, and we can meassure the pressure as

$$P = \rho k_B T + \frac{1}{3V} \sum_{i<j} \mathbf{F}_{ij} \cdot \mathbf{r}_{ij}, \tag{16}$$

which is valid for the micro-canonical ensemble. The results for a are shown for a starting temperature of 300 Kelvin in figures 7 and 8, with a time step of 0.005 (in MD units). We see that the temperature quickly drops, before becoming stable. The total energy fluctuations

Δt	Var(E)
0.001	0.0197
0.005	0.1208
0.01	1.6565

Table 1: The table shows a the variance in energy for a few values of Δt. It looks as the fluctuations become smaller as Δt decreases.

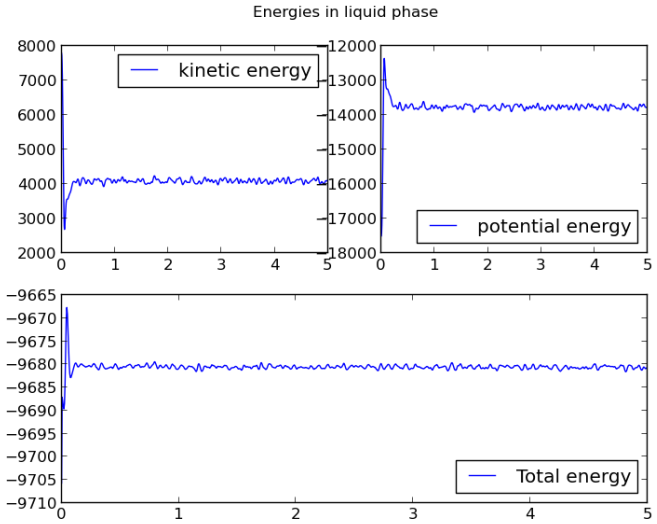


Figure 7: The figure shows the kinetic, potential and total energy of the system as a function of time, using an initial temperature of 300 Kelvin and a time step of 0.005 in MD units.

are alsmost are small. It is interesting to see how the fluctuations vary as a function of the time step. Table 1 shows the variance of the energies for different values of Δt.

Later we will implement a thermostat to our system, which will allow us to change the temperature of the system, and so we can plot the pressure as a function of the temperature. This is shown in figure 9. We see that there at one point is a sudden change in pressure while the temperature is relatively constant, and we can interpret this as a phase transition.

4.3 The diffusion constant

We want to characterize transport in a fluid by measuring the self-diffusion of an atom: We give an atom i a label, and measure its position as a function of time $\mathbf{r}_i(t)$. We find the diffusion constant from the mean square displacement of all atoms (we trace the motion of every atom):

$$\langle r^2(t) \rangle = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{r}(t) - \mathbf{r}_{\text{initial}})^2 \tag{17}$$

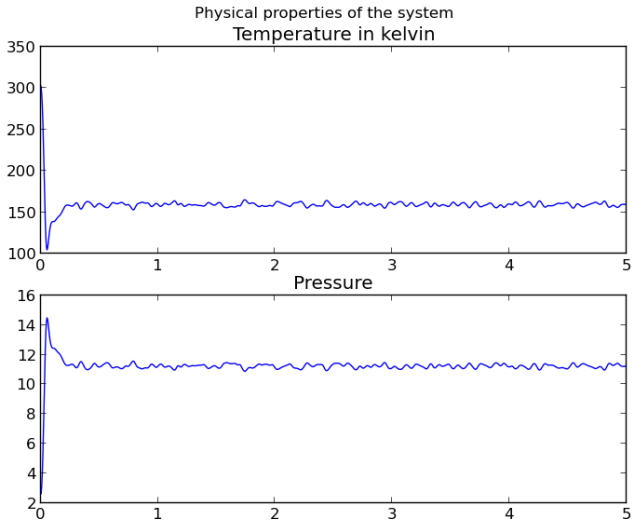


Figure 8: The figure shows the temperature and pressure of the system using an initial temperature of 300 Kelvin and a time step of 0.005 in MD units

From theoretical considerations of the diffusion process we can relate the diffusion constant in the liquid to the mean square displacement through:

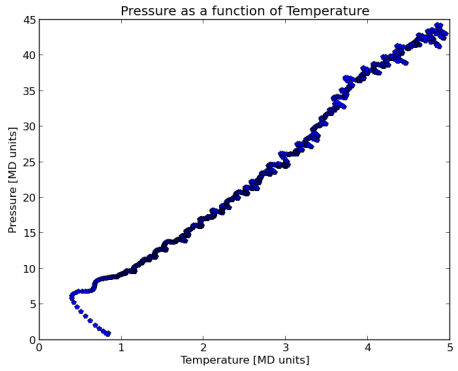
$$\langle r^2(t) \rangle = 6Dt \text{ when } t \to \infty \tag{18}$$

The average square displacement is plotted as a function of time in figure 10 for a starting temperature of 700 Kelvin (which gave an equilibriated temperature of around 320 Kelvin). We see a clear linear tendency, and we can measure D to be roughly $D \approx 0.059$ in MD units.

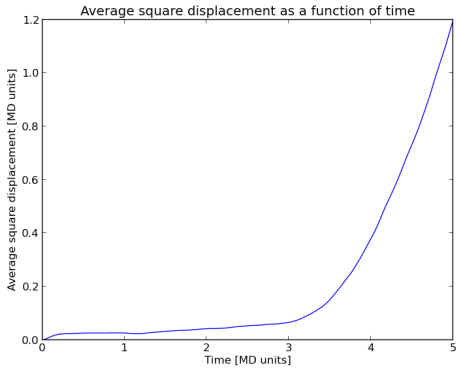
4.4 Microscopic structure - radial distribution functions

The radial distribution function $g(r)$, also called a pair correlation function, is a tool for characterizing the microscopic structure of a fluid. It is interpreted as the radial probability for finding another atom a distance r from an arbitrary atom, or equivalently, the atomic density in a spherical shell of radius r around an atom. It is commonly normalized by dividing it with the average particle density so that $\lim_{r\to\infty} g(r) = 1$.

The results for a solid and a liquid is shown in figure 11. We see that the case for a liquid looks smoother than that for a solid. This could be explained by the fact that there is a higher correlation between the particle positions in the solid (i.e. the particles are trapped in a grid).



(a) Pressure as a function of Temperature



(b) Mean square displacement

Figure 9: The figures show 9(a) the pressure as a function of temperature in a system that is linearly heating up, and 9(a) the mean square displacement. The fact that the mean square displacement starts increasing at the same time as the sudden increase in pressure points to a phase transition from solid to liquid.

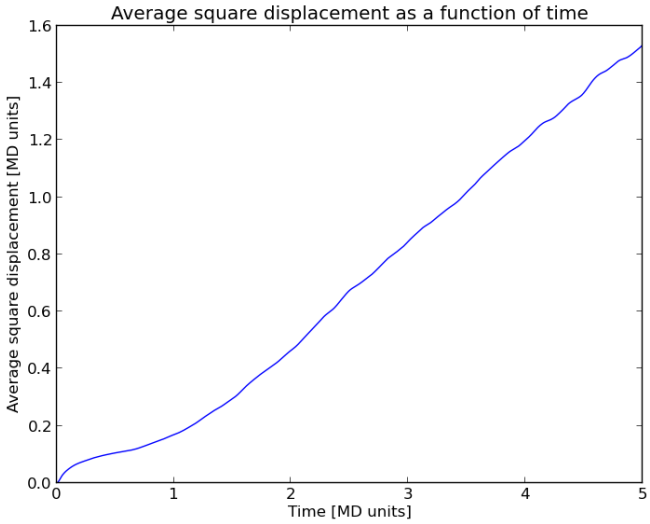
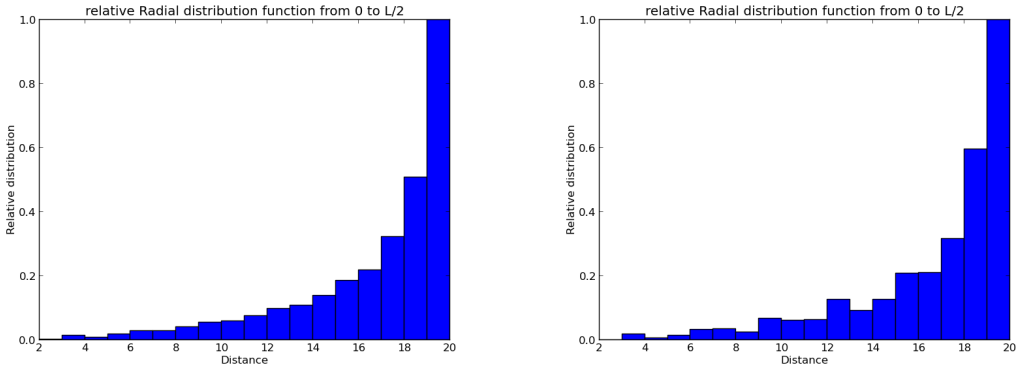


Figure 10: The figure shows the avg square displacement as a function of time in liquid Argon.



(a) radial distribution for a liquid

(b) radial distribution for a solid

Figure 11: The plots show the radial distribution for , a liquid, and , a solid

5 Thermostats

In order to simulate the canonical ensemble, interactions with an external heat bath must be taken into account. Many methods have been suggested in order to achieve this, all with their pros and cons. Requirements for a good thermostat are:

- Keeping the system temperature around the heat bath temperature
- Sampling the phase space corresponding to the canonical ensemble
- Tunability
- Preservation of dynamics

Here we will implement two thermostats; the Berendsen thermostat and the Andersen thermostat.

5.1 The Berendsen thermostat

Many thermostats work by rescaling the velocities of all atoms by multiplying them with a factor γ. The Berendsen thermostat uses

$$\gamma = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T_{\text{bath}}}{T} - 1\right)} \tag{19}$$

with τ as the relaxation time, tuning the coupling to the heat bath. Though it satisfies Fourier's law of heat transfer, it does a poor job at sampling the canonical ensamble. If we set $\tau = \Delta t$, it will keep the estimated temperature exactly constant. It should be put to 10-20 times this value. The resulting temperature is shown in figure 12. We see that after the usual dip, the temperature increases to the thermostat value (here set equal to the initial value of the system).

5.2 The Andersen thermostat

The Andersen thermostat simulates (hard) collisions between atoms inside the system and in the heat bath. Atoms which collide will gain a new normally distributed velocity with standard deviation $\sqrt{k_B T_{\text{Bath}}/m}$. For all atoms, a random uniformly distributed number in the interval [0,1] is generated. If this number is less than $\Delta t/\tau$, the atom is assigned a new velocity. In this case, τ is treated as a collision time, and should have about the same value as the τ in the Berendsen thermostat. The Andersen thermostat is very useful when equilibriating systems, but disturbs the dynamics of e.q. lattice vibrations.

The resulting temperature function when using the same parameters as for the Berendsen thermostat is shown in figure 13.

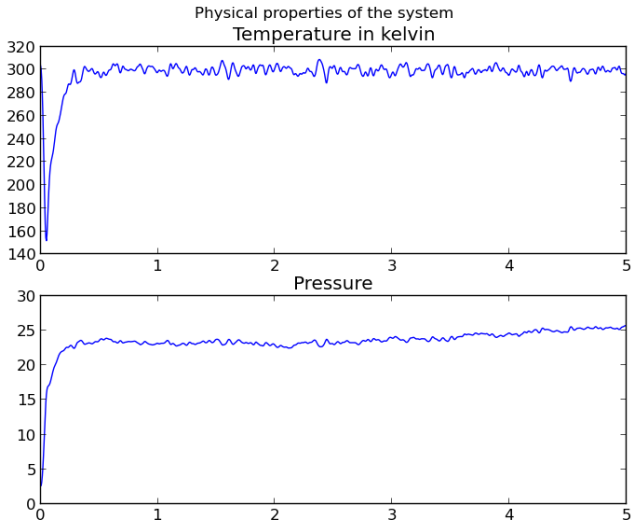


Figure 12: The figure shows the temperature of the system when controlled by a Berendsen thermostat at 300 Kelvin.

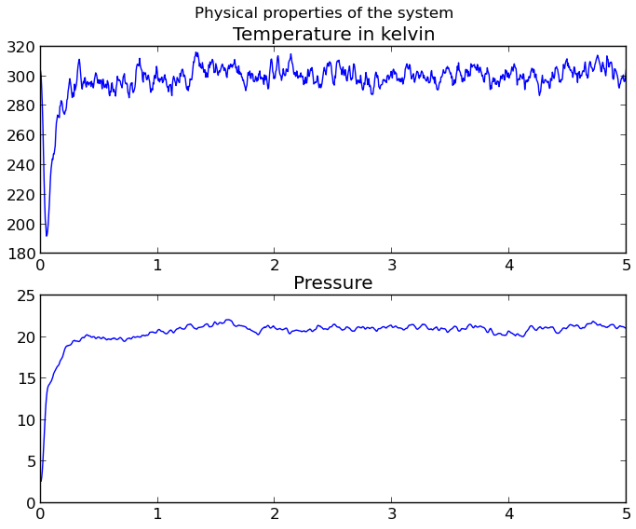


Figure 13: The figure shows the temperature of the system when controlled by a Andersen thermostat at 300 Kelvin.