

Project 1, FYS4460

Fredrik E Pettersen
f.e.pettersen@fys.uio.no

February 27, 2013

Contents

1	About the problem	3
2	The algorithm	3
3	Points of progress	3
4	Results	4
5	Stability and precision	10
6	Final comments	10

1 About the problem

The goal of this project is to model systems of Argon atoms using the Lennard-Jones potential.

2 The algorithm

To model the system we will set up an initial distribution of Argon atoms in a face-centered cubic lattice and give each atom an initial, random velocity drawn from the Boltzmann distribution. Luckily, the armadillo package for C++ has a function for doing this.

Since we are working with very small numbers ranging over several orders of magnitude it will be useful to rescale our units. Scaling the position so that $\sigma = 3.405\text{\AA}$ is the unit length and $\epsilon = k_B \times 119.8\text{K}$ is the unit energy gets us a long way. We can also use the Argon mass $m_{Ar} = 39.948\text{amu}$ ($1\text{amu} = 1.66053886 \times 10^{-27}\text{kg}$). As a means of making the program considerably more efficient we will divide our volume into cells of a pre-determined size containing some particles and neglect the single-pair forces between particles further away than 3σ . As we can see from figure 1 this is a rather good approximation. Furthermore the program will now include some single-pair forces between particles up to 6σ away from each other.

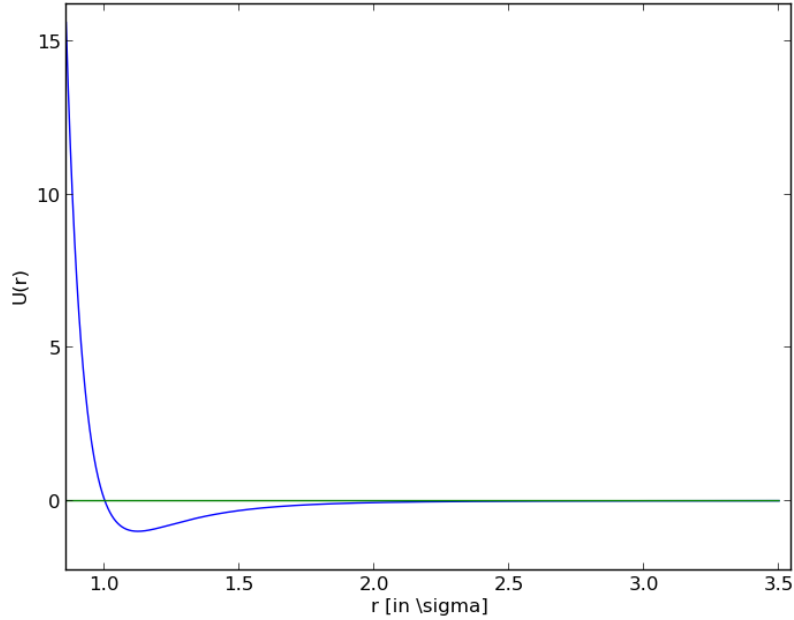


Figure 1: A plot of the Lennard-Jones potential vs the norm of the distance between two atoms.

Finally we will implement a thermostat.

3 Points of progress

Step 1 of this implementation is to simply implement Newton's 2. law using the integrator given in the project description and visualize this. Seeing as all the particles are given an initial random velocity, they just keep going in this direction.

Step 2 is to use the Lennard-Jones potential to find the force on an atom and the equilibrium interatomic distance. We find the single-pair force between two particles by the relation

$$\vec{F} = -\nabla U = 24\epsilon \left[2 \frac{\sigma^{12}}{r^{13}} - \frac{\sigma^6}{r^7} \right]$$

We find the equilibrium distance by setting $\vec{F} = 0$ and after some massage we find

$$r = \sigma \sqrt[6]{2}$$

Step 3 is to check the increase in speed we get for larger systems after implementing cells and neighbour lists. Put simply the increase is formidable for large systems, but not really noticable for small systems. I did one simulation of 131072 particles after implementing the cells which finished 800 timesteps over night. This simulation would have taken days without the implementation of cells.

The last step (not counting dealing with the results) is to implement thermostats on the system. this is done by rescaling the velocities of each atom in the equilibration phase of the simulation. We are asked to implement two thermostats, the Berendsen thermostat

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

and the Andersen thermostat where for each atom a uniform random number is generated at each timestep. If the number is less than $\frac{\Delta t}{\tau}$ the atom is assigned a new velocity.

Implementing the first thermostat allows the system to actually “break free” of the crystal lattice for realistic temperatures.

4 Results

First of all, the initial distribution of Argon is visualized in figure 2 for a lattice consisting of 32 atoms. This is a so called face centered cubic lattice, where one cube consists of $5 \times 5 \times 5 = 25$ Argon atoms.

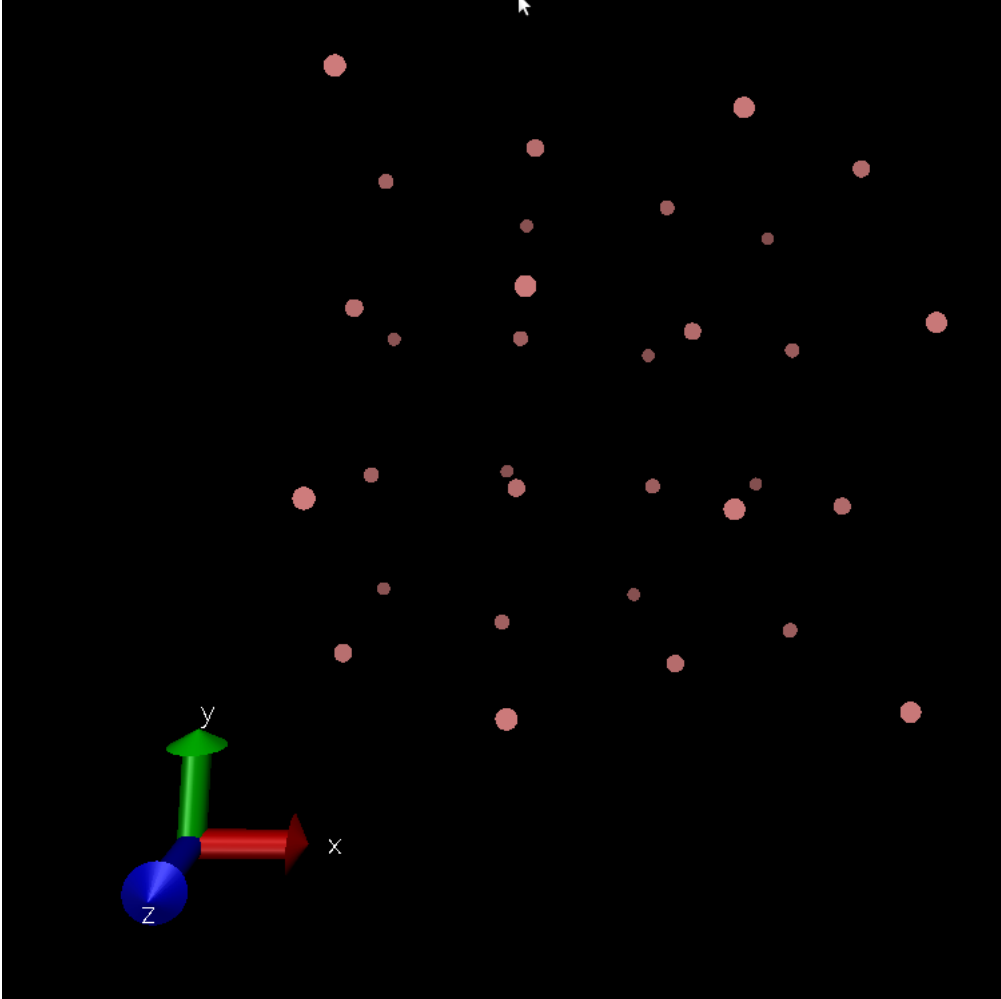


Figure 2: The initial configuration of Argon atoms makes a face-centered cubic lattice

Having placed the atoms in the correct formation, we give each atom an initial random velocity from the Boltzmann distribution depending on the temperature.

For the temperature $T = 119.8K = 1$ in MD units the velocity distribution has 0 mean and standard deviation 1. We can check this for the first resultfile (see figure 3).

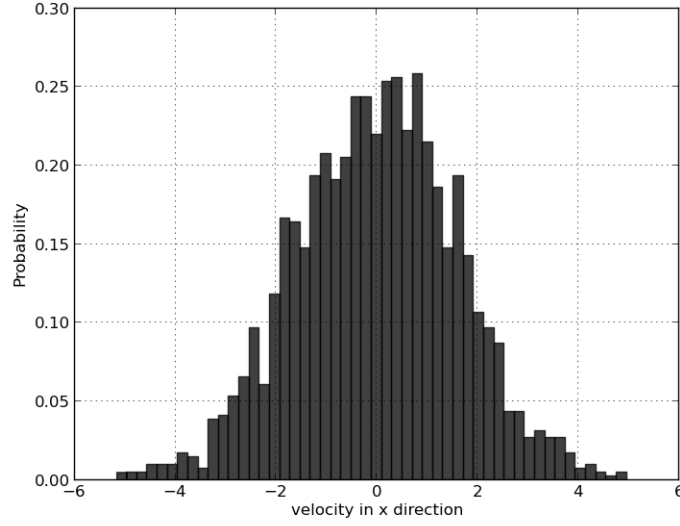


Figure 3: Velocity distribution at $t=0$.

Listed below are some plots of temperature and total energy for a few simulations. We see a clear difference in the mean square of the total displacement of an atom from before implementing the Berendsen thermostat to after this is implemented (figures 6 and 9)

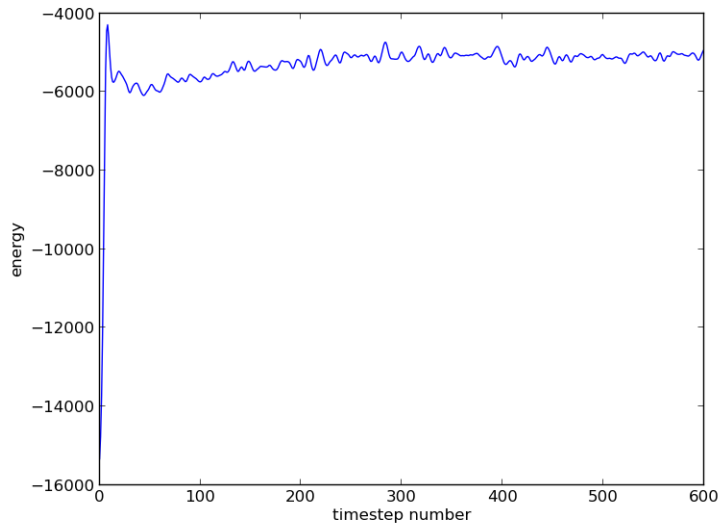


Figure 4: Total energy of a system of 2048 particles starting out at $T = 750K \simeq 6$ in MD units. $\Delta t = 0.005$

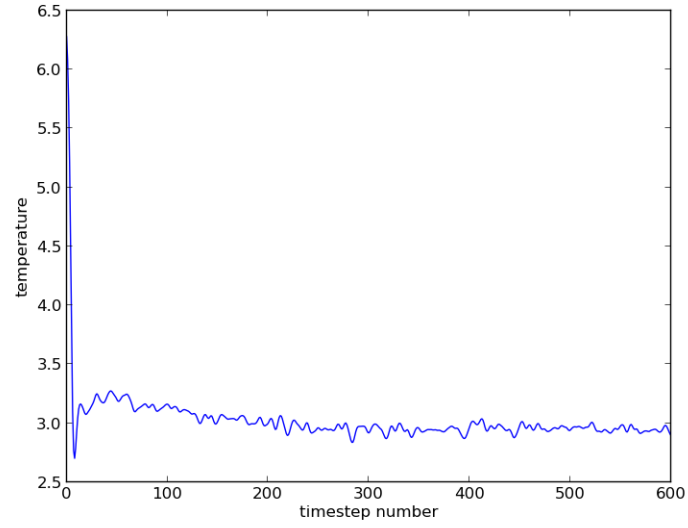


Figure 5: Temperature of a system of 2048 particles starting out at $T = 750K \simeq 6$ in MD units. $\Delta t = 0.005$

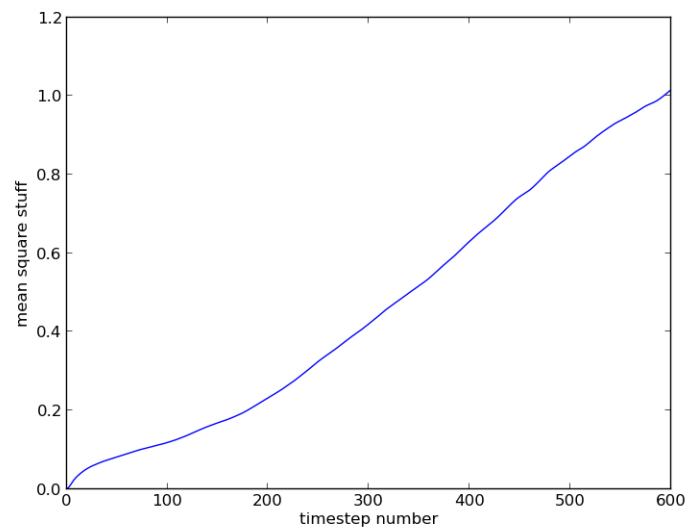


Figure 6: RMS of the total displacement from initial position. The slope of the graph is proportional to the Diffusion constant. Simulation of 2048 atoms, $T_0 = 750K$, no thermostat.

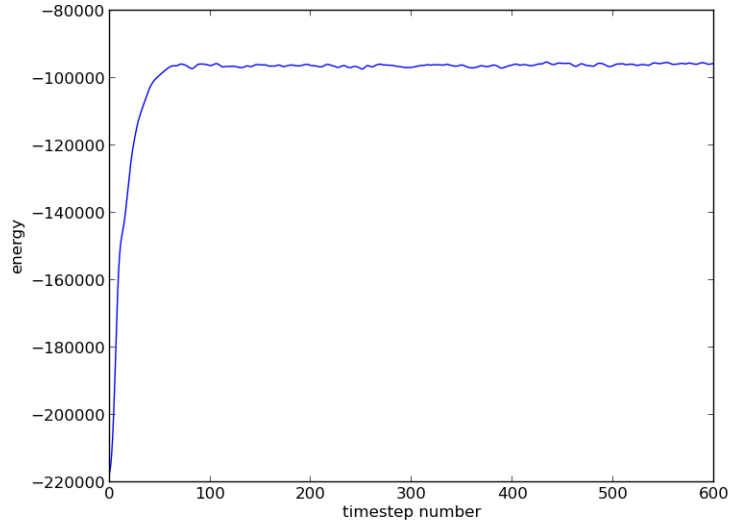


Figure 7: Temperature of a system of 16384 particles starting out at $T = 300K \simeq 2.5$ in MD units using the Berendsen thermostat. $\Delta t = 0.005$

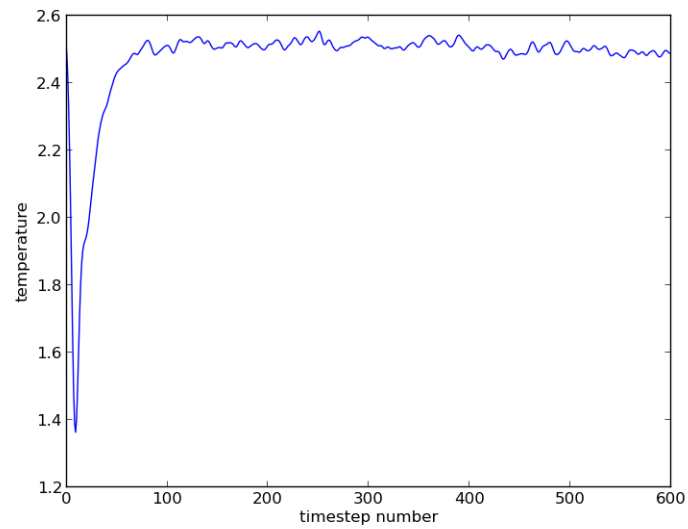


Figure 8: Temperature of a system of 16384 particles starting out at $T = 300K \simeq 2.5$ in MD units using the Berendsen thermostat. $\Delta t = 0.005$

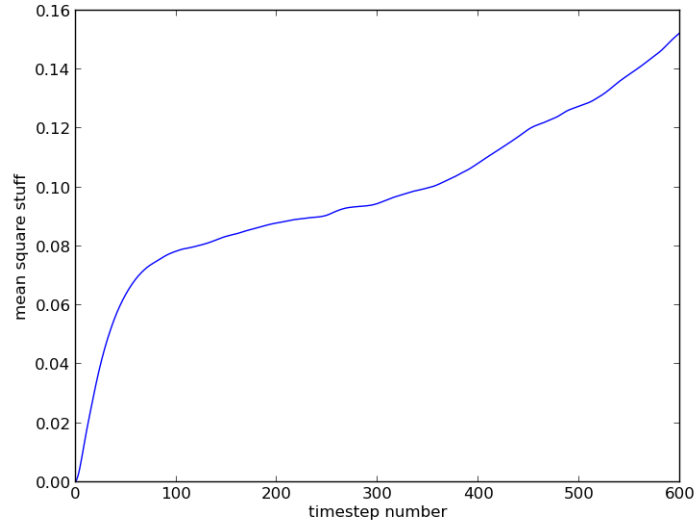


Figure 9: RMS of the total displacement from initial position. The slope of the graph is proportional to the Diffusion constant. Simulation of 16384 atoms, $T_0 = 300K$, Berendsen thermostat.

The following figures are the results of a simulation where the Andersen thermostat was implemented.

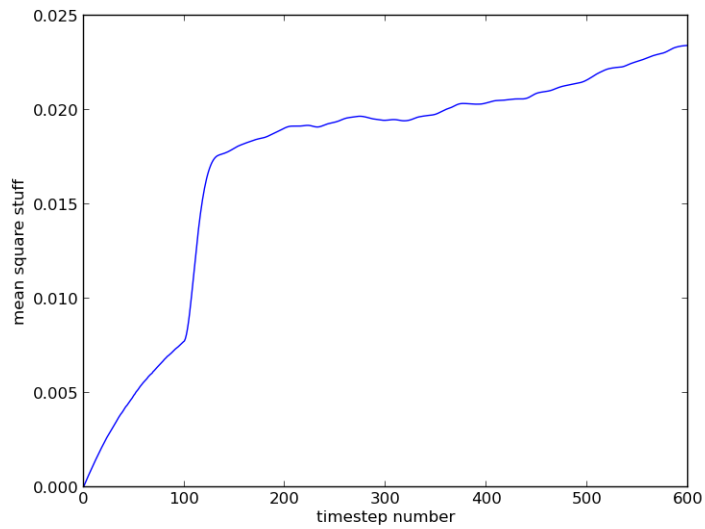


Figure 10: RMS of the total displacement from initial position. The slope of the graph is proportional to the Diffusion constant. Simulation of 16384 atoms, $T_0 = 200K$, Andersen thermostat.

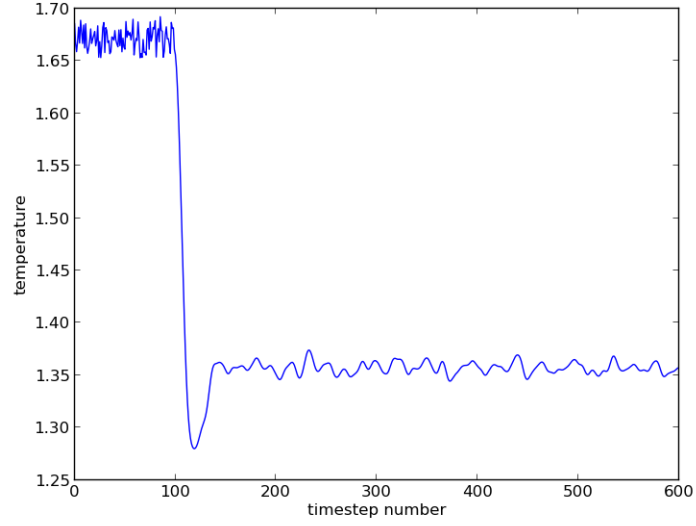


Figure 11: RMS of the total displacement from initial position. The slope of the graph is proportional to the Diffusion constant. Simulation of 16384 atoms, $T_0 = 200K$, Andersen thermostat.

There is a considerable difference between the $T(t)$ graphs using the Berendsen and the Andersen thermostat. I think it is a bit weird that the temperature drops once the thermostat is turned off (at 100 timesteps). There might be some problem with my implementation.

The radial distribution of particles is found by dividing the area where $r \in [0, \frac{L}{2}]$ into subsections, and counting the particles in these regions. As an added “safety” we average this over some timesteps.

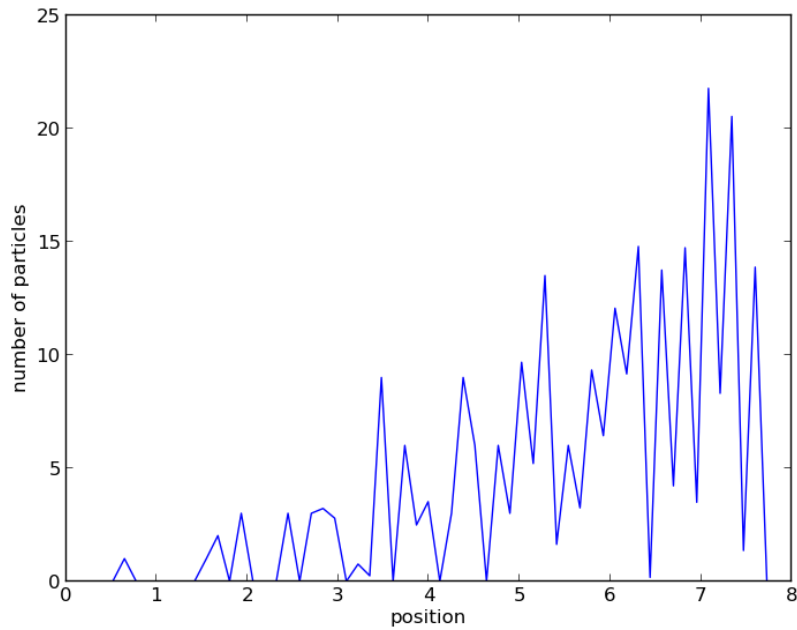


Figure 12: Radial distribution of atoms for a crystal structure. Simulation with 4000 particles and no thermostat.

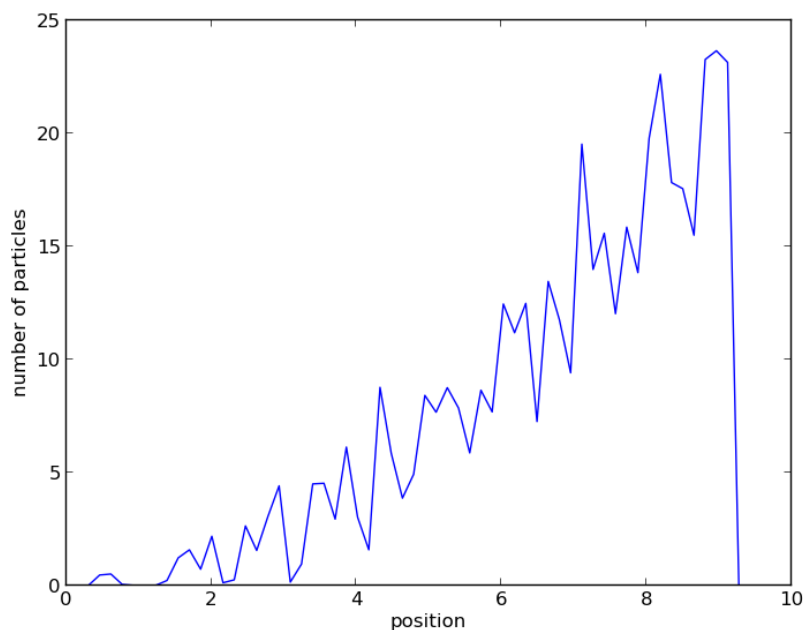


Figure 13: Radial distribution of atoms for a liquid phase. Simulation with 6912 particles and Andersen thermostat.

5 Stability and precision

We notice that the system “creates” some energy during the simulation. This is of course a non-physical artifact which depends on how large Δt we use compared to the forces between the atoms.

6 Final comments

I would like to point out that my program as it is could be more efficient if I’d have implemented Newton’s third law. As it stands, all single pair forces are calculated twice which of course is unnecessary. However the program seems to be working, and after 2 weeks of debugging I’m happy with that and I can afford to wait twice as long for my program to finish (After all it is not a factor 10).