Project 1

FYS4460 - Disordered systems and percolation

Anders Johansson
8th February 2018

a)

I have used the following workflow to see how the velocity distribution evolves with time:

- LAMMPs generates an fcc structure, and saves a data file.
- A python script reads the data file, and replaces the velocities (which are zero) with uniformly distributed velocities in a specified range.
- LAMMPs runs a simulation from the resulting data file.
- A python script uses ovito to parse the simulation data, makes histograms for each saved frame and computes the correlation.

When calculating the histograms, I have made sure the same bins are used for all frames by first finding the maximum velocity attained by any atom during the simulation, and then using equally sized bins in the range $[-v_{\max}, v_{\max}]$. One histogram is computed for each direction, and then the average of these is taken.

The correlation is computed by normalising the histograms and taking the dot product with the histogram computed from the final frame. As the velocity distribution approaches the final distribution, the correlation should approach 1.

From figure 1 on the following page it is clear that the velocity distribution rapidly changes from a uniform to a gaussian shape. Figure 2 on the next page indicates that this happens exponentially, i.e.

$$C(t) = 1 - C_0 \mathrm{e}^{-t/\tau}\,,$$

where $C(t)$ is the correlation, C_0 is the initial correlation and τ is a time constant. If this is the case,

$$\ln(1 - C) = \ln\left(C_0 \mathrm{e}^{-t/\tau}\right) = \ln(C_0) - t/\tau\,,$$

so when plotted on a logarithmic scale, $1 - C(t)$ should be a linear function with slope $-1/\tau$. The result is shown in figure 3 on page 3, and while the noise increases as the correlation approaches unity, it is clear from the first half of the graph that the trend is linear, confirming the exponential approach to 1.

The time constant, τ, can thus be estimated by picking out the linear part of the data in figure 3 on page 3 and finding the slope. While this sounds simple, picking out a linear bit of a graph is hard to program. Fortunately,

$$C(\tau) = 1 - C_0 \mathrm{e}^{-\tau/\tau} = 1 - C_0/\mathrm{e} \implies 1 - C(\tau) = C_0/\mathrm{e} \implies \frac{1 - C(\tau)}{1 - C(0)} = \frac{1}{\mathrm{e}}\,,$$

so the time constant can also be found by checking when $1 - C$ has reached $1/\mathrm{e}$ of its initial value, which is very simple to program. The result is

$$\tau = 6.1 \text{ ps}\,.$$

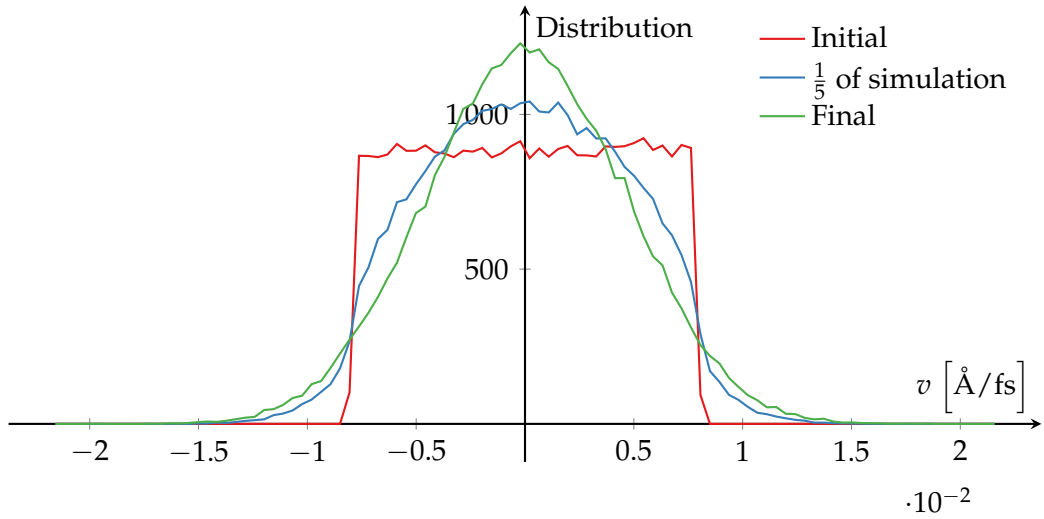

Figure 1: Distribution of particle velocities in the initial and final configurations, as well as after one fifth of the simulation time. The shape changes rapidly from uniform to gaussian.

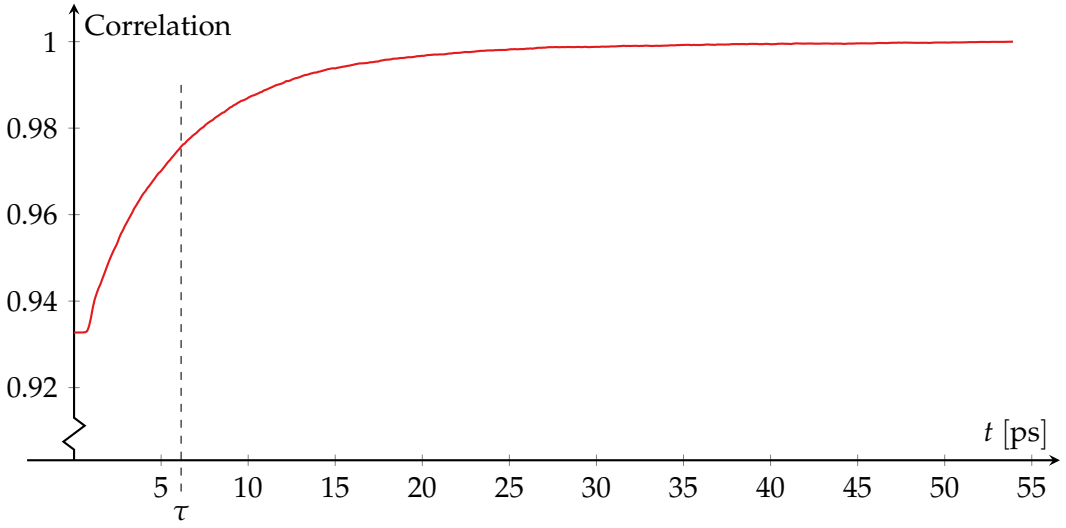

Figure 2: Correlation of the velocity distribution as a function of time. The approach to 1 appears to be exponential, as confirmed by figure 3 on the next page.

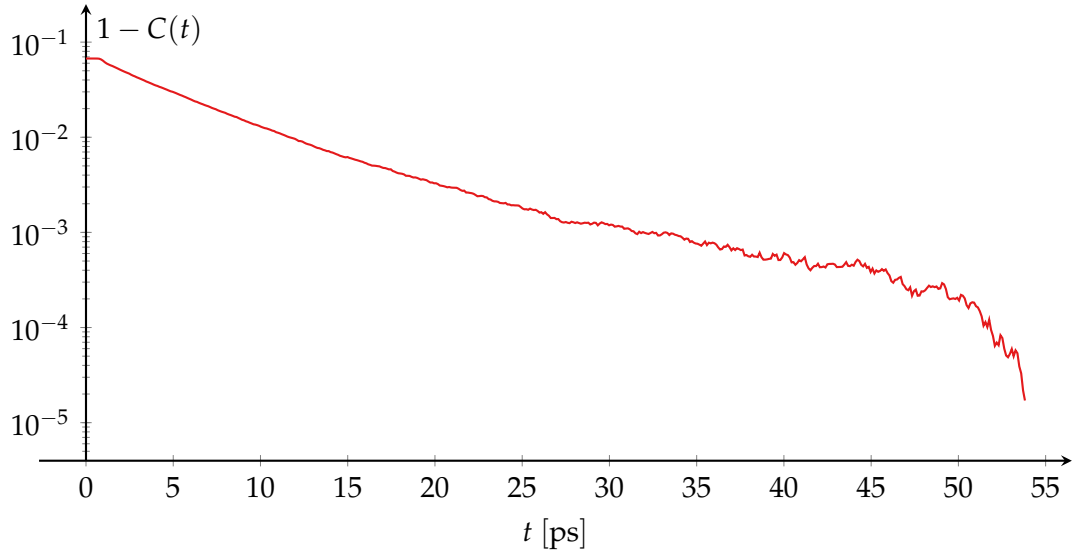

Figure 3: Deviation of the correlation from 1, plotted on a logarithmic scale. The linear trend of the first part indicates exponential decay. As the velocity distribution approaches the final distribution, the noise starts dominating the deviation.

b)

In order to find the time dependence of the total energy (which should be constant in an NVE-system) as well as the fluctuations, I have made a simple python script which goes through a set of time steps and runs a LAMMPS script for each time step. Between the runs, the energy is read from the log file and the standard deviation is calculated.

The time step starts at $\Delta t = 0.001 t_0$, where t_0 is the characteristic time for argon. As the time step is gradually increased, the system starts losing atoms, causing the simulation to crash. The largest time step used is $\Delta t = 0.018 t_0$.

As seen from figure 4 on the following page, the energy conservation is much worse when a large time step is used. Figure 5 on the next page is an attempt at quantifying the effect of the time step, but the results are hard to interpret.

c)

In order to find the time dependence of the temperature as well as the fluctuations, I have made a simple python script which goes through a set of system sizes and runs a LAMMPS script for each system size. Between the runs, the temperature is read from the log file and the standard deviation is calculated.

The size starts at $L_x = L_y = L_z = 10a$, where a is the chosen initial unit cell size for argon. The largest size used is $L_x = L_y = L_z = 20a$.

Figure 6 on page 5 shows the temperature as a function of time for the smallest and largest system sizes. While the equilibrium temperature and equilibration time are the same, the fluctuations are

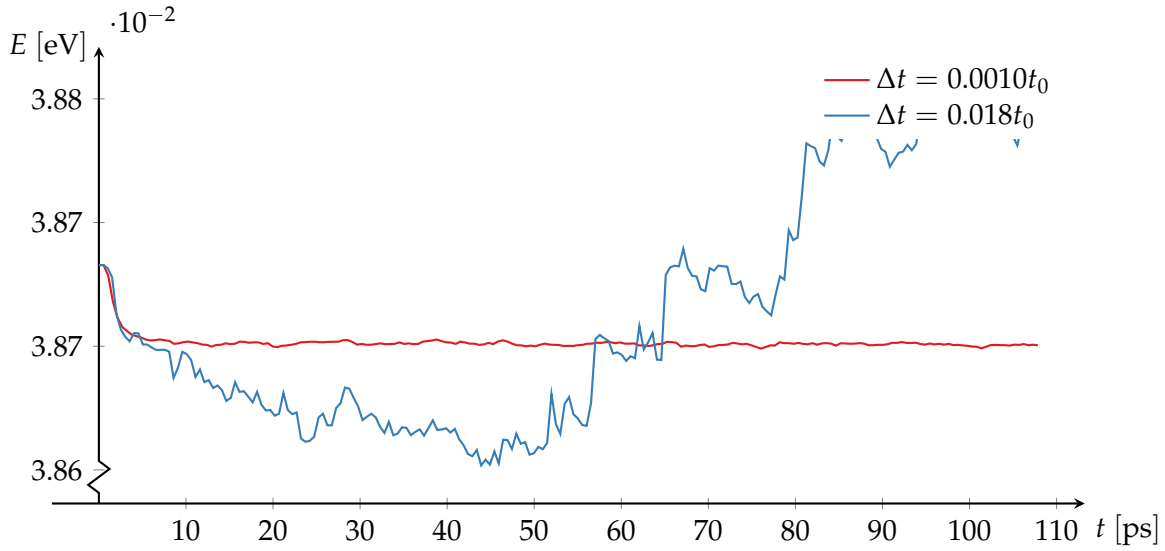

Figure 4: Total energy as a function of time for the largest and smallest time steps used. It is clear that the energy conservation is better when a small time step is used.

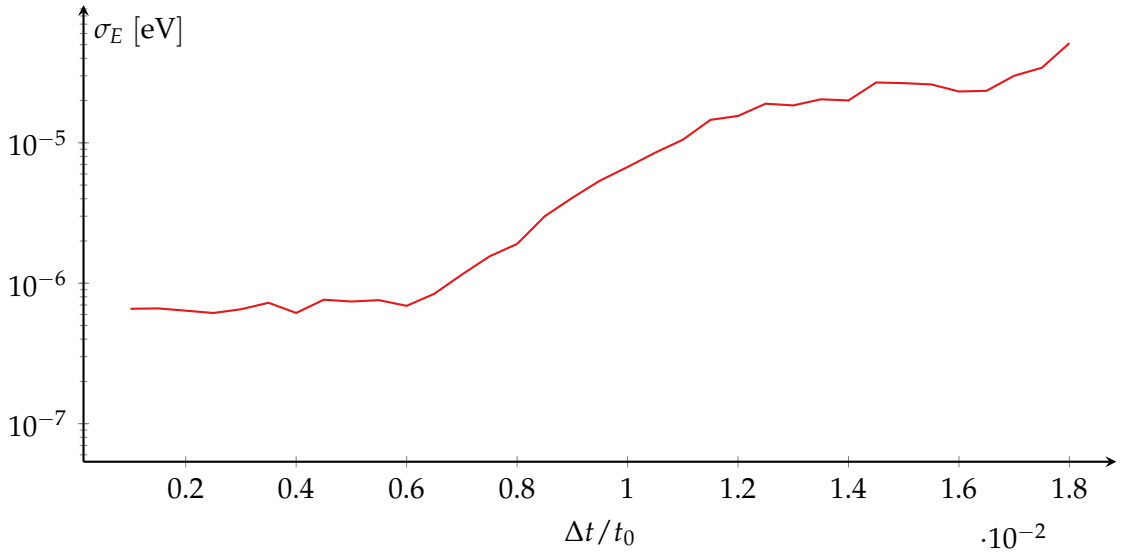

Figure 5: Standard deviation of the total energy as a function of the time step.

much greater when the system is smaller. Figure 7 on the next page shows the standard deviation as a function of system size. Note the logarithmic scale on both axes.

The linear trend of $\ln(\sigma_T)$ as a function of $\ln(L)$ indicates that the fluctuations are proportional to some power of the system size, as

$$\ln(\sigma_T) = C\ln(L) + D = \ln\left(L^C\right) + D \implies \sigma_T = \mathrm{e}^{\ln\left(L^C\right)+D} = \mathrm{e}^D \mathrm{e}^{\ln\left(L^C\right)} = \mathrm{e}^D L^C \propto L^C \,.$$

Numpy's polyfit function gives the result $C = -1.11 \approx -3/2$. This can be rewritten as

$$\sigma_T \propto \frac{1}{L^{3/2}} \propto \frac{1}{\left(N^{1/3}\right)^{3/2}} = \frac{1}{\sqrt{N}} \,,$$

which is a known theoretical result.

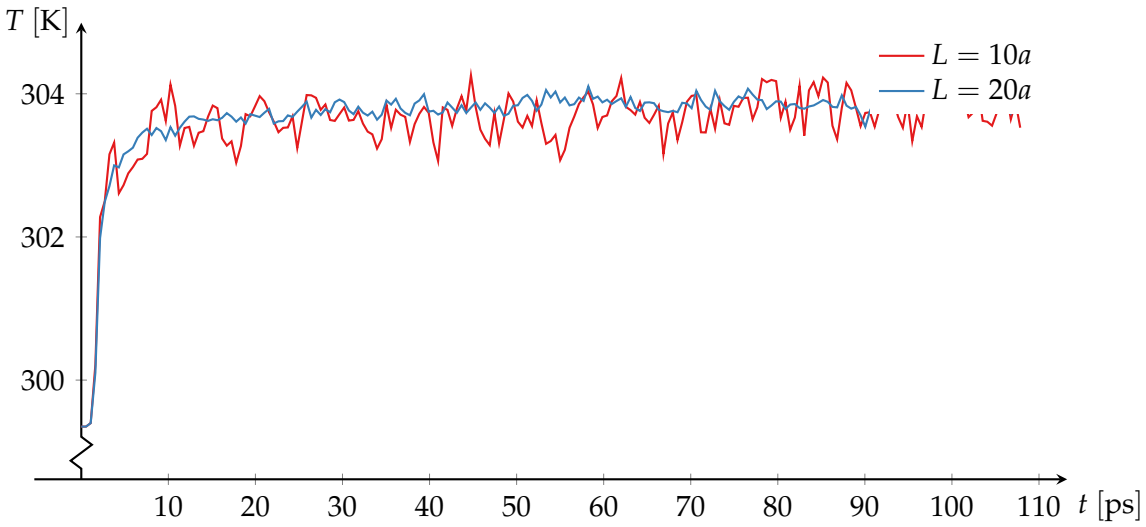

Figure 6: Temperature as a function of time for the largest and smallest system sizes used. It is clear that the temperature fluctuations decrease with system size, while the equilibration time and equilibrium temperature do not depend on the number of atoms.

d)

In order to find the temperature dependence of the pressure, I have made a simple python script which goes through a set of temperatures and runs a LAMMPS script for each temperature. Between the runs, the pressure is read from the log file, an equilibration time is found, and the mean pressure after the system has reached equilibrium is calculated.

Figure 8 on the following page shows the pressure as a function of time for the smallest and largest temperatures. Figure 9 on page 7 shows the pressure as a function of temperature. The result is very linear, as expected from the van der Waals equation of state,

$$\left(P + a\left(\frac{N}{V}\right)^2\right)(V - Nb) = Nk_\mathrm{B}T \,.$$

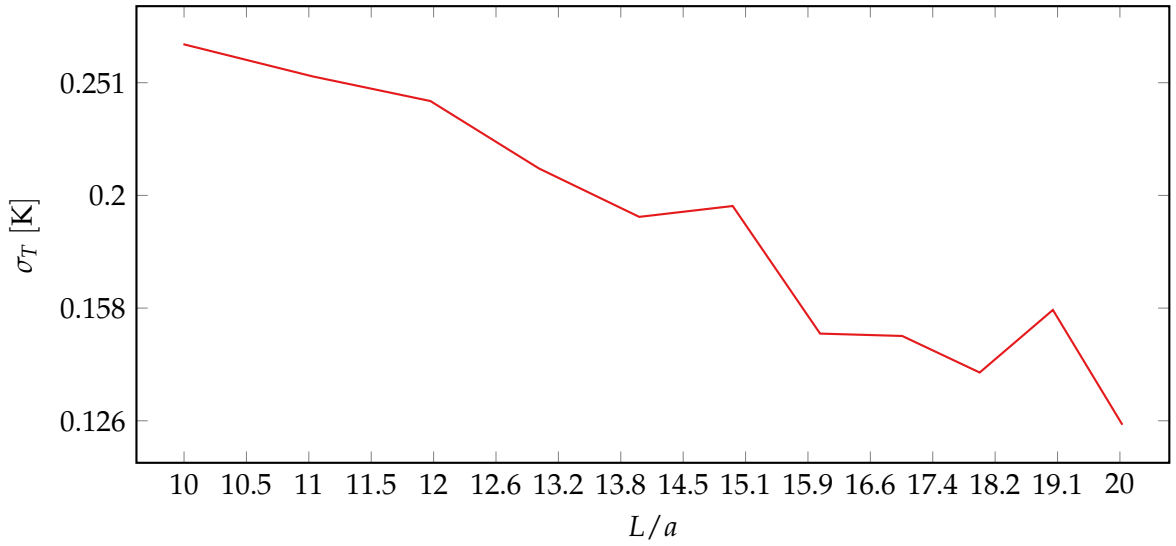

Figure 7: Standard deviation of the temperature as a function of the system size. Both axes are logarithmic, so the linear trend indicates that the fluctuations are proportional to some power of the system size.

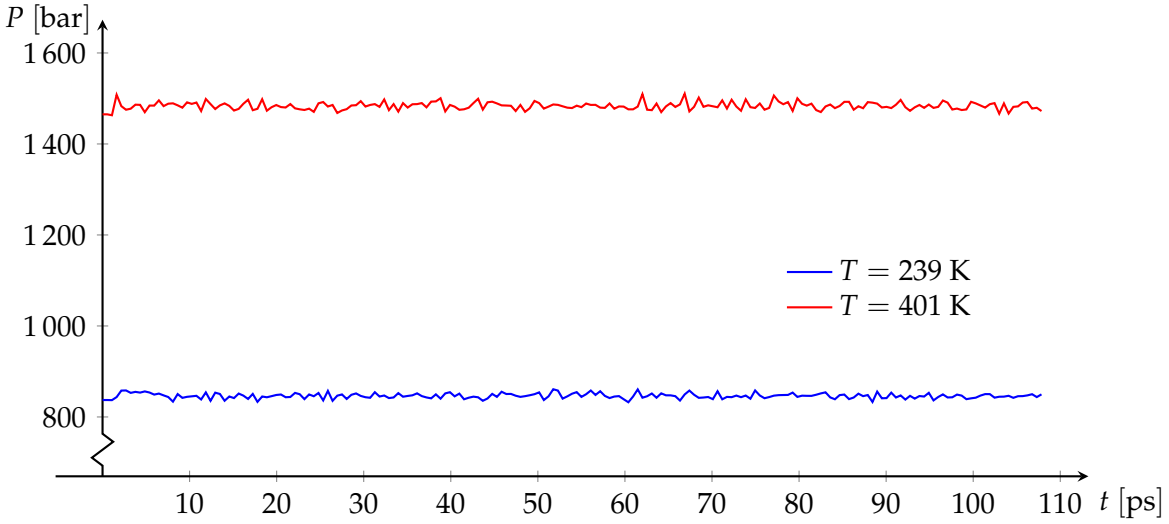

Figure 8: Pressure as a function of time for two different temperatures. The shapes of the graphs are approximately the same, but they stabilise at different values.

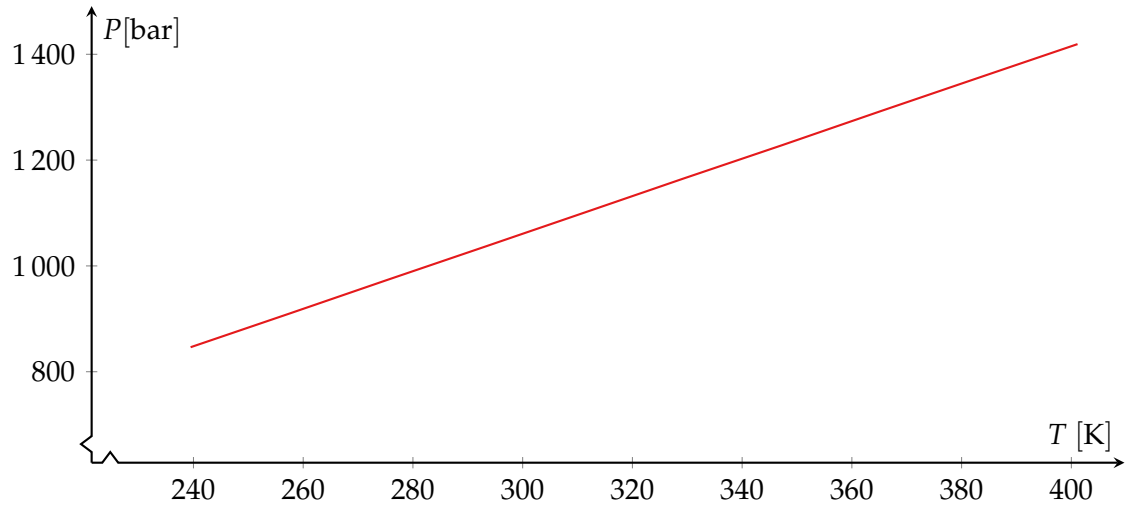

Figure 9: Pressure as a function of temperature. The result fits well with the expectation of a linear dependence.

e)

Ideally, the equation of state for the system should be the van der Waal equation of state,

$$\left(P + a\left(\frac{N}{V}\right)^2\right)(V - Nb) = Nk_{\mathrm{B}}T\,,$$

which is a modified version of the good ol' ideal gas law, $PV = Nk_{\mathrm{B}}T$. The modifications are $V \to V - Nb$, which represents the fact that the atoms take up some space, and $P \to P + a(N/V)^2$, which incorporates the added pressure due to collisions. Together, these changes make the equation suitable for describing a system where atoms collide elastically, but otherwise do not interact. In this part of the project, the atoms are modelled as interacting through short-range conservative forces, which should give a behaviour reasonably closed to the target system of the van der Waal equation of state.

By introducing the density, $\rho = N/V$, and dividing by N, the equation of state can be rewritten as

$$\left(P + a\rho^2\right)\left(\rho^{-1} - b\right) = \rho k_{\mathrm{B}}T \implies P = \frac{\rho k_{\mathrm{B}}T}{\rho^{-1} - b} - a\rho^2 = \rho^2\left(\frac{k_{\mathrm{B}}T}{1 - \rho b} - a\right).$$

The parameters a and b can be determined through running simulations for many pairs of (ρ, T). A fitting function from scipy determines the coefficients which make the predicted equation of state fit as well as possible with the obtained data.