Physics of Soft Condensed Matter Project

Vijayraj Shanmugaraj, 20171026

This is the report of the project and the corresponding code for the Physics of Soft Condensed Matter course, Spring '21.

Components of this project

The following tasks were given for the project:

- Generate a random configuration of 108 Argon atoms in a box of side length 18Å such that they obey periodic boundary conditions
- Minimize the potential energy of the above configuration, since the random configuration might have resulted in a higher energy state
- Create 2000 frames of the moving argon atoms, following Hamilton's equations of motion, generating trajectories and velocities of the atoms with time
- Calculate Mean square displacement, Velocity correlation function, Van Hove correlation function, and Dynamic Structure Factor of the system

This report will describe how each task was implemented, and discuss the results of the same.

Task 1 : Random configuration generation

Note: the code for this subsection is in **`init_config.py`**

For this, we first start out with an empty list of atoms. The algorithm goes on ad follows

Random configuration algorithm

while number of atoms < 108 *do*
    pick a random coordinate in space → c
    for atom *in* already placed atoms *do*
        if minimum image distance (c, atom) ≤ 3.4Å
            discard c; break
    if minimum image distance condition is satisfied for all placed atoms
        append c to list of atoms
    end
end

The minimum image vector of a point *p* (p_x , p_y , p_z) is the shortest vector from the concerned point *t* from the seven vectors p, ($p_x \pm 18$, p_y , p_z), (p_x , $p_y \pm 18$, p_z) , (p_x , p_y , $p_z \pm 18$). (Basically the shortest distance of p's image from all six sides of the bounding box.

Task 2 : Minimizing potential energy of the system

For this, we take the initial coordinates and we perform gradient descent as follows:
(For calculating V_i and F_i , the following formulas were used):

$$Vi = \sum_{j \neq i} V_{ij} = 4\varepsilon \times \sum_{j \neq i} \left(\frac{\sigma}{rij}\right)^{12} - \left(\frac{\sigma}{rij}\right)^{6}$$

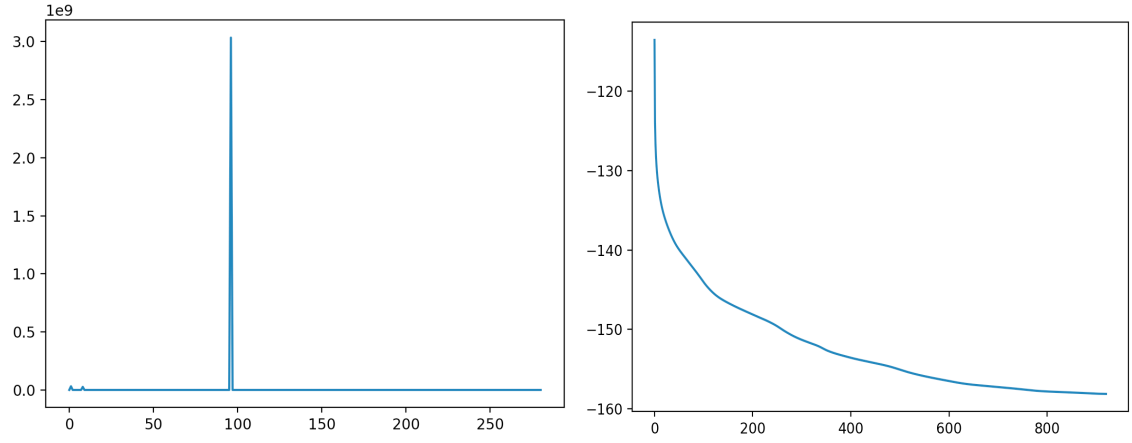
$$Fi = \left(\frac{-dVi}{d(rij)}\right)$$

F_i is the magnitude of force, it will be in the direction of r_{ij}.

Steepest descent algorithm

i = 0
while True *do*
    for every atom *do*
        Calculate $V_i(r_i)$, $F_i = -\nabla V_i(r_i)$
        $r_{i+1} \leftarrow r_i + \alpha F_i$
    end
    Calculate $U_i \leftarrow \sum V_{i+1}$
    if $| U_i - U_{i+1} | \leq \partial$, *break*
end

The initial configuration had an energy of -113/M kcal (where $M = 6.022 \times 10^{23}$). I experimented with 3 values of α: 1, 0.1, 0.01

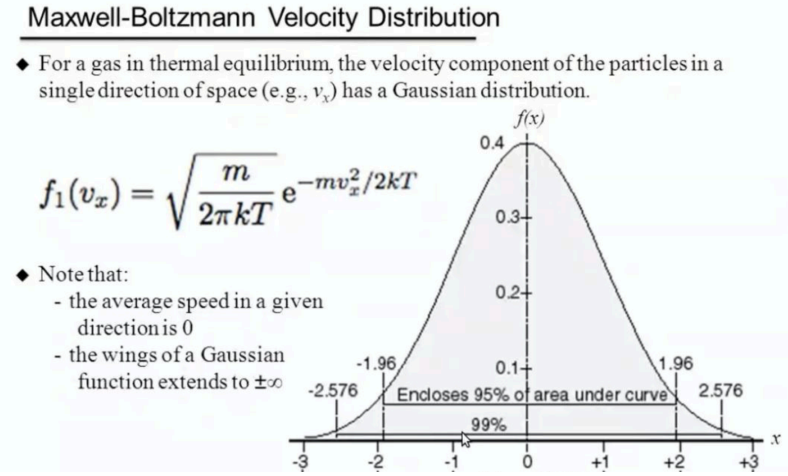


Right: α = 1, Left: α = 0.1

The issue with α (step size) = 1 was that the step size was too large, and hence the gradient descent was jumping too far across the solution space, and hence finally terminated without finding a maxima, in fact ending in a final state higher than the initial state (nearly at an energy of 0 kcal). Setting the step size at 0.01 took too much time, and did not terminate even after an hour (after about 3200 iterations, the energy of the system was at -137/M kcal, which is clearly not the expected minima). For step size 0.1, the system gave best results, converging satisfactorily at around -158/M kcal).

Task 3 : Frame generation

The initial velocities of all the atoms were sampled according to the normal distribution given below (basically, the velocity in each direction had μ = 0 and σ = √(kT/m), where k is the Boltzmann constant, m is the mass of a single argon atom, and T is the temperature.



Now for calculating the trajectory and velocity evolution with time, instead of solving the Hamiltonian and Langevin equations, we use the Velocity Verlet Algorithm. For each atom, we run this algorithm for 2000 (number of frames) iterations. (Here Δt is taken as 4 femtoseconds) . (*Note*: F_i is the sum of all F_{ij}'s, each in the direction of the minimum image vector of atom j w.r.t. atom i)

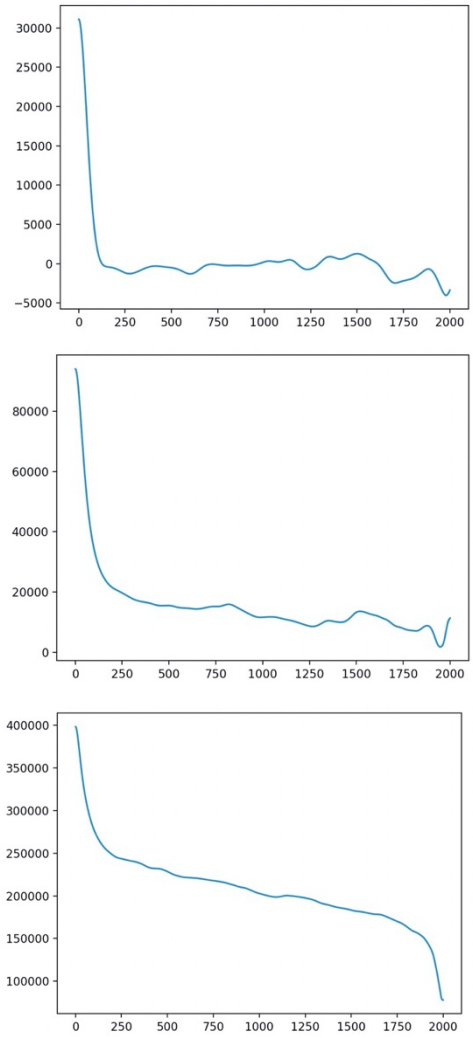
1. Calculate $\vec{v}\left(t+\frac{1}{2}\Delta t\right) = \vec{v}(t) + \frac{1}{2}\vec{a}(t)\,\Delta t$.
2. Calculate $\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}\left(t+\frac{1}{2}\Delta t\right)\,\Delta t$.
3. Derive $\vec{a}(t+\Delta t)$ from the interaction potential using $\vec{x}(t+\Delta t)$.
4. Calculate $\vec{v}(t+\Delta t) = \vec{v}\left(t+\frac{1}{2}\Delta t\right) + \frac{1}{2}\vec{a}(t+\Delta t)\Delta t$.

In my code I have eliminated intermediate steps and condensed the algorithm as shown below:

1. Calculate $\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}(t)\,\Delta t + \frac{1}{2}\vec{a}(t)\,\Delta t^2$.
2. Derive $\vec{a}(t+\Delta t)$ from the interaction potential using $\vec{x}(t+\Delta t)$.
3. Calculate $\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{1}{2}\left(\vec{a}(t) + \vec{a}(t+\Delta t)\right)\Delta t$.

Task 4 : Calculating function plots

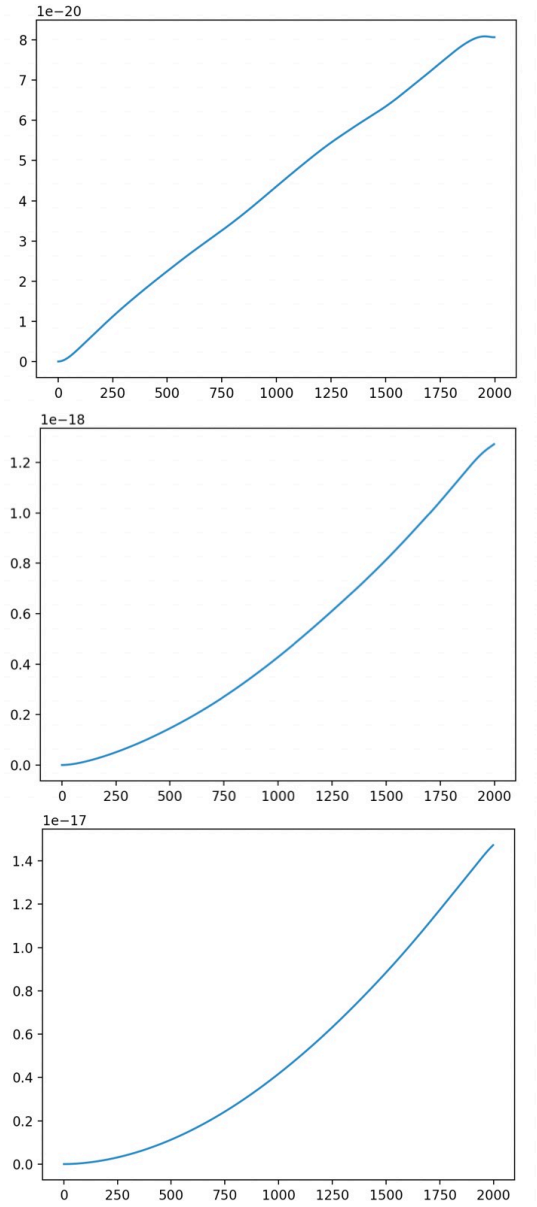
1. Velocity Correlation function



Plot 1: From top to bottom: 100K, 300K, 1000K

The velocity correlation plots clearly demonstrate the behaviour of argon atoms with respect to the temperature of the system. At lower temperatures, the effect of potential energy and collisions due to the other argon atoms can actually be observed, and it's clearly visible that the atoms change direction of their movement after a good number of frames (the velocity correlation function goes negative. But as temperature increases, we can see that the velocity does not dip, and the particles slowly tend to maintain velocities, especially at 1000K, where the velocity does dip into the negative side, showing that the atoms move almost undeviated.

2. Mean Square Displacement



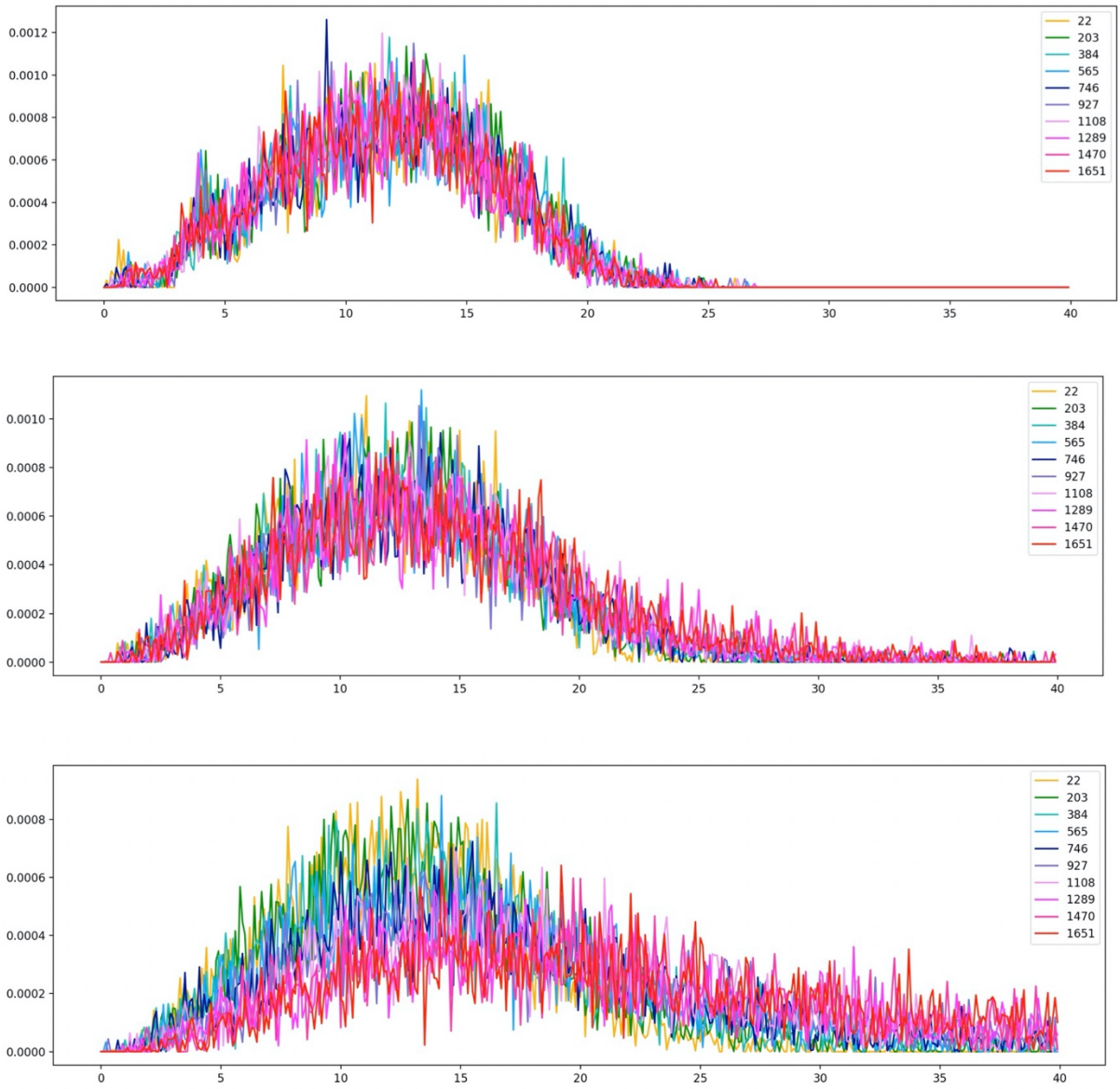
Plot 2: MSD (Displacement versus time gap plot for 100, 300 and 1000K;

Table below: Value of Diffusion constant (D) for different values of T

Temperature	*D (SI values)*
100K	1.43e-09
300K	3.81e-08
1000K	4.9e-07

The MSD vs time gap plots are as expected, almost tending to a straight line, in accordance with the formula $<X^2> = A + 6DT$. D also increases with temperature. This is intuitive since the mean square displacement will increase with temperature, due to the higher kinetic energy as a result of higher temperatures.

3. Van Hove Function



Plot 3: Van Hove Function plot for T = 100, 300, 1000K

The Van Hove correlation function is defined as the probability of finding a particle at a distance 'r' at time t, given that there was a particle at the origin at time t. Now at lower temperatures (at 100K), we can see that the probability of finding a particle at time t at a distance > 26Å is zero. However, this probability increases as temperature increases. At 300K, the probability of finding an atom at distance 40Å is non-zero, and at 1000K, the probabilities at distance 40Å is actually significant, and the distinction between peaks for different time gaps are clear. For Δt = 88fs, the peak of G(r, Δt) is significantly high at around 12-13 Å. As the time gap increases, the probability of finding atoms at various distances becomes more uniform, with the peak at ≅ 20 Å. It is clear that due to higher temperatures, atoms tend to move away from the origin more often, and the above graphs demonstrate the argument very well.

4. Dynamic structure factor

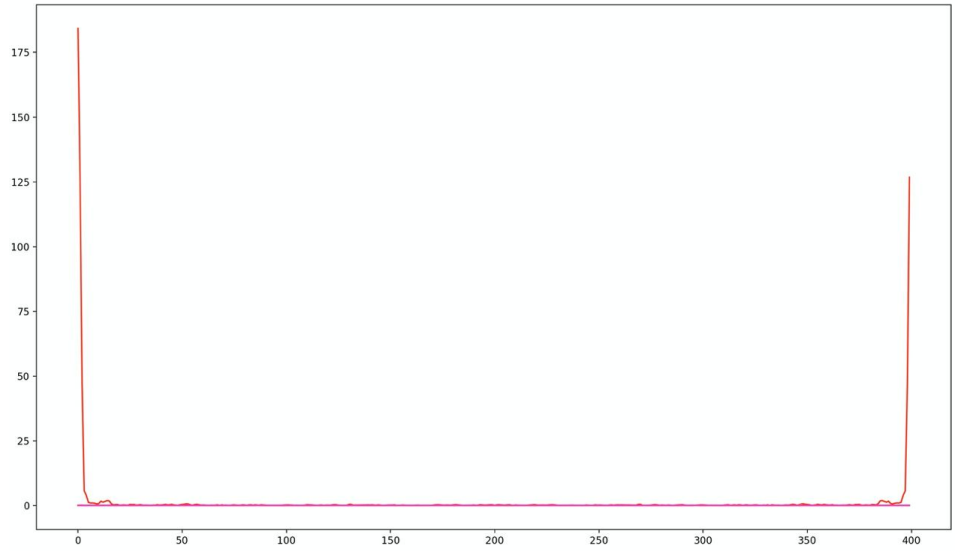
The algorithm to calculate the DSF was as follows:

F(k, t) = Fourier transform of the Van Hove function.

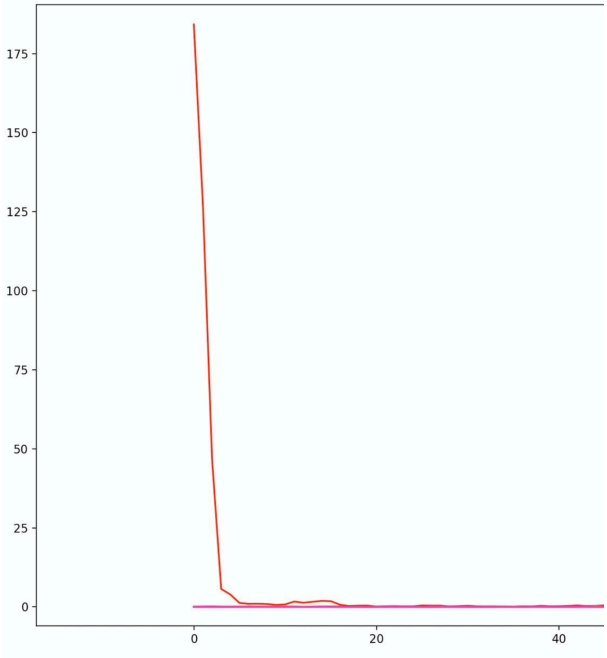
I calculated the Vanhove function for r from 1 to 40Å (G_r_t being the 2D array for the same) and for time gap t = 0 to 1999. I had then taken the discrete Fourier transform of each row (since we needed the Fourier transform wrt space for different values of time. Then I did a time Fourier transform (Since S(k, w) = time Fourier transform of F), and then plotted the value of S(k) (absolute value of the complex number) vs k for different w.

```
def S():
    G_r_t = np.zeros((len(d), 40))

    for time in tqdm(range(len(d))):
        for r in range(40):
            G_r_t[time][r] = G(r, time)

    F_k_t = np.array([np.fft.fft(row) for row in G_r_t])
    F_k_t_T = F_k_t.T

    S_k_w = np.array([np.fft.fft(row) for row in F_k_t_T])
```

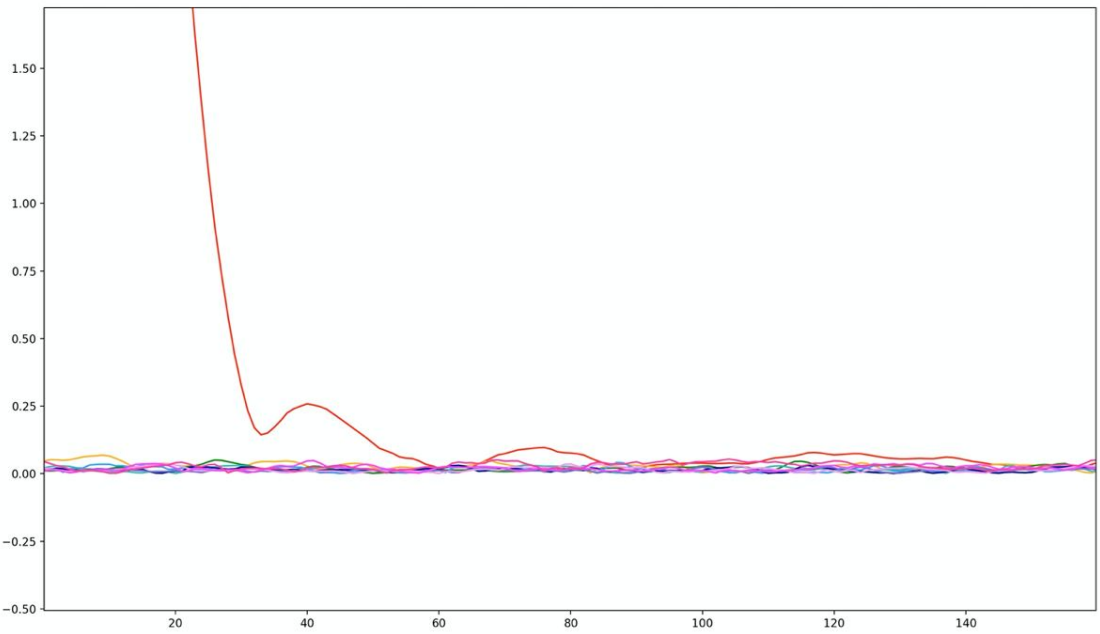
The plot I got was as follows:



On zooming into the first end, this is what we see



The red line corresponds to ω = 0. For other values of ω, the DSF seems to oscillate around the 0 point.



The above plot was the plot for 1000K.