Molecular Dynamics of Lennard-Jones System

Abeer Karandikar and Sharvil Sankriti
UG 24

Prof. Bikram Phookun

Graduate Assistant: Philip Cherian

December 14, 2023

Abstract

In this project we simulate a system of particles interacting through the short-range Lennard-Jones potential. We simulate the system at constant temperatures which can be set as desired in order to get phase transitions. We detect the presence of phase transitions through obtaining the Radial Distribution Function. In an attempt to move to larger systems we modify the simulation function using the nearest neighbour method to implement a cut-off radius on the potential.

1 Introduction

Computational Molecular Dynamics (CMD) is a powerful and interdisciplinary approach in the realm of scientific research that plays a pivotal role in unraveling the mysteries of molecular behavior. At its core, CMD leverages advanced computational techniques to simulate the dynamic motion of molecules and study their interactions at the atomic level. By employing principles from physics, chemistry, and computer science, CMD enables scientists to explore the intricate dance of atoms and molecules, providing valuable insights into the fundamental processes that govern the behavior of matter.

In essence, CMD serves as a virtual laboratory where researchers can investigate the thermodynamics, kinetics, and structural properties of molecular systems without the constraints of traditional experimental methods. This computational approach has proven instrumental in various scientific fields, including chemistry, materials science, and biochemistry, offering a unique perspective that complements experimental observations.

In this project, we attempted to understand and implement the concepts of basic molecular dynamic simulations, through the apparatus of the Lennard-Jones gas, building upon understanding derived from the Classical Ideal Gas. In this report, we discuss the techniques involved, the computational efficiency and the theoretical background that goes into such a simulation.

2 Theoretical Background

2.1 The Classical Ideal Gas

The classical ideal gas model serves as a fundamental framework for understanding the behavior of gases under varying conditions. The basic physics behind the description of the gases involve the following assumptions

- The particles do not interact with each other through a potential field
- When the particles collide with each other, these collisions are elastic and no energy or momentum is lost
- the particles are point particles and occupy negligible volume compared to the size of the container
- the particles are free particles and move about randomly.

However, these assumptions lead to theoretical predictions that are far from those observed in the lab. This is because most of these assumptions are not true practically and hence, there are many different kinds of corrections to the understanding of the behaviour of gases.

2.2 The Lennard-Jones Gas

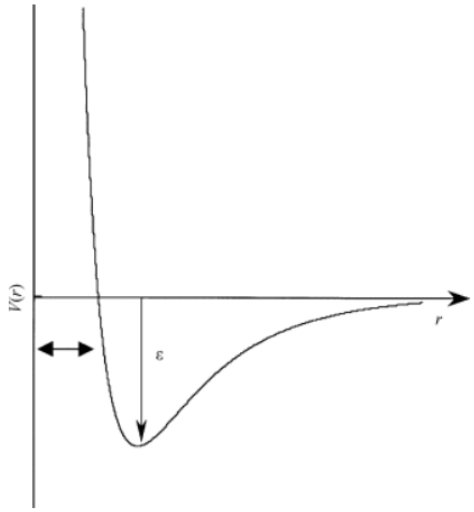

Figure 1: The Lennard-Jones potential. Image source: https://www.sciencedirect.com/topics/physics-and-astronomy/lennard-jones-potential

The Lennard-Jones potential, named after John Lennard-Jones who introduced it in 1924, stands as a widely used mathematical model for describing the interaction energy between pairs of neutral atoms or molecules.

The Lennard jones potential is a short range potential field that graphs itself mathematically, such that at small separations, particles repel each other, and intermediate distances, they attract each other and at larger distances, particles effectively do not interact with each other.

The potential is described as a function of separation by the equation

$$u(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{1}$$

And in our project, we try to simulate the motion of the particles in a gas that interact with each other through this potential field.

2.3 Radial Distribution Function

In this project, we are looking at a Lennard-Jones gas and simulating its dynamics at various temperatures. Thus, as we move to different temperatures we observe phase transitions. We need to find a reliable way of detecting the occurrence of such phase transitions. The Radial Distribution Function ($g(r)$) is a way of looking at the different spacial orderings seen in the different phases. In crystalline solids, for example, the location of each particle is very highly correlated to that of all the other particles (Patharia, Beale 2011).

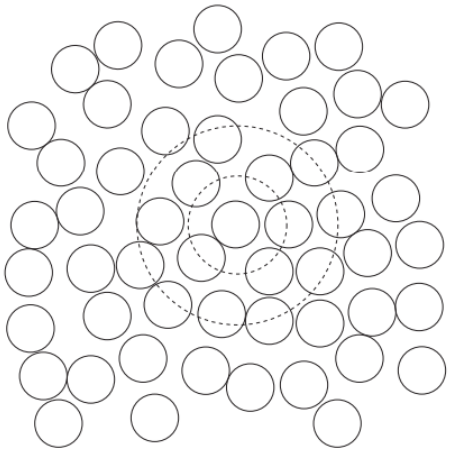

Figure 2: The Number of Particles present in a Shell. Image Source: *Statistical Mechanics, Third Edition* Patharia, Beale 2011

The RDF is a measure of the number density of particles in a shell of thickness $r + dr$ drawn around each particle and then averaging over all particles. Thus we get the following normalisation condition for $g(r)$,

$$\rho \int g(r) d\mathbf{r} = N - 1 \tag{2}$$

For an ideal gas we consider no correlations thus the radial distribution gives $g(r) = 1$ for all r while for a Lennard-Jones gas we have, for $r \longrightarrow 0$, $g(r) \longrightarrow 0$. This is due to the repulsive force between the particles when they are close to each other. For particles farther away, the correlation drops off and hence we get for $r \longrightarrow \infty$, $g(r) \longrightarrow 1$. The methods used to computationally obtain the RDF is detailed in the later sections. The RDF can also be used to calculate various physical variables such as pressure, potential energy of the system being simulated. We get the following relation between the pressure (P) and temperature (expressed in units of energy so actually $k_B T$, but here written as T) of the system,

$$\frac{PV}{NT} = 1 - \frac{\rho}{2Td} \int g(r) r \frac{du(r)}{dr} d\mathbf{r} \tag{3}$$

where d is the number of dimensions hence, here 2. We get the potential energy per particle as,

$$\frac{U}{N} = \frac{\rho}{2} \int g(r) u(r) d\mathbf{r} \tag{4}$$

where, the potential energy between the particle and all other particles in a shell of thickness $r + dr$ is $u(r)\rho g(r)$ (Gould, Tobochnik, Christian 2016).

The Lennard-Jones potentials are used to describe noble gases. Where the attractive term is due to the dipoles created temporarily by fluctuations in the electron cloud. This term falls as the inverse sixth power of the separation between the atoms and the repulsive term is due to the overlap of the electron clouds when atoms are close to each other this term decays as the inverse twelfth power of the separation between the atoms (Adams 2001).

3 Aim Of the Project

The aim of our project is as follows

- To simulate the molecular dynamics of the Lennard Jones system
- To simulate a system where the desired temperature can be provided by the user and be maintained by the system
- to Observe solidification of our system at low temperatures
- To observe phase transitions within the system (Solids, Liquids and Gases)
- To improve the algorithm in order to make it possible to simulate the molecular dynamics of a larger number of molecules

4 Techniques involved

4.1 Simulating a Lennard-Jones Gas

We set some of the parameters from equation (2.2) to make simulations more convenient, we have $m = 1$, $\epsilon = 1$, $\sigma = 1$. Thus we use the following expression for the potential,

$$u(r) = 4\left[\left(\frac{1}{r}\right)^{12} - \left(\frac{1}{r}\right)^{6}\right] \tag{5}$$

To simulate the dynamics of the particles we use an integration routine called the Verlet method. The algorithm is quite simple and reads as follows (Verlet 1967):

1. Obtain the intermediate position $\longrightarrow$ after a half time-step the position is updated Using previous position and previous velocity. As,

$$r_i(t + 0.5\Delta t) = r_i(t) + \frac{\Delta t}{2} \times v_i(t)$$

2. Update new velocity $\longrightarrow$ after a full time-step, using previous velocity and acceleration, the new velocity is obtained. As,

$$v_i(t + \Delta t) = v_i(t) + \Delta t \times a_i(t + \Delta t/2)$$

3. Obtain the new position $\longrightarrow$ After a full step, using new velocity and intermediate position, the new positions are obtained as,

$$r_i(t + \Delta t) = r_i(t + \Delta t/2) + \frac{\Delta t}{2} \times v_i(t + \Delta t)$$

Using this integration routine we can keep track of the motion of all the particles. We conduct the simulation in what can be thought of a L×L box. Now, we have to take care of what happens to a particles when they reach the edge of this box.We do this by introducing periodic boundary conditions. Which means that if a particles reaches an edge, say the right edge, it is reintroduced back into the box from the opposite, here left, edge.

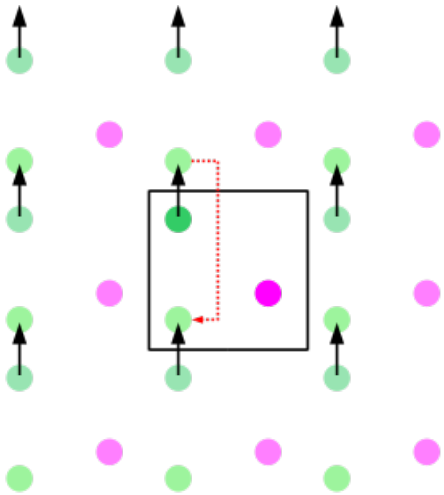

Figure 3: Schematic Representation of the Periodic Boundary Conditions Implemented. Image Source: https://en.wikipedia.org/wiki/Periodic-boundary-conditions

We implement these as follows:

1. if $x > L \longrightarrow x \longrightarrow x - L$
2. if $x < 0 \longrightarrow x \longrightarrow x + L$

4.2 Andersen Thermostat

In our simulation we wanted to simulate our system at predefined temperature. So that we could simulate it at various temperatures. One way of keeping a thermodynamic system at a predefined constant temperature is to keep it in contact with a heat bath. We simulate this computationally by using a method called the Andersen Thermostat. In our simulation we need to define two quantities ν- the mean rate at which each particle suffers a stochastic collision and T- the desired temperature (Andersen 1980). The Andersen thermostat is implemented as follows:

1. After each time-step a particle is randomly selected and is assigned a velocity drawn from the Maxwell-Boltzmann distribution with a probability $\nu\Delta t$.
2. This process is repeated till equilibrium is attained.
3. To "cool" the system to lower temperatures we need to increase the value ν i.e. increasing the thermal contact with the heat bath in order to get to a low temperature within the time of the simulation.

4.3 Computing the Radial Distribution Function

At this stage, we could simulate the Lennard-Jones system at predetermined constant temperatures which allowed us to simulate the system in different phases. To detect the phase of the system we used the Radial Distribution Function. The behaviour and usefulness of the RDF has been already described now we look at how we calculated the RDF computationally,

1. The function takes the thickness of the shell for which the RDF is to be evaluated, the position array obtained from the simulation and a time-step which determines the configuration for which the RDF is to be calculated.
2. The input parameters set the configuration of particles for which the RDF is to be evaluated then we go over all pairs of particles and calculate the separation between each pair.
3. We then bin these separations into bins of thickness dr the number of particles in each bin is in essence what we need to get the RDF
4. The number of particles in each bin is normalised by dividing it by the thickness of the shell that bin corresponds to.
5. This process is repeated and averaged over several time-steps to get the RDF

Listing 1: Function to calculate the Radial Distribution Function.

```
    @njit
def compute_RDF(dr,s,posa):
    # accumulate data for n(r)
    L1=((L)**2+(L)**2)**0.5 #largest possible distance between
       two particles
    g= L1/dr #number of bins
    r=np.array([0.]) #array to store the separations
#     r_count=np.zeros(shape=(N-1,int(g)))
    for i in range(N - 1):
         for j in range(i + 1, N):
                rn=pbc_distance(posa[s][j]-posa[s][i])
                r1=np.sqrt(rn[0]**2+rn[1]**2) #Getting the
                   separation between a pair of particles
                r=np.append(r,r1) #storing it in an array
    r = r[0:]
    hist, binedges = np.histogram(r, int(g))#dividing the
       values of separation in bins of thickness dr
```

```
15      return hist, binedges
```

4.4 Modified Simulation Function

The simulation of the dynamics of such a system can get computationally time intensive due to the way the pairwise interaction if the particles is calculated. At each time-step, we go over all the possible pairs and calculate the force between the two particles. This is an inefficient method which can be improved upon. The Lennard-Jones potential is a short-range potential and therefore falls off very rapidly (inverse sixth power of the separation) thus, we do not need to consider the pair-wise interaction between all the particles. Beyond the certain distance from a given particle, the interaction can simply be ignored. Before we can implement the cut-off radius, the potential needs to be modified. This is because of a sudden drop in the value of the potential due to the cut-off introduces a discontinuity which needs to be dealt with. Thus we modify the potential as,

$$\bar{u}(r) = u(r) - u(r_c) - \left(\frac{du(r)}{dr}\right)_{r=r_c} (r - r_c) \tag{6}$$

Now the cut-off radius can be implemented. In order to do so we need to keep track of the neighbours of any given particle at any given time-step. We did so as follows:

1. The function take a cut-off radius (slightly greater than the cut-off radius for the potential in (4.4)) and the positions of the particles at a particular time-step.
2. We start with an array of zeros for each particles then one by one go over all other particles and calculate the separation between the two particles. If the particle lies inside the cut-off radius the array element with the index of that particle is set to one. Thus we get an array of zeros and ones for each particle where an element being one means that the corresponding particle lies within the cut-off radius.
3. for each particle we loop over all particles including itself. The element of the array corresponding to itself is always set to zero for all particles.
4. Such lists are made and kept track of for all particles.

Listing 2: Function to obtain the neighbour list for a configuration of particles.

```
1     @njit
2 def computeNeighbourList(r_cutoff, pos):#Keeping track of the
    neighbours at a timestep
3     cumulativeNeighbours = np.zeros((N, N))#Cumulative list of
        the neighbour list of all particles
```

```
    #r=np.array([0.])
    for i in range(N):
        numberInList = np.zeros(N) #starting with zeros
        for j in range(N):
            rn=pbc_distance(pos[j]-pos[i])
            r1=np.sqrt(rn[0]**2+rn[1]**2)
            #r=np.append(r,r1)
            if r1 < r_cutoff:
                if j == i:
                    numberInList[j] = 0 #the element for the
                       particle ittself is always zero
                else:
                    numberInList[j] =+ 1 #If the particle is
                       within the cutoff, making the
                       corresponding element 1
        cumulativeNeighbours[i] = numberInList #list of all
           lists
    return cumulativeNeighbours
```

In the modified simulate function we can now use these neighbour lists to make the computation faster. We do so by firstly, for each particle looping over i.e. considering only the particles which are in its neighbour list i.e. are within the cut-off radius. We can make another modification by noting that it took approximately 17 time-steps for the particles to by a distance 0.2 away from each other. Thus we do not really need to keep calculating the neighbour list for each time-step. It is enough for us to update the neighbour list only about every 15^{th} time-step. After including these two changes, we get a simulation function which takes much less CPU time which will be discussed further in the next section (Gould, Tobochnik, Christian 2016).

5 Results

5.1 The Simulation Function

The simulation function simulates the dynamics of the Lennard-Jones system at a specified constant temperature. The images below show the initial and final configurations as well as energy conservation for one such sample run.

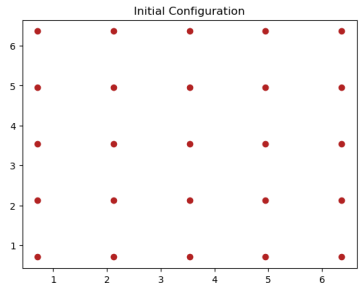

Figure 4: Initial configuration for a sample simulation

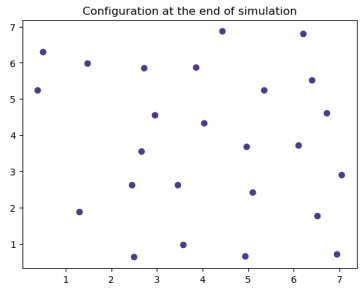

Figure 5: Final configuration for a sample simulation

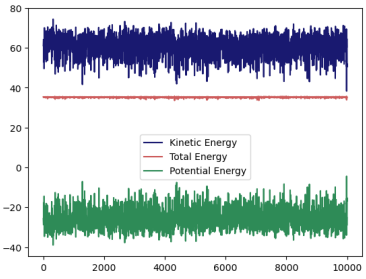

Figure 6: Energy conservation over a simulation

We can use this to simulate the system at a low temperature by increasing the parameter controlling the degree of thermal contact with the virtual heat bath ν to obtain the following final configurations showing the formation of a solid with ordering and crystalline structure.

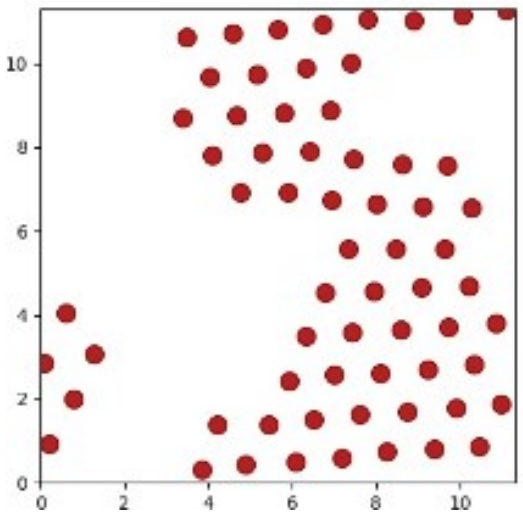

Figure 7: Solidification in an 8×8 lattice

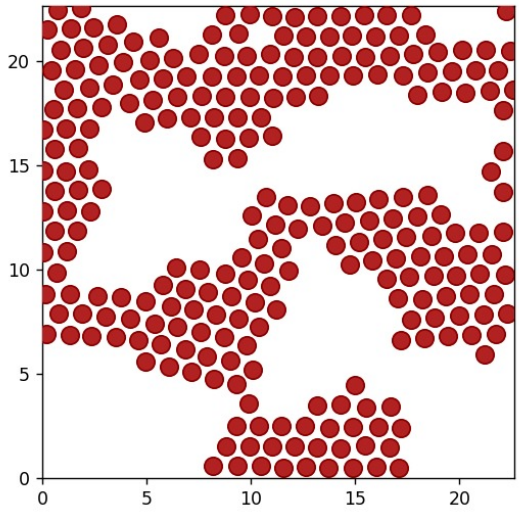

Figure 8: Solidification in an 16×16 lattice

5.2 Results Using he Radial Distribution Function

We calculated the RDF for various temperatures to see the nature of the graph for various phases. We obtained the following plots for the temperatures 0.1, 0.5, 2. The phases are distinguished by the second and third peaks in the graphs of the RDF. Solids have characteristic sharp second and third peaks. These are also seen for liquids but to a lesser degree and as we move to the gaseous phase the second and third peaks begin to vanish. This has been experimentally detected in supercritical Argon (Ghosh, Krishnamurthy 2018).

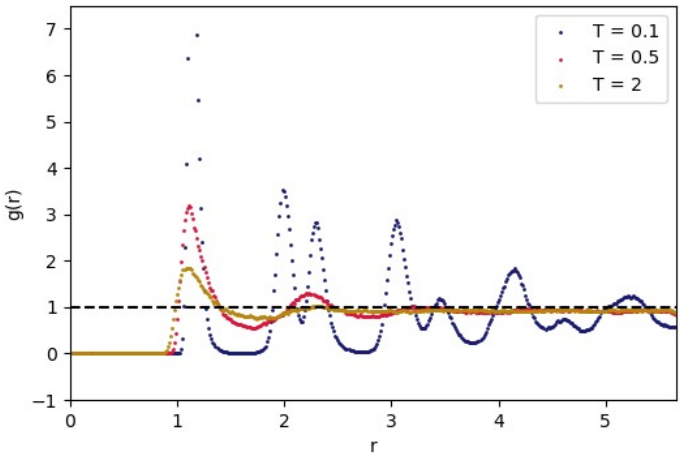

Figure 9: Graph showing the RDF obtained at various temperatures- 0.1, 0.5, 2

The following graph shows how the RDF evolves over time:

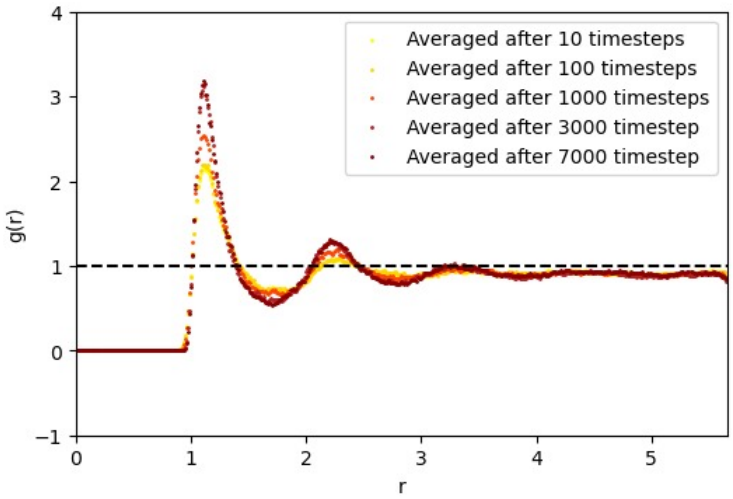

Figure 10: Graph showing RDF averaged over different range of time-steps. (3000) each time. To get a sense of how the RDF evolves.

We can compare the results obtained using the RDF calculated computationally by comparing them to the analytical results we have for pressure and energy. For pressure we use equation (2.3) to compare the behaviour of pressure with temperature analytically expected to that obtained computationally.

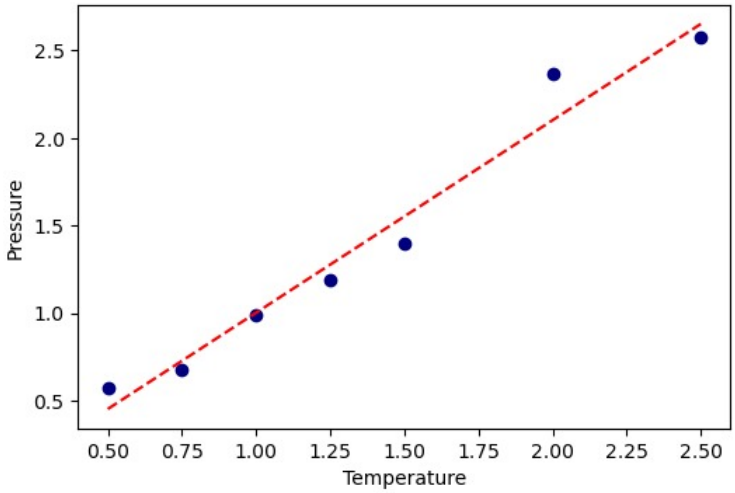

Figure 11: Graph showing the relation between the pressure and temperature obtained by calculating the pressure using the RDF

Rearranging the terms in equation (2.3), we get,

$$P = \rho T - \frac{\rho^2}{2d} \int g(r) \frac{du(r)}{dr} d\mathbf{r} \tag{7}$$

We worked with a system with $\rho = 0.5$ while the slope obtained from the graph is 0.55 which agrees with the expectation. We also observe a negative intercept. The possible reasons and implications of which are discussed in the next section.

For the potential energy, the magnitude of the potential energy obtained using the simulation is 107.341 while the value obtained from the radial distribution function is 120.251

5.3 Comparing Computational Times for the Simulation Functions

For simulating larger lattice sizes we observed the modified simulate function to be taking consistently lower CPU times compared to those taken by the old one. The graph below shows the same.

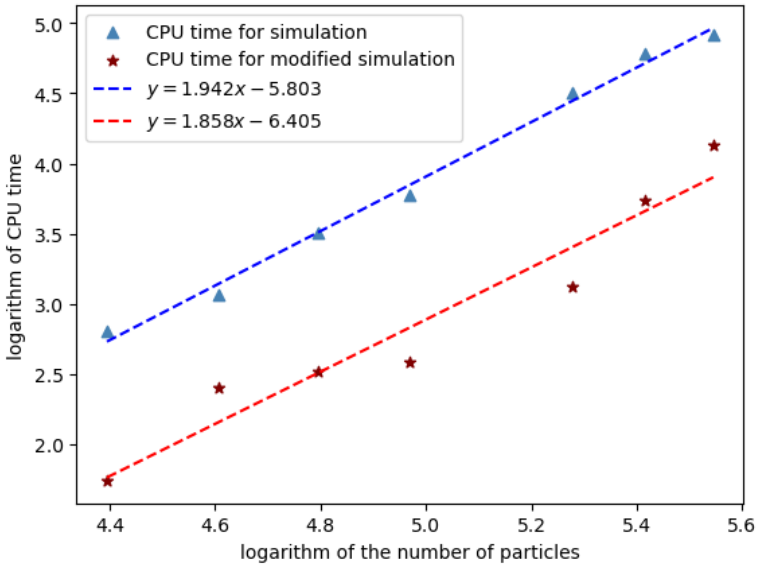

Figure 12: Comparison of CPU times for the original and modifies simulation functions. The log-log plot gives the dependence of CPU time on the lattice size

What we note here is that even though the times are smaller, the dependence still seems to be close to the order N^2. It is only slightly lesser that that for the old simulate function. The difference is expected to be seen much more starkly for larger lattice sizes but doing so at times leads to the old simulate function failing to finish running. Even as the modified simulation function gives us much faster computation times and thus the ability to simulate larger lattices which give more realistic systems to work with, the results obtained from it are not physical. There are consistently blow ups in the potential energy which we have been attempting to address which we suspect lead to these incorrect results. .

6 Discussion

6.1 New Simulate Function

We faced difficulties in trying to get the modified simulation function to work. Initially, it turned out that implementing the cut off radius was much slower than the older simulate function which just looped over all the particles a total of N^2 times. Hence, the function clearly was not doing the job it was meant to do, even though it ran.

On going through the code more carefully and adjusting for loops that did not have to be there (Redundant code), we managed to have the modified simulate function work much

faster than the old simulate function. We were capable of running a simulation for a 16×16 particle lattice which was tedious to do with the older simulate function been used. However, the results obtained were not physically sensible. The potential reasons identified for this are

- Improper enforcement of Periodic Boundary Conditions
- Blow ups in the velocity of the particles

6.1.1 Improper Enforcement of Periodic Boundary Conditions

As the name suggests, this error can occur due to not enforcing boundary conditions correctly. Periodic Boundary Conditions (PBC's) are enforced in general at the start of the loops before the implementation of the Verlet Scheme, and on obtaining the new positions of the particles at a particular time step, to ensure that all positions recorded obey the boundaries of the system.

However, on observing the Potential Energy Versus Time plot, we concluded that this is not likely to be the issue with our simulation.

6.1.2 Potential energy Blow Ups

On implementing the cut off radius, a new issue arose which was the discontinuity in the potential energy expression. If we imagine a particle crossing the cut off limit, it will initially, while being within the cut off distance, will contribute to the energy of the particle at the center, but on crossing the cut off distance, the interaction will immediately be ignored. Hence, there is a discontinuity in the potential energy as a particle crosses through the cut off distance.

Discontinuities might be a problem as blow ups in the potential energy would lead to blow ups in the acceleration, which in turn leads to the particles coming in close proximity to one another and being rapidly repelled. This motion is far too rapid to be captured in the time-step defined.

On plotting the Potential energy as a function of time, we observed that the potential energy did in fact, blow up at certain places, and hence, we understand this to be the issue causing the New Simulate function not to work.

6.2 Calculating Pressure from the RDF

The equation used to derive pressure from the Radial Distribution function, as discussed above is

$$\frac{PV}{NT} = 1 - \frac{\rho}{2Td} \int g(r) r \frac{du(r)}{dr} d\mathbf{r} \tag{8}$$

From the above equation, using the parameters we inputted into our simulation, we expected the slope of the $PvsT$ plot to be 0.5.

We observed a slope of 0.55 well in agreement with our expectations. However, we also obtained a negative intercept on the plot. The question then arises, is a negative intercept reasonable to expect from the above equation or any other physical consideration?

The implications of a negative intercept is that at absolute zero, the Pressure of the system is negative implying that there is a net imploding force or that the particles are pushed away from the walls of the container and towards the middle. It also seems to imply that the particles attain order and the particles do not collide with the walls anymore, implying immobility at the most, and at the very least, a tendency of the particles to stay away from the walls.

Mathematically speaking, it might be reasonable to expect a negative intercept as on closely observing the parameters that constitute the intercept, we realise the parameter that decides whether, for a particular r, the intercept is positive or negative is $du(r)dr$. We then make the observation that when this parameter is negative (Repulsive force), the intercept would be positive, and when this parameter is positive, the intercept would be negative. Now, we make the claim that when this parameter is negative, the intercept becomes zero, as the repulsive force functions upto a distance r where the Radial distribution function $g(r)$ returns zero particles. That is the product $g(r)du(r)dr$ becomes zero. And for larger distances, the intercept then becomes negative. Hence, we can reason that we expect the intercept to be negative.

7 References

1. Pathria, R. K. (2017, February 21). Statistical Mechanics. Elsevier.
2. Gould, H., Tobochnik, J., Christian, W. (1988, January 1). An Introduction to Computer Simulation Methods. Addison Wesley Publishing Company.

3. J.B. Adams, in Encyclopedia of Materials: Science and Technology, 2001

4. Verlet, L. (1967, July 5). Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. Physical Review, 159(1), 98–103. https://doi.org/10.1103/physrev.159.98

5. Andersen, H. C. (1980, February 15). Molecular dynamics simulations at constant pressure and/or temperature. The Journal of Chemical Physics, 72(4), 2384–2393. https://doi.org/10.1063/1.439486

6. Ghosh, K., Krishnamurthy, C. V. (2018, January 22). Structural behavior of supercritical fluids under confinement. Physical Review E, 97(1).