Computational Physics: Problem Set 11

Mohammad Ghoraishi
99100788

June 16, 2023

1 Main Simulation Functions

1.1 positionInit() Function

This function takes the initial conditions that we have specified for the system, then arranges the particles into a crystal form on the left side of the box.

1.2 velocityInit() Function

This function takes the maximum value of velocity as an input, then assigns a random speed to each of the particles within the desired range. It then returns two arrays containing the velocities.

1.3 force() Function

This function takes the two arrays of positions in X and Y, then it calculates the force on each particle and puts it all into Fx and Fy arrays for forces in the directions of X and Y respectively. We use periodic boundary conditions and shift the position of the particles exiting the box to re-enter the box from the other side. We also calculate the total potential energy of the system in this function to save time. We also use the cutoff values of force and potential too keep everything consistent. This function returns Fx, Fy and the potential energy as output.

1.4 positionUpdate() Function

This function takes the two arrays of position, the two arrays of velocity and the two arrays of force as input, then it updates the position of the particles by one step using the velocity-verlet method and returns the new position arrays as output.

1.5 velocityUpdate() Function

This function takes the two arrays of velocity, the two arrays of force and two arrays of force in the previous time-step as input, then it updates the velocities of all of the particles and also calculates the kinetic energy of the system to

save time. It then returns the updated velocity arrays and the kinetic energy as output.

2 Trajectory

To show the trajectory of the particles, we use the @animate macro in julia outside of a for loop and run the simulation inside the loop using the velocity-verlet method. For practical purposes, we generate one frame of the animation after 100 time steps which significantly reduces our run time. The resulting gif is named trajectory.gif in the files.

3 Particles on the Left Side

3.1 leftSide() Function

This is a simple function that calculates the number of particles on the left side of the box by testing if the X position is less than half the length of the box or not. This function returns the number of particles on the left side as a number. The resulting graph is as follows:

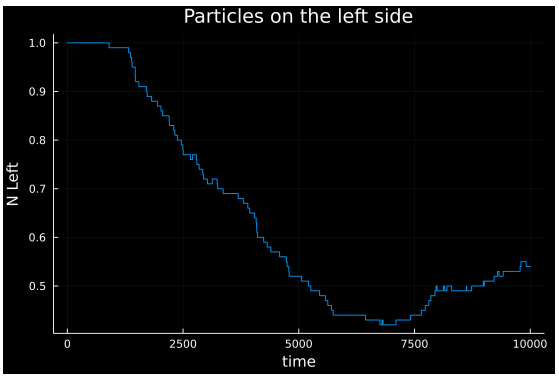

Figure 1: Particles on the left side of the box.

4 Energy Conservation

We already have functions that calculate the values of potential and kinetic energies of the system. So now we can just put these values into arrays and sum them to get the value of the total energy of the system. We have plotted the resulting graph below that shows energy is conserved:

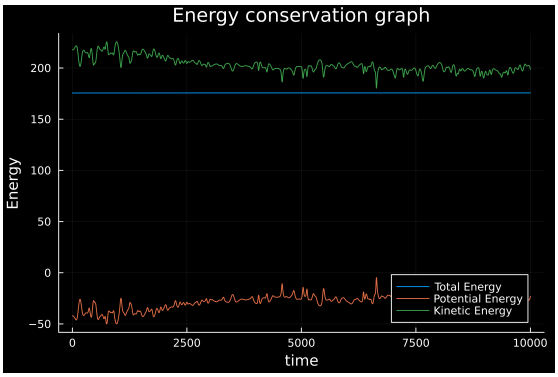

Figure 2: Energy Conservation over time.

5 Pressure and Temperature

We use the virial Formulas for pressure and temperature to calculate them. We already have the kinetic energy of the system so calculating the temperature is really straight forward but we use the function below to calculate the pressure.

5.1 pressure() Function

This function takes the two position arrays, The two Force arrays and the temperature as input, then it calculates the pressure using the virial expansion and returns the pressure. The resulting graphs are as follows:

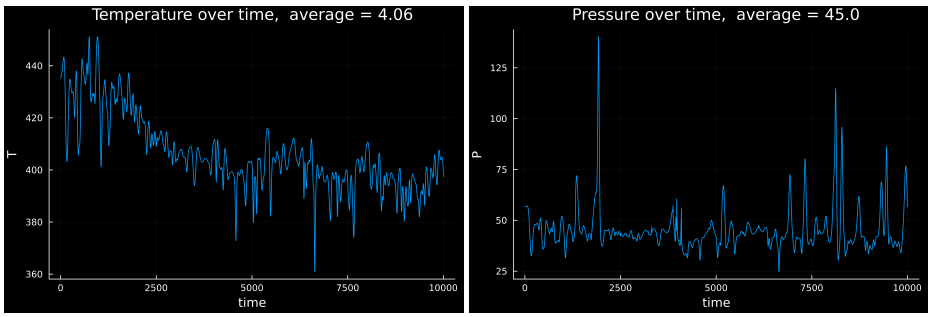

Figure 3: Temperature and Pressure values over time.(The averages are also on the top of the graphs)

5.2 Van der Waals

The Van der Waals equation can be written as follows:

$$P = \frac{1}{V/N - b}T - \frac{N^2}{V^2}a$$

So by plotting pressure against temperature and fitting a line, we can calculate the constants. By running the simulation 10 times for different values of T we get the following graph:

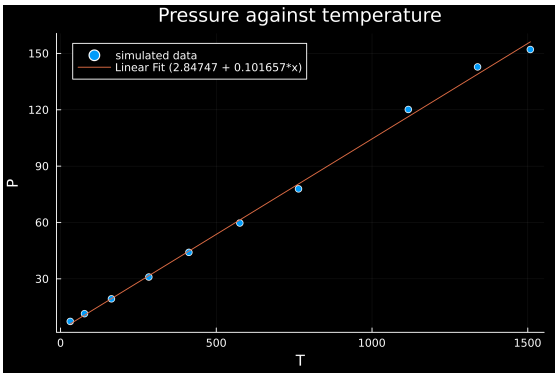

Figure 4: Pressure against temperature.

Using the Values of the fitted line, the constants are as follows:

6 Scaling Energy

In this section we run the same code as the trajectory loop but this time we multiply the velocities by 0.9999 each step which simulates energy loss. The trajectory of this system is named scaling speeds.gif and scaling speeds 2.gif which simulate the transition from gas to fluid and gas to solid respectively. The energy over temperature graph is also as follows and we can observe the phase change in the beginning of the plot with the change in slope.

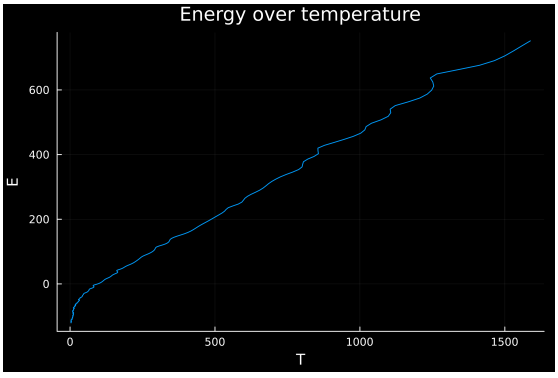

Figure 5: Energy against temperature and phase change.