Draft version October 26, 2022

PHY407 Lab-06 Report

Raina Irons[1]

[1] *Wrote and solved Q1,Q3*

QUESTION 1 - MOLECULAR TRAJECTORIES

PART (A)

We have the Lennard-Jones potential,

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^1 2 - \left(\frac{\sigma}{r}\right)^6\right], \tag{1}$$

where ϵ is the depth of the potential well. To find the force between the two particles with one centered at the origin, we take the gradient of the potential in spherical coordinates.

$$\begin{aligned} F &= -\nabla V \\ &= -\frac{\partial V}{\partial r}\hat{r} \\ &= -4\epsilon\frac{\partial}{\partial r}\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6\right]\hat{r} \\ &= -4\epsilon\left[-\left(\frac{12\sigma^{12}}{r^{13}}\right) + \left(\frac{6\sigma^6}{r^7}\right)\right]\hat{r} \end{aligned}$$

As the potential does not depend on ϕ or θ, we take the partial only with respect to r. We express $r = \sqrt{x^2+y^2}$ in 2D, and note that $x = r\cos\theta$, $y = r\sin\theta$.

$$\begin{aligned} F_x &= 4\epsilon\left[\left(\frac{12\sigma^{12}}{(x^2+y^2)^{13/2}}\right) - \left(\frac{6\sigma^6}{(x^2+y^2)^{7/2}}\right)\right]\cos\theta\hat{x} \\ &= 4\epsilon\left[\left(\frac{12\sigma^{12}}{(x^2+y^2)^{13/2}}\right) - \left(\frac{6\sigma^6}{(x^2+y^2)^{7/2}}\right)\right]x(x^2+y^2)^{-1/2}\hat{x} \\ &= 4\epsilon\left[\left(\frac{12\sigma^{12}}{(x^2+y^2)^{13}}\right) - \left(\frac{6\sigma^6}{(x^2+y^2)^{7}}\right)\right]x\hat{x} \\ &= \left[\left(\frac{48\epsilon\sigma^{12}}{(x^2+y^2)^{13}}\right) - \left(\frac{24\epsilon\sigma^6}{(x^2+y^2)^{7}}\right)\right]x\hat{x} \end{aligned}$$

Similarly, we can do the same for the y-component, which leaves us with:

$$F_y = \left[\left(\frac{48\epsilon\sigma^{12}}{(x^2+y^2)^{13}}\right) - \left(\frac{24\epsilon\sigma^6}{(x^2+y^2)^{7}}\right)\right]y\hat{y} \tag{2}$$

From here, we can use $F = ma$ for each respective component, which leaves us with,

$$a_x = \left[\left(\frac{48\epsilon\sigma^{12}}{(x^2+y^2)^{13}}\right) - \left(\frac{24\epsilon\sigma^6}{(x^2+y^2)^{7}}\right)\right]\frac{x}{m}\hat{x}, \tag{3}$$

$$a_y = \left[\left(\frac{48\epsilon\sigma^{12}}{(x^2+y^2)^{13}}\right) - \left(\frac{24\epsilon\sigma^6}{(x^2+y^2)^{7}}\right)\right]\frac{y}{m}\hat{y}, \tag{4}$$

and we can use this in our Verlet method. For more generalized co-ordinates where the position of one particle does not start at the origin, we can adapt the positions to be $(x-a)^2$ and $(y-b)^2$ to get the total position.

PART (B)

The pseudo-code for the Verlet method is as follows:

```
# Pseudocode
# Import  libraries
# Define given timestep and generate time domain
# Define the inital positions for each simulation
# Define the Verlet method in a function where we need
# inital position, and the time vectors to perform the simulation
# Plot the result
# Show x,y plot and the evolution of each component over time
# Show the energy plots for the first simulation, and demonstrate how energy is conserved

# The Verlet function
#   Initialize constants
#   Set up the inital conditions on the position arrays of both particles
#   Compute the first iteration of the velocity components at dt/2
#   Start for loop over len(t)-1
#   Compute the the next position instance from previous mid-point velocity calculation
#   Compute k vector from new position
#   Compute mid-point velocity from new k vector
#   Return the positions vectors
```

We can implement this code with the following given initial conditions.

1. $\vec{r_1} = [4, 4]$, $\vec{r_2} = [5.2, 4]$
2. $\vec{r_1} = [4.5, 4]$, $\vec{r_2} = [5.2, 4]$
3. $\vec{r_1} = [2, 3]$, $\vec{r_2} = [3.5, 4.4]$

Below are the plots from each simulation given the above initial conditions.

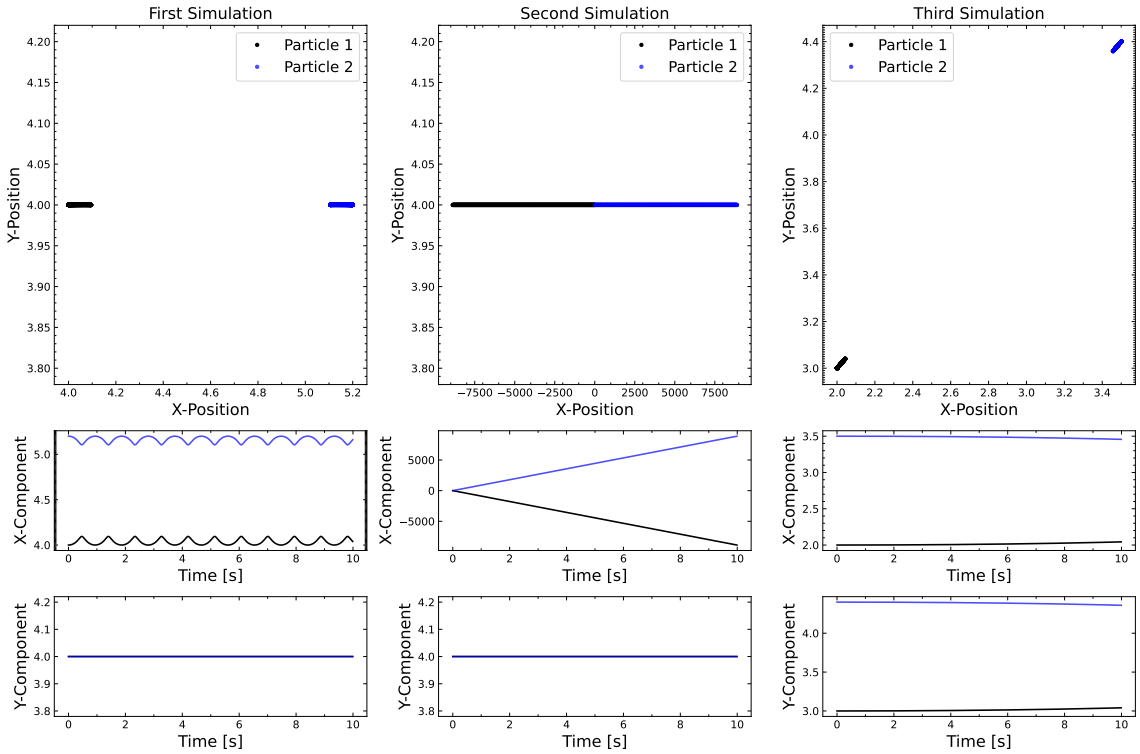

Figure 1: *Left column*: The first simulation results in what appears to be oscillatory behaviour in the x-component. There is no change in vertical position due to the initial positions of both particles being parallel with respect to the x-axis. *Center column*: The second simulation starts with both particles very close together, which triggers a divergence in the positions and no oscillations in the time frame given. *Right column*: Here we do not appear to have any oscillations in the trajectory of both particles, only slowly attracting one another over time and coming close.

PART (C)

We see from the three simulations oscillatory is evident in Simulation 1. We can demonstrate the energy conservation of this behaviour by plotting the energy over time of the two particles. Figure 2 exhibits conservation of energy of both particles in simulation one. The energy of the system is not evidently dissipating over time, resulting in a constant total energy for the system.

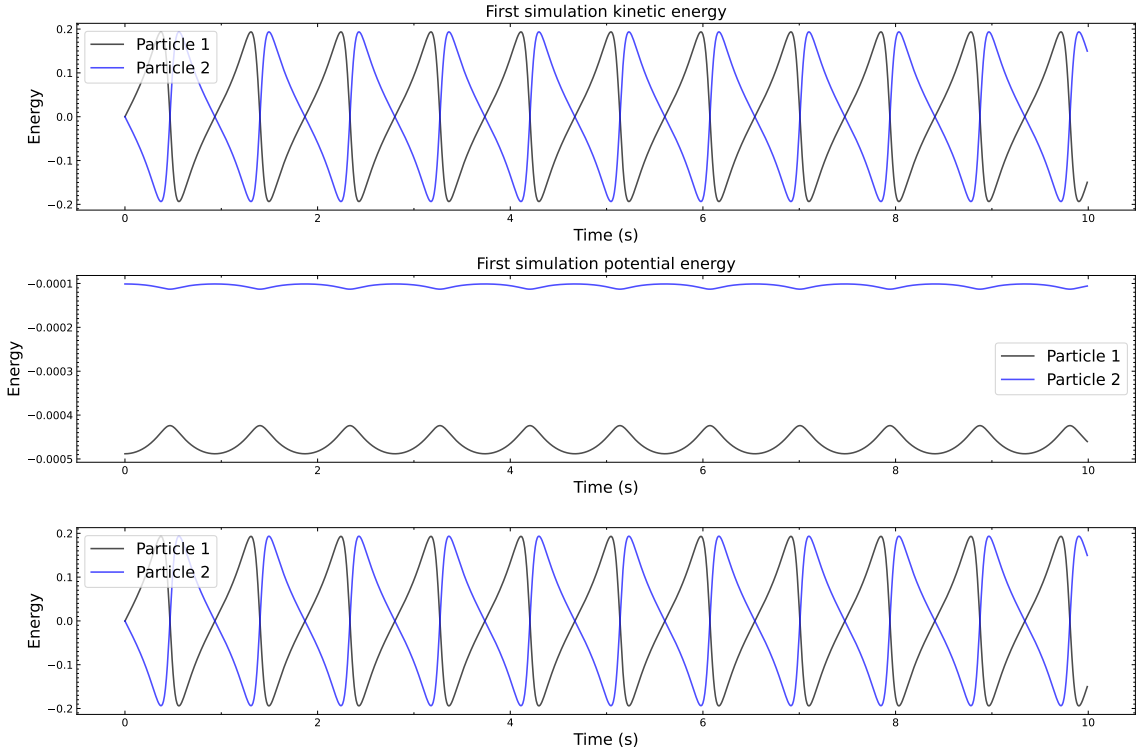

Figure 2: *Top*: The kinetic energies of both particles in simulation one. Here we see the energy oscillates back and forth. *Middle*: The potential energy of both particles in simulation one. *Bottom*: Simulation one exhibits oscillatory motion in the x component. Here we see the total of each particle energy oscillates.

QUESTION 2 - MOLECULAR DYNAMICS SIMULATION

PART (A)

We now modify our Verlet function in order to account for 16 particles evenly distributed along a 4X4 grid in the xy-plane. The basic formulae for this is outlined in the pseaudo-code below.

```
# Import libraries
# Set N number of particles to be simulated, the distances between each particle
# Set the initial positions of all particles at vertices of 4X4 grid, COM located at (0,0)
# Set the initial velocity as zero for all particles to 0
# Define timestep dt = 0.01, and create time array
# Modify the Verlet method for multibody in a function
# Plot the simulation on xy-plane of the grid
# Plot the energy as a function of time from each particle

# The modified Verlet function:
#   Initialize the physical constants
#   Initialize x,y positions, full-step vx,vy, half-step vx,vy as 2D numpy array, with N particles on row a
#   and time t on column axis
#   Initialize the kinetic and potential energy as an array
#   Compute the first iteration of the half-step velocity components at dt/2
#   for each particle
#   Compute the energy at t = 0
#   Start a time for-loop from 1 to len(t):
#       Compute the the positions from prev. half-step velocity for all particles
#       Calculate acceleration and potential energy ith particle from j particles
#       Compute k vector from this acceleration
#       Compute full-step velocity from this k vector
#       Compute half-step velocity from this k vector
#       Calculate kinetic energy using the current full-step velocity
#   Sum kinetic and potential energies
#   Return positions and velocities in x,y components, time, and energies
```

The basic setup of our system is given as follows.

```
N = 16
Lx = 4.0
Ly = 4.0
dx = Lx/sqrt(N)
dy = Ly/sqrt(N)
x_grid = arange(dx/2, Lx, dx)
y_grid = arange(dy/2, Ly, dy)
xx_grid, yy_grid = meshgrid(x_grid, y_grid)
x_initial = xx_grid.flatten()
y_initial = yy_grid.flatten()
```

From which we can implement the above pseudo-code and get the following result.

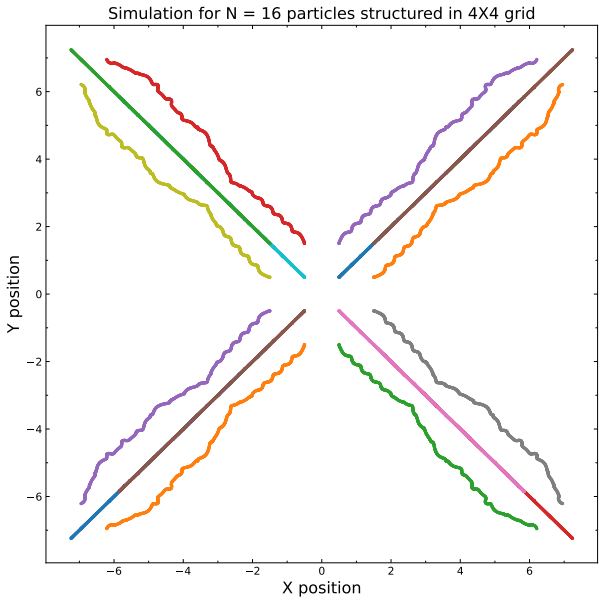

Figure 3: Here we see the evolution of our 16 particle system over time. The center portion of the plot is the starting 4X4 grid for each of our particles, progressing outward as time evolves. We see the outermost particles gravitate and repel each other repeatedly.

PART (B)

Unfortunately, we were not able to achieve a plot demonstrating conservation of energy for the 16 particle system. The energy deviated frequently from the median of the total system energy, and did not exhibit any obvious oscillatory patterns. Something is wrong with our 16 particle simulation, but the culprit is unknown.

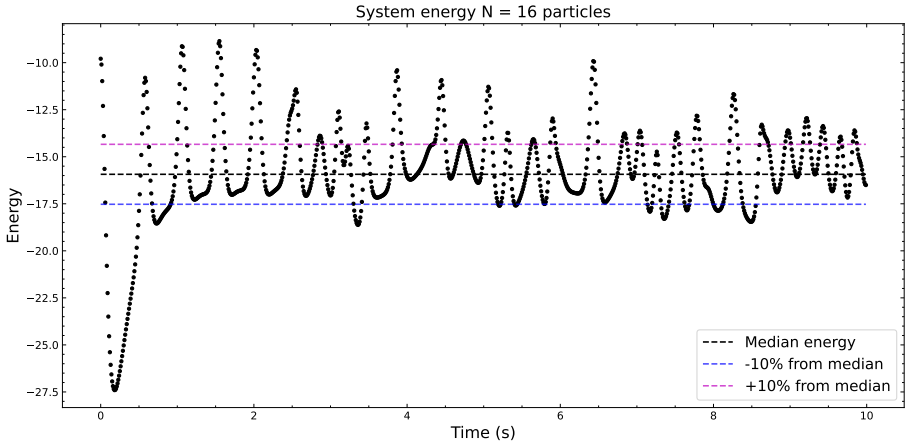

Figure 4: Here we show the total system energy for the 16 particle system. Unfortunately there is no obvious behaviour demonstrating conservation of energy with this system, as there are no repeated oscillations in the energy.