PHY407 (Lab 6: Ordinary Differential Equations, Pt. I)

Utkarsh Mali[1] and Aslesha Pokhrel[1,2]

[1]Department of Physics, University of Toronto
[2]Department of Computer Science, University of Toronto

October 23, 2020

Contents

Work Allocation:
Aslesha: Question 2, Question 3
Utkarsh: Question 1, Question 2

Question 1: Modeling Space Garbage

(1b) Here we are asked to separate two systems of second order differential equations into simpler first order linear differential equations. When separating these equations into first order linear differential equations we use the terms `xdot` and `ydot` to represent the other pair of linear differential equations obtained from the reduction of order. Notice that since our equations of motion are not dependent on any specific value of t, it can be taken to be a dummy variable when computing the ODE computation.

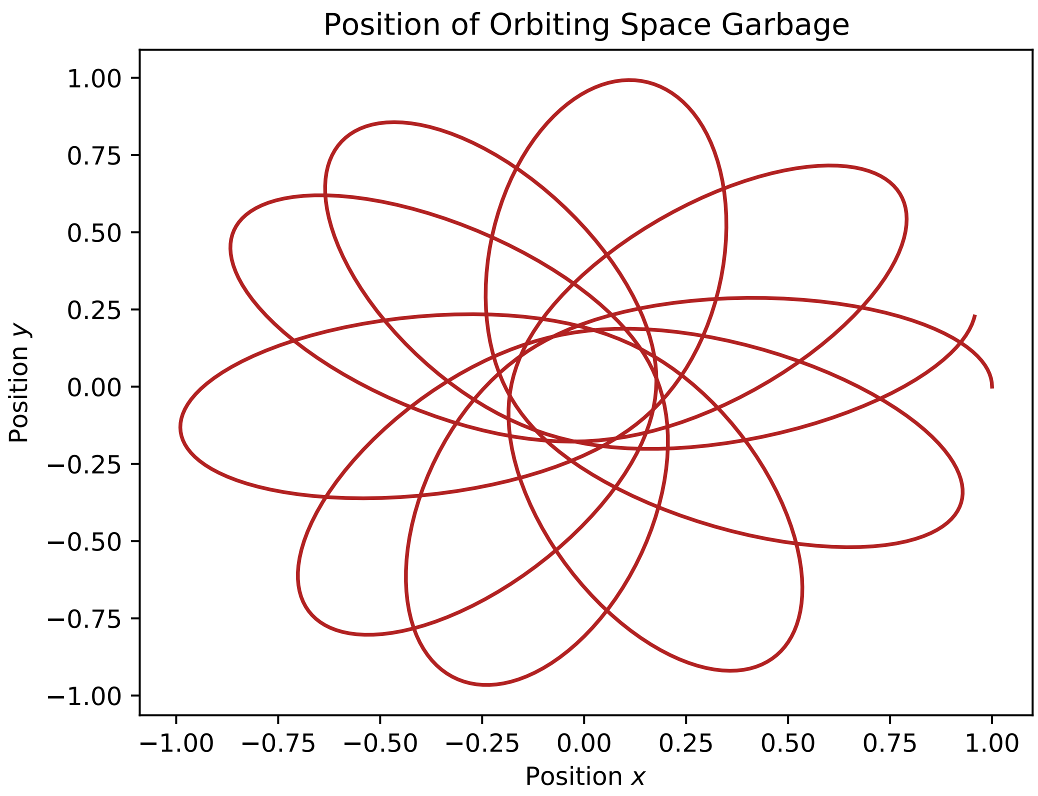

Figure 1: Here we observe the plot showing the orbit of our space garbage in a given time frame. We observe behaviour analogous to an ellipsoid but with some additional behaviour offsets. As also note that the orbit is not exactly $\frac{1}{r^2}$ since that is not an exact term in our differential equation.

```
PSEUDOCODE:
# Question 1b
# Define units
# Define functions, after reducing them to first order
    # We first set our r vector equal to our position and velocities as required
    # We then calculate our distance r as defined in the question
    # Then set parameters of constants to be multiplied to the numerator
    # Depending on which equation is being solved, set the value of the
    # Set the appropriate value of denominator as defined in the question
    # Compute fraction for x ODE
    # compute fraction for y ODE
    # return array of updated positions and velocities.

# Set initial conditions of time
# Set initial conditions of position and velocity
# Set N and h
# Setting arrays for iteration
# Set r vector


# Apply RK4 method
    # append points first
    # Compute RK4 method loop
    # Notice all the t’s being used are dummy variables which do not affect the computation
    # update r for next iteration

# Plotting the figure requested in the question
# Plot an additional speed figure for aesthetic
```

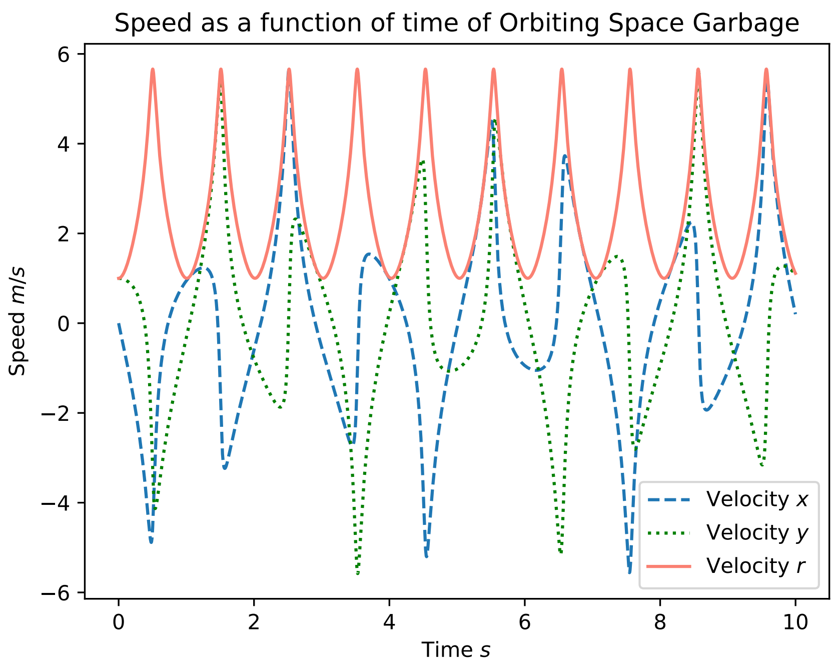

Figure 2: Additional plot showing the speed of the object which provided interesting insights upon brief review.We observe periodic change in the r velocity representing typical orbital motion.

Question 2: Molecular Dynamics Pt1

(a) **Here we are asked to consider a molecular dynamics simulation in which we have 2 particles.** We found expressions for x and y using the derivative method and then applied the `Verlet` method to obtain the value of our plots.

(b) **Now we are asked to show pseudo code and the plots obtained from our 3 initial conditions.** The three plots are shown below:

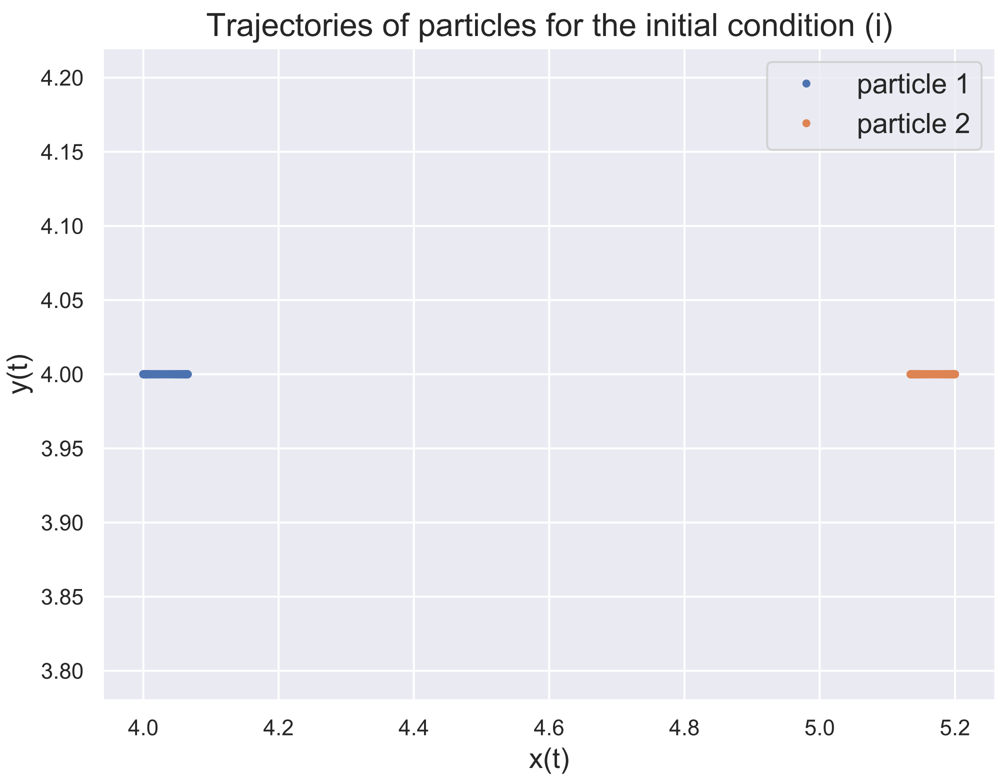

Figure 3: Plot showing trajectory of the 2 particles with initial condition $\vec{r_1} = [4, 4], \vec{r_2} = [5.2, 4]$

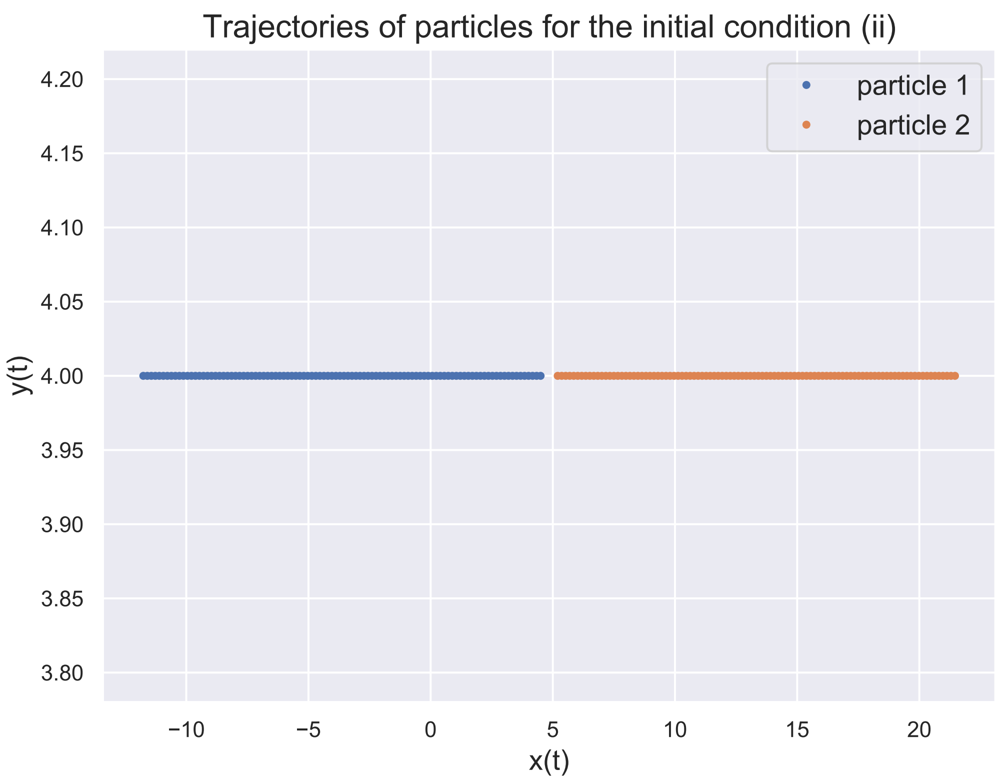

Figure 4: Plot showing trajectory of the 2 particles with initial condition $\vec{r_1} = [4.5, 4], \vec{r_2} = [5.2, 4]$

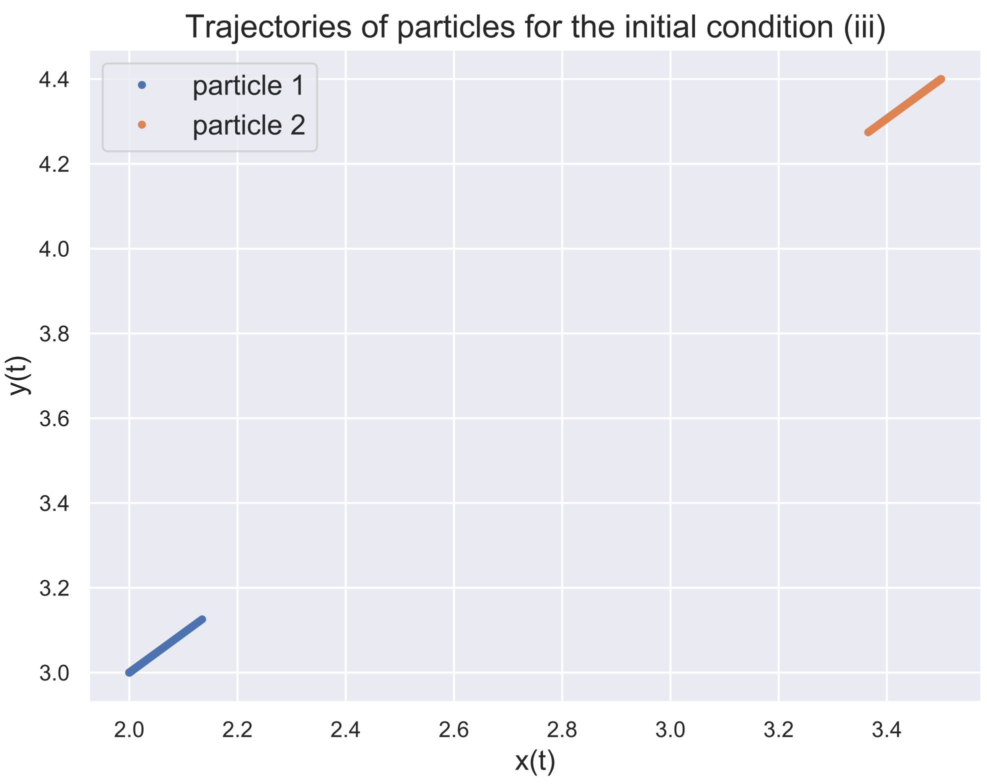

Figure 5: Plot showing trajectory of the 2 particles with initial condition $\vec{r_1} = [2, 3], \vec{r_2} = [3.5, 4.4]$

Similarly, the pseudocode used to calculate the trajectory of the two particles given their initial conditions is given below:

```
PSEUDO-CODE:

# import packages
# set constants
# mass of the molecule
# x and y components of accelerations found in part (a)
    # the distance between the two particles
    # use F = V(r) = ma where m = 1 to find a.
    # Multiply by x2-x1 / r for x component
    # Multiply by y2-y1 / r for y component
    # return acceleration


# Define run verlet funciton
    # set given values
    # set step size dt
    # set the number of steps

    # initialize arrays to hold results
    # initialize initial values of r1, and r2
    # for the first step only. Represent v(t + h/2) in eq7 by v_temp

    # apply eqn 8 - 11 repeatedly in a for loop
        # apply equation 8
        # apply equation 9. K is a list [kx, ky]
        # apply equation 10
        # apply equation 11
        # return updated values

# First initial condition q (i)
# Create plot
# Plot both particles
# Make pretty and display

# Second initial condition q (ii)
# Create plot
# Plot both particles
# Make pretty and display

# Third initial condition q (iii)
# Create plot
# Plot both particles
# Make pretty and display
```

(c) **Out of the three initial conditions shown above, the first initial condition (i) $\vec{r_1} = [4, 4], \vec{r_2} = [5.2, 4]$ leads to the oscillatory motion for both the particles.** This is confirmed by the periodic motion seen for both the particles in the plot below:

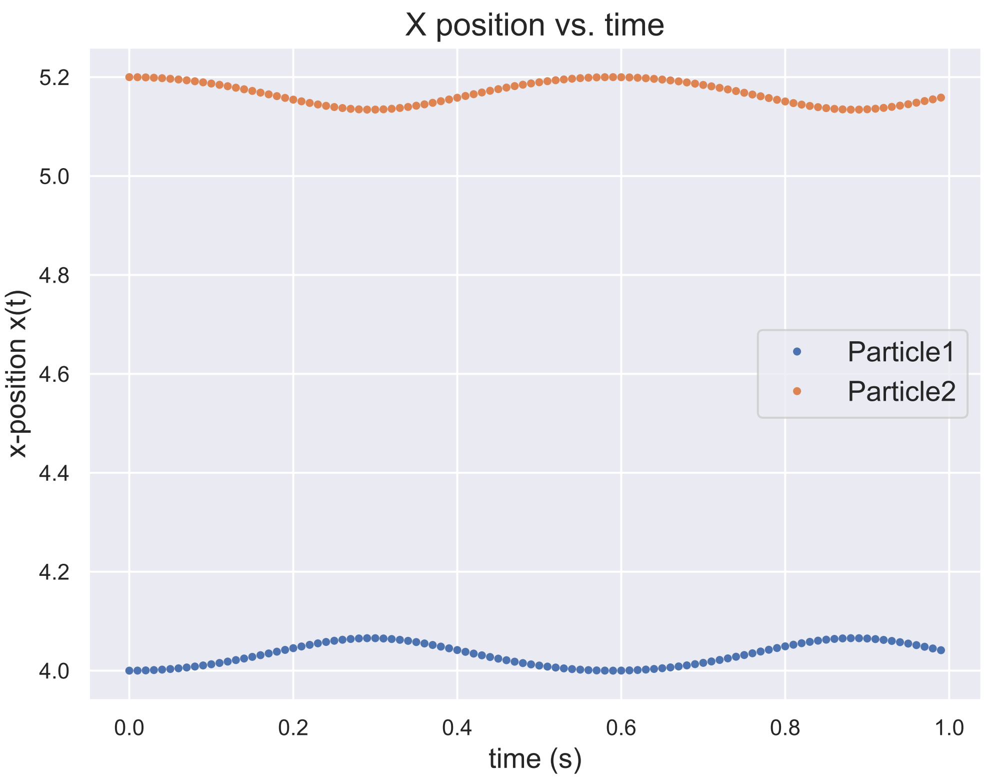

Figure 6: Plot showing the x-motion of the two particles over time. As shown in the plot above, the motion of both the particles is oscillatory.

We see this oscillatory motion for the case (i) because the total energy is conserved for the entire system and since the particles initially start in a potential bowl, it does not have enough energy to overcome this potential which leads to a periodic motion of the particles.

Question 3: Molecular Dynamics Pt2

(a) **Here we are asked to update our pseudocode in Q2 to so that we can simulate motion of an arbitrary number of particles.In this case we are asked to simulate $N = 16$ particles.** We updated our pseudocode in Q2 expanding our system to $N = 16$ particles. We first set the initial position of particles to be evenly spaces in a square domain of size L. We then assigned the initial velocities of all particles to 0. We then calculated the distance and acceleration. The Verlet method was then used to update the position and velocity. The trajectories of all the sixteen particles are shown in the plot below:

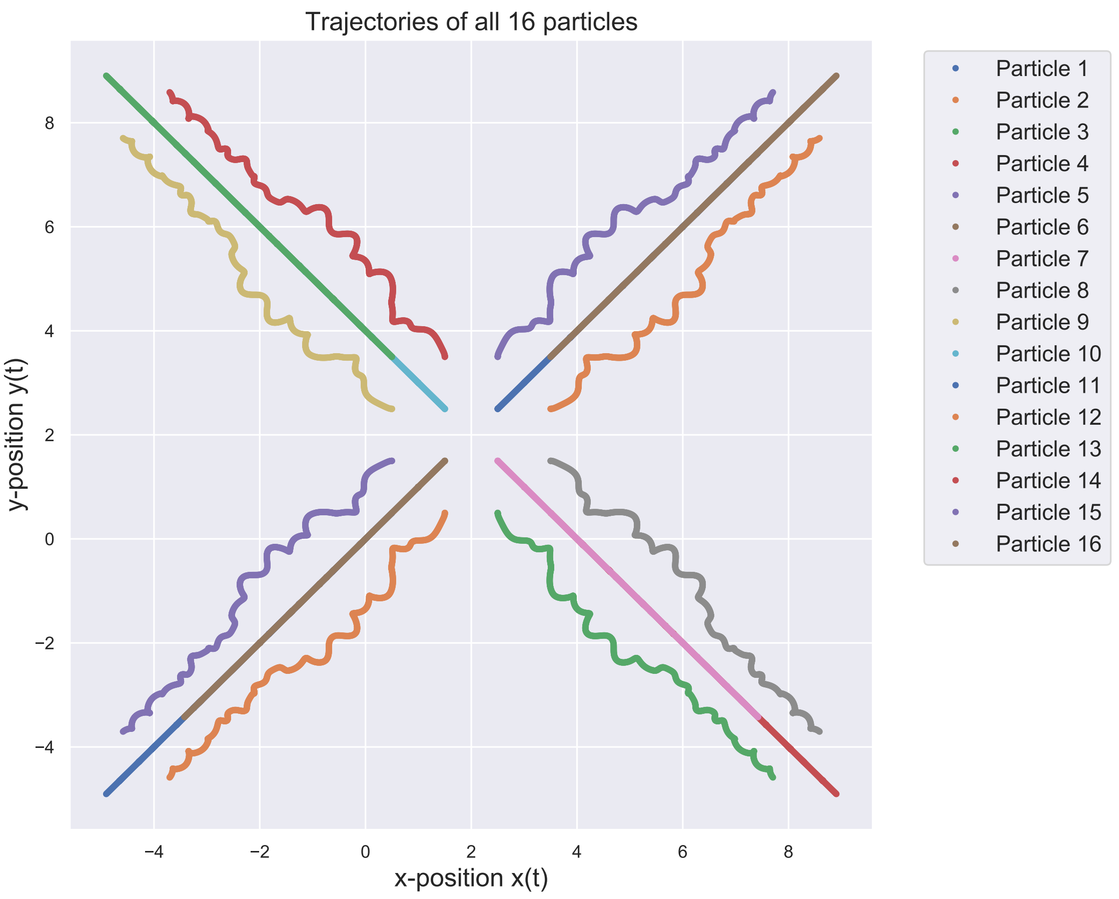

Figure 7: Plot showing trajectories for all 16 particles.

The pseudo-code for simulating the trajectories of N particles is shown belo:

```
PSEUDOCODE:
# import packages

#define function for total acceleration
   # Given length, Lx, Ly
```

```
    # initialize x and y comp of the acceleration on each particle i

    # accumulate the acceleration due to all the interaction for particle i through
        for loop
        # only accumulate the acceleration if the particle is not i itself
            # set xi, yi, xj and yj
            # calculate the acceleration given by the formula 24 * ( r^(-7) - 2 *
                r^(-13)) where r = sqrt((xj - xi)^2 + (yj - yi)^2)
            # multiply by x-dispacement/distance(xi, yi, xj, yj) for x component of
                acceleration
            # multiply by y-dispacement/distance(xi, yi, xj, yj) for y component of
                acceleration
    # return x-y components of the new acceleration

# run verlet algorithm
    # num_particles, num_steps and h (step size) is given in the argument
    # setting the initial positions of the particles to be evenly spaced in a
    # square domain using the code snippet given in the handout
    # copy code from lab guide
    # initialize arrays to hold the results

    # empty 3D array of size (num_particles, num_steps, 2) for velocity. Initialize
        to 0
    # empty 3D array of size (num_particles, num_steps, 2) for distance
    # empty 2D array of size (num_particles, 2) to store v(t + h/2) in eq7 for each
        particle

    # set the initial position in r array
    # update initial x pos
    # update initial y pos

    # find v_temp in eq 7
        # use the helper function of "acceleration" to get acceleration x-y components
        # update x-component of v_temp for each particle
        # update y-component of v_temp for each particle

    # run Verlet method for num_steps time steps with for loop
        # calculate equation 8 of Verlet method
        # update x comp of distance r
        # update y comp of distance r

        # calculate equation 9, 10 and 11 in a loop for all the particles
            # use the helper "acceleration" function to get acceleration x-y components
                # calculate kx and ky of equation 9

            # calculate equation 10
                # update x-comp of v
                # update y-comp of v

            # calculate equation 11
                # update x-comp of v(t + h/2)
                # update y-comp of v(t + h/2)
    # return v, r

# plot result function
    # set num particles
    # set num steps
    # set h
    # get velocity and postion arrays using run_verlet method

    # Plot the trajectory for all particles
    # make graph pretty


# run the program
```

(b) **Now we are asked to compute the energy of the system by adding both the potential and kinetic energy at each time step. We should find that energy is being conserved.** After computing the energy of the system, we found that the total energy was conserved to within about 1%, as expected.

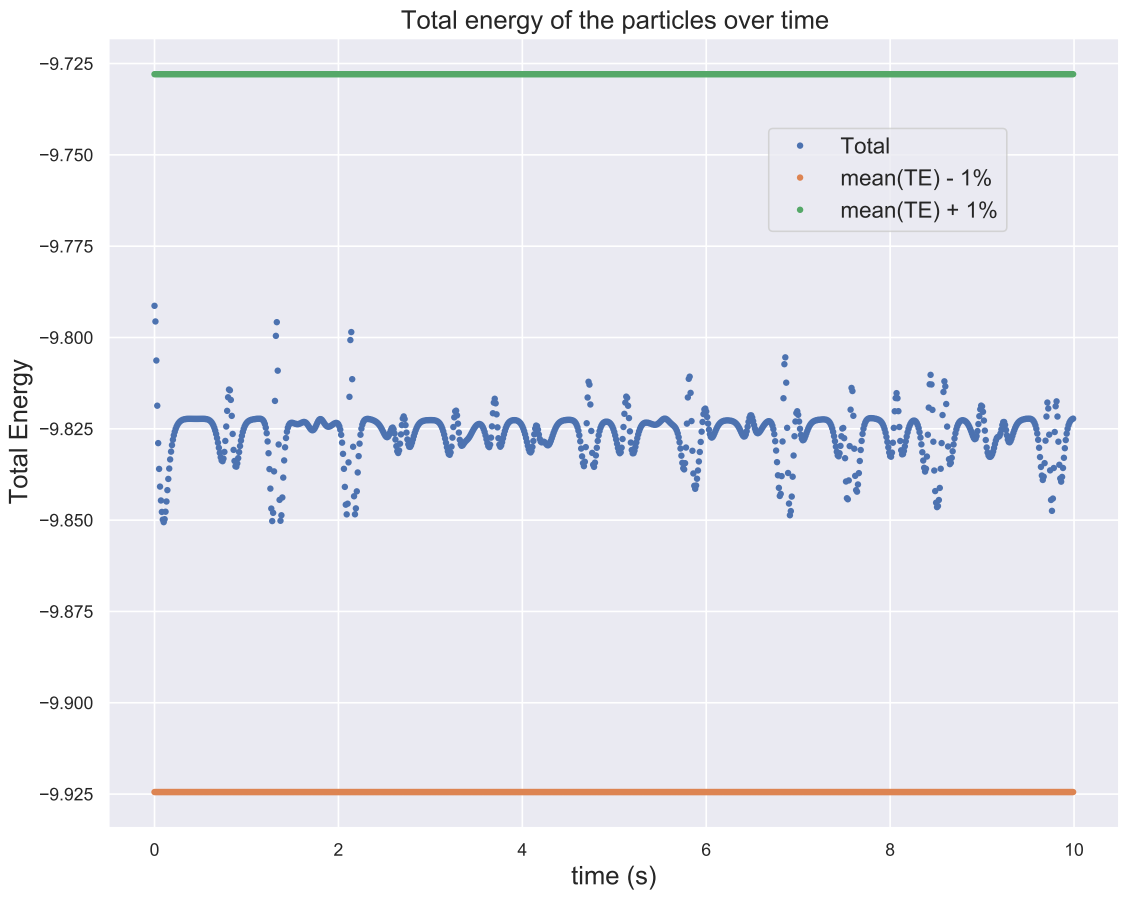

Figure 8: Plot showing energy of the entire system.As observed we find that **energy is indeed conserved** to within ±1%.

Similarly, the plot showing the potential, kinetic and total energy over time is also given below:

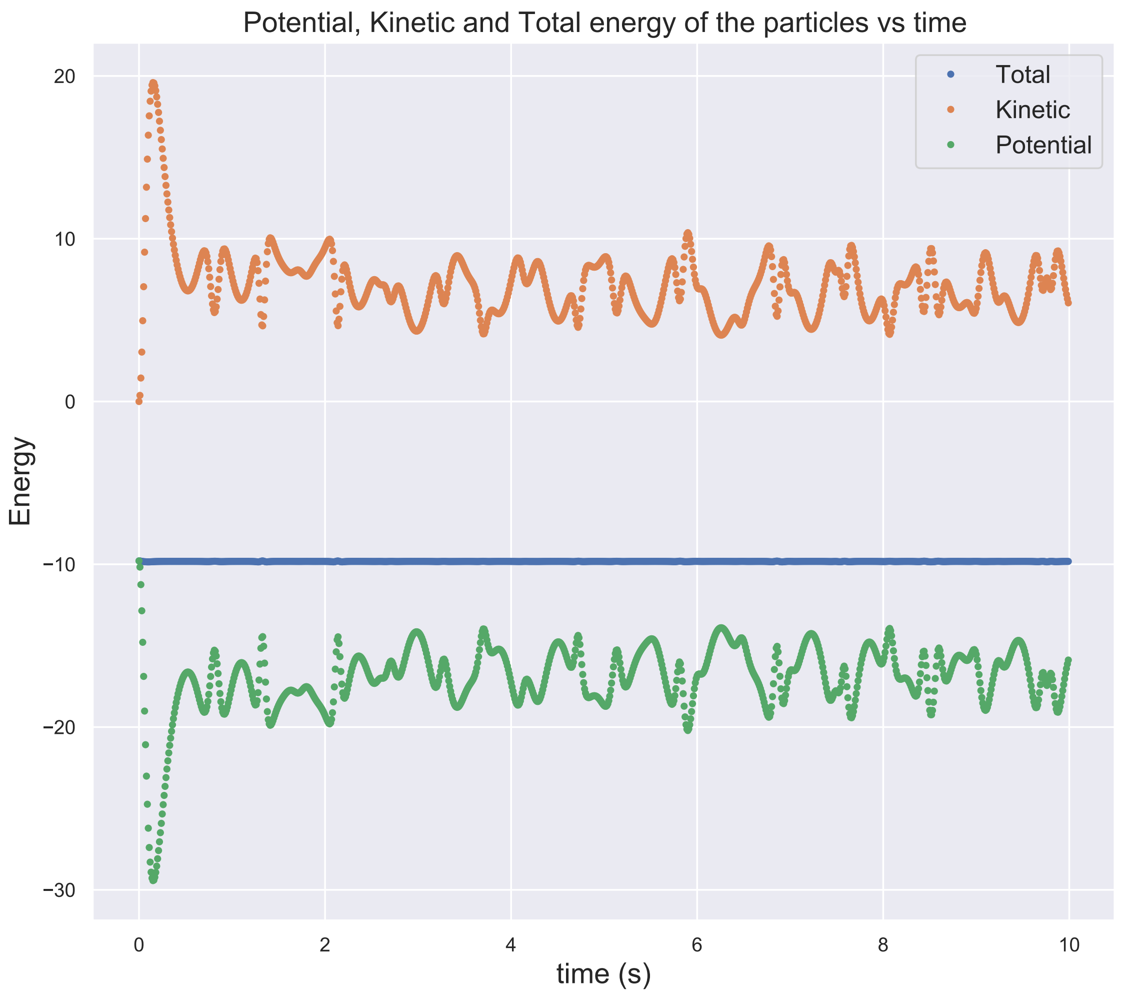

Figure 9: Plot showing the kinetic, potential and total energy of the entire system over time.

(c) **Finally, our task is to implement the periodic boundary conditions. We need to make the particles that exit the domain, re-enter at a specific spot, and accordingly make changes to the forces of the particles.** Finally, we plot the trajectory of all the particles in the same graph.

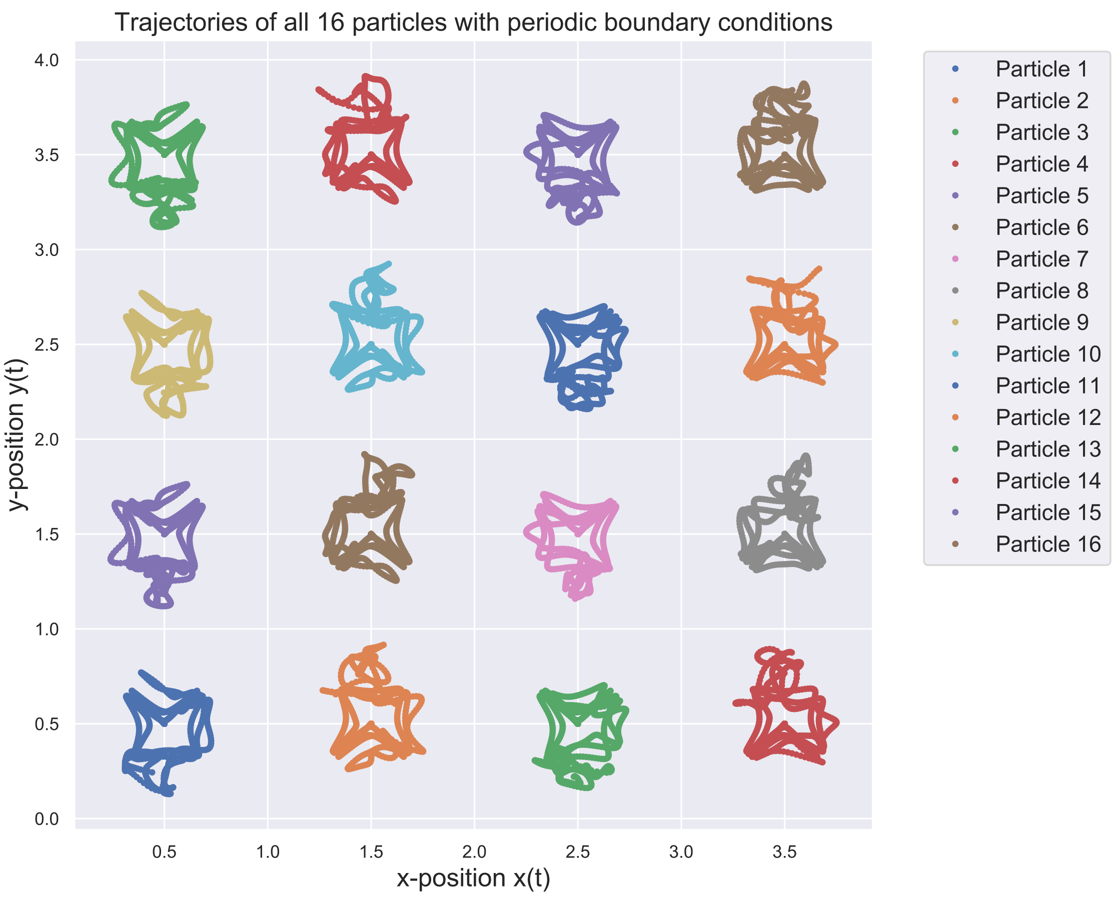

Figure 10: Plot showing trajectories of the particles with boundary conditions applied.

We can see from fig. 10 that over the course of the simulation, the motion of each of the particle is contained within a specific region i.e. the position of the particle does not vary widely. The noise we see in the movement of the particle is primarily due to the error and limitation in the numerical methods. This shows that by implementing the periodic boundary conditions, we can simulate the molecular motion where the simulation process is not affected by the interactions of the molecules with the box and the molecules also do not move far apart from each other. This is different from the result we obtained in fig. 7 where the molecules move really far apart from each other as the time passes due to a lack of the boundary conditions.