Name: **Lab# 10** **Your ID #:**

Molecular dynamics (MD)

Please answer the below questions:

Q1 (5pts): <u>Problem 1. (Shooting Method).</u>

Use shooting method to solve the below question.

$$\ddot{y} - 4y + 4x = 0$$

$$y(0) = 0, \;\; y(1) = 2$$

(a) Verify the solution using the same values as the ones used in our lecture.
(b) Plot the solution with the two values on a graph.
(c) Try to use different guess values and resolve the problem.

Q1 (10pts): <u>Problem 2. (Molecular dynamics (MD))</u>

The below Python code is intended to simulate a molecular dynamics (MD) system using the Verlet algorithm. This is a simulation of particles in a 3D box using the Lennard-Jones potential. Answer the following question:

1. What is the purpose of this code?
2. What are the initial conditions for the simulation?
3. What is the Verlet algorithm and how is it used in this simulation?
4. How are the positions and velocities of the particles updated at each time step?
5. What is the purpose of the force() function and how is it used in the simulation?
6. How are the particle positions plotted at each time step?
7. What is the purpose of the plt.pause() function in the code?
8. How does the simulation end?
9. What if the N=100 ?
10. Can you create a movie of the positions updated at each time.

==

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Constants
N = 100  # number of particles
L = 10.0  # length of the simulation box
dt = 0.01  # time step
t = 0.0  # initial time
nsteps = 100  # number of time steps
```

```
# Initial conditions
r = np.random.rand(N, 3) * L
v = np.zeros((N, 3))
a = np.zeros((N, 3))
m = 1.0

# Force calculation
def force(r):
    f = np.zeros((N, 3))
    for i in range(N):
        for j in range(i+1, N):
            rij = r[i] - r[j]
            fij = 24 * (2 / np.power(np.linalg.norm(rij), 14) - 1 / np.power(np.linalg.norm(rij), 8)) *
rij
            f[i] += fij
            f[j] -= fij
    return f

# Plot initial particle positions
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(r[:,0], r[:,1], r[:,2])
ax.set_xlim([0, L])
ax.set_ylim([0, L])
ax.set_zlim([0, L])

# Update particle positions at each time step and plot
for step in range(nsteps):
    r_half = r + v * dt/2
    f = force(r_half)
    a = f / m
    r = r + v * dt + 0.5 * a * dt**2
    f_new = force(r)
    v = v + 0.5 * (f + f_new) * dt / m
    t += dt

    ax.clear()
    ax.scatter(r[:,0], r[:,1], r[:,2])
    ax.set_xlim([0, L])
    ax.set_ylim([0, L])
    ax.set_zlim([0, L])
    plt.pause(0.01)

plt.show()
```

===