ME 3030

Modelling and Simulation

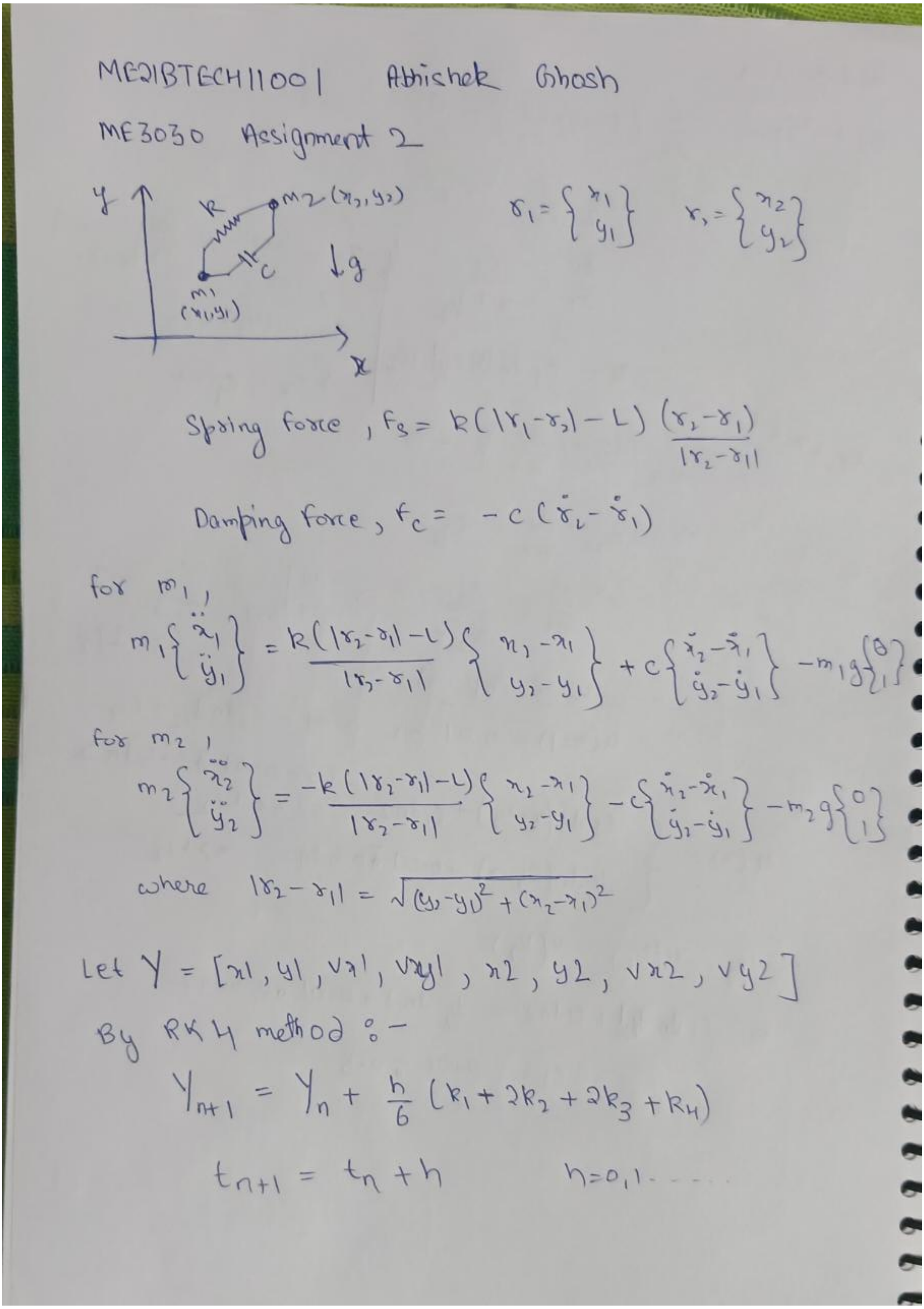
Assignment 2

ME21BTECH11001

Abhishek Ghosh

Question 1

a)



ME21BTECH11001 Abhishek Ghosh

ME3030 Assignment 2

$$r_1 = \begin{Bmatrix} x_1 \\ y_1 \end{Bmatrix} \qquad r_2 = \begin{Bmatrix} x_2 \\ y_2 \end{Bmatrix}$$

Spring force, $F_s = k(|r_1 - r_2| - L)\dfrac{(r_2 - r_1)}{|r_2 - r_1|}$

Damping force, $F_c = -c(\dot{r}_2 - \dot{r}_1)$

for m_1,

$$m_1 \begin{Bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \end{Bmatrix} = \frac{k(|r_2 - r_1| - L)}{|r_2 - r_1|} \begin{Bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{Bmatrix} + c \begin{Bmatrix} \dot{x}_2 - \dot{x}_1 \\ \dot{y}_2 - \dot{y}_1 \end{Bmatrix} - m_1 g \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

for m_2,

$$m_2 \begin{Bmatrix} \ddot{x}_2 \\ \ddot{y}_2 \end{Bmatrix} = -\frac{k(|r_2 - r_1| - L)}{|r_2 - r_1|} \begin{Bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{Bmatrix} - c \begin{Bmatrix} \dot{x}_2 - \dot{x}_1 \\ \dot{y}_2 - \dot{y}_1 \end{Bmatrix} - m_2 g \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

where $|r_2 - r_1| = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$

Let $Y = [x1, y1, vx1, vy1, x2, y2, vx2, vy2]$

By RK4 method :-

$$Y_{n+1} = Y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$t_{n+1} = t_n + h \qquad n = 0, 1 \ldots$$

where $k_1 = f(t_n, y_n)$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right)$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right)$$

$$k_4 = f(t_n + h, y_n + h k_3)$$

$$K.E = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m v_2^2 \qquad v_1 = \begin{Bmatrix} \dot{x}_1 \\ \dot{y}_1 \end{Bmatrix}$$

$$\text{Spring PE} = \frac{1}{2} k \left(|r_2 - r_1| - L\right)^2 \qquad 2\, v_2 = \begin{Bmatrix} \dot{x}_2 \\ \dot{y}_2 \end{Bmatrix}$$

$$PE = m_1 g y_1 + m_2 g y_2$$

The MATLAB code for the following RK4 is:-

b)

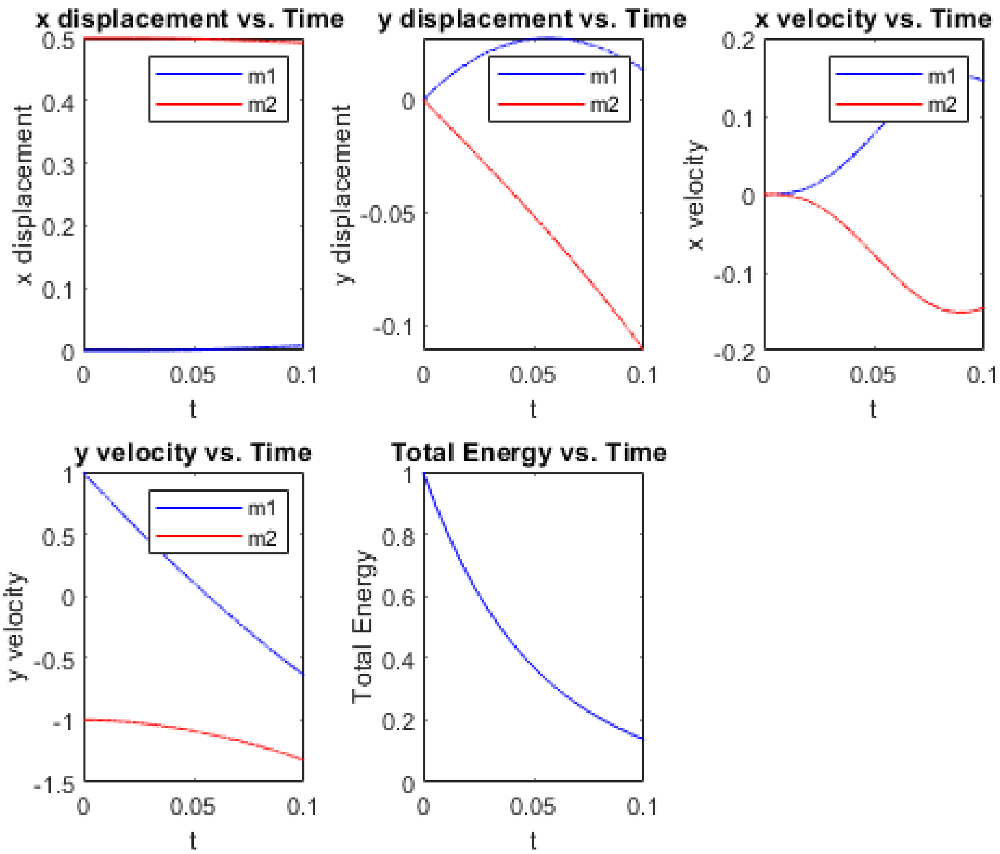
In presence of gravity & damping : -

```
% ME3030 Assignment 2
% Abhishek Ghosh ME21BTECH11001
% Define system parameters
m1 = 1.0;       % Mass of m1 in kg
m2 = 1.0;       % Mass of m2 in kg
k = 1000.0;     % Spring stiffness in N/m
c = 5.0;        % Damping coefficient in Ns/m
l = 0.5;      % Free length of the spring in m
g = 9.81;     % Acceleration due to gravity in m/s^2

% Initial conditions
x1_i = 0.0;
y1_i = 0.0;
vx1_i = 0.0;
vy1_i = 1.0;

x2_i = 0.5;
y2_i = 0.0;
vx2_i = 0.0;
vy2_i = -1.0;

% Time step
dt = 1.0e-5;
```

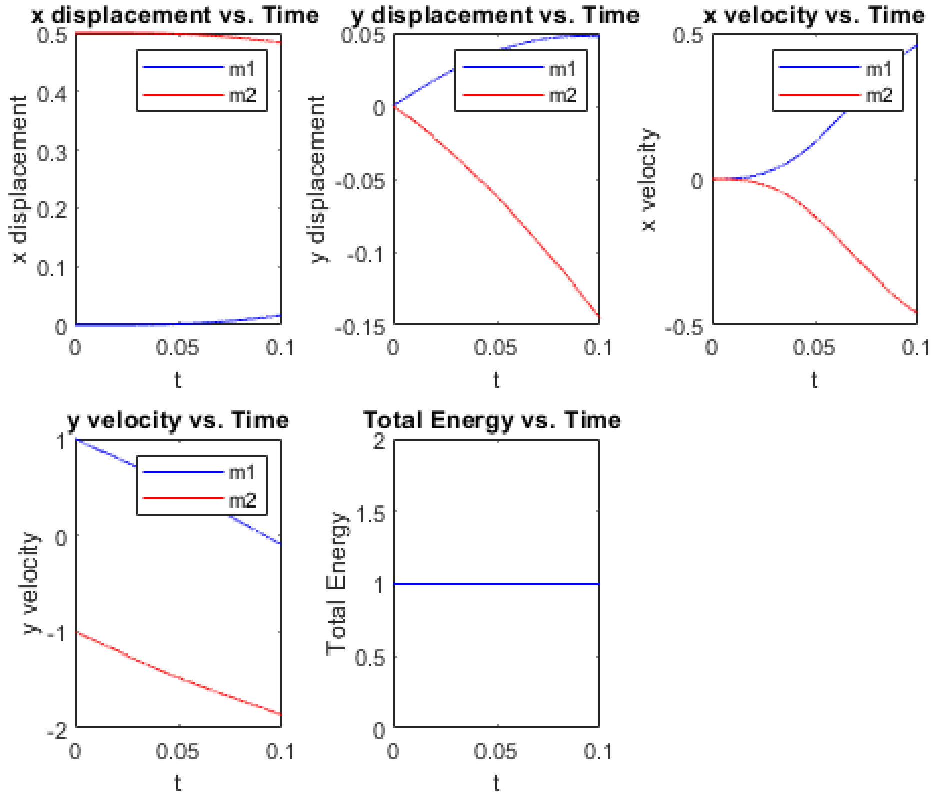
```
% Number of time steps
num_steps = 10000;

% Time array
time = (0:num_steps-1) * dt;

% Initialize arrays to store positions and velocities
x1 = zeros(1, num_steps);
y1 = zeros(1, num_steps);
vx1 = zeros(1, num_steps);
vy1 = zeros(1, num_steps);

x2 = zeros(1, num_steps);
y2 = zeros(1, num_steps);
vx2 = zeros(1, num_steps);
vy2 = zeros(1, num_steps);

% Initialize distance between masses array
distance_between_masses = zeros(1, num_steps);

% Initialize kinetic energy, spring potential energy, gravitational potential
energy and total energy array
kinetic_energy = zeros(1, num_steps);
spring_pot_energy = zeros(1, num_steps);
grav_pot_energy = zeros(1, num_steps);
total_energy = zeros(1, num_steps);

% Set initial conditions
x1(1) = x1_i;
y1(1) = y1_i;
vx1(1) = vx1_i;
vy1(1) = vy1_i;

x2(1) = x2_i;
y2(1) = y2_i;
vx2(1) = vx2_i;
vy2(1) = vy2_i;

distance_between_masses(1) = sqrt((x2(1) - x1(1))^2 + (y2(1) - y1(1))^2);
kinetic_energy(1) = 0.5*m1*((vx1(1))^2 + (vy1(1))^2) + 0.5*m2*((vx2(1))^2 +
(vy2(1))^2);
spring_pot_energy(1) = 0.5*k*((distance_between_masses(1) - l)^2);
grav_pot_energy(1) = m1*g*y1(1) + m2*g*y2(1);
total_energy(1) =  kinetic_energy(1) + spring_pot_energy(1) + grav_pot_energy(1);

% disp(total_energy(1));

states = [x1_i; y1_i; vx1_i; vy1_i; x2_i; y2_i; vx2_i; vy2_i];


% Using Runge Kutta method (RK4) :-

for i = 1:num_steps-1

    % update the states by RK4 method
    k1 = dt * calculate_derivatives(time(i), states, m1, m2, k, c, l, g);
    k2 = dt * calculate_derivatives(time(i) + dt/2, states + k1/2, m1, m2, k, c, l,
g);
    k3 = dt * calculate_derivatives(time(i) + dt/2, states + k2/2, m1, m2, k, c, l,
g);
    k4 = dt * calculate_derivatives(time(i) + dt, states + k3, m1, m2, k, c, l, g);

    states = states + (k1 + 2*k2 + 2*k3 + k4) / 6;

    x1(i+1) = states(1);
    y1(i+1) = states(2);
    vx1(i+1) = states(3);
```

```
    vy1(i+1) = states(4);
    x2(i+1) = states(5);
    y2(i+1) = states(6);
    vx2(i+1) = states(7);
    vy2(i+1) = states(8);

    % Update distance
    distance_between_masses(i+1) = sqrt((x2(i+1) - x1(i+1))^2 + (y2(i+1) -
y1(i+1))^2);

    % Update energy
    kinetic_energy(i+1) = 0.5*m1*((vx1(i+1))^2 + (vy1(i+1))^2) +
0.5*m2*((vx2(i+1))^2 + (vy2(i+1))^2);
    spring_pot_energy(i+1) = 0.5*k*((distance_between_masses(i+1) - l)^2);
    grav_pot_energy(i+1) = m1*g*y1(i+1) + m2*g*y2(i+1);
    total_energy(i+1) =  kinetic_energy(i+1) + spring_pot_energy(i+1) +
grav_pot_energy(i+1);
end

% round off the energies
for i = 1:num_steps
    kinetic_energy(i) = round(kinetic_energy(i), 3);
    spring_pot_energy(i) = round(spring_pot_energy(i), 3);
    grav_pot_energy(i) = round(grav_pot_energy(i), 3);
    total_energy(i) = round(total_energy(i), 3);
end

% Plot results in 1 plot
figure;

% x vs time plot
subplot(2,3,1);
plot(time, x1, 'b',time,x2,'r');
xlabel('t');
ylabel('x displacement');
legend('m1', 'm2');
title('x displacement vs. Time ');

% y vs time plot
subplot(2,3,2);
plot(time, y1, 'b',time,y2,'r');
xlabel('t');
ylabel('y displacement');
legend('m1', 'm2');
title('y displacement vs. Time');

% vx vs time plot
subplot(2,3,3);
plot(time, vx1, 'b',time,vx2,'r');
xlabel('t');
ylabel('x velocity');
legend('m1', 'm2');
title('x velocity vs. Time');

% vy vs time plot
subplot(2,3,4);
plot(time, vy1, 'b',time,vy2,'r');
xlabel('t');
ylabel('y velocity');
legend('m1', 'm2');
title('y velocity vs. Time');

% Energy vs time plot
subplot(2,3,5);
plot(time, total_energy, 'b');
xlabel('t');
ylabel('Total Energy');
title('Total Energy vs. Time ');
```

```
% function to calculate derivatives
function derivatives = calculate_derivatives(~, states, m1, m2, k, c, l, g)
    x1 = states(1);
    y1 = states(2);
    vx1 = states(3);
    vy1 = states(4);
    x2 = states(5);
    y2 = states(6);
    vx2 = states(7);
    vy2 = states(8);

    distance = sqrt((x2 - x1)^2 + (y2 - y1)^2);
    spring_force = k * (distance - l);
    damper_force_x = c * (vx2 - vx1);
    damper_force_y = c * (vy2 - vy1);

    ax1 = (spring_force * (x2 - x1)) / (m1 * distance) + damper_force_x / m1;
    ay1 = (spring_force * (y2 - y1)) / (m1 * distance) + damper_force_y / m1 - g;

    ax2 = (spring_force * (x1 - x2)) / (m2 * distance) - damper_force_x / m2;
    ay2 = (spring_force * (y1 - y2)) / (m2 * distance) - damper_force_y / m2 - g;

    derivatives = [vx1; vy1; ax1; ay1; vx2; vy2; ax2; ay2];
end
```



c) If c=0 (The total energy becomes constant without damping)



d) In absence of gravity (g=0) :-