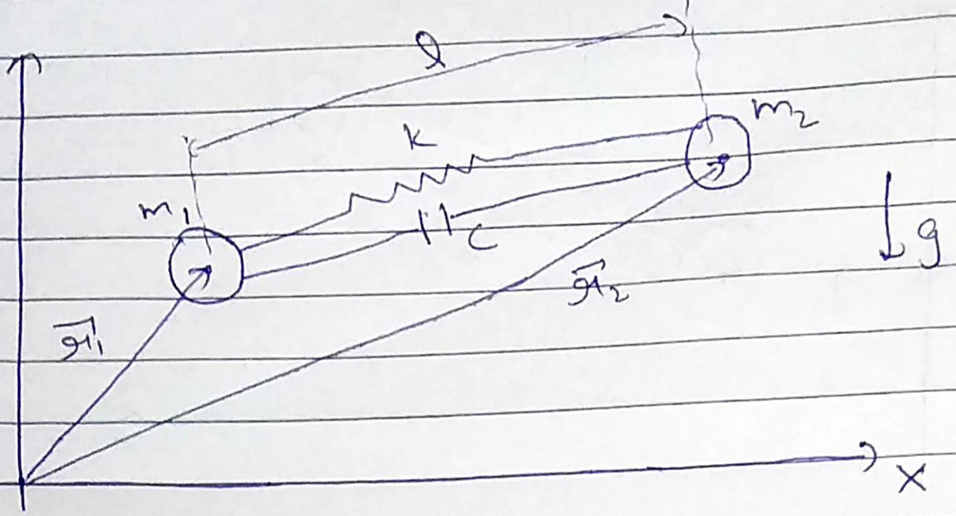
ME 3030
Assignment - 1

Q



Spring force

$$\vec{F}_s = k\left(|\bar{r}_2-\bar{r}_1|-l\right)\left(\frac{\bar{r}_2-\bar{r}_1}{|\bar{r}_2-\bar{r}_1|}\right)$$

Damping force $\quad \bar{F}_d = c(\dot{r}_2-\dot{r}_1)$

Gravity force $\quad \vec{F}_g = -m\vec{g}$

$$r_1 = \begin{Bmatrix} x_1(t) \\ y_1(t) \end{Bmatrix} \quad r_2 = \begin{Bmatrix} x_2(t) \\ y_2(t) \end{Bmatrix} \quad \dot{r}_1 = \begin{Bmatrix} \dot{x}_1(t) \\ \dot{y}_1(t) \end{Bmatrix}$$

$$\dot{r}_2 = \begin{Bmatrix} \dot{x}_2(t) \\ \dot{y}_2(t) \end{Bmatrix}$$

Writing equations for m_1:

$$m_1\begin{Bmatrix} \ddot{x}_1(t) \\ \ddot{y}_1(t) \end{Bmatrix} = k\,\frac{(|\bar{r}_2-\bar{r}_1|-l)}{|\bar{r}_2-\bar{r}_1|}\begin{Bmatrix} x_2-x_1 \\ y_2-y_1 \end{Bmatrix} + c\begin{Bmatrix} \dot{x}_2-\dot{x}_1 \\ \dot{y}_2-\dot{y}_1 \end{Bmatrix} - m_1\begin{Bmatrix} 0 \\ g \end{Bmatrix}$$

Writing equations for m_2:

$$m_2\begin{Bmatrix} \ddot{x}_2(t) \\ \ddot{y}_2(t) \end{Bmatrix} = k\left(\frac{|\bar{r}_2-\bar{r}_1|-l}{|\bar{r}_2-\bar{r}_1|}\right)\begin{Bmatrix} x_1-x_2 \\ y_1-y_2 \end{Bmatrix} - c\begin{Bmatrix} \dot{x}_2-\dot{x}_1 \\ \dot{y}_2-\dot{y}_1 \end{Bmatrix} - m_2\begin{Bmatrix} 0 \\ g \end{Bmatrix}$$

Assignment 3
Aaryan, CO21BTECH11001

The problem is solved using Newton raphson's method combined with RK4 integration technique.

Given the values of variables:

$$c = 5.0$$
$$g = 9.8$$
$$final\_time = 2.0s$$

Initial guess taken for the velocities:

$$\dot{x}_1(0) = 1.0$$
$$\dot{y}_1(0) = -1.0$$
$$\dot{x}_2(0) = -1.0$$
$$\dot{y}_2(0) = 1.0$$

The value of initial velocities turn out to be:

$$\dot{x}_1(0) = 30.7753$$
$$\dot{y}_1(0) = -1.6437$$
$$\dot{x}_2(0) = -24.9153$$
$$\dot{y}_2(0) = 22.6025$$

```
% 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 = 9.8;     % Acceleration due to gravity in m/s^2

% Positions in order x1, y1, x2, y2
init_position = [0.0; 0.0; 0.5; 0.0];
final_position = [1.0; 1.0; 1.0; 1.5];

% Guess for initial_velocities in order vx1, vy1, vx2, vy2
v = [1.0; -1.0; -1.0; 1.0];

% Small change
dv = 1.0e-3;

% Convergence criteria
eps = 1e-2;

initial_time = 0.0;
final_time = 2.0;

while true

    temp_final_position = spring_mass_rk4_solve(init_position, v,
 initial_time, final_time, m1, m2, k, c, l, g);

    f = temp_final_position - final_position;

    if (max(abs(f))) < eps
        temp_final_position
        break
    end

    J = zeros(4, 4);

    for i = 1:4
        temp_v = v;
        temp_v(i) = temp_v(i) + dv;
        temp_final_position_dv = spring_mass_rk4_solve(init_position, temp_v,
 initial_time, final_time, m1, m2, k, c, l, g);
        J_col = zeros(4, 1);
        for j = 1:4
            derivative = (temp_final_position_dv(j) -
 temp_final_position(j)) / dv;
            J_col(j) = derivative;
        end
        J(:, i) = J_col;
    end
```

```
    v = v - J \ f;

end

v

function final_position = spring_mass_rk4_solve(init_position, init_velocity,
 init_time, final_time, m1, m2, k, c, l, g)

    % Initial conditions
    x1_initial = init_position(1);
    y1_initial = init_position(2);
    vx1_initial = init_velocity(1);
    vy1_initial = init_velocity(2);

    x2_initial = init_position(3);
    y2_initial = init_position(4);
    vx2_initial = init_velocity(3);
    vy2_initial = init_velocity(4);

    % Time step
    dt = 1.0e-5;

    % Number of time steps
    num_steps = round((final_time - init_time) / dt + 1);

    % 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);

    % Set initial conditions
    x1(1) = x1_initial;
    y1(1) = y1_initial;
    vx1(1) = vx1_initial;
    vy1(1) = vy1_initial;

    x2(1) = x2_initial;
    y2(1) = y2_initial;
    vx2(1) = vx2_initial;
    vy2(1) = vy2_initial;

    arr = [x1(1) ; y1(1) ; vx1(1) ; vy1(1) ; x2(1) ; y2(1) ; vx2(1) ; vy2(1)];

    % Using RK4 method
```

```
    for i = 1:num_steps-1

        k1 = dt * calculate_k_util(arr, m1, m2, k, c, l, g);
        k2 = dt * calculate_k_util(arr + 0.5 * k1, m1, m2, k, c, l, g);
        k3 = dt * calculate_k_util(arr + 0.5 * k2, m1, m2, k, c, l, g);
        k4 = dt * calculate_k_util(arr + k3, m1, m2, k, c, l, g);

        arr = arr + (k1 + 2.0 * k2 + 2.0 * k3 + k4) / 6.0;

        x1(i+1) = arr(1);
        y1(i+1) = arr(2);
        vx1(i+1) = arr(3);
        vy1(i+1) = arr(4);

        x2(i+1) = arr(5);
        y2(i+1) = arr(6);
        vx2(i+1) = arr(7);
        vy2(i+1) = arr(8);

    end

    final_position = [arr(1); arr(2); arr(5); arr(6)];

end

function calculated_k = calculate_k_util(arr, m1, m2, k, c, l, g)
    x1 = arr(1);
    y1 = arr(2);
    vx1 = arr(3);
    vy1 = arr(4);
    x2 = arr(5);
    y2 = arr(6);
    vx2 = arr(7);
    vy2 = arr(8);

    distance = sqrt((x1 - x2)^2 + (y1 - y2)^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;

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

temp_final_position =

```
    0.9973
    0.9975
    1.0052
    1.5092


v =

   30.7753
   -1.6437
  -24.9153
   22.6025
```

Published with MATLAB® R2023a