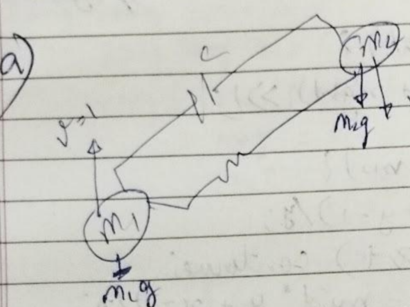
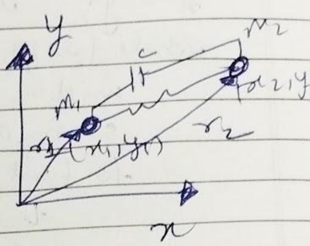
ME3030 Assignment 2
CO21BTECH11004

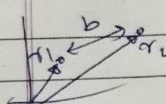
Que

Name- Darpan Gaur
Roll No- CO21BTECH11004

Que a)







$\Rightarrow$ $b = r_2 - r_1$, $\hat{b} = \dfrac{r_2 - r_1}{|r_2 - r_1|}$ direction

$|b| = |r_2 - r_1|$

Extension/Compression in spring $= |r_2 - r_1| - l$

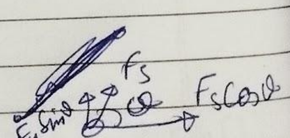
Spring Force $\Rightarrow k\left(|r_2 - r_1| - l\right)\hat{b}$

Damping Force $\Rightarrow c(\dot{r}_2 - \dot{r}_1)$

Gravitational Force $= -mg$

Eq $\Rightarrow$ $m\ddot{r} = k\left(|r_2 - r_1| - l\right)\hat{b} + c(\dot{r}_2 - \dot{r}_1) - mg$

In x-y plane
x → Take x component
y → Take y component



$$\cos\theta = \frac{x_2 - x_1}{\underbrace{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}_{|r_2 - r_1|}}$$

$$\sin\theta = \frac{y_2 - y_1}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}$$

Equations

DATE: / /
PAGE:

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

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

Solving Using 4^{th} order Runge Kutta (RK4)

Put,
$$k_1 = dt \cdot f(t_n, y_n)$$
$$k_2 = dt \cdot f\left(t_n + \frac{dt}{2}, y_n + \frac{k_1}{2}\right)$$
$$k_3 = dt \cdot f\left(t_n + \frac{dt}{2}, y_n + \frac{k_2}{2}\right)$$
$$k_4 = dt \cdot f\left(t_n + dt, y_n + k_3\right)$$

$$y_{n+1} = y_n + \left(\frac{k_1 + 2k_2 + 2k_3 + k_4}{6}\right)$$

In this question, there are eight variables

$z_1 = x_1$, $z_2 = x_2$
$z_3 = y_1$, $z_4 = y_2$
$z_5 = \dot{x}_1$, $z_6 = \dot{x}_2$
$z_7 = \dot{y}_1$, $z_8 = \dot{y}_2$

Make ↳ Array of variables of $z_1, \ldots z_8$
and make find_f → function
for returning derivative

& update

$$\text{variables} = \text{variables} + \left(\frac{k_1 + 2k_2 + 2k_3 + k_4}{6}\right)$$

replace f by find f function
which returns array of derivatives
corresponding eight variables

Multivariable Newton Rapson form

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix}^{i+1} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix}^{i} - \underbrace{\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix}^{-1}}_{\text{Jacobian}} \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{Bmatrix}^{i}$$

Using Newton Rapson

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Here we find the final coordinates using the initial coordinates and guess initial velocity.

As we are dealing with multiple variables so using Jacobian for derivative.

$\boxed{\text{Recipe}}$

Find final position using guess → fpos
Find final position using perturbated guess → fposb

f = fpos − final position − given
fb = fposb − final position − given

~~[illegible]~~ Form Jacobian

$v = v - J/f$

J → 4×4 matrix
as four function & four variables

solving using Newton Rapson

Stop when difference between guessed final position & initial final position is less than ~~a~~ a tolerance

Code

```
% Name           :- Darpan Gaur
% Roll Number    :- CO21BTECH11004

% Constants
l = 0.5;
m1 = 1.0;
m2 = 1.0;
c = 5.0;
k = 1000.0;
g = 9.81;
Ti = 0.0;
Tf = 2.0;

% Given boundary conditions
inPos = [0.0; 0.0; 0.5; 0.0];
fPos = [1.0; 1.0; 1.0; 1.5];

% Guess for velocity
v = [5.0; 10.0; -1.0; 5.0];

% Small change
dv = 1.0e-3;

% Tolerance
eps = 1e-3;
eps2 = 1e-6;

while true

   pos = implicit_euler_solve(inPos, v, Ti, Tf, m1, m2, k, c, l, g,
eps2);

   err = pos - fPos;
   if (max(abs(err))) < eps
       fprintf("Final Position: [x1 y1 x2 y2]'\n");
```

```
        disp(pos);
        break
    end
    J = zeros(4, 4);

    for i = 1:4
        v_new = v;
        v_new(i) = v_new(i) + dv;
        pos_dv = implicit_euler_solve(inPos, v_new, Ti, Tf, m1, m2,
k, c, l, g, eps2);
        J_col = zeros(4, 1);
        for j = 1:4
            derivative = (pos_dv(j) - pos(j)) / dv;
            J_col(j) = derivative;
        end
        J(:, i) = J_col;
    end

    v = v - J \ err;
end

format longg
fprintf("Final velocity: [vx1 vy1 vx2 vy2]'\n");
disp(v);
function fPos = implicit_euler_solve(inPos, init_velocity,
init_time, Tf, m1, m2, k, c, l, g, eps)
    dt = 1.0e-4;    % Time step (you may need to adjust this)
    % Small change in values for calculating numerical derivative
    small_change = 1.0e-4;
    num_steps = round((Tf - init_time) / dt + 1);


    % Initialize arrays to store positions and velocities
    x1 = zeros(1, num_steps);
    y1 = zeros(1, num_steps);
    x1_1 = zeros(1, num_steps);
```

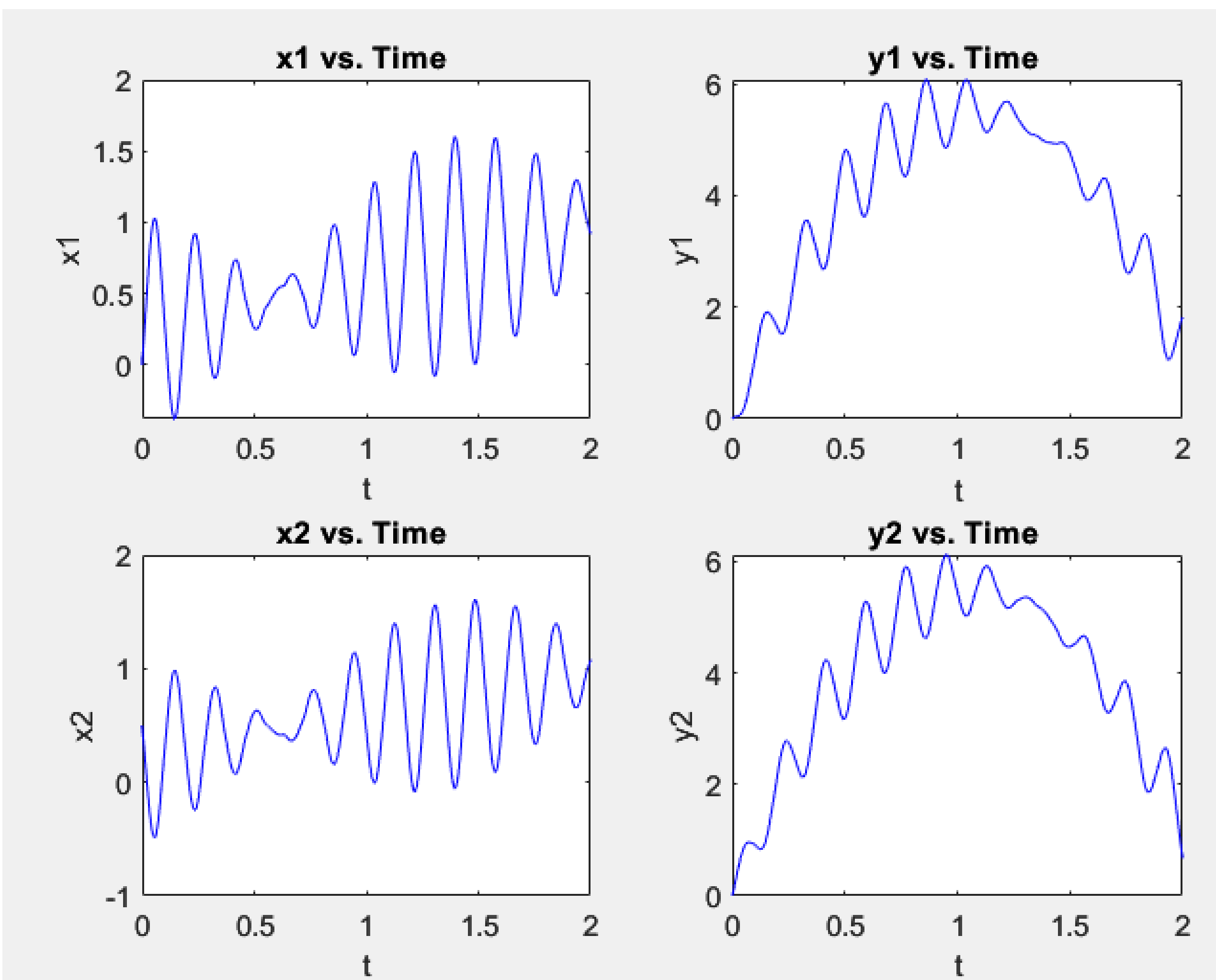
```
    y1_1 = zeros(1, num_steps);

    x2 = zeros(1, num_steps);
    y2 = zeros(1, num_steps);
    x2_1 = zeros(1, num_steps);
    y2_1 = zeros(1, num_steps);

    % Set initial conditions
    x1(1) = inPos(1);
    y1(1) = inPos(2);
    x1_1(1) = init_velocity(1);
    y1_1(1) = init_velocity(2);

    x2(1) = inPos(3);
    y2(1) = inPos(4);
    x2_1(1) = init_velocity(3);
    y2_1(1) = init_velocity(4);

    variables = [x1(1) ; y1(1) ; x1_1(1) ; y1_1(1) ; x2(1) ; y2(1) ;
x2_1(1) ; y2_1(1)];

    % Using implicit euler method
    for i = 1:num_steps-1

        guess = variables;
        f_values = find_f(guess, variables, dt, m1, m2, k, c, l, g);
        while (max(abs(f_values)) > eps)
            J = zeros(8, 8);

            for j = 1:8
                temp = guess;
                temp(j) = temp(j) + small_change;
                temp_f_values = find_f(temp, variables, dt, m1, m2,
k, c, l, g);
                J_col = zeros(8, 1);
                for ind = 1:8
```

```
                    derivative = (temp_f_values(ind) - f_values(ind))
/ small_change;
                    J_col(ind) = derivative;
                end
                J(:, j) = J_col;
            end
            guess = guess - J \ f_values;
            f_values = find_f(guess, variables, dt, m1, m2, k, c, l,
g);
        end
        variables = guess;
        x1(i+1) = variables(1);     y1(i+1) = variables(2);
        x1_1(i+1) = variables(3);   y1_1(i+1) = variables(4);
        x2(i+1) = variables(5);     y2(i+1) = variables(6);
        x2_1(i+1) = variables(7);   y2_1(i+1) = variables(8);
    end
    fPos = [variables(1); variables(2); variables(5); variables(6)];
end
function values = find_f(variables, prev_arr, dt, m1, m2, k, c, l,
g)
    x1 = variables(1);        y1 = variables(2);
    x1_1 = variables(3);      y1_1 = variables(4);
    x2 = variables(5);        y2 = variables(6);
    x2_1 = variables(7);      y2_1 = variables(8);
    b = sqrt((x1 - x2)^2 + (y1 - y2)^2);
    fSprinf = k * (b - l);
    fD_x = c * (x2_1 - x1_1);
    fD_y = c * (y2_1 - y1_1);
    x1_2 = (fSprinf * (x2 - x1)) / (m1 * b) + fD_x / m1;
    y1_2 = (fSprinf * (y2 - y1)) / (m1 * b) + fD_y / m1 - g;
    x2_2 = -(fSprinf * (x2 - x1)) / (m2 * b) - fD_x / m2;
    y2_2 = -(fSprinf * (y2 - y1)) / (m2 * b) - fD_y / m2 - g;
    temp = [x1_1 ; y1_1 ; x1_2 ; y1_2 ; x2_1 ; y2_1 ; x2_2 ; y2_2];
    values = variables - prev_arr - dt * temp;
end
```

Multiple answers possible for initial velocity based on condition and guess

- For **c = 5.0, g = 9.81**
  - guess = [5 10 -1 5] (vx1 vy1 vx2 vy2)
  - **Initial velocity came :- [191.348,**
    -2246698.1366,
    -190.598,
    2246719.007]

- For **c=0, g=9.81**
  - guess = [10 10 -10 5] (vx1 vy1 vx2 vy2)
  - **Initial velocity came :- [25.34, 3.47, -24.59, 17.39]**
  - Putting then in Assignent 2 code and plotting the x1, x2, y1, y2 we get x1 = 1.0 , y1 = 1.0, x2 = 1.0, y2 = 1.5, at t=2, which was given for final state.



- For **c=0, g=0**
  - guess = [5 5 -5 5] (vx1 vy1 vx2 vy2)
  - **Initial velocity came :- [15.789, -0.3237, -15.0397, 1.5737]**
  - Putting then in Assigent 2 code and plotting the x1, x2, y1, y2 we get x1 = 1.0 , y1 = 1.0, x2 = 1.0, y2 = 1.5, at t=2, which was given for final state.