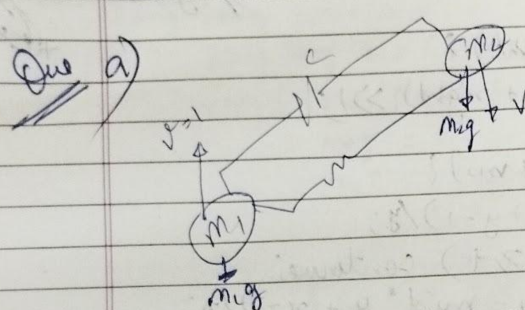
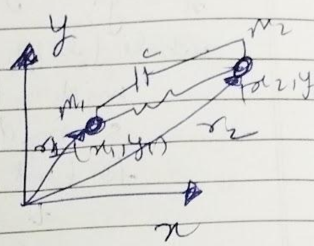
ME3030 Assignment 2
CO21BTECH11004

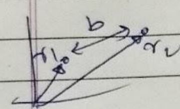
Que

Name- Darpan Gaur
Roll No- CO21BTECH11004

Que a)







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

$|b| = |r_2 - r_1|$

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

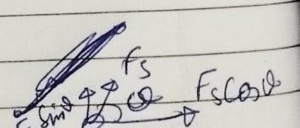
Spring Force $\Rightarrow k(|r_2 - r_1| - l)\hat{b}$

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

Gravitational Force $= -mg$

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

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



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

Equations

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

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

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

Put, $k_1 = dt . f(t_n, y_n)$

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

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

$k_4 = dt . f\left(t_n + dt, y_n + k_3\right)$

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

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

variables = 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

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

PAGE:

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.

[Recipe]

\# Find final position using guess → fpos

\# Find final position using perturbated guess → fposb

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

\# Form Jacobian

\# $v = v - J/f$

\# $J \rightarrow 4\times4$ matrix
as four function & four variables

solving using Newton Rapson

\# Stop when difference between guessed final position & initial final position is less than tolerance

Code

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

% constants
l = 0.5;
m1 = 1.0;
m2 = 1.0;
c = 0.0;
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 = [10.0; 10.0; -10.0; 5.0];

% Small change
dv = 1.0e-3;

% tolerance
eps = 1.0e-3;

while true
   pos = rk4_solve(inPos, v, Ti, Tf, m1, m2, k, c, l, g);
   err = pos - fPos;
   % L1 scheme for error
   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 = rk4_solve(inPos, v_new, Ti, Tf, m1, m2, k, c, l, g);
        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

fprintf("Final velocity :- [vx1 vy1 vx2 vy2]' \n")
disp(v)

function finalPos = rk4_solve(inPos, init_velocity, Ti, Tf, m1, m2, k, c,
l, g)

    dt = 0.00001;

    % Number of time steps
    num_steps = round((Tf - Ti) / 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);
    x1_2 = zeros(1, num_steps);
    y1_2 = zeros(1, num_steps);

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

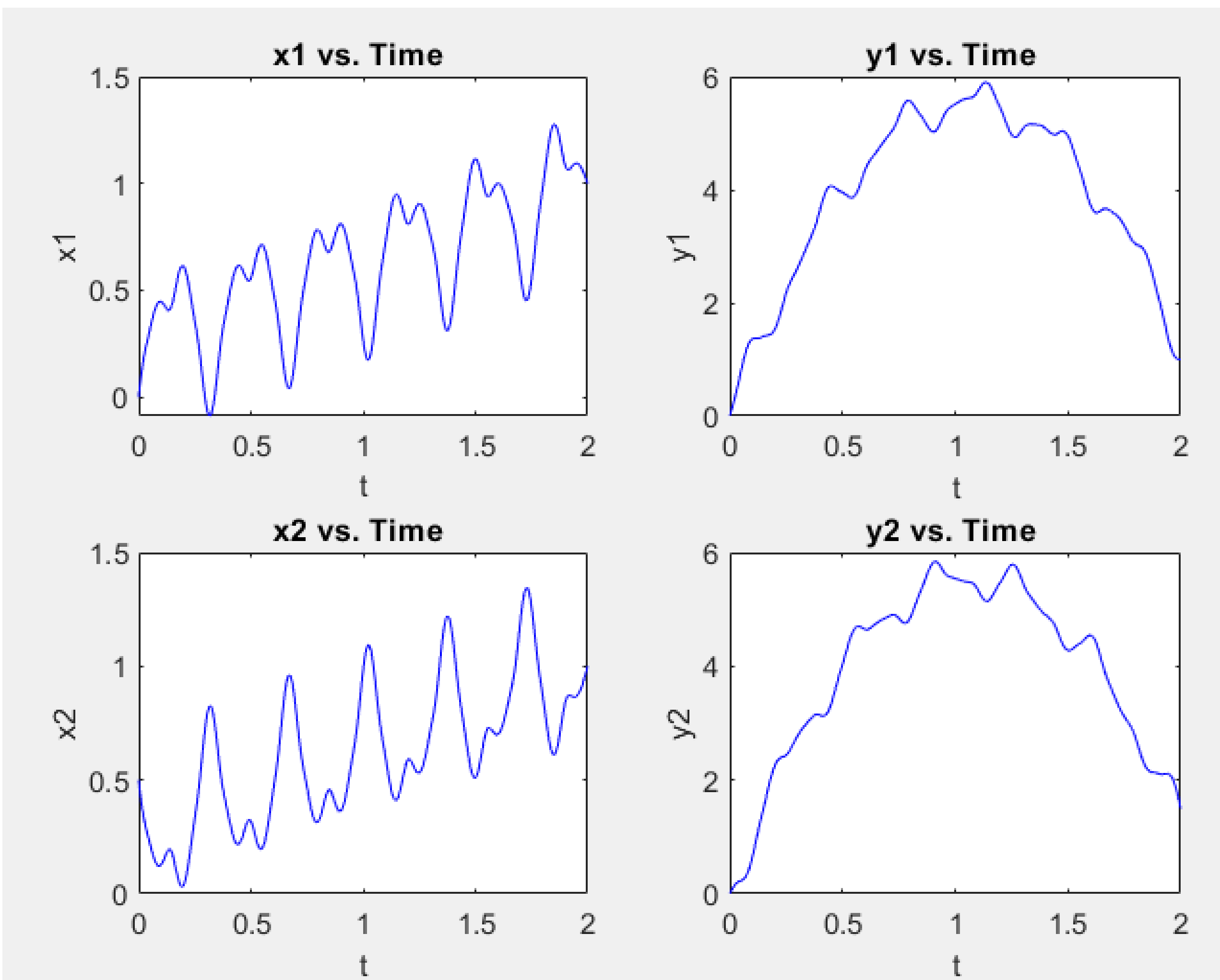
```
   y1_1(1) = init_velocity(2);
   x2(1) = inPos(3);
   y2(1) = inPos(4);
   x1_2(1) = init_velocity(3);
   y1_2(1) = init_velocity(4);

   variables = [x1(1) ; y1(1) ; x1_1(1) ; y1_1(1) ; x2(1) ; y2(1) ;
x1_2(1) ; y1_2(1)];

   % Using RK4 method
   for i = 1:num_steps-1
       k1 = dt * find_f(variables, m1, m2, k, c, l, g);
       k2 = dt * find_f(variables + 0.5 * k1, m1, m2, k, c, l, g);
       k3 = dt * find_f(variables + 0.5 * k2, m1, m2, k, c, l, g);
       k4 = dt * find_f(variables + k3, m1, m2, k, c, l, g);
       variables = variables + (k1 + 2.0 * k2 + 2.0 * k3 + k4) / 6.0;
       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);
       x1_2(i+1) = variables(7); y1_2(i+1) = variables(8);
   end
   finalPos = [variables(1); variables(2); variables(5); variables(6)];
end

function f = find_f(variables, 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);
   fSpring = k * (b - l);
   fD_x = c * (x2_1 - x1_1);
   fD_y = c * (y2_1 - y1_1);
   x1_2 = (fSpring * (x2 - x1)) / (m1 * b) + fD_x / m1;
   y1_2 = (fSpring * (y2 - y1)) / (m1 * b) + fD_y / m1 - g;
   x2_2 = -(fSpring * (x2 - x1)) / (m2 * b) + fD_x / m2;
   y2_2 = -(fSpring * (y2 - y1)) / (m2 * b) + fD_y / m2 - g;

   f = [x1_1 ; y1_1 ; x1_2 ; y1_2 ; x2_1 ; y2_1 ; x2_2 ; y2_2];
```

```
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 :- [24.0148, 9.8036, -18.0844, 10.8610]**

- For **c=0, g=9.81**
  - guess = [10 10 -10 5] (vx1 vy1 vx2 vy2)
  - **Initial velocity came :- [9.3737, 13.6445, -8.6237, 7.2255]**
  - 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 :- [12.5065, 0.4621, -11.7565, 0.7879]**
    - 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.