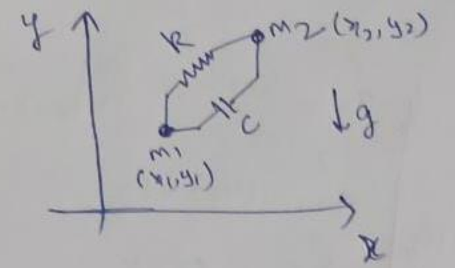
ME3030

Assignment 4

ME21BTECH11001

Abhishek Ghosh



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

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,

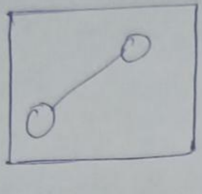
$$m_1 \begin{Bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \end{Bmatrix} = \frac{k(|r_2 - r_1| - L)}{|r_2 - r_1|} \begin{Bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{Bmatrix} + c \begin{Bmatrix} \dot{x}_2 - \dot{x}_1 \\ \dot{y}_2 - \dot{y}_1 \end{Bmatrix} - m_1 g \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

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

$\dot{x}_1(0) = \alpha_1$
$\dot{x}_2(0) = \alpha_2$
$\dot{y}_1(0) = \beta_1$
$\dot{y}_2(0) = \beta_2$ $\longrightarrow$  $\longrightarrow$

Newton Rhapson method :-

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

$$\begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{Bmatrix}^{+} = \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{Bmatrix}^{-} - \begin{bmatrix} \frac{\partial f_1}{\partial \alpha_1} & \frac{\partial f_1}{\partial \alpha_2} & \frac{\partial f_1}{\partial \beta_1} & \frac{\partial f_1}{\partial \beta_2} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_4}{\partial \alpha_1} & \frac{\partial f_4}{\partial \alpha_2} & \frac{\partial f_4}{\partial \beta_1} & \frac{\partial f_4}{\partial \beta_2} \end{bmatrix}^{-1} \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{Bmatrix}^{-}$$

The implicit Euler method computes the approximations using

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

```
% ME21BTECH11001 Abhishek Ghosh
% 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 = [0.5; 0.5; 0.5; 0.5];

% Small change in velocity
dv = 1.0e-3;

% Convergence criteria for final position
epsilon = 1e-2;

% Convergence criteria for next values in the integrator
eps_next_vals = 1e-6;

initial_time = 0.0;
final_time = 2.0;

while true

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

    f = temp_final_position - final_position;

    if (max(abs(f))) < epsilon
        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 = implicit_solve(init_position, temp_v,
initial_time, final_time, m1, m2, k, c, l, g, eps_next_vals);
        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
```

```matlab
function final_position = implicit_solve(init_position, init_velocity, init_time,
final_time, m1, m2, k, c, l, g, epsilon)

    % Initial conditions
    x1_i = init_position(1);
    y1_i = init_position(2);
    vx1_i = init_velocity(1);
    vy1_i = init_velocity(2);

    x2_i = init_position(3);
    y2_i = init_position(4);
    vx2_i = init_velocity(3);
    vy2_i = init_velocity(4);

    % Time step
    dt = 1.0e-4;

    % Small change in values for calculating numerical derivative
    small_change = 1.0e-4;

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


    % 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_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;

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

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

        guess = arr;
        f = calculate_derivative(guess, arr, dt, m1, m2, k, c, l, g);

        while (max(abs(f)) > epsilon)
            % Jacobian
            J = zeros(8, 8);

            for j = 1:8
                temp = guess;
                temp(j) = temp(j) + small_change;
                current = calculate_derivative(temp, arr, dt, m1, m2, k, c, l, g);
                J_col = zeros(8, 1);
                for ind = 1:8
                    derivative = (current(ind) - f(ind)) / small_change;
                    J_col(ind) = derivative;
                end
                J(:, j) = J_col;
```

```
            end

            guess = guess - J \ f;

            f = calculate_derivative(guess, arr, dt, m1, m2, k, c, l, g);

        end

        arr = guess;

        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 calculate_der = calculate_derivative(arr, prev_arr, dt, 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;

    temp = [vx1 ; vy1 ; ax1 ; ay1 ; vx2 ; vy2 ; ax2 ; ay2];
    calculate_der = arr - prev_arr - dt * temp;
end
```

```
temp_final_position =

    1.0000
    0.9960
    1.0000
    1.4960
```

```
v =

   1.0e+06 *

    0.0002
   -2.2467
   -0.0002
    2.2467
```

The above values are for vx1,vy1,vx2,vy2