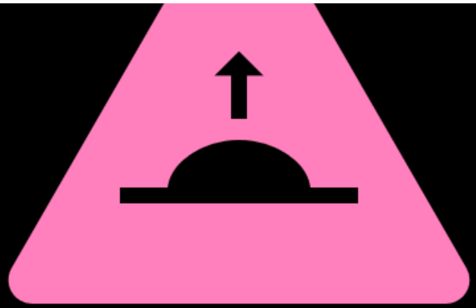
ENGINEERING A SMOOTHER RIDE:

VIBRATION REDUCTION IN DRIVER SEAT

Pavan Prudhvi - 234103427
Harshith - 234103420
Anjeet - 234103407

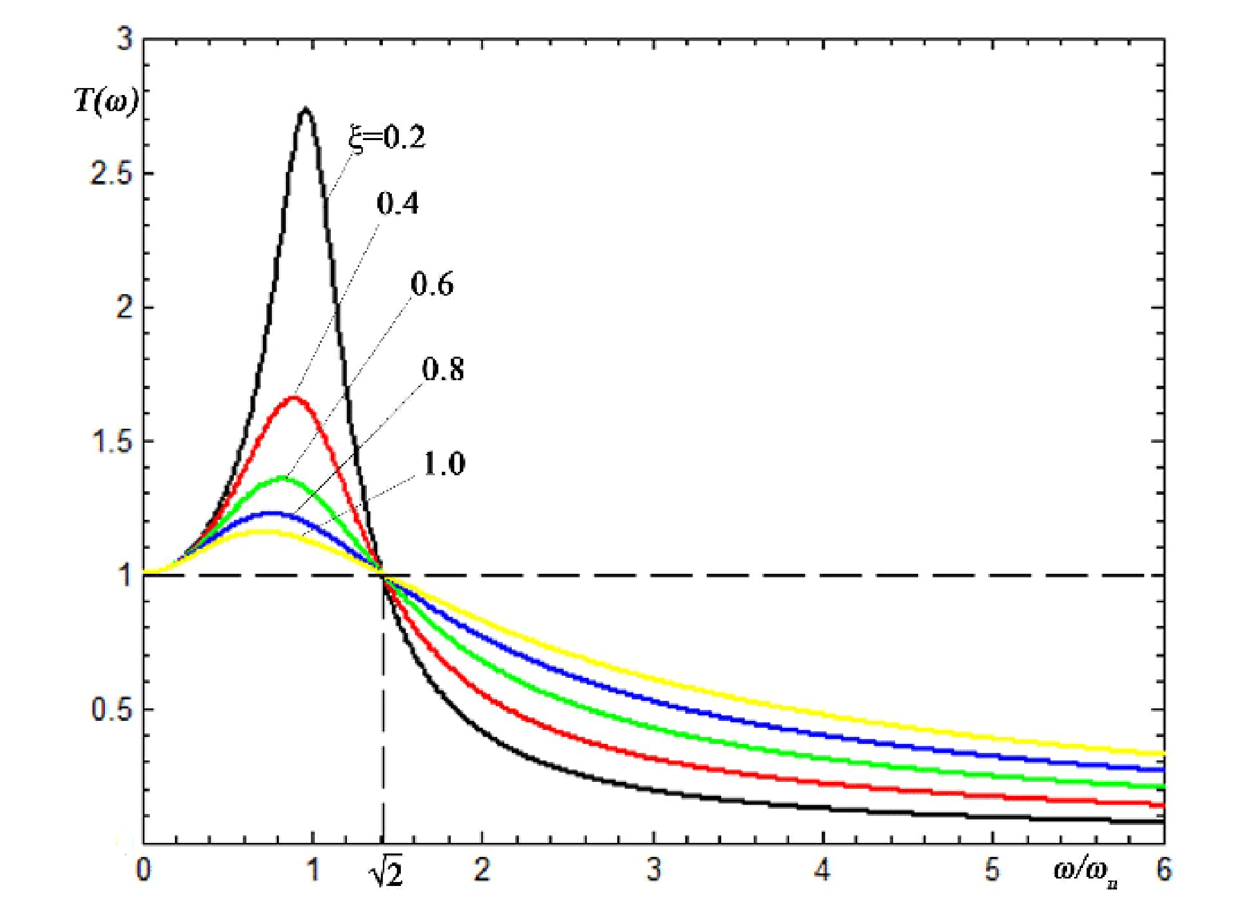
Problem Statement:



Excessive vibration of a driver's seat in a vehicle can have several effects on the driver, potentially leading to discomfort, fatigue, and even health issues over time.

To improve the driver’s ride quality, the suspension systems of the vehicle were optimized and controlled. However, the structure of the semi-active or active suspension systems was very complicated and expensive. Thus, it was limited in application on all vehicles and the driver's ride quality was also limited.

Problem :



$$\frac{\omega}{\omega_n} > \sqrt{2}$$

$$\omega_n = \sqrt{(\frac{k}{m})}$$

- We can reduce natural frequency by reducing the stiffness.
- Reducing stiffness will leads to reduction in load bearing capacity of the system.

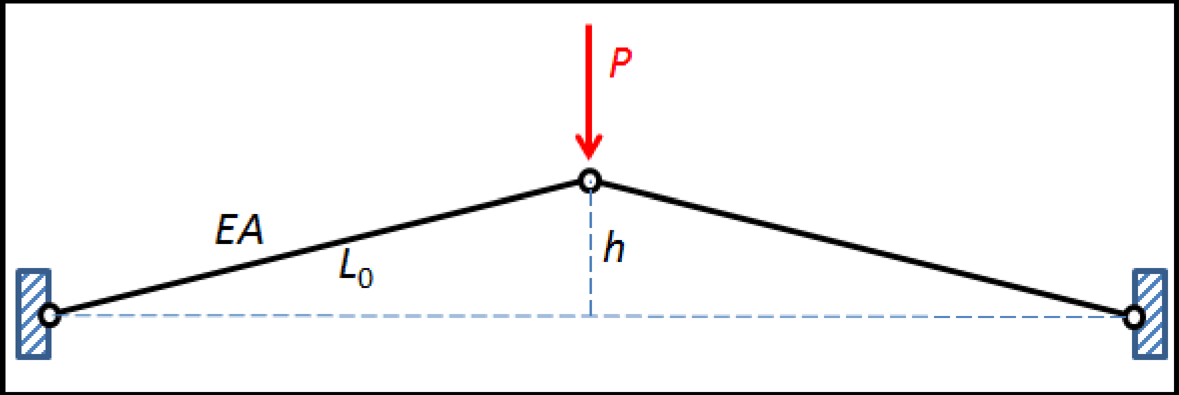
Moving Towards the Solution:

To enhance the driver's ride quality .

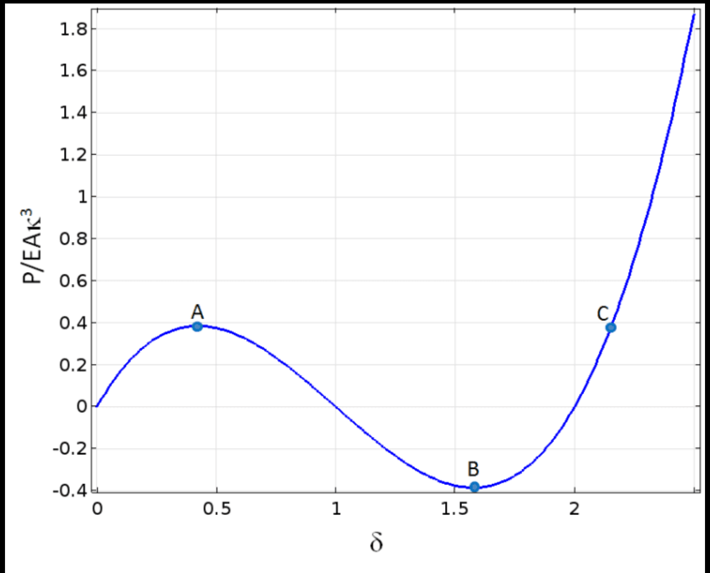
The seat suspension with negative stiffness structure was added to reduce the vibrations of the driver’s seat.

Negative Stiffness Mechanism

- In a typical structure, an increase in force causes an increase in displacement.
- Negative-stiffness mechanisms are those that can, exhibit increasing displacement with decreasing force during some region of their force-displacement relationship.
- It is a passive approach for achieving low vibration environments and isolation against low frequency vibrations
- Negative Stiffness Mechanism(NSM) reduces effective stiffness of the system without reducing the weight bearing capacity and leads to High-Static Low-Dynamic stiffness



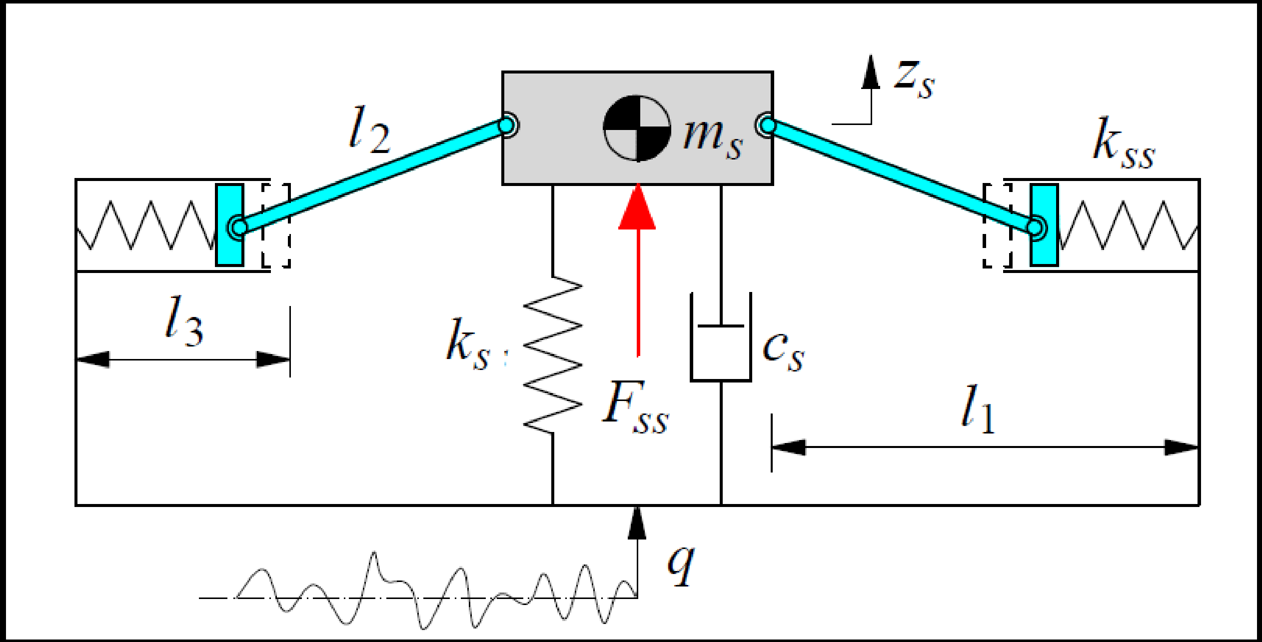
Two-bar structure under compression



Between points A and B, displacement is increasing while force is decreasing. Thus, the structure's stiffness is negative in that region.

Force (P/EAκ3) vs Displacement (δ) of a two-bar structure under compression.

Negative Stiffness Structure(NSS) system:



Z = z_s - q

$$m_s\ddot{z} + c_s\dot{z} + \ k_s z + 2k_{ss}pz = m_s(\omega^2 Y sin(\omega t))$$

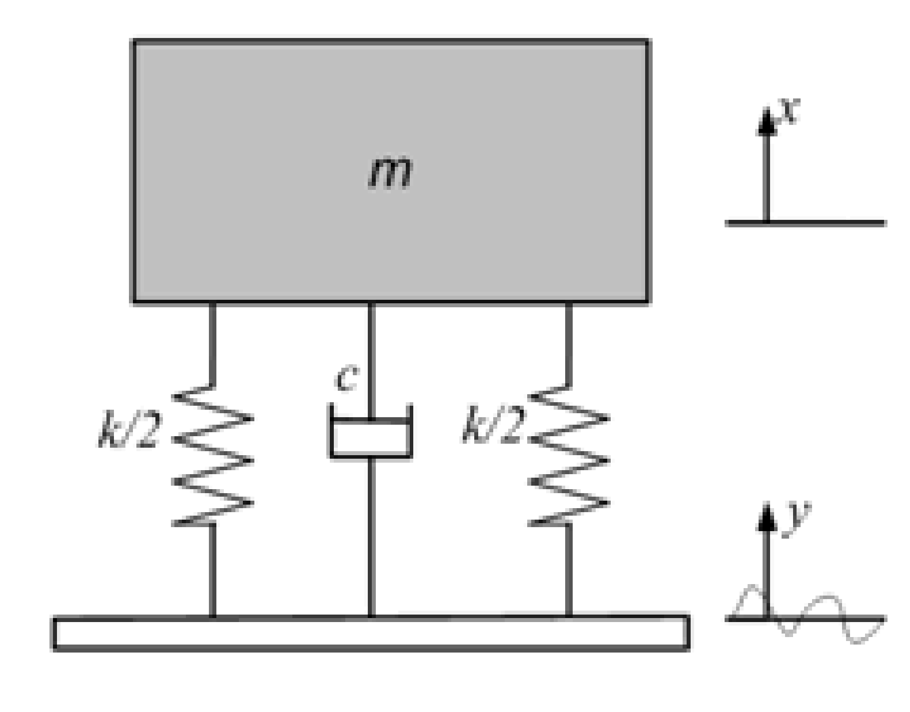
$$Where, p = \frac{l_3 - l_1}{\sqrt{(l_2 - z^2)}} + 1$$

Calculation

- The isolation efficiency of the suspension system was evaluated via its root mean square displacement (z_{ws}).
- ${z_{ws}}^2 = T^{-1} * \int_0^T z^2$ dt
- In order to evaluate the isolation efficiency of the seat's NSS and SS system, the smaller values of the z_{ws} are chosen as the objective functions.
- Root Mean Square Displacement (RMSD)-SS: 0.024531
- Root Mean Square Displacement (RMSD)-SS: 0.024531

SS system

- We can first assume the seat and suspension to consist of a simple mass, spring and damper system.



Equation of motion:

$$m\ddot{z} + c\dot{z} + kz = m\left(\omega^2 Y sin(\omega t)\right) \quad | \quad Z = x - y$$

MATLAB code – ode45

```
% Solve the first differential equation: mz'' + cz' + k1z + 2k2pz = m(ω^2)Ysin(ωt)
function dzdt1 = myODE3(t, z)
    % Define the parameters for the first equation
    m1 = 85; % Mass
    c1 = 250; % Damping coefficient
    k1 = 25000; % Spring constant
    k2 = 13600; % Another spring constant
    omega1 = 3.14; % Frequency
    Y1 = 1; % Amplitude
    l1 = 0.23; % Constant
    l2 = 0.17; % Constant
    l3 = 0.14; % Constant
    % Calculate p
    p = (l3- l1) / sqrt(l2^2 - z(1)^2) + 1;
    % Define the first differential equation
    dzdt1 = [z(2); (m1 * (omega1^2) * Y1 * sin(omega1 * t) - c1 * z(2) - k1 * z(1) - 2
* k2 * p * z(1)) / m1];
end
```

```
% Solve the second differential equation: mz'' + cz' + k1z = m(ω^2)Ysin(ωt)
function dzdt2 = myODE4(t, z)
    % Define the parameters for the second equation
    m2 = 85; % Mass
    c2 = 250; % Damping coefficient
    k2 = 25000; % Spring constant
    omega2 = 3.14; % Frequency
    Y2 = 1; % Amplitude

    % Define the second differential equation
    dzdt2 = [z(2); (m2 * (omega2^2) * Y2 * sin(omega2 * t) - c2 * z(2) - k2 * z(1)) /
m2];
end

% Define the parameters for the first equation
    m1 = 85; % Mass
    c1 = 250; % Damping coefficient
    k1 = 25000; % Spring constant
    k2 = 13600; % Another spring constant
    omega1 = 3.14; % Frequency
    Y1 = 1; % Amplitude
    l1 = 0.23; % Constant
    l2 = 0.17; % Constant
    l3 = 0.14; % Constant
```

MATLAB code – ode45

```
% Solve the first ODE using ode45

[t1, Z1] = ode45(@myODE3, tspan, z0);

% Solve the second ODE using ode45
[t2, Z2] = ode45(@myODE4, tspan, z0);

% Extract z values for both equations
z_values1 = Z1(:, 1);
z_values2 = Z2(:, 1);
```

```
% Create a single graph to compare the results of both equations
figure;
plot(t1, z_values1, 'b', 'LineWidth', 2);

hold on;
plot(t2, z_values2, 'r', 'LineWidth', 2);
xlabel('Time (t)');
ylabel('z(t)');
title('Comparison of Two Differential Equations');
legend('Equation 1', 'Equation 2');
```

MATLAB code – RK4 method

```
function [t, z] = R2(myODE, tspan, initial_conditions, h)
    t_initial = tspan(1);
    t_final = tspan(2);
    % Number of steps
    num_steps = round((t_final - t_initial) / h);
    % Initialize arrays to store results
    t = zeros(num_steps, 1);
    z = zeros(num_steps, length(initial_conditions));
```

```
t(1) = t_initial;
    z(1, :) = initial_conditions;
 for i = 1:num_steps
        % Current time
        ti = t(i);
        % Current state
        zi = z(i, :);
       k1 = h * myODE(ti, zi)';
        k2 = h * myODE(ti + h/2, zi + k1/2)';
        k3 = h * myODE(ti + h/2, zi + k2/2)';
        k4 = h * myODE(ti + h, zi + k3)';
        % Update time and state
        t(i + 1) = ti + h;
        z(i + 1, :) = zi + (k1 + 2*k2 + 2*k3 + k4)/6;
    end
end
```

```matlab
function dzdt1 = myODE3(t, z)
    m1 = 85; % Mass
    c1 = 250; % Damping coefficient
    k1 = 25000; % Spring constat
    k2 = 13600; % Another spring constant
    omega1 = 3.14; % Frequency
    Y1 = 1; % Amplitude
    l1 = 0.23; % Constant
    l2 = 0.17; % Constant
    l3 = 0.14; % Constant
p = (l3- l1) / sqrt(l2^2 - z(1)^2) + 1;
    % Define the first differential equation
    dzdt1 = [z(2); (m1 * (omega1^2) * Y1 * sin(omega1 * t) - c1 * z(2) - k1 * z(1) - 2 * k2 * p * z(1)) / m1];
end
```

```
function [t, z] = R3(myODE, tspan, initial_conditions, h)
t_initial = tspan(1);
    t_final = tspan(2);

    % Number of steps
    num_steps = round((t_final - t_initial) / h);

    % Initialize arrays to store results
    t = zeros(num_steps, 1);
    z = zeros(num_steps, length(initial_conditions));

    % Set initial conditions
    t(1) = t_initial;
    z(1, :) = initial_conditions;
```

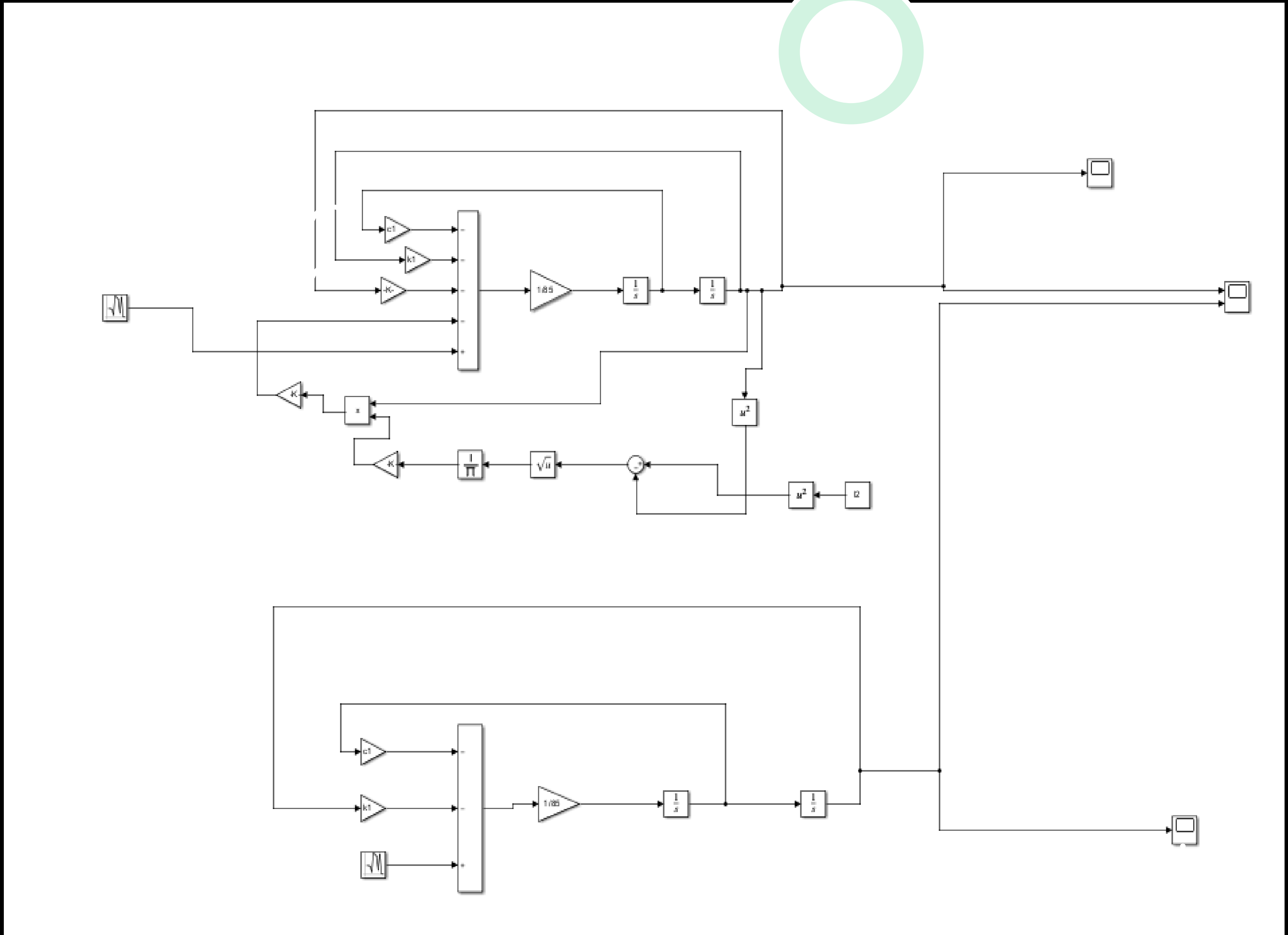
```
% Runge-Kutta integration
    for i = 1:num_steps
        % Current time
        ti = t(i);
        % Current state
        zi = z(i, :);
        % Runge-Kutta coefficients
        k1 = h * myODE(ti, zi)';
        k2 = h * myODE(ti + h/2, zi + k1/2)';
        k3 = h * myODE(ti + h/2, zi + k2/2)';
        k4 = h * myODE(ti + h, zi + k3)';
        % Update time and state
        t(i + 1) = ti + h;
        z(i + 1, :) = zi + (k1 + 2*k2 + 2*k3 + k4)/6;
    end
end
```

```
• % Solve the second differential equation: mz'' + cz' + k1z = m(ω^2)Ysin(ωt)
• function dzdt2 = myODE4(t, z)
•     % Define the parameters for the second equation
•     m2 = 85; % Mass
•     c2 = 250; % Damping coefficient
•     k2 = 25000; % Spring constant
•     omega2 = 3.14; % Frequency
•     Y2 = 1; % Amplitude
•
•     % Define the second differential equation
•     dzdt2 = [z(2); (m2 * (omega2^2) * Y2 * sin(omega2 * t) - c2 * z(2) - k2 * z(1)) /
m2];
• end
```

```matlab
% Define time span and initial conditions
tspan = [0, 10];
initial_conditions = [0, 0]; % [z_initial, z'_initial]
h = 0.01; % Step size
% Call the Runge-Kutta function for the first ODE
[t1, z1] = R2(@myODE3, tspan, initial_conditions, h);

% Call the Runge-Kutta function for the second ODE
[t2, z2] = R3(@myODE4, tspan, initial_conditions, h);
% Plot the results for both ODEs in one plot
figure;
plot(t1, z1(:, 1), 'b-', 'LineWidth', 2);
hold on;
plot(t2, z2(:, 1), 'r-', 'LineWidth', 2);
hold off;
```

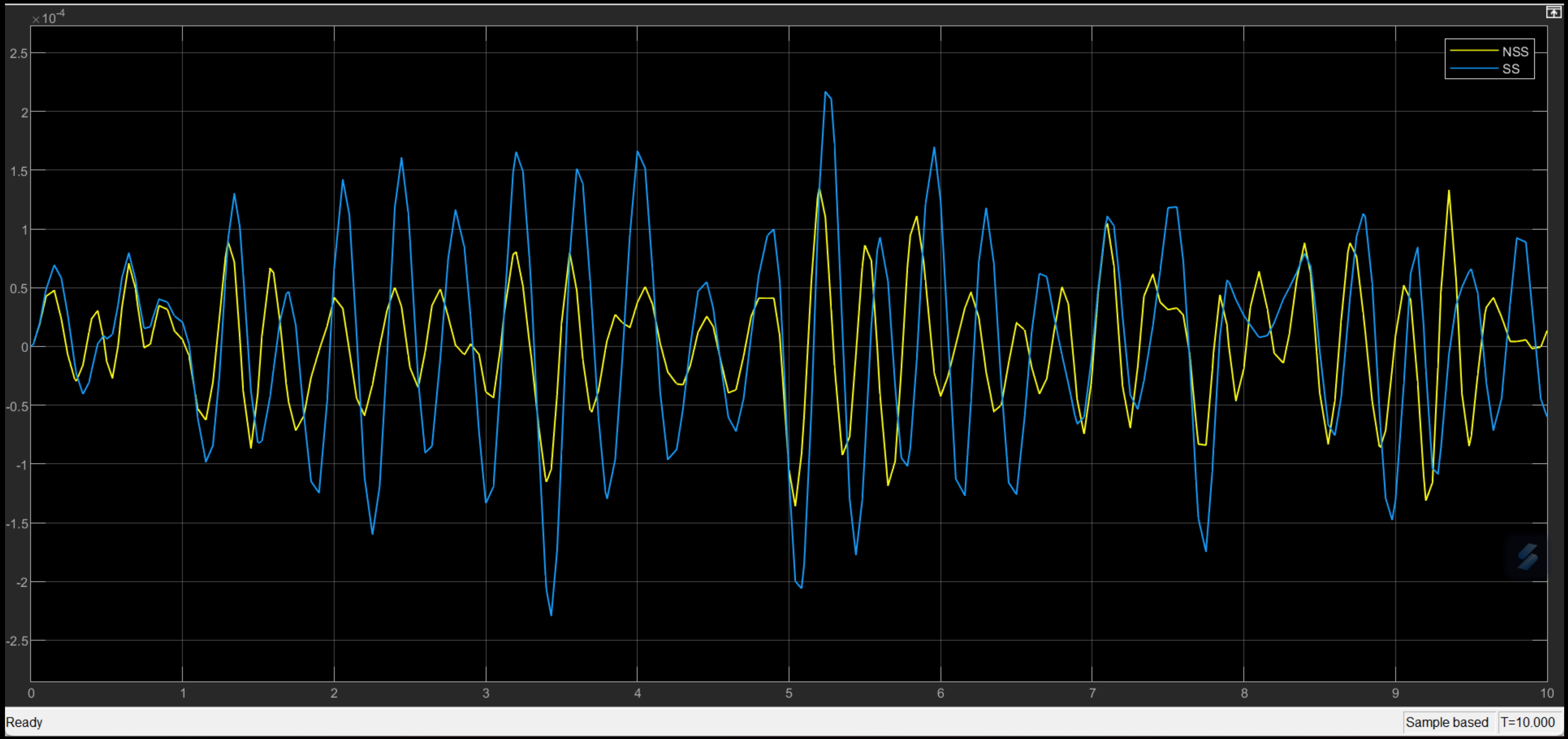
```matlab
xlabel('Time');
ylabel('Displacement');
title('Comparison of Two ODEs using Runge-Kutta 4th Order Method');
legend('show');
grid on;

% Calculate root mean square displacement (RMSD)
T = t(end);  % Total time span
rmsd = sqrt(trapz(t, z(:, 1).^2) / T);

% Display the calculated RMSD
disp(['Root Mean Square Displacement (RMSD): ', num2str(rmsd)]);
```

S I M U L I N K M O D E L :



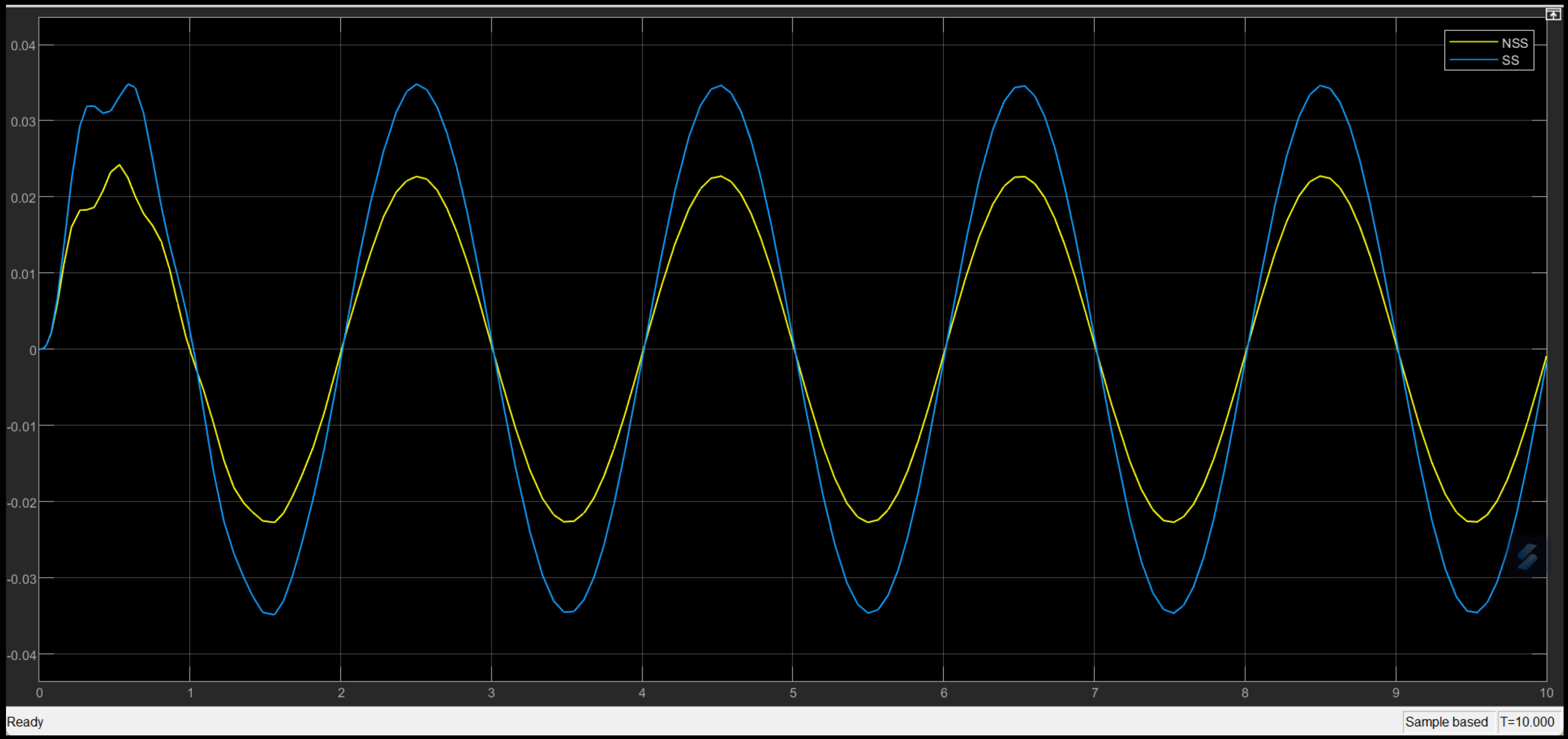
Results:

Random Excitation for both NSS an SS using Simulink



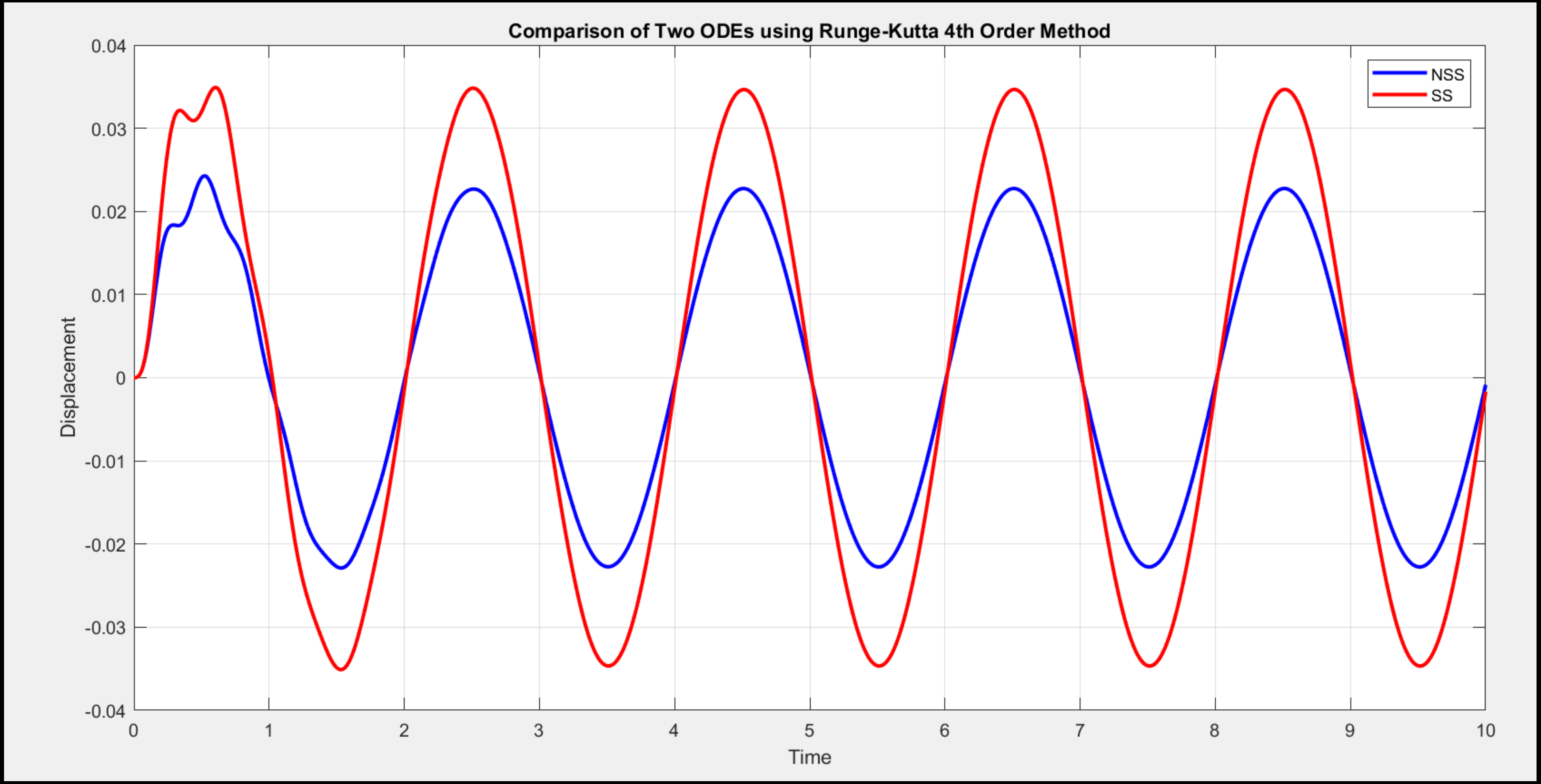
Results:

Sinusoidal input for both NSS and SS system



Results:

Using MATLAB code and sinusoidal excitation as input



Conclusion

Root Mean Square Displacement (RMSD)-NSS: **0.016081**

Root Mean Square Displacement (RMSD)-SS: **0.024531**

The calculation results of the (Zws) shows that by using NSS there is reduction of **52.546%** comparing to the SS system.

The results show that the driver's seat ride comfort and isolation efficiency of the seat’s NSS are better than that of SS

References

- ISO 2631-1. Mechanical vibration and shock – evaluation of human exposure to whole body vibration– Part 1: General requirements,” Geneve, Switzerland, 1997. Workers exposed to whole-body vibration (WBV) can be at increased risk for musculoskeletal disorders including low back problems, neck problems, and muscle fatigue. The ISO 2631-1 (1997) is a widely accepted standard for WBV assessment and provides guidelines on how to properly measure and interpret WBV exposure in relation to human health and comfort
- A new design of seat suspension using different models of negative stiffness structure
- Negative stiffness devices for vibration isolation applications: A review   Huan Li, Yancheng Li, and Jianchun Li

THANK YOU!