Bangladesh University of Engineering and Technology

Department of Electrical and Electronic Engineering

Course No. EEE 212

Course Title: Numerical Techniques Laboratory

A project on

"Spring Mass System"

Developed by-

1. Kafi Anan ID.2006040
2. Md. Zobayer Nabil ID.2006041

Course Teachers:

1. Bejoy Sikder, Lecturer, EEE, BUET
2. Md. Meftahul Haque, Teaching Assistant, EEE, BUET

Problem Statement:

Numerically calculate and simulate the position and velocity of a mass attached to a Single and Two Degree of Freedom Spring-Mass system. **All the mathematical formulations and derivations has been attached in the presentation slides and hence has been omitted from the report to avoid redundancy. For mathematical background, you are requested to go through the slides.**

Objective:

In this project, the inputs are

- Mass of the object, m
- Spring constant, k
- Exciting function F(t) (amplitude and frequency/angular frequency)
- Damping constant, c
- Initial position and velocity as initial condition

The outputs are

- Graph of position (respect to time)
- Graph of Velocity (respect to time)

Achievement:

After simulating the project, we are able to study

- Free vibration of one and two degrees of freedom spring mass system in both damped and no damped condition

– Forced vibration of one degree of freedom spring mass system in both damped (under damped, over damped, and critically damped) and no damped condition

Project Code:

❖ Free Vibration of one degree of freedom spring mass system with no damp condition

Code used for solving the Differential Equation:

```
clc;
clear all;
close all;
global m k
mass = "Mass of body: ";
m = input(mass);
sprnConst = "Spring constant: ";
k = input(sprnConst);

dt = .005;
t = 0:dt:5;
c1 = "Initial velocity: ";
vel0 = input(c1);
c2 = "Iniitial displacement: ";
disp0 = input(c2);
y0 = [vel0 disp0]; %[vel disp]
[tsol,ysol] = ode45('odetest1',[0:dt:5],y0);
plot(t,ysol(:,2));
xlabel('Displacement')
ylabel('metres')
figure
plot(t,ysol(:,1));
xlabel('Velocity')
ylabel('metre/sec')
T = 2*pi*sqrt(m/k)
figure
```

Code used for the animation of the motion of the body attached to the spring:

```
% creates mass plot
mass = plot(ysol(1,2), 0, 'bo', 'MarkerSize', 30, 'MarkerFaceColor', 'b');

% sets axis limits
axis([min(ysol(:,2))-0.1 max(ysol(:,2))+0.1 -0.1 0.1]);

% loops through time steps and update position of mass
time = t;
for i = 1:length(time)
    set(mass, 'XData', ysol(i,2));
    pause(dt);
end
```

Here, the mass plot is initialized at ysol(1,2) which is the initial position of the objected connected to the spring. The axis limits are set using the 'axis' function and the limits of x-axis is set using the 'max' and 'min' functions, otherwise the animation would not be in a perfect range.

The function used:

```
function dy = odetest1(t,y)
global m k
dy = [-k*y(2)/m;y(1)];

```

❖ Forced Vibration of one degree of freedom spring mass system under undamped condition

Code to solve the Differential Equation:

```
clc;
clear all;
close all;

global m k w_f F0

mass = "Mass of body: ";
m = input(mass);
sprnConst = "Spring constant: ";
k = input(sprnConst);
for_w = "Angular frequency of exciting force: ";
w_f = input(for_w);
F_amp = "Amplitude of exciting force: ";
F0 = input(F_amp);

dt = .005;
t = 0:dt:3;
c1 = "Initial velocity: ";
vel0 = input(c1);
c2 = "Iniitial displacement: ";
disp0 = input(c2);
y0 = [vel0 disp0]; %[vel disp]
[tsol,ysol] = ode45('odetest2',0:dt:3,y0);
plot(t,ysol(:,2));
xlabel('Displacement')
ylabel('metres')
figure
plot(t,ysol(:,1));
xlabel('Velocity')
ylabel('metre/sec')
T = 2*pi*sqrt(m/k)
figure
```

Code used for the animation of the motion of the body attached to the spring:

```
% creates mass plot
mass = plot(ysol(1,2), 0, 'bo', 'MarkerSize', 30, 'MarkerFaceColor', 'b');

% sets axis limits
axis([min(ysol(:,2))-0.1 max(ysol(:,2))+0.1 -0.1 0.1]);

% loops through time steps and update position of mass
time = t;
for i = 1:length(time)
    set(mass, 'XData', ysol(i,2));
    pause(dt);
end
```

The function used:

```
function dy = odetest2(t,y)
global m k w_f F0

f1 = F0*sin(2*pi*w_f*t);
dy(1) = (f1 - k*y(2))/m;
dy(2) = y(1);
dy=dy';
```

❖ Free Vibration of one degree of freedom spring mass system with damped condition

The code used to solve the Differential Equation:

```
clc;
clear all;
close all;

global m k c
mass = "Mass of body: ";
m = input(mass);
sprnConst = "Spring constant: ";
k = input(sprnConst);
dampC = "Damping constant: ";
c = input(dampC);

dt = .005;
t = 0:dt:2;
c1 = "Initial velocity: ";
vel0 = input(c1);
c2 = "Iniitial displacement: ";
disp0 = input(c2);
y0 = [vel0 disp0]; %[vel disp]
[tsol,ysol] = ode45('odetest3',0:dt:2,y0);
plot(t,ysol(:,2));
grid on;
xlabel('Displacement')
ylabel('metres')
figure
plot(t,ysol(:,1));
grid on;
xlabel('Velocity')
ylabel('metre/sec')
T = 2*pi*sqrt(m/k)
Damping_ratio = c/(2*sqrt(m*k))
```

Code used for the animation of the motion of the body attached to the spring:

```
% create mass plot
mass = plot(ysol(1,2), 0, 'bo', 'MarkerSize', 30, 'MarkerFaceColor', 'b');

% set axis limits
axis([min(ysol(:,2))-0.1 max(ysol(:,2))+0.1 -0.1 0.1]);

% loop through time steps and update position of mass
time = t;
for i = 1:length(time)
    set(mass, 'XData', ysol(i,2));
    pause(dt);
end
```

Function used:

```
function dy = odetest3(t,y)
global m k c

dy = [-k*y(2)/m-c*y(1)/m ; y(1)];
```

❖ Forced Vibration of one degree of freedom spring mass system with damped condition

Code to solve the Differential Equation:

```
clc;
clear all;
close all;

global m k c w_f F0

mass = "Mass of body: ";
m = input(mass);
sprnConst = "Spring constant: ";
k = input(sprnConst);
dampC = "Damping constant: ";
c = input(dampC);
for_w = "Angular frequency of exciting force: ";
w_f = input(for_w);
F_amp = "Amplitude of exciting force: ";
F0 = input(F_amp);

dt = .005;
t = 0:dt:3;
c1 = "Initial velocity: ";
vel0 = input(c1);
c2 = "Iniitial displacement: ";
disp0 = input(c2);
y0 = [vel0 disp0]; %[vel disp]
[tsol,ysol] = ode45('odetest4',0:dt:3,y0);
plot(t,ysol(:,2));
grid on;
xlabel('Displacement')
ylabel('metres')
figure
plot(t,ysol(:,1));
grid on;
xlabel('Velocity')
ylabel('metre/sec')
T = 2*pi*sqrt(m/k)
figure;
```

Code used for the animation of the motion of the body attached to the spring:

```
% create mass plot
mass = plot(ysol(1,2), 0, 'bo', 'MarkerSize', 30, 'MarkerFaceColor', 'b');

% set axis limits
axis([min(ysol(:,2))-0.1 max(ysol(:,2))+0.1 -0.1 0.1]);

% loop through time steps and update position of mass
time = t;
for i = 1:length(time)
    set(mass, 'XData', ysol(i,2));
    pause(dt);
end
```

Function used:

```
function dy = odetest4(t,y)
global m k c w_f F0
f1 = F0*sin(2*pi*w_f*t);

dy = [f1/m - k*y(2)/m - c*y(1)/m; y(1)];
```

❖ Free Vibration of two degree of freedom spring mass system with no damp condition

Code to solve the system of Differential Equations:

```
clc;
clear all;
close all;

m1 = input('Mass of first object: ');
m2 = input('Mass of second object: ');
M = [m1 0;0 m2];
k1 = input('First spring constant: ');
k2 = input('Second spring constant: ');
k3 = input('Third spring constant: ');
K = [k1+k2 -k2;-k2 k2+k3];
[modeShape freq] = eig(K,M);

A00 = zeros(2);
A10 = eye(2);
CC = [A00 A10; -inv(M)*K A00];
global CC;
max_freq = max(sqrt(diag(freq))/(2*pi));
dt = 1/(max_freq*20);
time = 0:dt:500*dt;

c1 = input('Initial displacement of first body: ');
c2 = input('Initial displacement of second body: ');
c3 = input('Initial velocity of first body: ');
c4 = input('Initial velocity of second body: ');

y0 = [c1 c2 c3 c4];
[tsol,ysol] = ode23('odetest2DOF1',time,y0);
plot(time,ysol(:,1:2),'linewidth',1);
xlabel('Time(s)');
ylabel('Displacement(m)')
grid on
figure
plot(time,ysol(:,3:4),'linewidth',1);
xlabel('Time(t)');
ylabel('Velocity(m/s)');
grid on
```

Code used for the animation of the motion of the body attached to the spring:

```
figure
axis([min(ysol(:,2))-1 max(ysol(:,2))+1 -0.1 0.1]);
grid on
hold on
mass1 = plot(ysol(1,1), 0, 'bo', 'MarkerSize', 30, 'MarkerFaceColor', 'b');
mass2 = plot(ysol(1,2), 0, 'ro', 'MarkerSize', 20, 'MarkerFaceColor', 'r');

for i = 1:length(time)
    set(mass1, 'XData', ysol(i,1), 'YData', 0);
    set(mass2, 'XData', ysol(i,2)+.5, 'YData', 0);
    pause(dt);
end
```

As there are two bodies in the spring-mass system, we need to initialize the position of both the bodies after creating the mass blocks. Then, in the for loop, through iteration, the position of both the blocks are updated and is viewed through animation by the 'set' function.

Function used:

```
function dy = odetest2DOF1(t,y)
global CC;
dy = CC*y;
```

❖ Free Vibration of two degree of freedom spring mass system with damped condition

The code to solve the system of Differential Equations:

```
clc;
clear all;
close all;

m1 = input('Mass of first object: ');
m2 = input('Mass of second object: ');
M = [m1 0;0 m2];
k1 = input('First spring constant: ');
k2 = input('Second spring constant: ');
k3 = input('Third spring constant: ');
c1 = input('Viscous damping for first portion: ');
c2 = input('Viscous damping for second portion: ');
c3 = input('Viscous damping for third portion: ');
C = [c1+c2 -c2;-c2 c2+c3];
K = [k1+k2 -k2;-k2 k2+k3];
[modeShape freq] = eig(K,M);

A00 = zeros(2);
A10 = eye(2);
CC = [A00 A10; -inv(M)*K -inv(M)*C];
global CC;
max_freq = max(sqrt(diag(freq))/(2*pi));
dt = 1/(max_freq*20);
time = 0:dt:200*dt;

c1 = input('Initial displacement of first body: ');
c2 = input('Initial displacement of second body: ');
c3 = input('Initial velocity of first body: ');
c4 = input('Initial velocity of second body: ');

y0 = [c1 c2 c3 c4];
[tsol,ysol] = ode23('odetest2DOF1',time,y0);
plot(time,ysol(:,1:2),'linewidth',1);
xlabel('Time(s)');
ylabel('Displacement(m)')
grid on
figure
```

```
plot(time,ysol(:,3:4),'linewidth',1);
xlabel('Time(t)');
ylabel('Velocity(m/s)');
grid on

figure
axis([min(ysol(:,2))-1 max(ysol(:,2))+1 -0.1 0.1]);
grid on
hold on
mass1 = plot(ysol(1,1), 0, 'bo', 'MarkerSize', 30, 'MarkerFaceColor', 'b');
mass2 = plot(ysol(1,2), 0, 'ro', 'MarkerSize', 20, 'MarkerFaceColor', 'r');

% Loop through the time steps and update the position of the masses
for i = 1:length(time)
    set(mass1, 'XData', ysol(i,1), 'YData', 0);
    set(mass2, 'XData', ysol(i,2)+.5, 'YData', 0);
    pause(dt);
end
```

The same process has been followed here too to animate both the objects.

Function used:

```
function dy = odetest2DOF2(t,y)
global CC;
dy = CC*y;

```

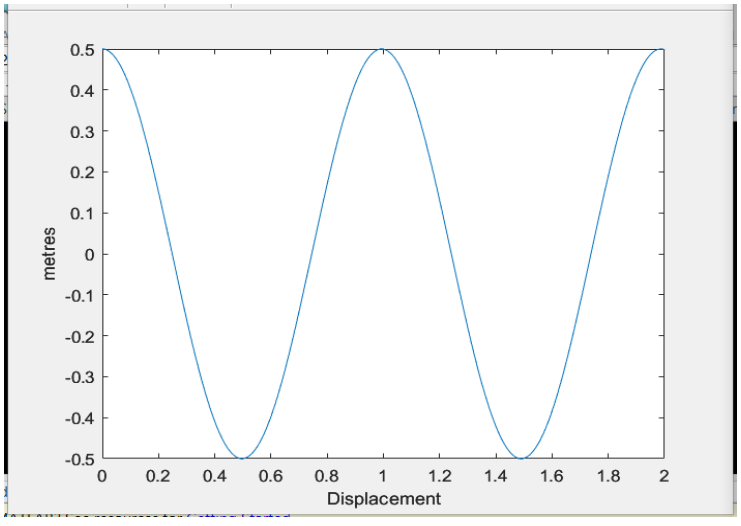
Project Output:

- ♦ Free Vibration of one degree of freedom spring mass system with no damp condition

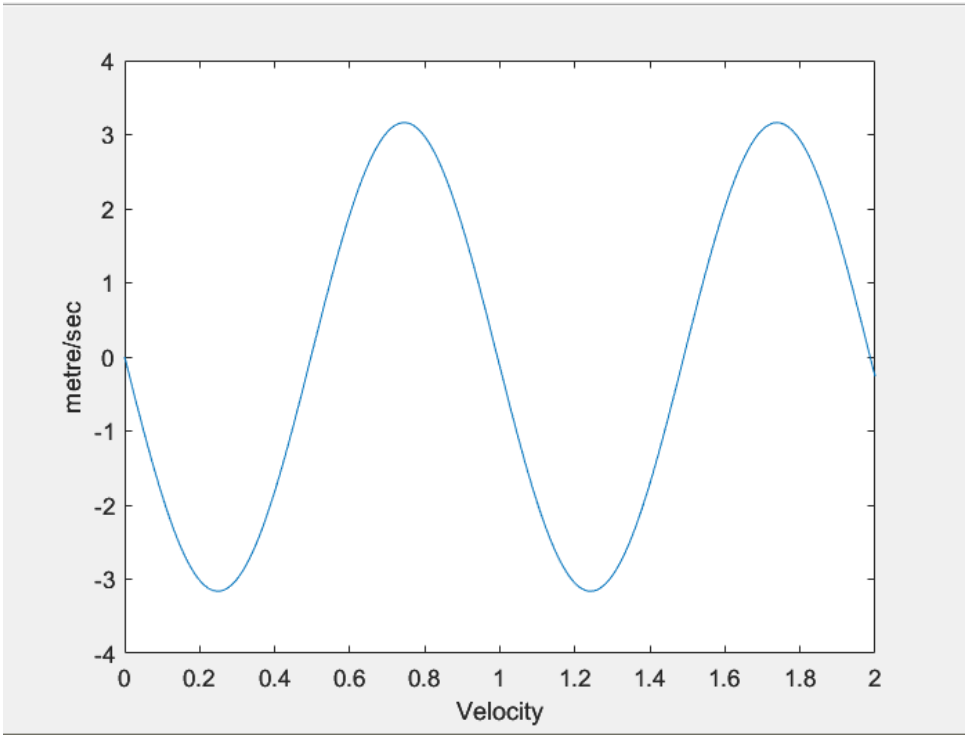
Input

```
Mass of body: 5
Spring constant: 200
Initial velocity: 0
Iniitial displacement: 0.5

T =

    0.9935
```

Displacement



Velocity

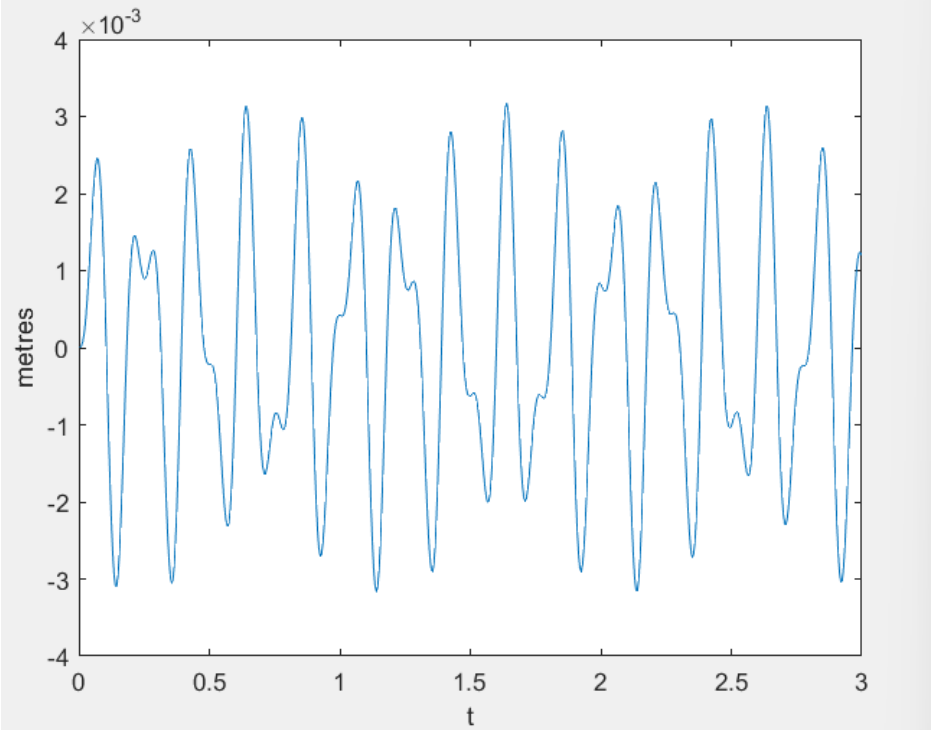


- **Forced Vibration of one degree of freedom spring mass system with no damp condition**

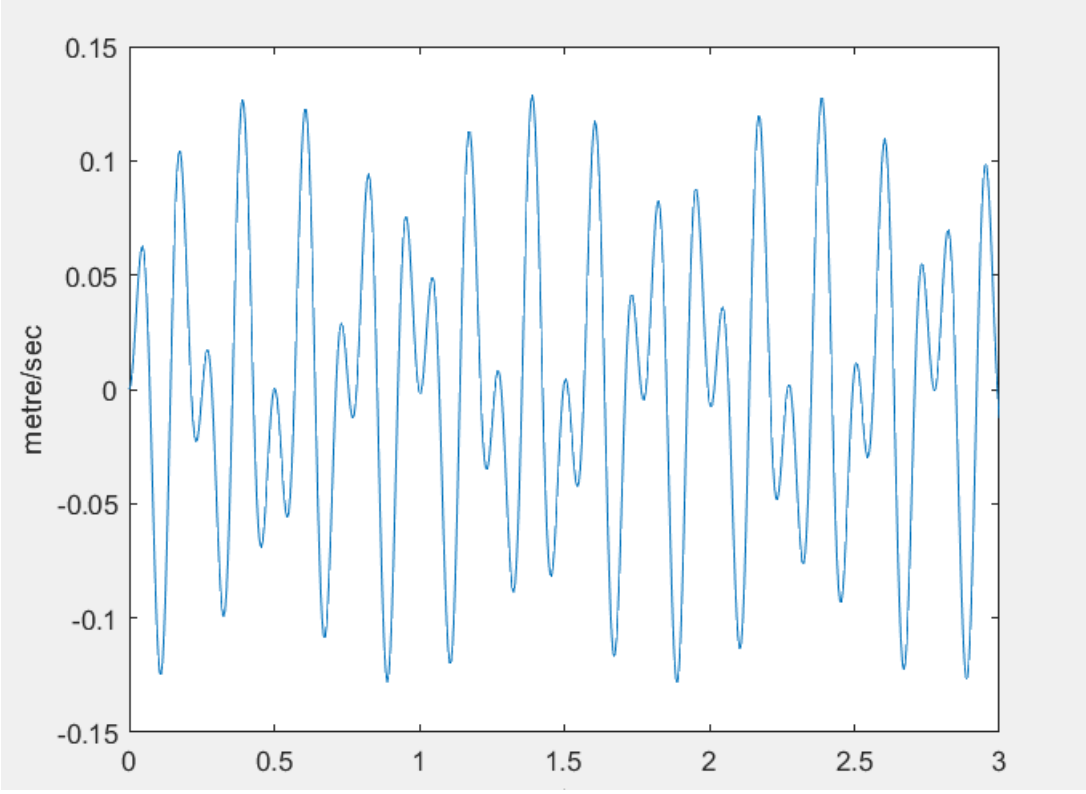
Input

```
Mass of body: 2
Spring constant: 2000
Angular frequency of exciting force: 9
Amplitude of exciting force: 5
Initial velocity: 0
Iniitial displacement: 0
```

Displacement



Velocity



The output graph is expected to be like this because it should be a superposition of both natural and forced frequency.

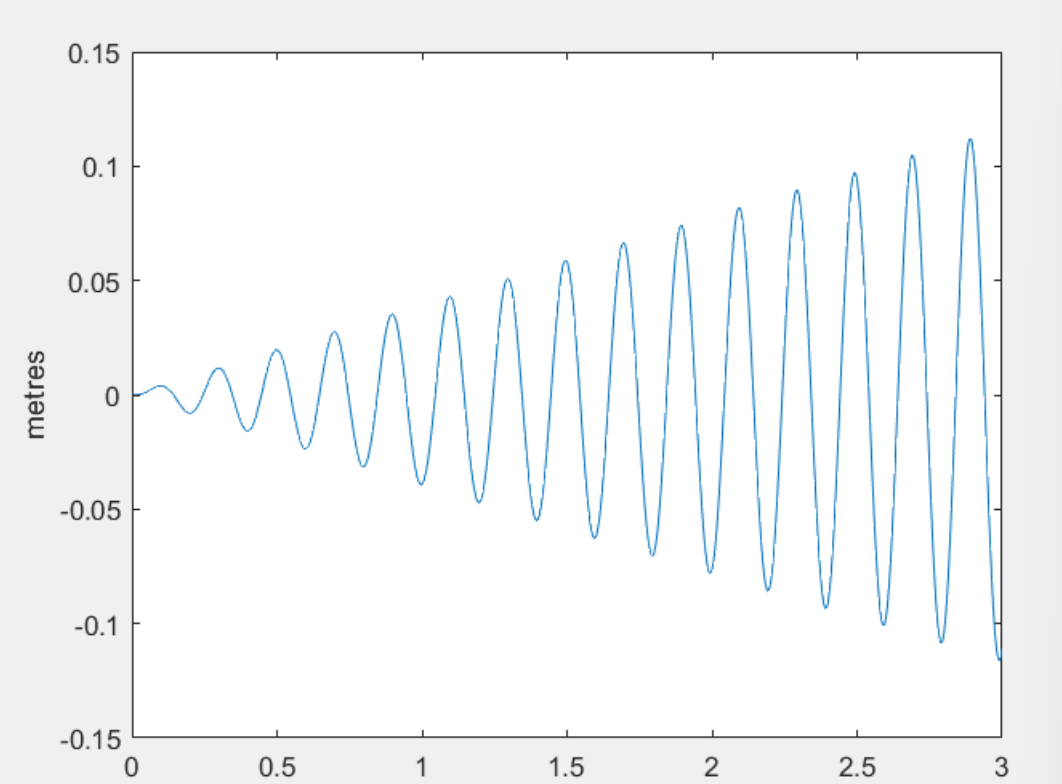
And as we know, if the forced frequency and the natural frequency becomes almost equal, then RESONANCE occurs , that is, the Amplitude of the resultant wave keeps increasing with time.

Input Prompt

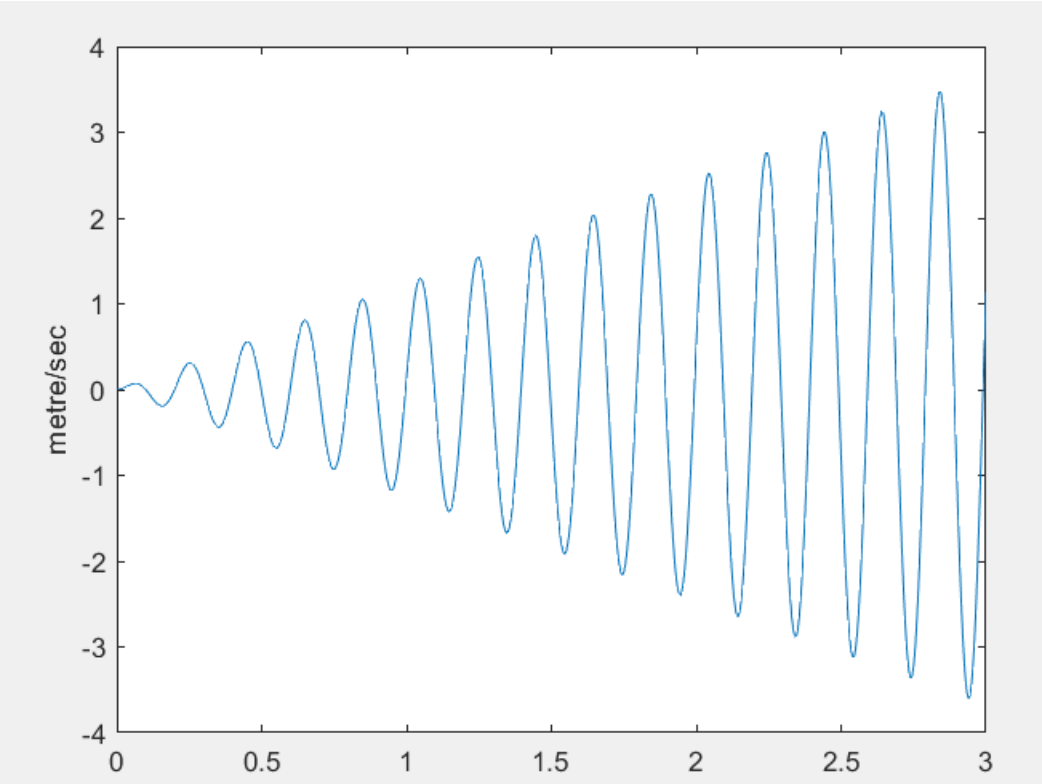
```
Mass of body: 2
Spring constant: 2000
Angular frequency of exciting force: 5
Amplitude of exciting force: 5
Initial velocity: 0
Iniitial displacement: 0
```

As we can see, the natural frequency in this case is around 5Hz and the input Forced frequency is also set to 5hz. So we will see the condition of Resonance

Displacement in case of Resonance



Velocity in case of Resonance:



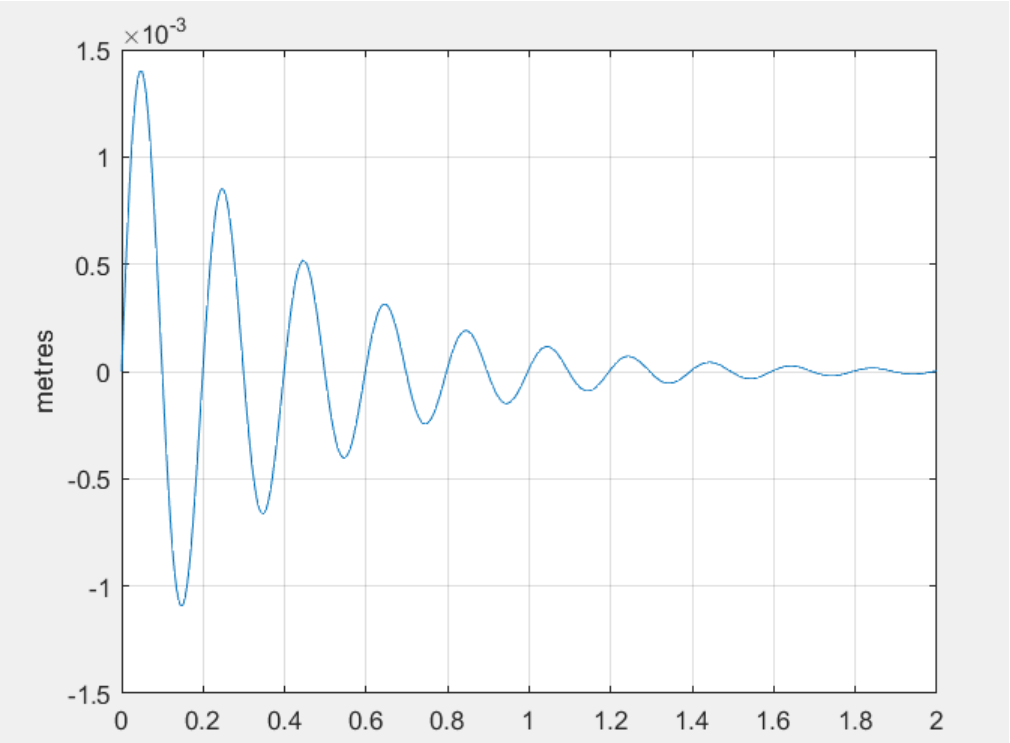
- **Free Vibration of one degree of freedom spring mass system with damped condition**

The input is given such that the output is obtained in UNDER-DAMPED condition:

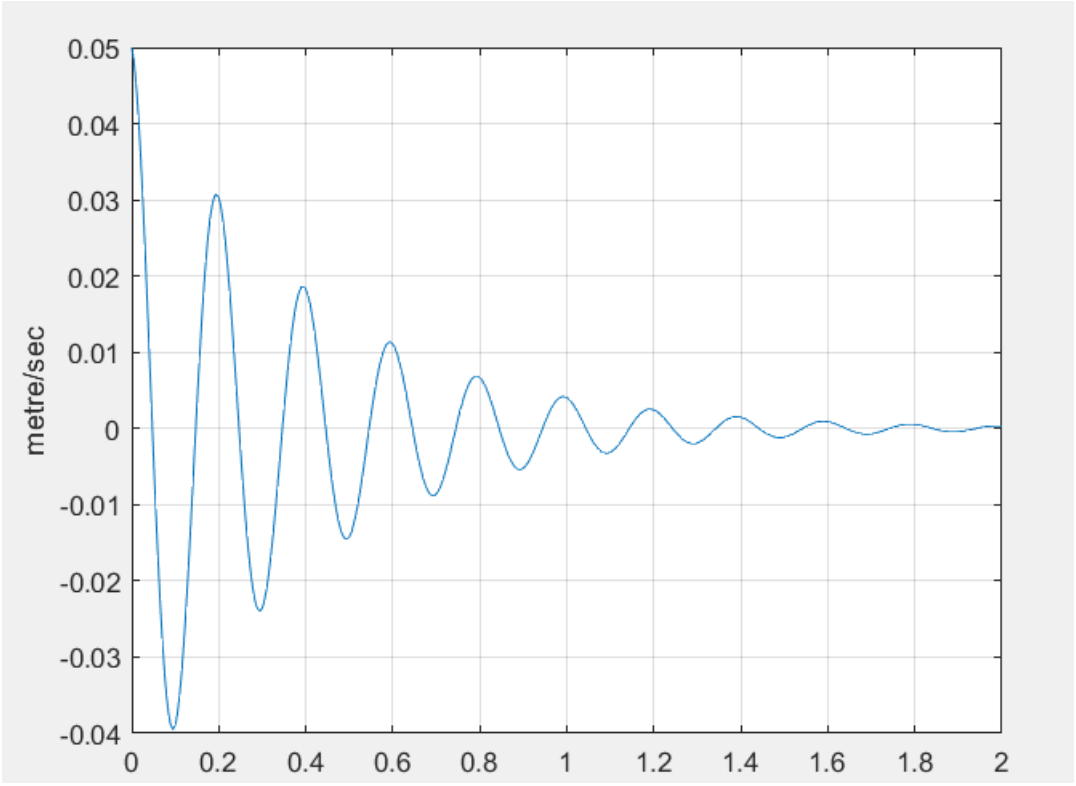
Input

```
Mass of body: 2
Spring constant: 2000
Damping constant: 10
Initial velocity: .05
Iniitial displacement: 0

T =

    0.1987


Damping_ratio =

    0.0791
```

Displacement



Velocity

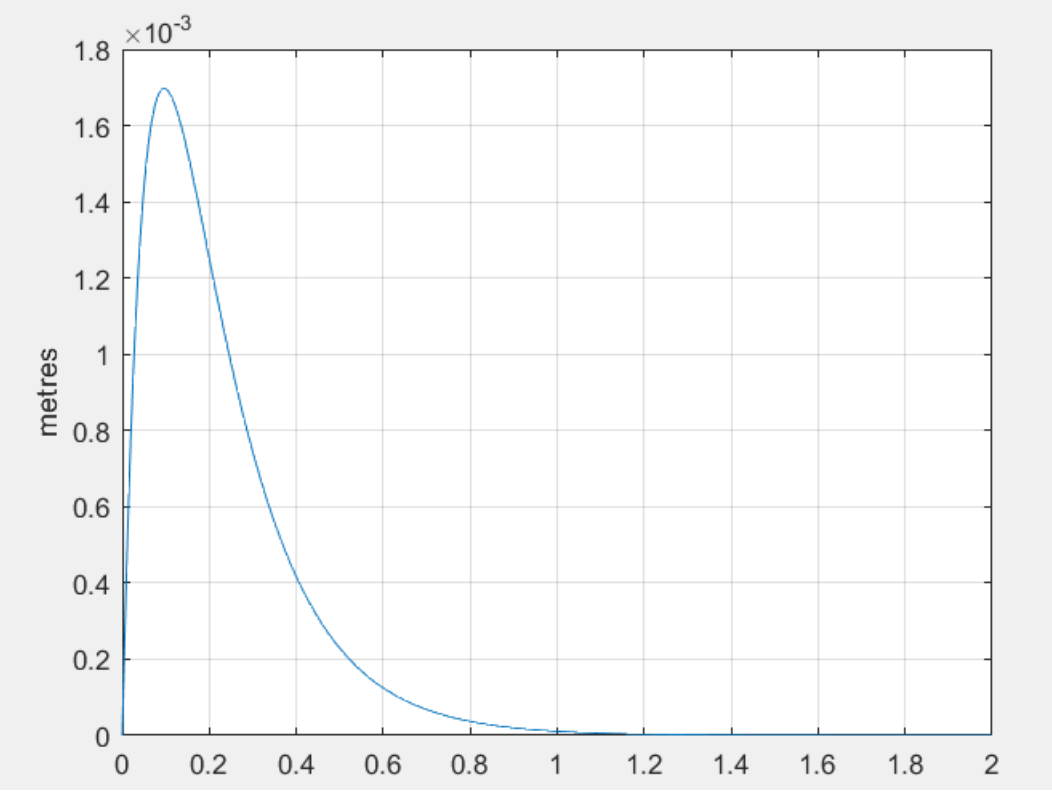


The input is set such that the output is obtained in OVER-DAMPED condition:

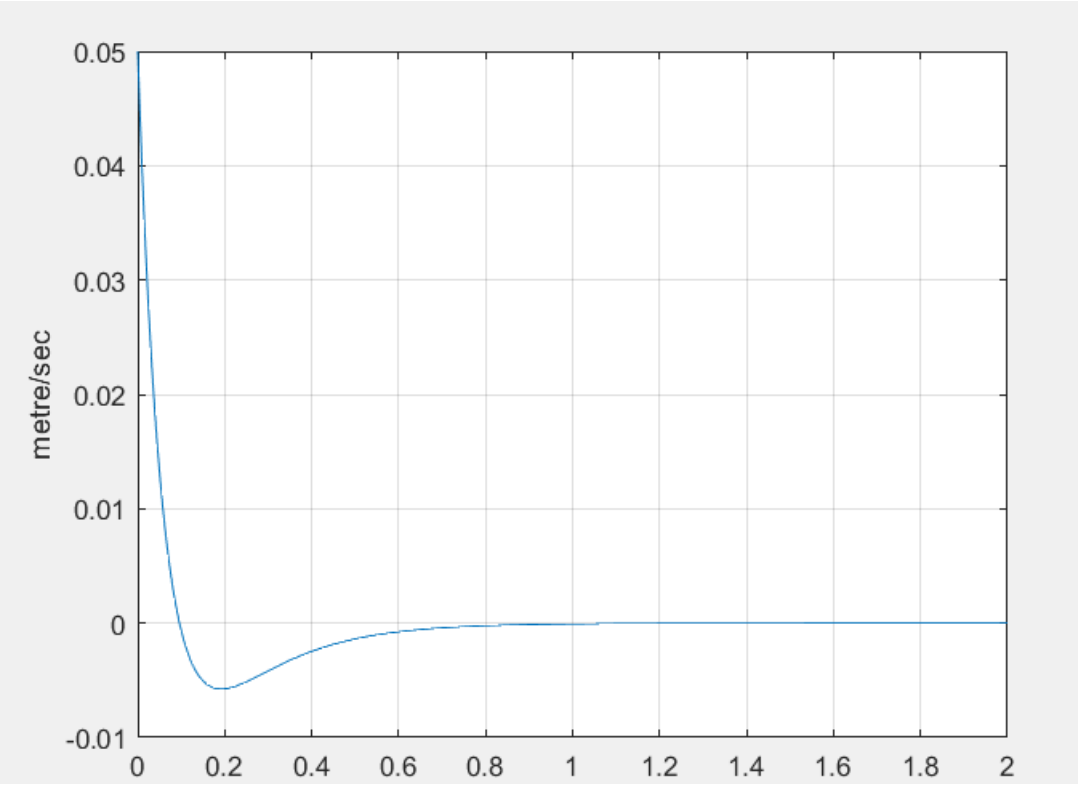
Input

```
Mass of body: 2
Spring constant: 200
Damping constant: 45
Initial velocity: .05
Iniitial displacement: 0

T =

    0.6283


Damping_ratio =

    1.1250
```

Displacement



Velocity



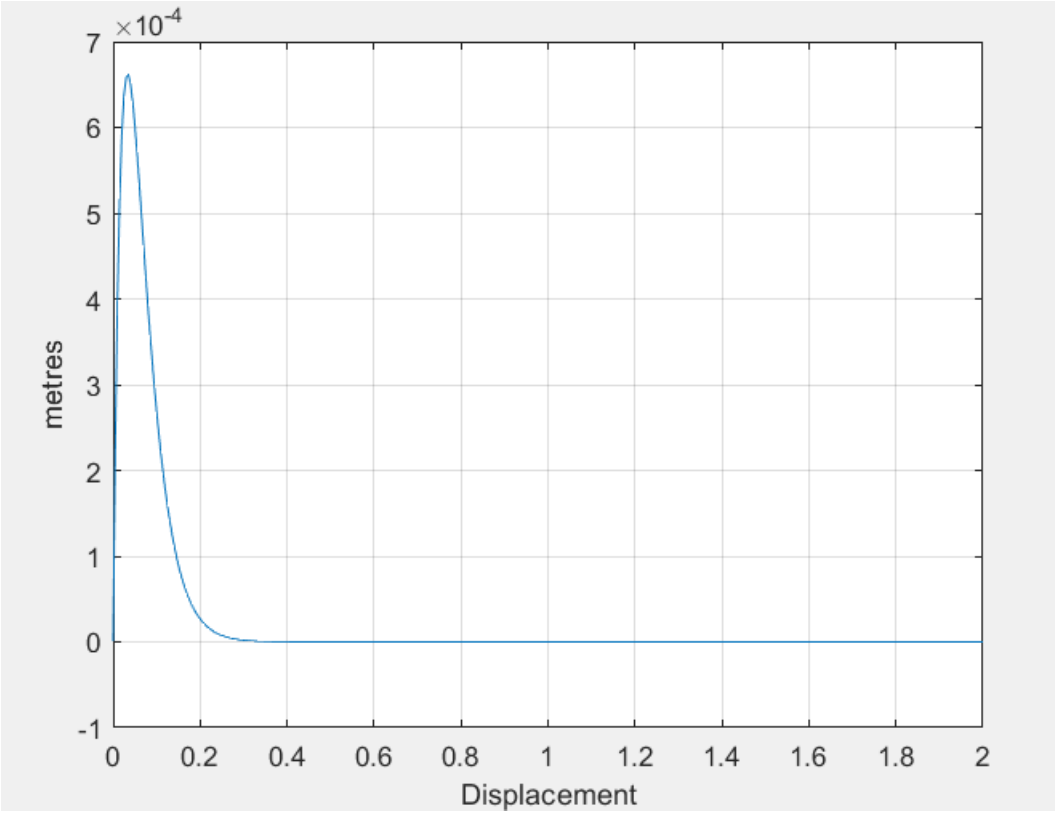
The input is set such that the output is obtained in CRITICALLY-DAMPED condition

Input

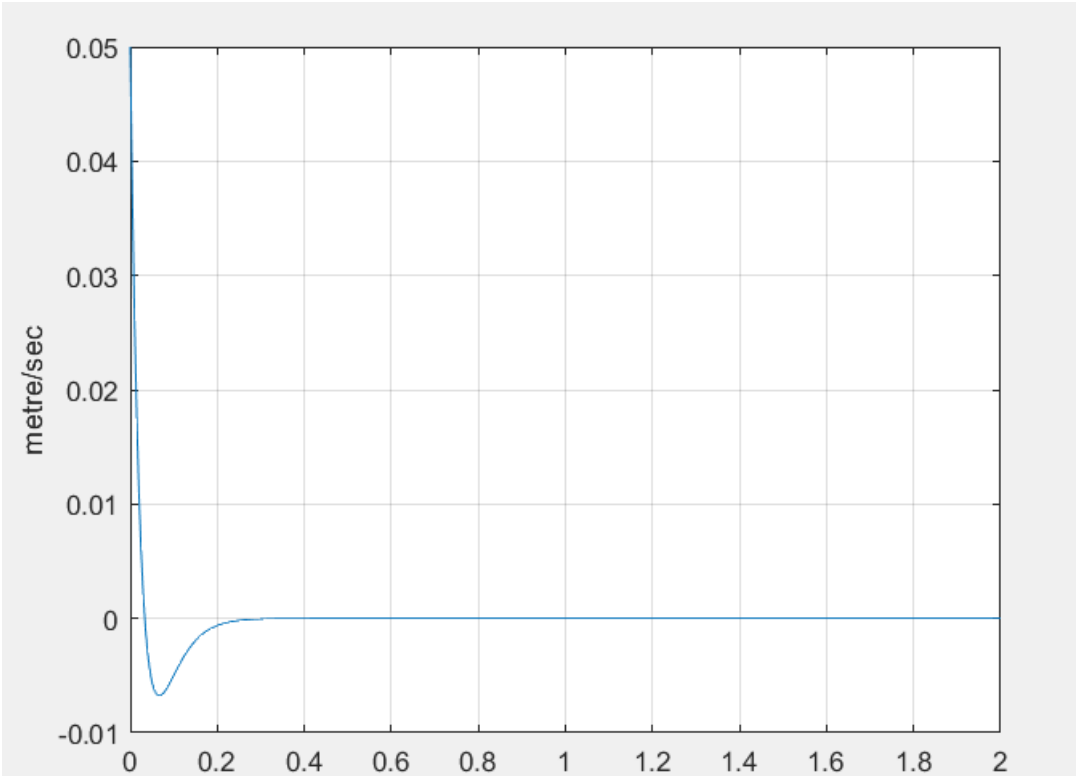
```
Mass of body: 1
Spring constant: 900
Damping constant: 60
Initial velocity: .05
Iniitial displacement: 0

T =

    0.2094


Damping_ratio =

     1
```

As we can see the damping ratio is 1. So it is in critically damped condition

Displacement



Velocity

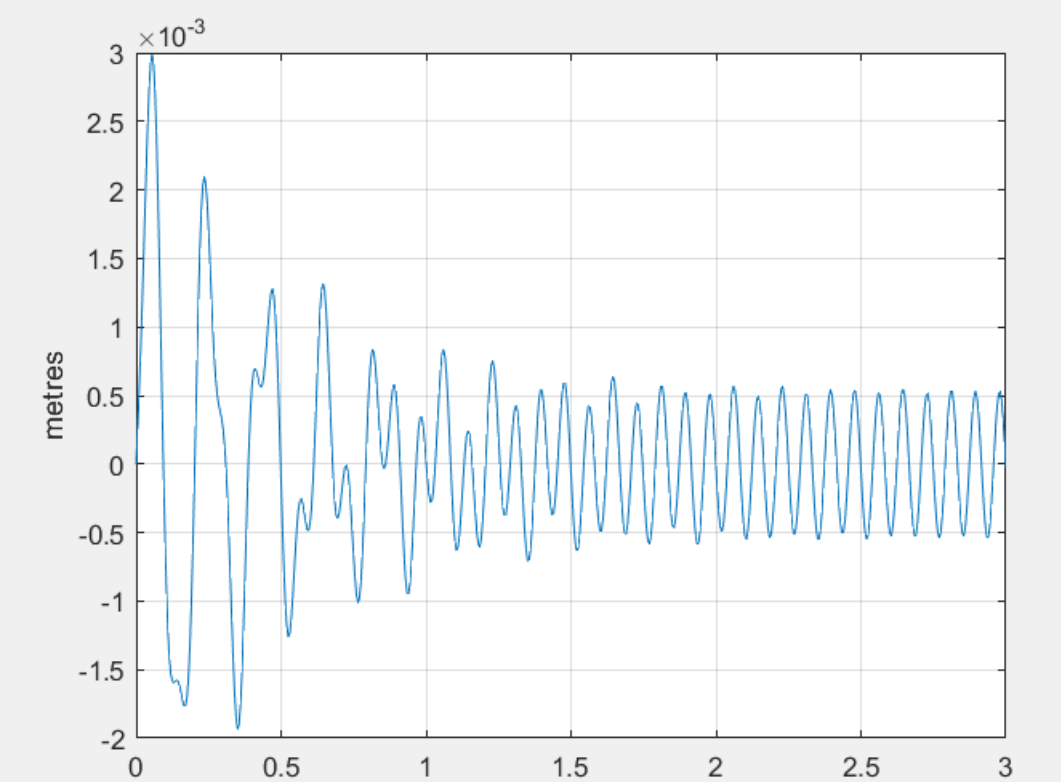


- **Forced Vibration of one degree of freedom spring mass system with damped condition**

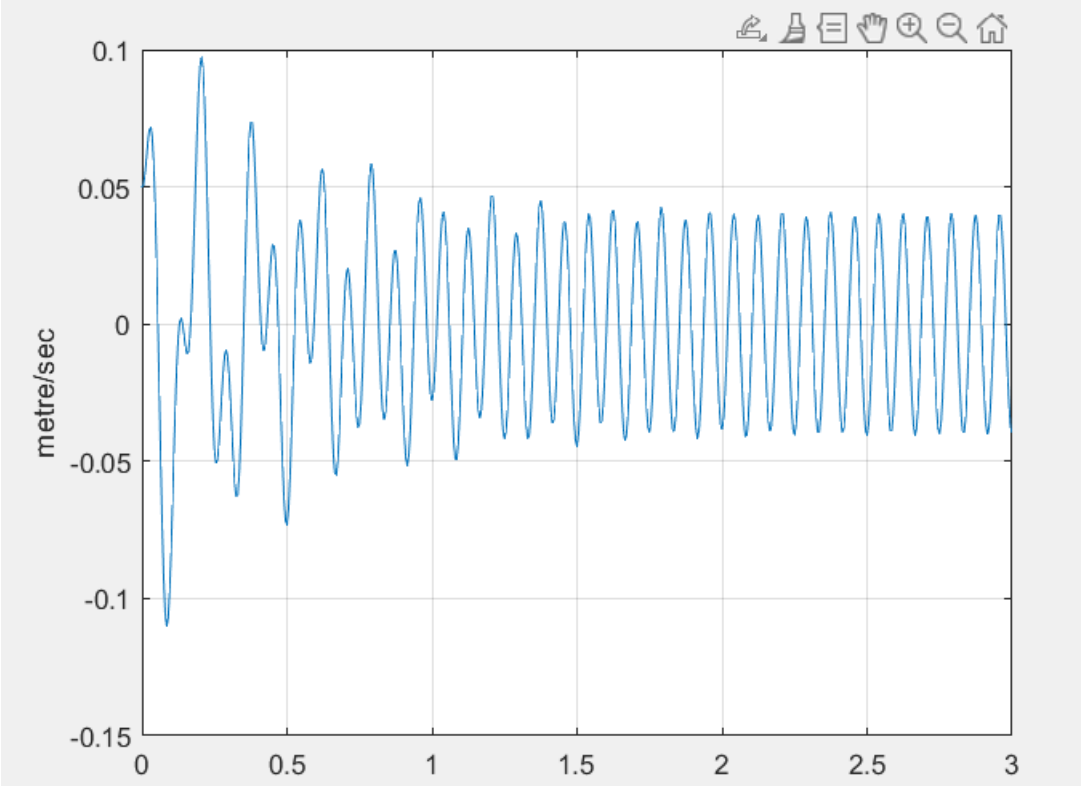
Input

```
Mass of body: 2
Spring constant: 2000
Damping constant: 8
Angular frequency of exciting force: 12
Amplitude of exciting force: 5
Initial velocity: .05
Iniitial displacement: 0

T =

    0.1987
```

Displacement



Velocity



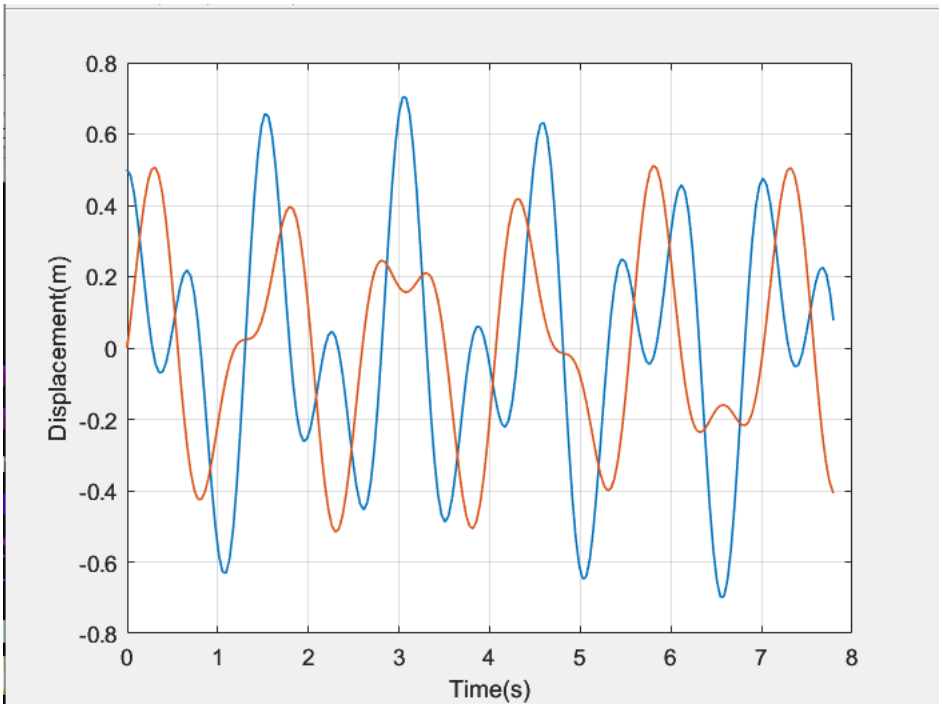
As we can see, the first portion of the graph is unstable. This is actually the **TRANSIENT** part because it is in damped condition, the transient part slowly dies out. And the emergence of the transient part was due to the applied force. The later part of the wave is the **STEADY STATE.**

♦ Free Vibration of two degree of freedom spring mass system with no damp condition

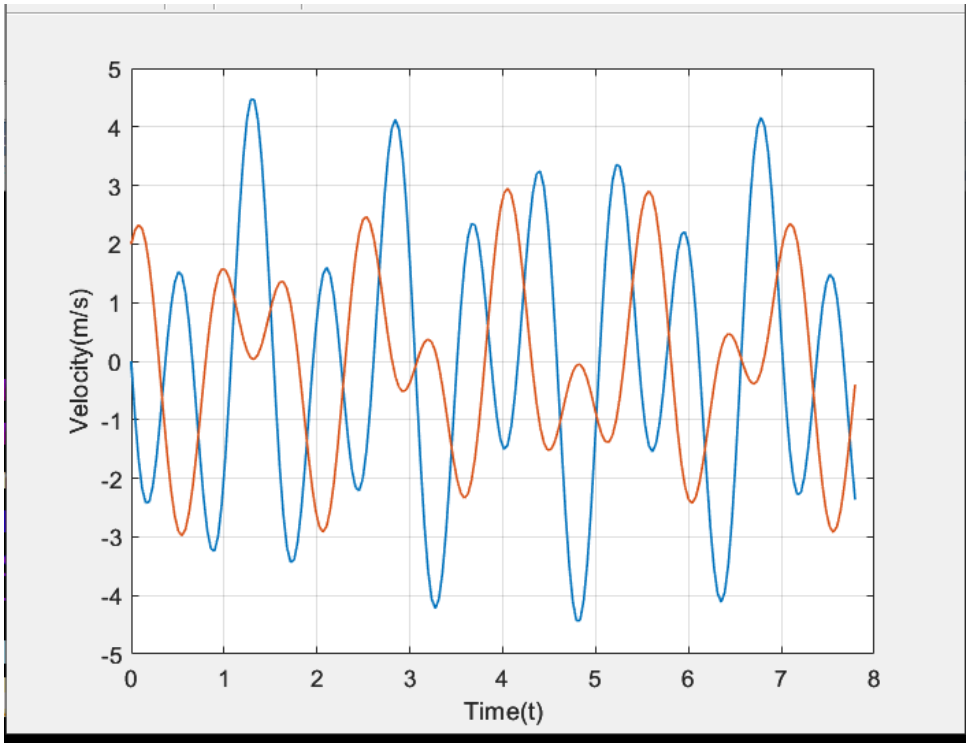
Input

```
Mass of first object: 10
Mass of second object: 20
First spring constant: 200
Second spring constant: 300
Third spring constant: 400
Warning: The value of local variables may have been changed to match the gl
global before you use that variable.
> In TDOFfreeNOdamp (line 17)
Initial displacement of first body: 0.5
Initial displacement of second body: 0
Initial velocity of first body: 0
Initial velocity of second body: 2
>>
```

Displacement



Velocity

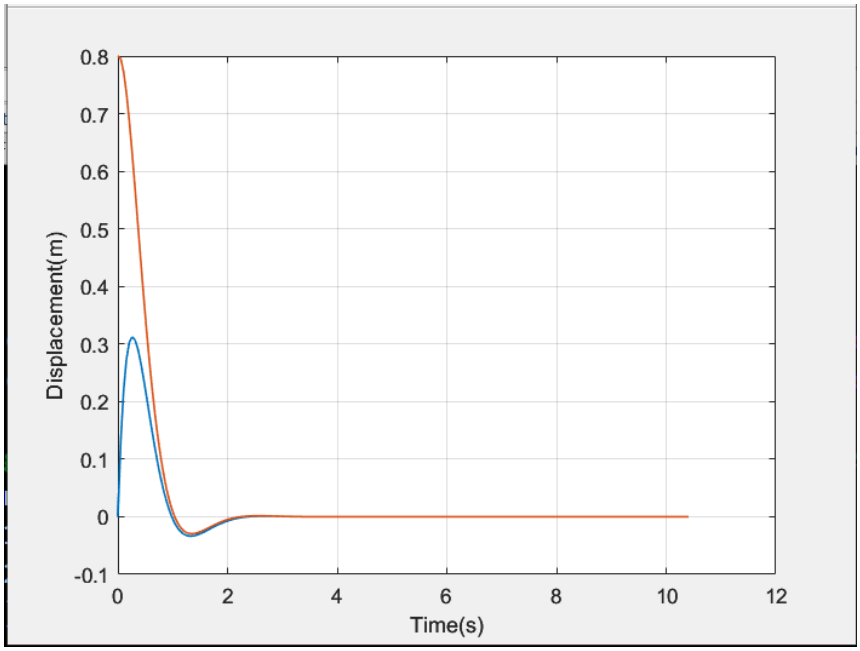


- ♦ Free Vibration of two degree of freedom spring mass system with damped condition

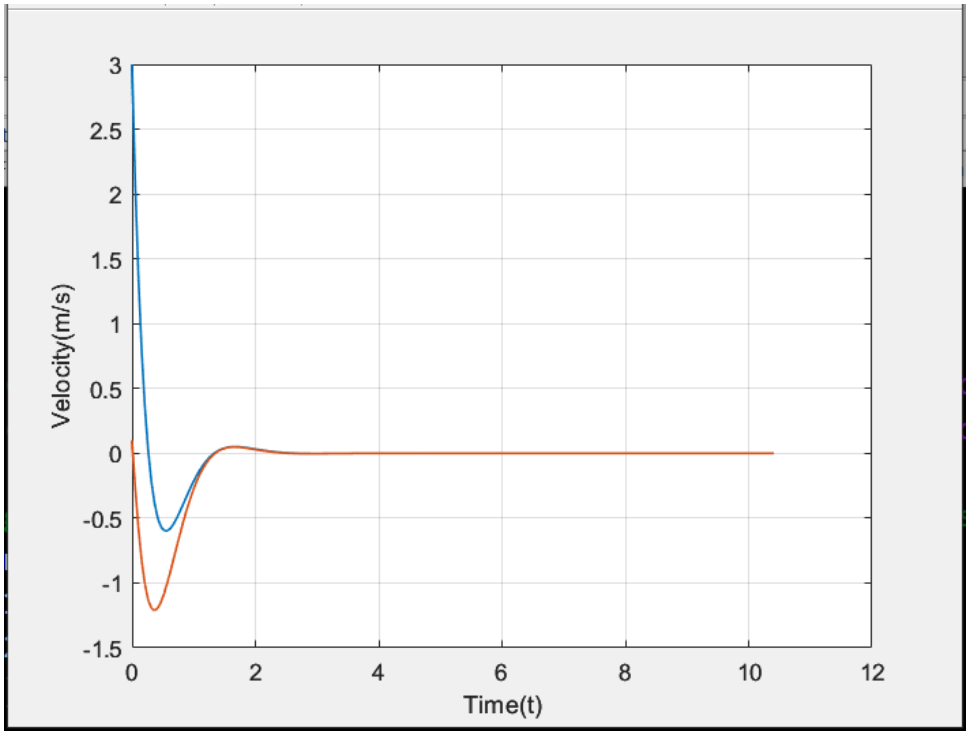
Input

```
Mass of first object: 10
Mass of second object: 15
First spring constant: 100
Second spring constant: 150
Third spring constant: 200
Viscous damping for first portion: 50
Viscous damping for second portion: 60
Viscous damping for third portion: 70
Warning: The value of local variables may have been c
global before you use that variable.
> In TDOFfreeDamp (line 21)
Initial displacement of first body: 0
Initial displacement of second body: 0.8
Initial velocity of first body: 3
Initial velocity of second body: 0.1
>>
```

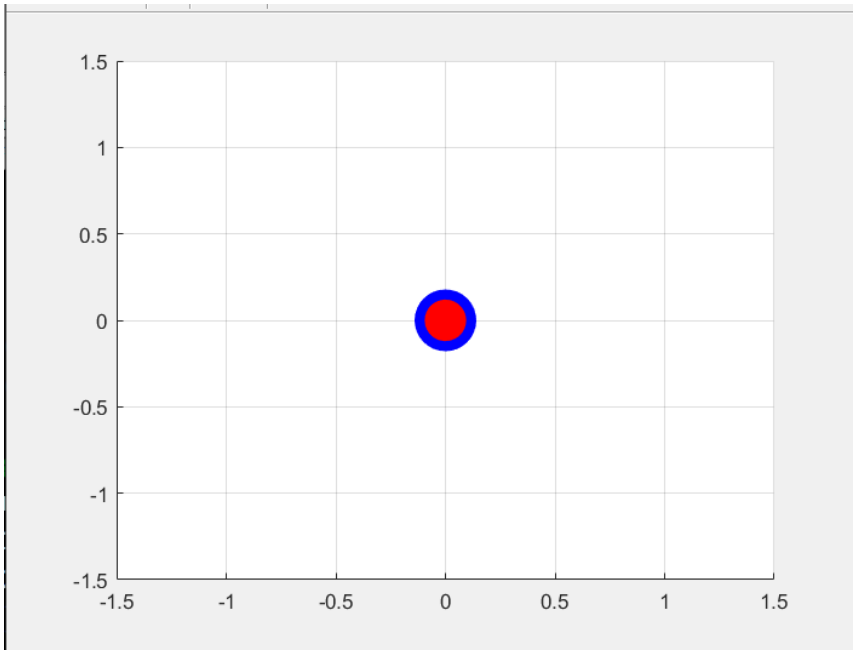
Displacement



Velocity



Position



This is how the animation diagram looks like in steady phase.

Contribution:

The project "Single and Two Degree of Freedom Spring-Mass System" was proposed by Kafi Anan, who also developed the concepts and mathematical derivations. Zobayer Nabil created the one degree of freedom codes, while Kafi Anan developed the two degree of freedom codes, graphs, and animation. Kafi Anan also prepared the presentation slides and reviewed and edited them with the help of Zobayer Nabil. Initially, Zobayer Nabil drafted the report, which was later reviewed and edited by Kafi Anan.

Conclusion:

The project on "Single and Two Degree of Freedom Spring-Mass System" was carried out with precision and thoroughness, resulting in successful completion. The simulations for all cases of one degree of freedom spring-mass system were performed perfectly, while the free vibration of the two-degree-of-freedom system, both with and without damping, was also carried out impeccably. The results obtained from the simulations were compared with theoretical results, and the level of similarity between them was remarkable. Therefore, the project can be considered as a success, achieving its intended objectives with excellence.

PROJECT LINK:

https://buetedu-my.sharepoint.com/:u:/g/personal/2006040_eee_buet_ac_bd/EYgq8dv9rbhAif32F2gfBtMBpEyTH67kgl9cX60H6VSmoQ?e=L6WiG3