Bangladesh University of Engineering and Technology
Department of Electrical and Electronic Engineering
Dhaka-1205, Bangladesh

Course No: EEE212

Course Title: Numerical Techniques Laboratory

Project Report on "Damped Spring-mass System"

Submitted by,

Name: Al Fattah Siddique

Student ID: 1906129

Section: B2

Date of Submission: 22-2-22

Project Introduction:

In this project of regarding spring-mass system, we had to plot the position and velocity of the mass attached to the system with respect to time. Here I have calculated the position and velocity numerically. Differential equation of a springmass system is

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = F(t)$$

***Now**, simplifying this equation, we get*

$$X''(t) = \frac{F(t)}{m} - \frac{b}{m}x'(t) - \frac{k}{m}x(t)$$

Now let's consider $x(t) = u$

$X'(t) = v$

$x''(t) = v'$

Now, $v' = \frac{F(t)}{m} - \frac{b}{m}v - \frac{k}{m}u$ -------(1)

And $u' = v$ -------(2)

To solve these two equations

$$\begin{pmatrix} v' \\ u' \end{pmatrix} = \begin{pmatrix} -\frac{b}{m} & -\frac{k}{m} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix} + F(t)\begin{pmatrix} \frac{1}{m} \\ 0 \end{pmatrix}$$

$u_{n+1} = u_n + u'dt$

$v_{n+1} = v_n + v'dt$

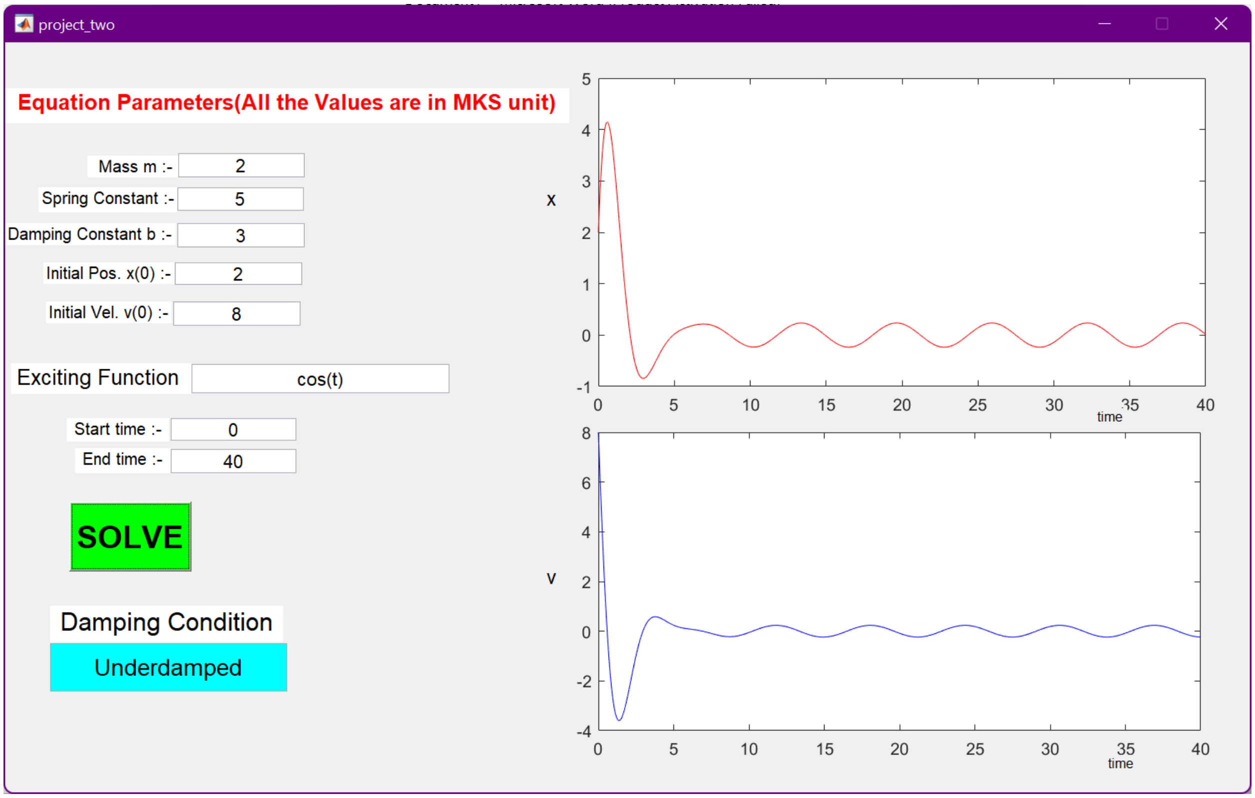
Next, we have to build a user interface where the user will be able to give inputs in MKS units. For a second order spring mass system the inputs will be an exciting function **F(t)**, the mass of the object **m**, damping constant **b**, spring constant **k**, and the initial conditions of position(u0) and velocity(v0). The user will be able to see the graph of **Position(x) vs. time** and **velocity (dx/dt) vs. time** and damping condition when he will click the solve button.

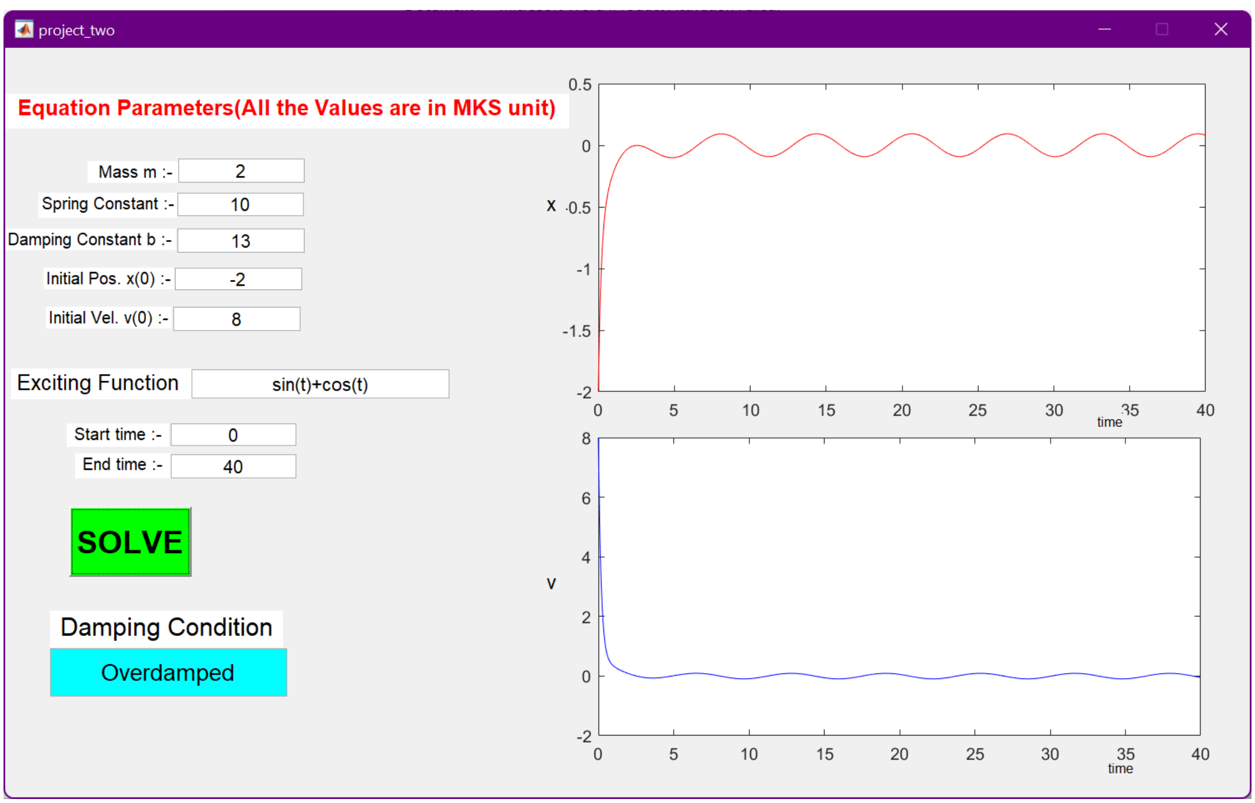
Main Code:

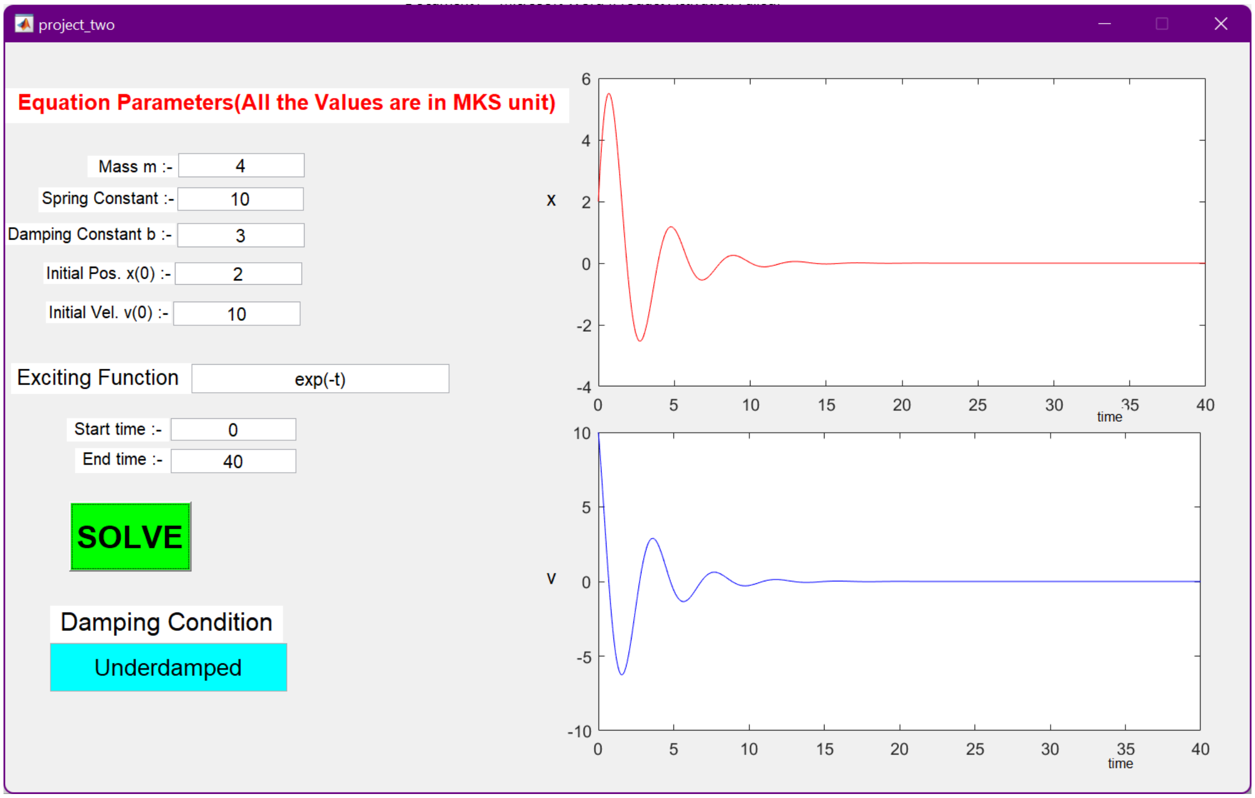
```
s=get(handles.fun,'string');

d=strcat('@(t)',s);
f=str2func(d);
x0=str2num(get(handles.u,'string'));
dx0=str2num(get(handles.v,'string'));
m=str2num(get(handles.m,'string'));
b=str2num(get(handles.b,'string'));
k=str2num(get(handles.k,'string'));
tstart=str2num(get(handles.ts,'string'));
tend=str2num(get(handles.tend,'string'));
dt=0.0001;

t=tstart:dt:tend;  %y=f(t)
len=(tend-tstart)/dt;
us=zeros(1,len);
vs=zeros(1,len);
A=[-(b/m) -(k/m);1 0];
M=[(1/m);0];
UV=[dx0;x0];
us(1)=x0;
vs(1)=dx0;
for i=1:len
   UVp=A*UV+f(t(i))*M;
```

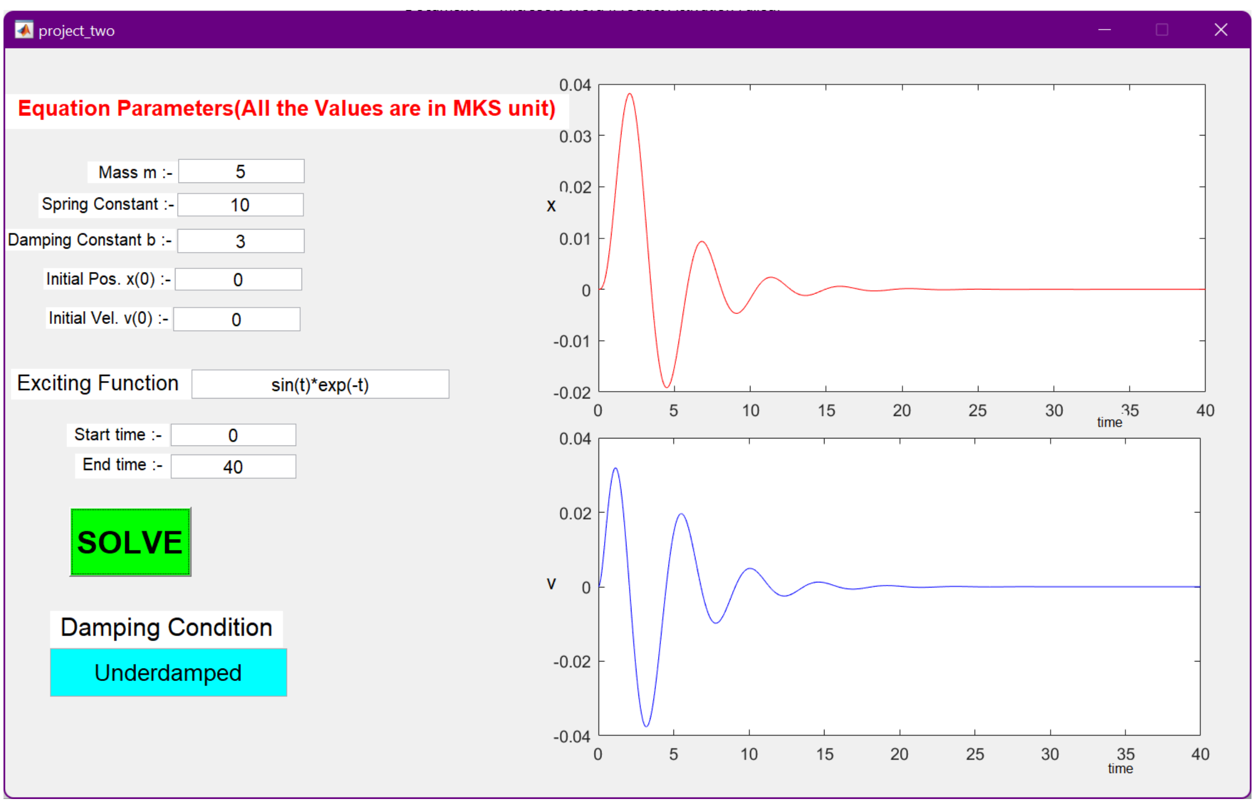
```
    UV=UV+UVp.*dt;
    vs(i+1)=UV(1);
    us(i+1)=UV(2);
end
if b==sqrt(4*m*k)
    disp("critically damped")
    s=1;
end
if b>sqrt(4*m*k)
    disp("overdamped")
    s=2;
end
if b<sqrt(4*m*k)
    disp("Underdamped")
    s=3;
end

axes(handles.axes1);
plot(tstart:dt:tend,us,'r')

axes(handles.axes2);
plot(tstart:dt:tend,vs,'b')

if s==1
    set(handles.testoutput, 'string','Critically damped');
end
if s==2
    set(handles.testoutput, 'string','Overdamped');
end
if s==3
    set(handles.testoutput, 'string','Underdamped');
end
```

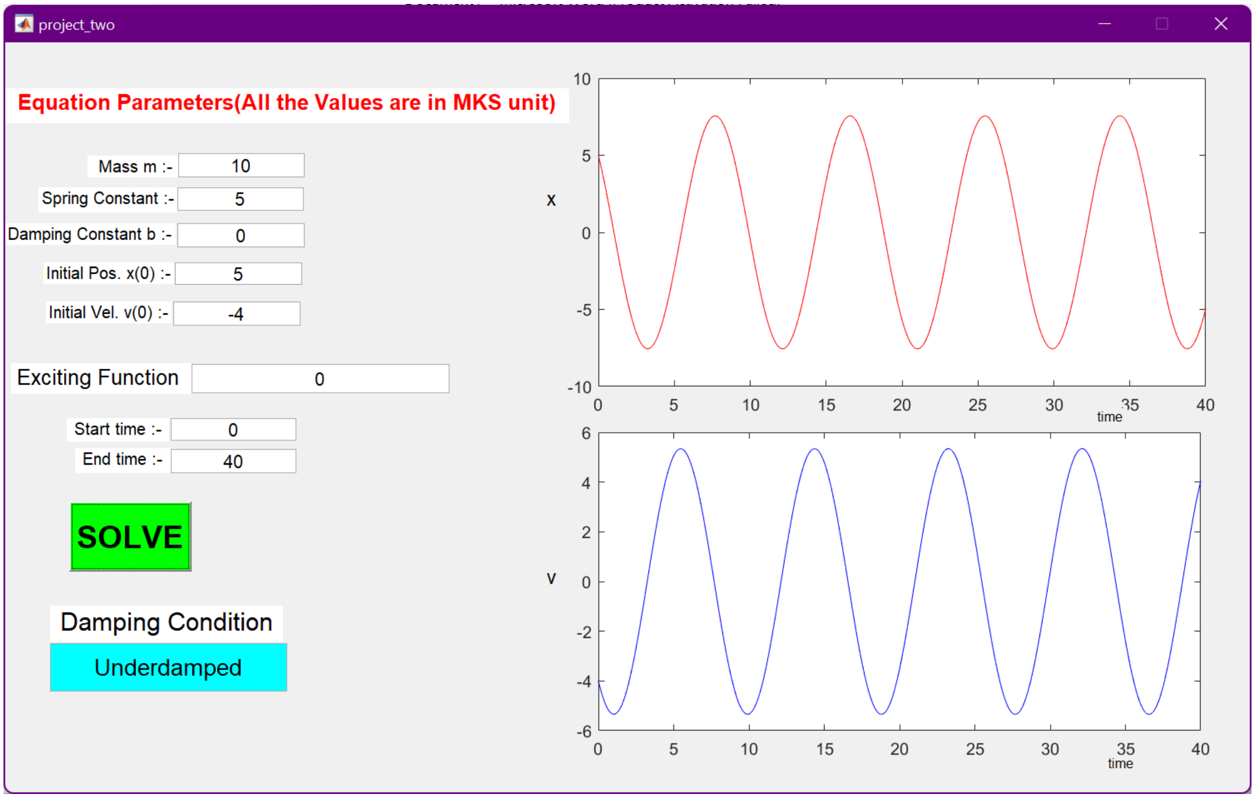
Output for 10 Test Cases:

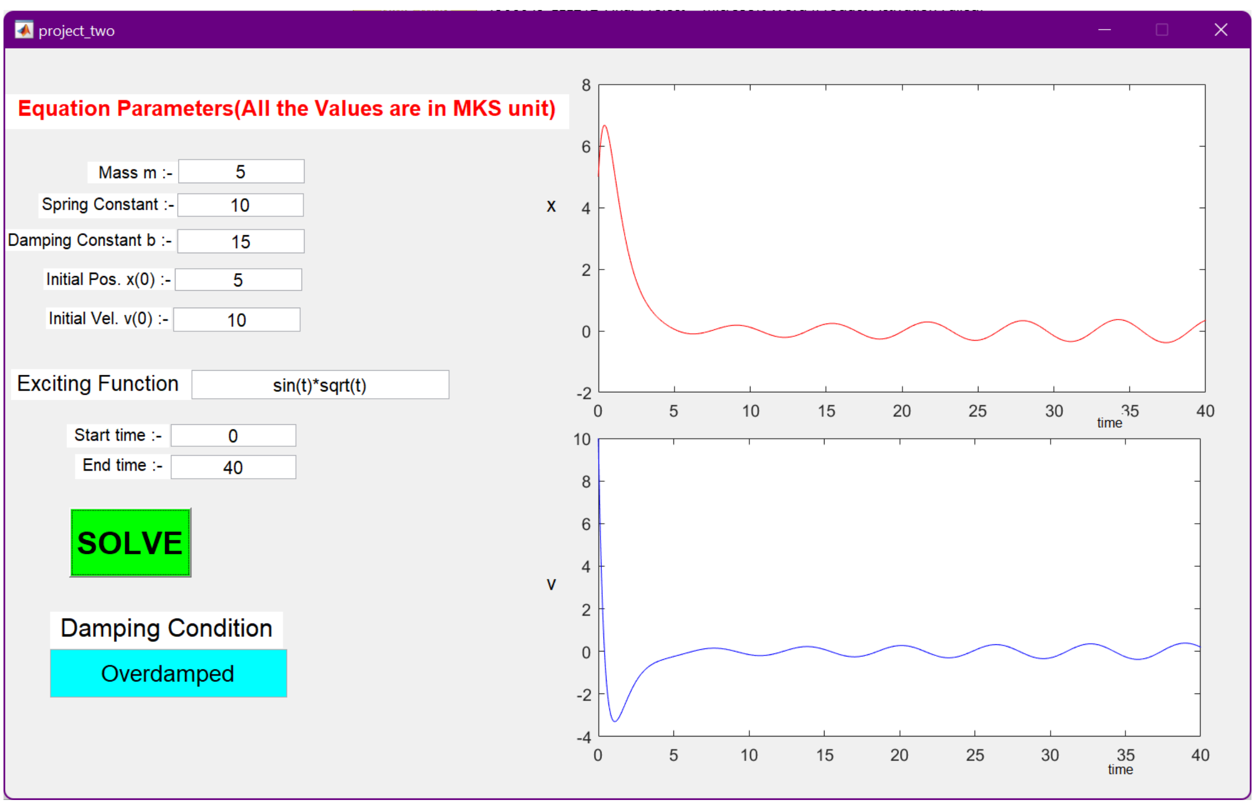


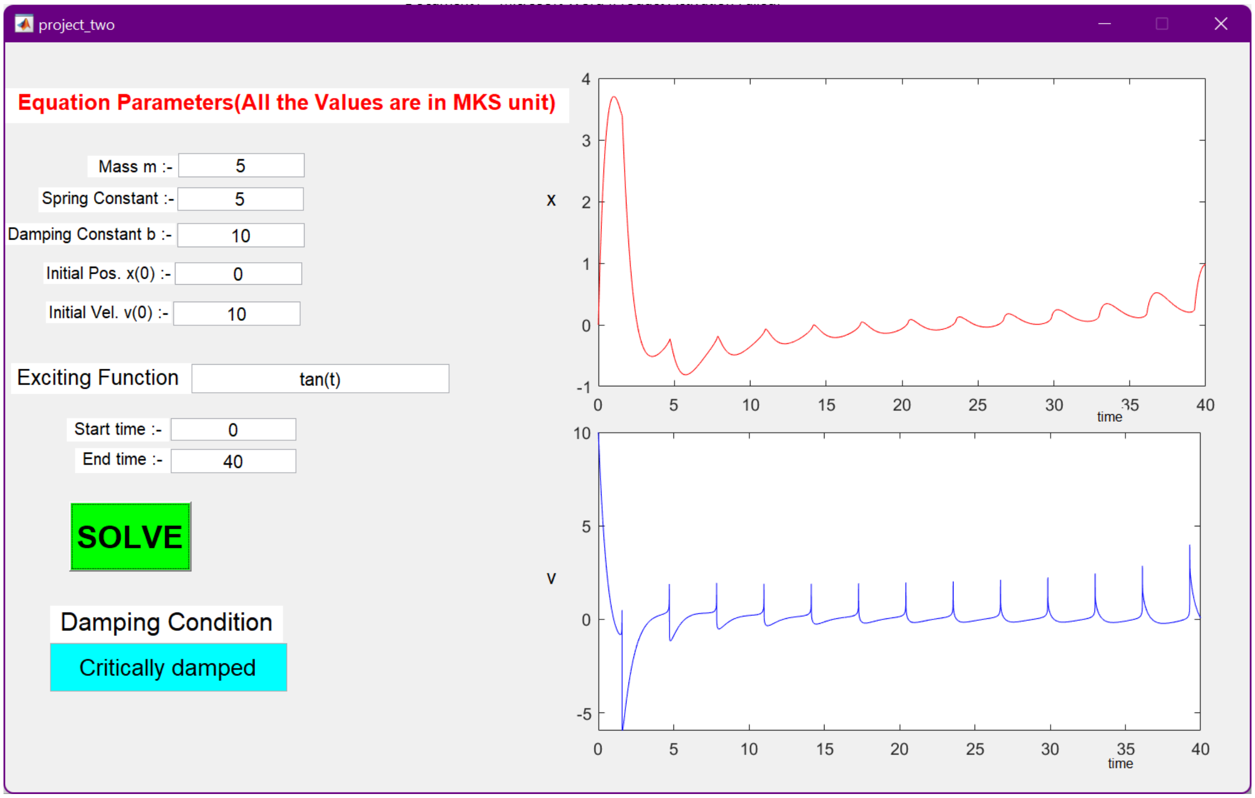


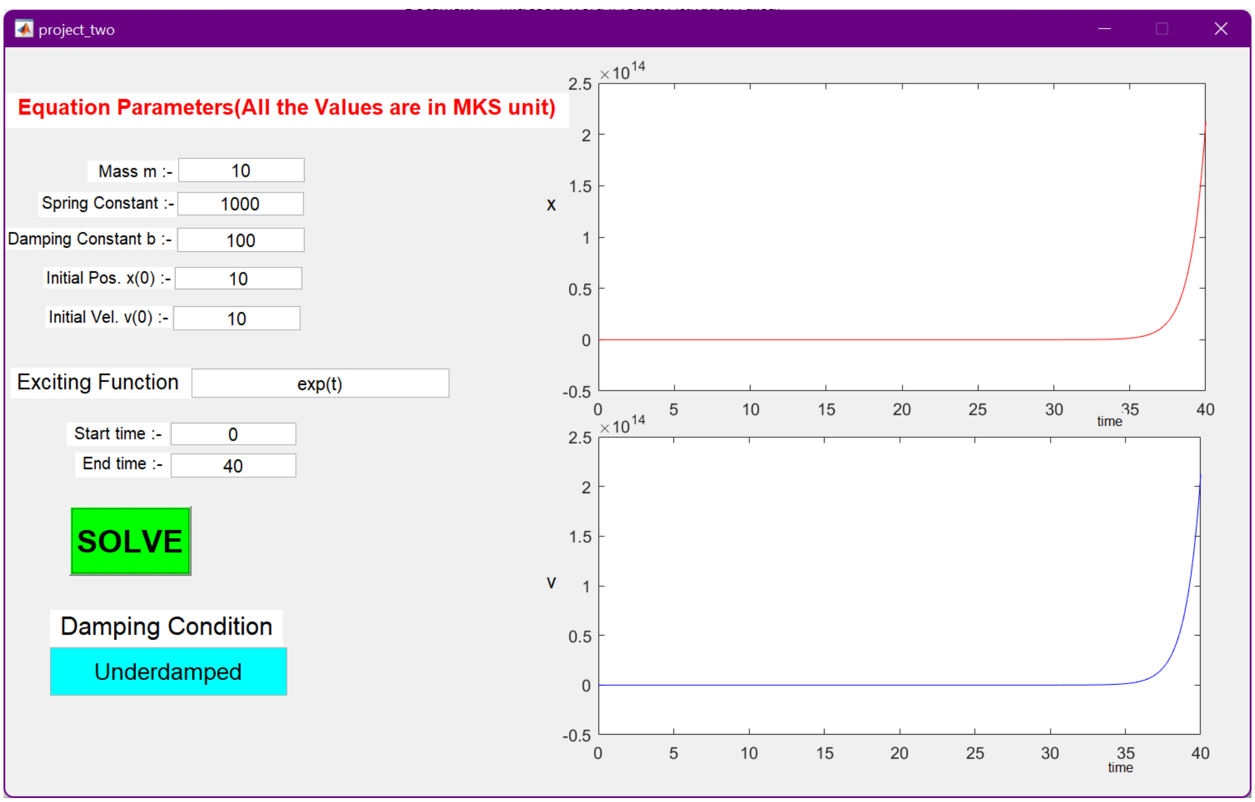

project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :-
4
Spring Constant :-
10
Damping Constant b :-
3
Initial Pos. x(0) :-
2
Initial Vel. v(0) :-
10
Exciting Function
exp(-t)
Start time :-
0
End time :-
40
SOLVE
Damping Condition
Underdamped
x
v
time
time

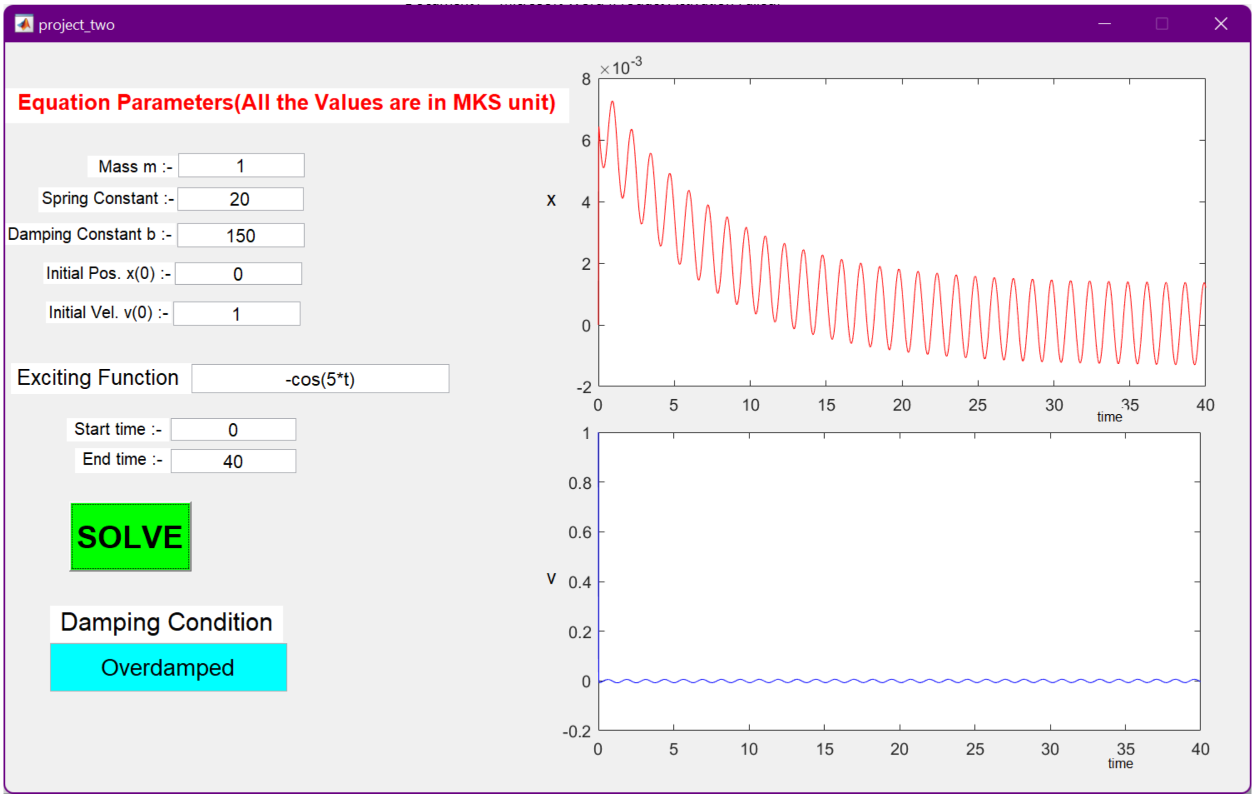

project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :-
5
Spring Constant :-
10
Damping Constant b :-
3
Initial Pos. x(0) :-
0
Initial Vel. v(0) :-
0
Exciting Function
sin(t)*exp(-t)
Start time :-
0
End time :-
40
SOLVE
Damping Condition
Underdamped
x
v
time
time

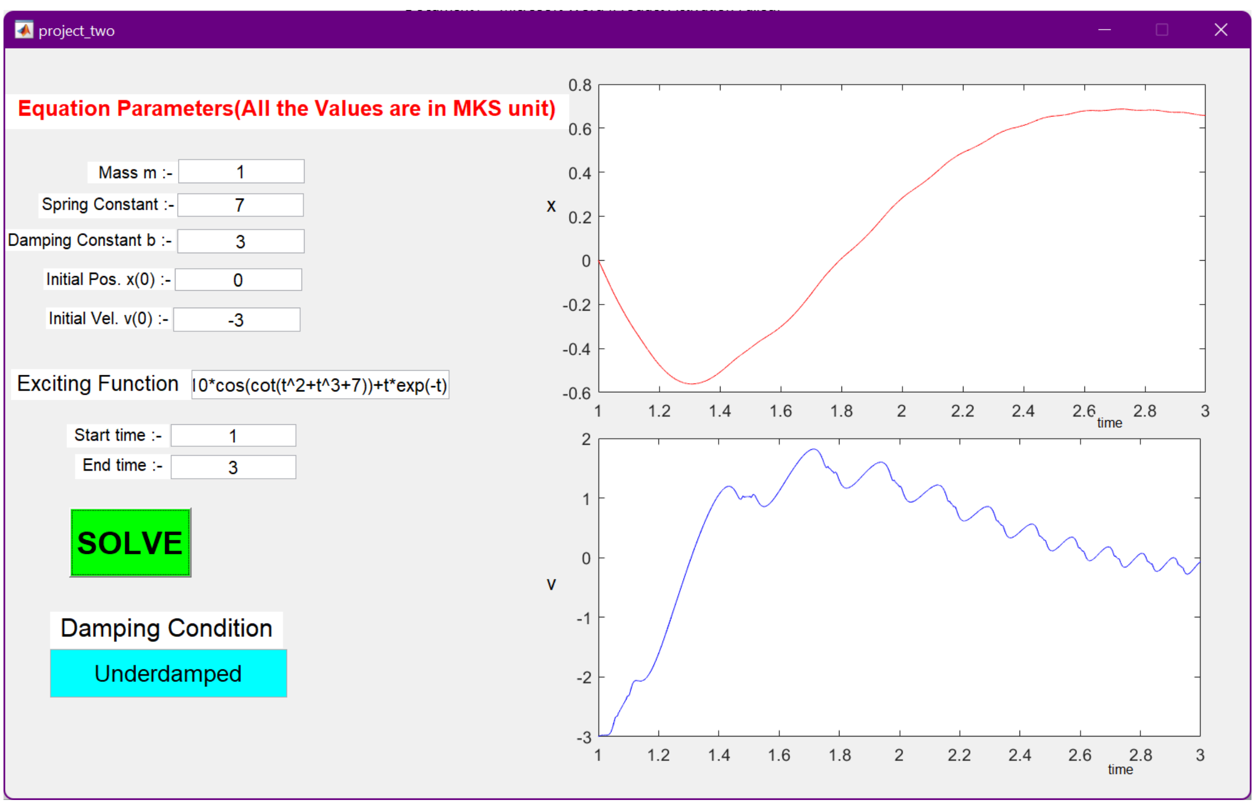

project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :- 10
Spring Constant :- 5
Damping Constant b :- 0
Initial Pos. x(0) :- 5
Initial Vel. v(0) :- -4
Exciting Function 0
Start time :- 0
End time :- 40
SOLVE
Damping Condition
Underdamped
x
v
time
time


project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :- 5
Spring Constant :- 10
Damping Constant b :- 15
Initial Pos. x(0) :- 5
Initial Vel. v(0) :- 10
Exciting Function sin(t)*sqrt(t)
Start time :- 0
End time :- 40
SOLVE
Damping Condition
Overdamped
x
v
time
time


project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :-
5
Spring Constant :-
5
Damping Constant b :-
10
Initial Pos. x(0) :-
0
Initial Vel. v(0) :-
10
Exciting Function
tan(t)
Start time :-
0
End time :-
40
SOLVE
Damping Condition
Critically damped
x
v
time
time


project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :-
10
Spring Constant :-
1000
Damping Constant b :-
100
Initial Pos. x(0) :-
10
Initial Vel. v(0) :-
10
Exciting Function
exp(t)
Start time :-
0
End time :-
40
SOLVE
Damping Condition
Underdamped
×10^14
×10^14
x
v
time
time


project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :- 1
Spring Constant :- 20
Damping Constant b :- 150
Initial Pos. x(0) :- 0
Initial Vel. v(0) :- 1
Exciting Function -cos(5*t)
Start time :- 0
End time :- 40
SOLVE
Damping Condition
Overdamped
x
v
time
×10^-3


project_two
Equation Parameters(All the Values are in MKS unit)
Mass m :- 1
Spring Constant :- 7
Damping Constant b :- 3
Initial Pos. x(0) :- 0
Initial Vel. v(0) :- -3
Exciting Function 10*cos(cot(t^2+t^3+7))+t*exp(-t)
Start time :- 1
End time :- 3
SOLVE
Damping Condition
Underdamped
x
v
time

Conclusion:

In this project, the goal was to provide the user with a simple and easy-to-use program to simulate the position and velocity for different time instant. Anyone can simulate the position and velocity of the system by using the interface for his/her need. This simulation also shows us the damping condition of the system.The person has nothing to do with the code. Thus, this project has a great advantage over simulating the position and velocity of a spring-mass damped system.