# BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering

***Course No:*** *EEE 212*

***Course Title:*** *Numerical Technique Laboratory*

## A MATLAB Project Report on Spring-Mass System

## Submitted to


- **Dr. Hafiz Imtiaz**

Assistant Professor, Department of EEE

Bangladesh University of Engineering & Technology

- **Shaimur Salehin Akash**

Lecturer, Department of EEE

Bangladesh University of Engineering & Technology



**Submitted By :** Jonaidul Islam Sikder

**Student ID** **:** 1906041

**Section** **:** A-2

**Department** **:** EEE


**Date of Submission :** 14 February, 2022

## Introduction

A spring that is attached to a mass comprises a spring mass system that can be mathematically modeled by a second order differential equation. Within elastic limits, the system abides by Hooke's law for elasticity from and through its implementation the spring constant, denoted by **k**, can be derived. The 2$^{nd}$ order differential equation can be derived applying Newton's 3$^{rd}$ law of motion. When the system is free from any retarding forces acting upon it and the mass vibrates being unaffected by any external force, the motion is called **free vibration** and the exciting function is 0. The energy stored in the system can be dissipated and consequently the simple harmonic oscillation is obstructed or stopped. This phenomenon is called damping and there can be three cases of damping : Under-damping, Critical damping and Over-damping. In a mechanical system like a spring-mass system, viscous drag of the medium that the system is situated in can contribute to damping. In this project, the displacement and velocity of a spring-mass system under various user-defined inputs along with the damping conditions have been evaluated via the construction of a GUI with the help of a MATLAB code.

## Problem Statement

For this project , I have been assigned with the task of numerically calculating and simulating the position and the velocity of a mass attached to a spring-mass system. Determining and displaying the damping condition of the system, I have been instructed to build a suitable GUI in order to simulate the given problem. I have been given the following second order ordinary differential equation of a spring mass system :

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

Here,

m = Mass of the object

b = Damping constant

k = Spring constant

$F(t)$ = Exciting Function

These data have to be taken as inputs from the user alongside the initial conditions ( initial displacement and initial velocity ) and the problem has to solved and simulated on that basis. As per the instructions, outputs for 10 different cases have to be attached in the project report.

## Numerical Procedure :

The most important task needed to be accomplished in order to prepare this project is solving the given 2nd order Ordinary Differential Equation ( ODE ) numerically on MATLAB. Before coding, it is necessary to proceed with the problem manually. In order to fulfill this objective, the 2nd order ODE has been converted into a 1st order ODE so that it can be solved numerically through the implementation of Euler's method. Now let us assume that,

$$\frac{dx}{dt} = a \dots\dots\dots\dots\dots\dots (1)$$

$$\therefore \frac{d^2x}{dt^2} = \frac{da}{dt}$$

Therefore, the given ODE can be re-written as,

$$m\frac{da}{dt} + ba + kx = F(t)$$

$$\text{Or, } \frac{da}{dt} = -\frac{b}{m}a - \frac{k}{m}x + \frac{F(t)}{m} \dots\dots\dots\dots\dots (2)$$

Now we can implement Euler's Method in order to solve the 1st order ODE in equation no. (2)

Let us assume our time step size is **h**. The initial time and the final time alongside **h** will be taken from the users as input. The number of steps needed will be,

$$n = \frac{t_{final} - t_{initial}}{h} \dots\dots\dots\dots\dots (3)$$

The value of h taken as input from the users has been stored under the variable **h1** in the code. Relatively small values of h allow decreasing room for errors. The value of n should be rounded because the number of iterations need to be an integer.

Initial displacement ( x0 ) and initial velocity ( v0 ) shall be taken as inputs from the users, too.

Two different functions have been assumed as equations (1) and (2).

$$f1(t, x, a) = a = \frac{dx}{dt} \dots\dots\dots\dots\dots (4)$$

$$f2(t, x, a) = -\frac{b}{m}a - \frac{k}{m}x \dots\dots\dots\dots\dots (5)$$

Now we know that, for a given pair of ODE's such as,

$$\frac{dy}{dt} = f(t,y,z)\ ; y(t0) = y0 \text{ and } \frac{dz}{dt} = g(t,y,z)\ ;\ z(t0) = \ z0$$

$y_{n+1} = y_n + ( h \text{ x } f(t_n, y_n, z_n ) )$ and $z_{n+1} = z_n + ( h \text{ x } g(t_n, y_n, z_n ) )$

Therefore, we can write that,

$$x_{n+1} = x_n + h\frac{dx}{dt} = x_n + \big( h \text{ x } f1(t_n, x_n, a_n)\big) \dots\dots\dots\dots\dots (5)$$

$$a_{n+1} = a_n + h\frac{da}{dt}$$

$$= a_n + \left( h \text{ x } \left( \frac{Exciting\ Function}{m} \right) \right)$$

$$+ ( h \text{ x } f2(t_n, x_n, a_n) \dots\dots\dots\dots\dots (6)$$

Lastly, our independent variable time will be updated as,

$t_{n+1} = t_n + h \dots\dots\dots\dots\dots (7)$ ; [ We know that if the independent variable is x, it is updated according to the formula $x_n = x_0 + nh$ ]

Now, for the damping conditions, it is known that,

- If $b^2$-4mk < 0, then the system is underdamped.
- If $b^2$-4mk = 0, then the system is critically damped.
- If $b^2$-4mk > 0, then the system is overdamped.

Following the aforementioned mathematical procedures, the code has been written on MATLAB and Guide platform has been used in order to simulate the problem via a GUI.

## Algorithm :

1. Take inputs from user ( The value of mass of the object, m ; the value of the damping constant, b ; the value of the spring constant, k ; the exciting function ; initial displacement and initial velocity, initial and final times of observation and the desired time step size )
2. Let, $\frac{dx}{dt} = a$, which leads to $\frac{d^2x}{dt^2} = \frac{da}{dt}$
3. Let, $f1(t, x, a) = a = \frac{dx}{dt}$ and $f2(t, x, a) = -\frac{b}{m}a - \frac{k}{m}x$
4. $n = \frac{t_{final} - t_{initial}}{h1}$
5. n_rounded = round(n)
6. $t(1) = t_{initial}$
7. x(1)=x0
8. a(1)=a0
9. Start with i=1 and Do

   [ x(i+1) = x(i) + ( h1*$f$1( t(i), x(i), a(i) ) )

   a(i+1) = a(i) + ( ( h1*string(t(i) )) / m ) + ( h1*$f$2( t(i), x(i), a(i) ) )

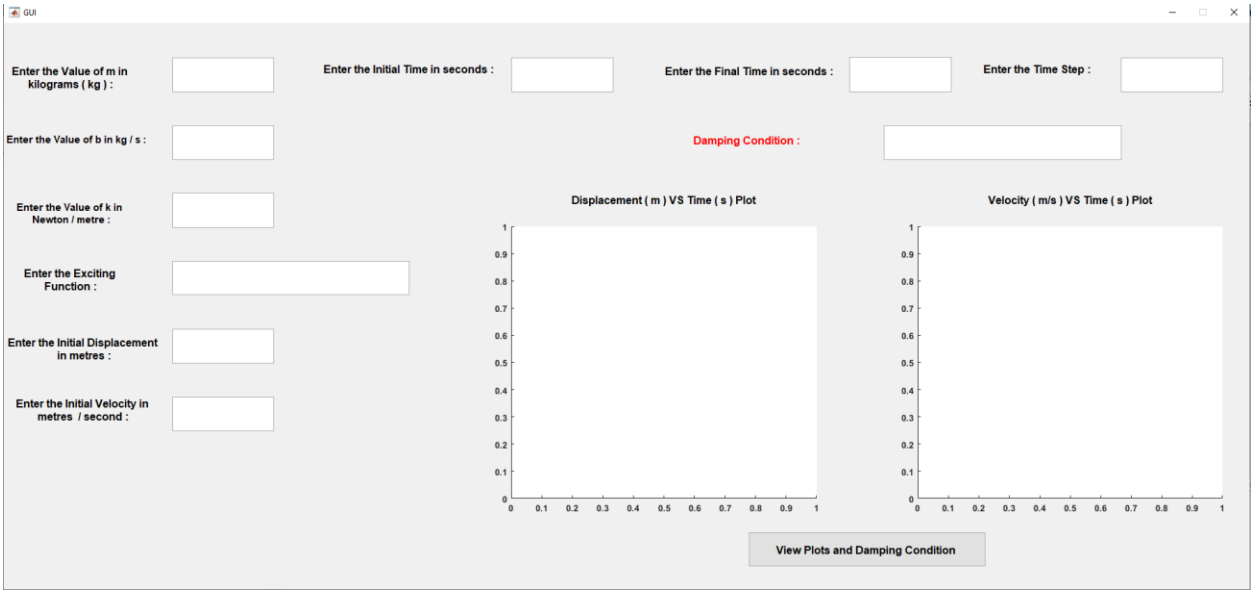
   t(i+1) = t(i) + h1 ]

   Until the value of i reaches round_n
10. If $b^2$-4mk < 0, show that the damping condition is underdamped

    Else if $b^2$-4mk = 0, show that the damping condition is critically damped

    Else if $b^2$-4mk > 0, show that the damping condition is overdamped
11. Plot the displacement VS time graph
12. Plot the velocity VS time graph

# Graphical User Interface ( GUI ) Layout




The value of m, b, k, initial displacement, initial velocity and the exciting function will be taken as inputs from the users alongside initial and final time and the time step size. For convenience, the vitality for SI units has been highlighted by mentioning them in the brackets. The code will take these inputs and generate what damping condition the spring-mass system is acting under and it will also generate the displacement vs time and the velocity vs time plots which have been labeled accordingly.

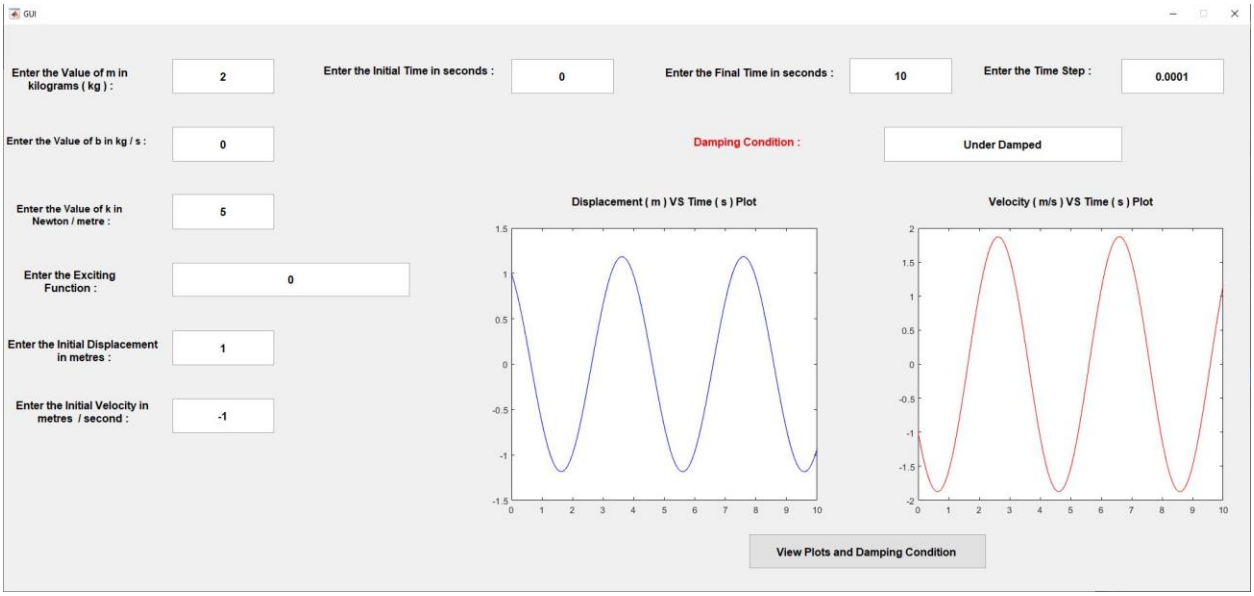
# MATLAB Code on Editor Window

```
282
283     % --- Executes on button press in pushbutton1.
284     function pushbutton1_Callback(hObject, eventdata, handles)
285     % hObject    handle to pushbutton1 (see GCBO)
286     % eventdata  reserved - to be defined in a future version of MATLAB
287     % handles    structure with handles and user data (see GUIDATA)
288
289
290 -   m=str2num(get(handles.edit1,'string'));
291
292 -   b=str2num(get(handles.edit2,'string'));
293
294 -   k=str2num(get(handles.edit3,'string'));
295
296 -   Exciting_Function=get(handles.edit4,'string');
297
298 -   EF=strcat('@(t)',Exciting_Function);
299
300 -   string=str2func(EF);
301
302 -   x0=str2num(get(handles.edit5,'string'));
303
304 -   v0=str2num(get(handles.edit6,'string'));
305
306 -   t_initial=str2num(get(handles.edit7,'string'));
307
308 -   t_final=str2num(get(handles.edit8,'string'));
309
310 -   h=str2num(get(handles.edit9,'string'));
311
312
```

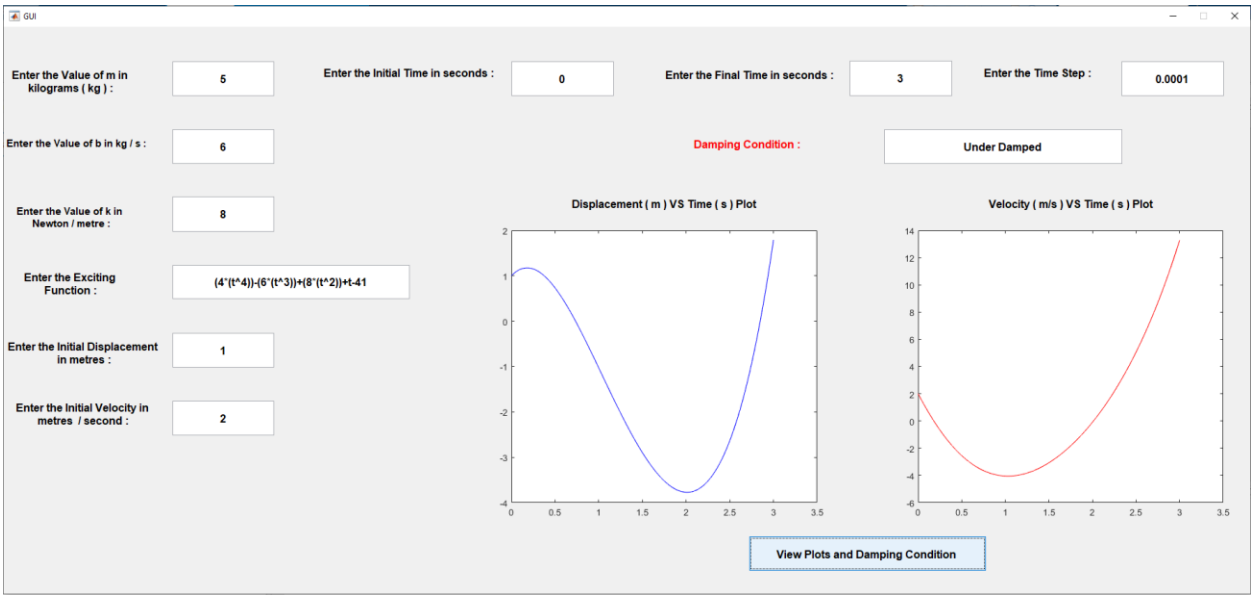
```
f1=@(t, x, a) a;

f2=@(t, x, a) (1/m)*(-k.*x-b.*a);


h1=h;

n=(t_final-t_initial)/h1;

n_rounded = round(n);

t(1)=t_initial;

x(1)=x0;

a(1)=v0;


for i=1:n_rounded

x(i+1)=x(i)+(h1*f1(t(i), x(i), a(i)));

a(i+1)=a(i)+((h1*string(t(i)))/m)+(h1*(f2(t(i), x(i), a(i))));

t(i+1)=t(i)+h1;

end


if ( (b^2) - ( 4*m*k ) < 0 )
    set(handles.edit10, 'string', 'Under Damped');

elseif ( (b^2) - ( 4*m*k ) == 0 )
    set(handles.edit10, 'string', 'Critically Damped');

elseif ( (b^2) - ( 4*m*k ) > 0 )
    set(handles.edit10, 'string', 'Over Damped');

end

plot(handles.axes1,t,x,'b');
plot(handles.axes2,t,a,'r');

```

## Outputs for Different Cases :

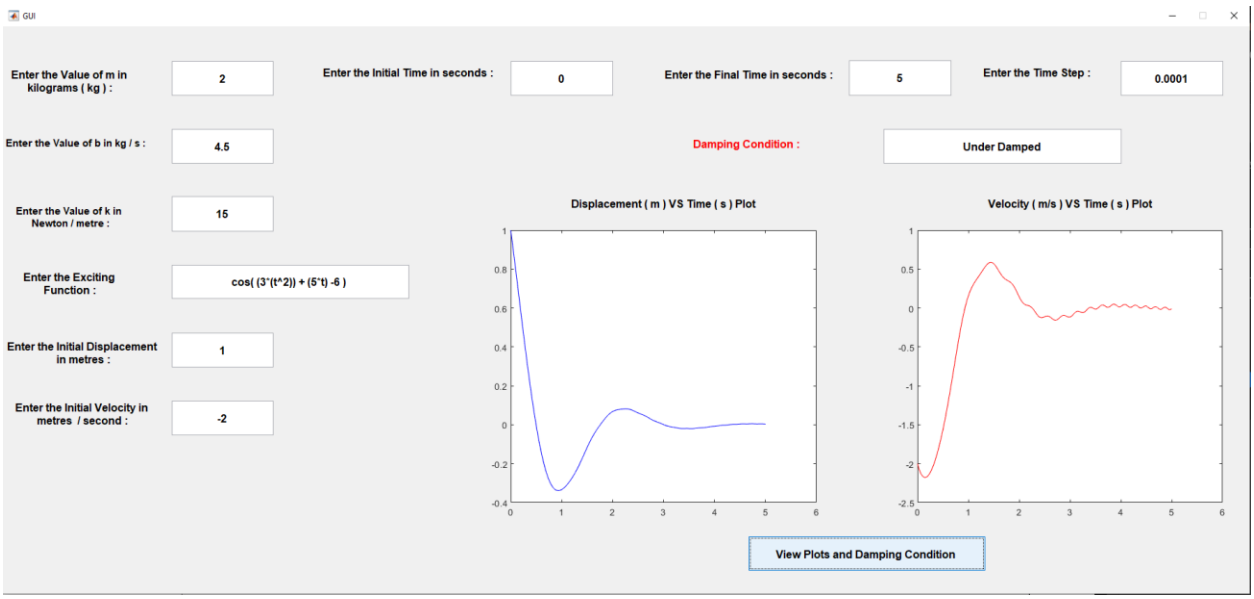
1. No damping i.e., the system is under damped. The exciting function is 0.



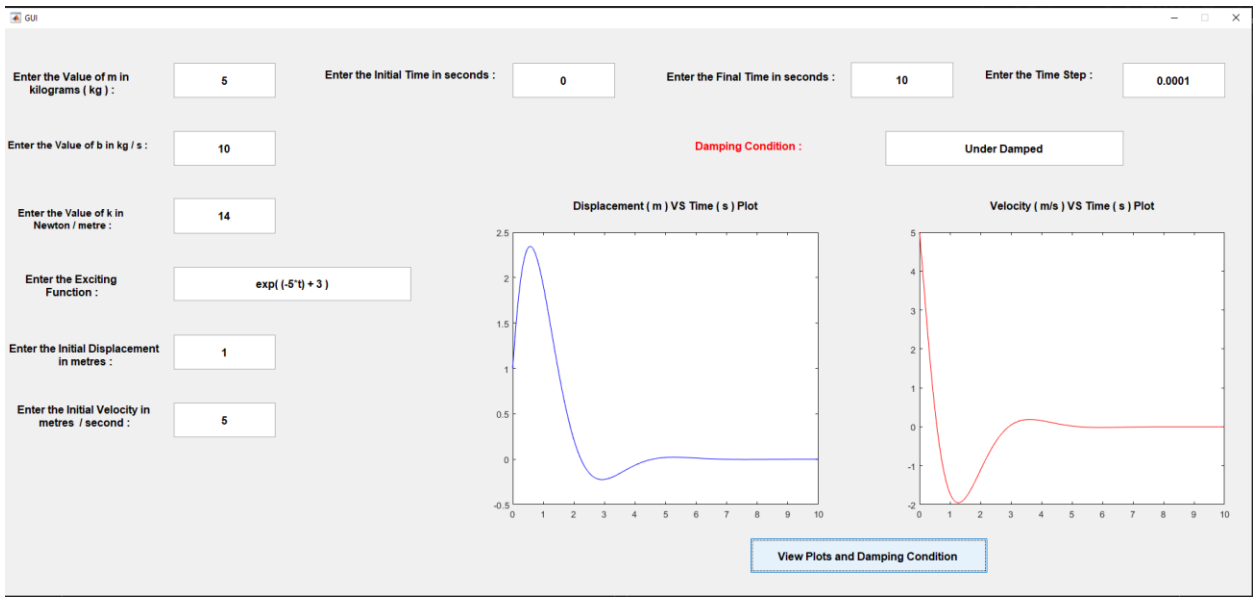
**2.** The system is under damped. The exciting function is a polynomial expression.



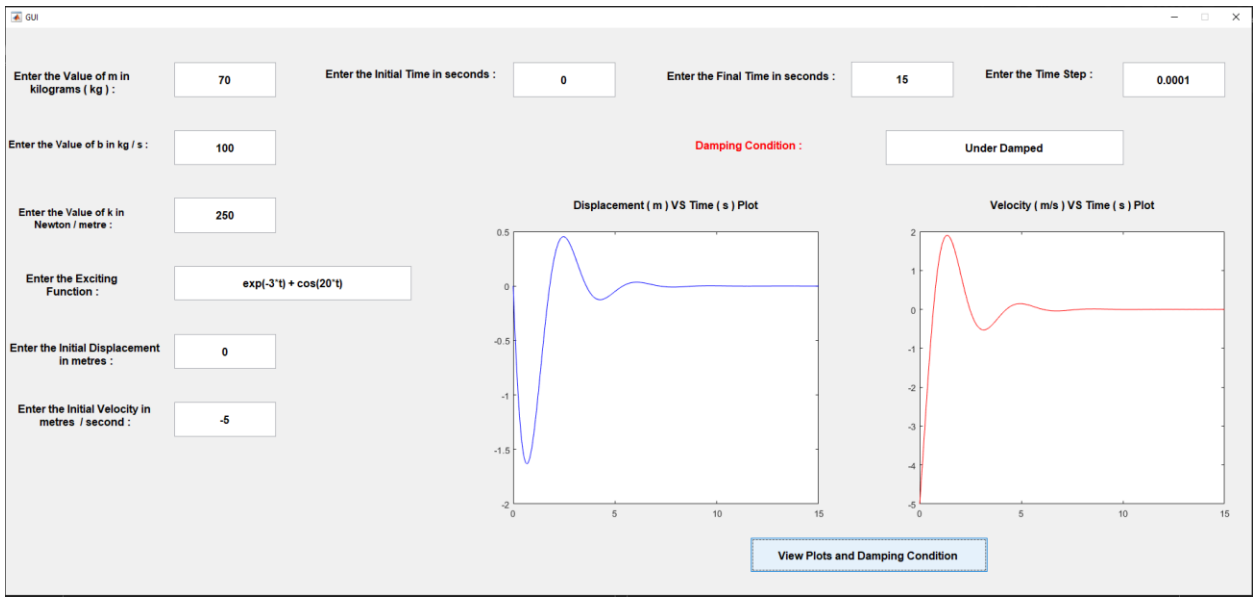
**3.** The system is under damped. The exciting function is a sinusoidal expression.



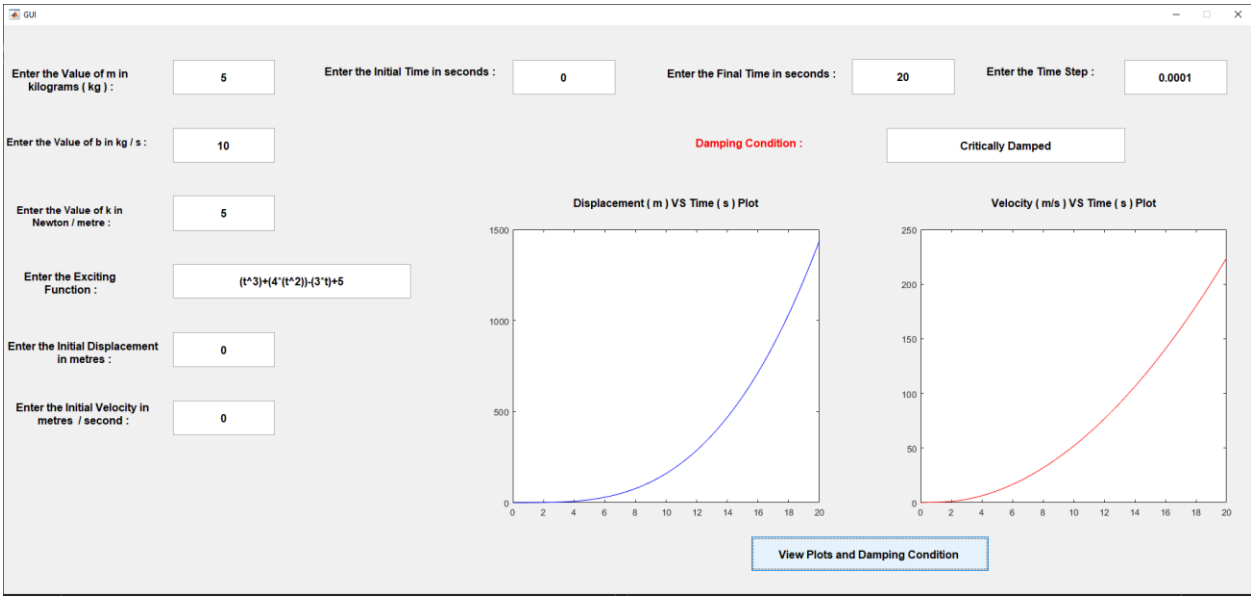
**4.** The system is under damped. The exciting function is an exponential expression.



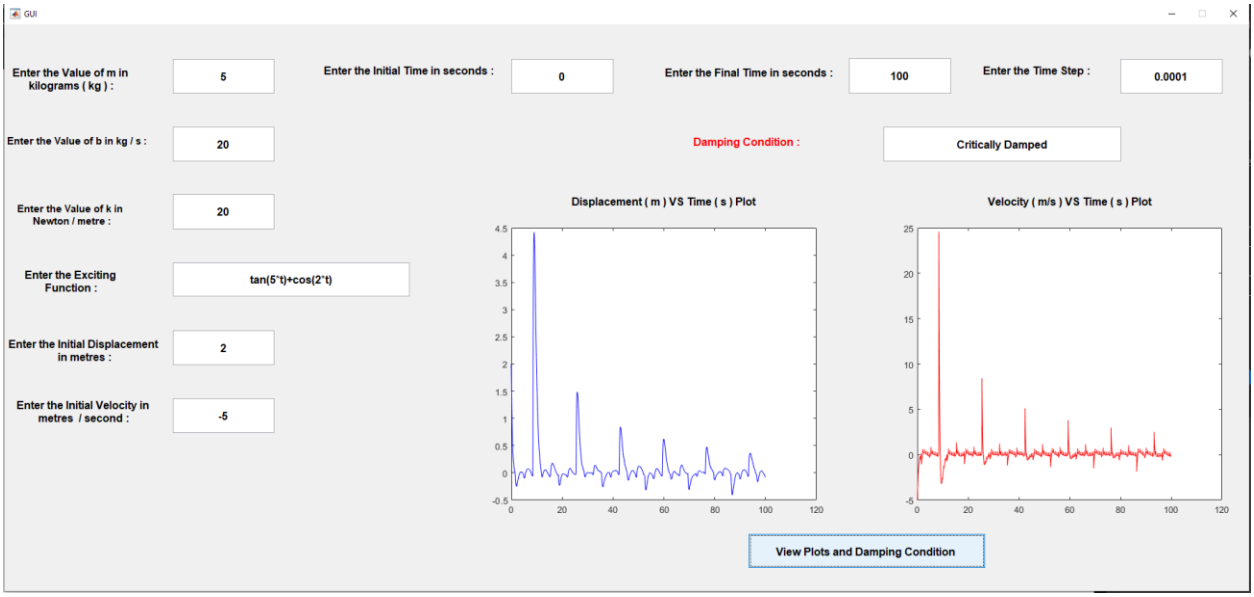
**5.** The system is under damped. The exciting function contains an exponential expression and a sinusoidal expression.



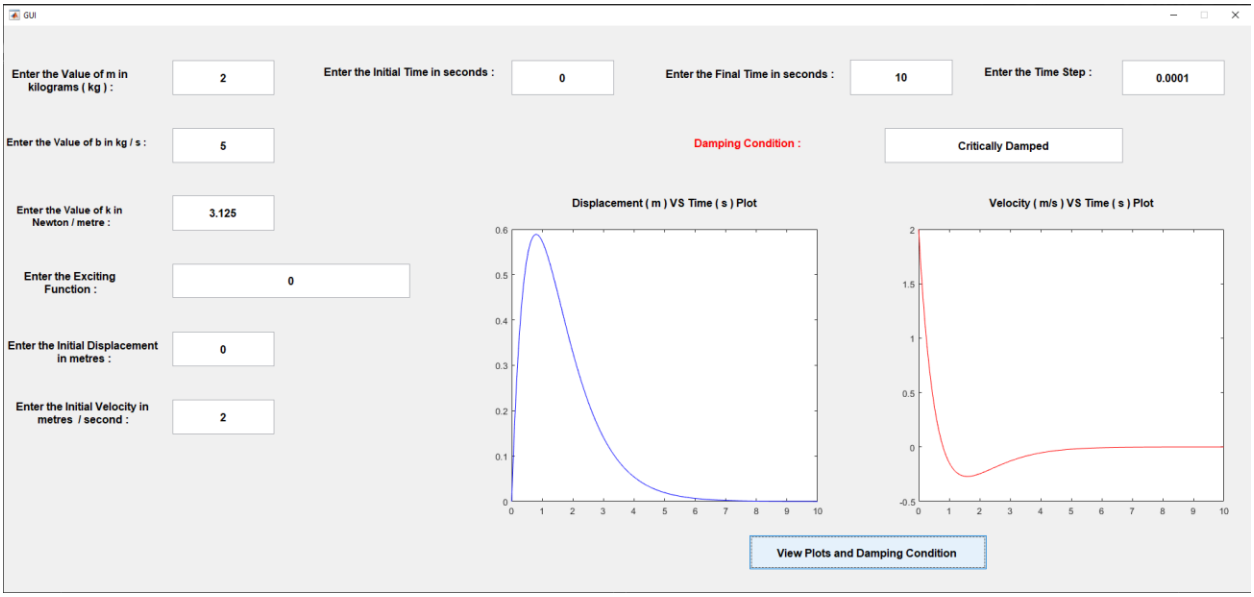
**6.** The system is critically damped. The exciting function is a polynomial expression.



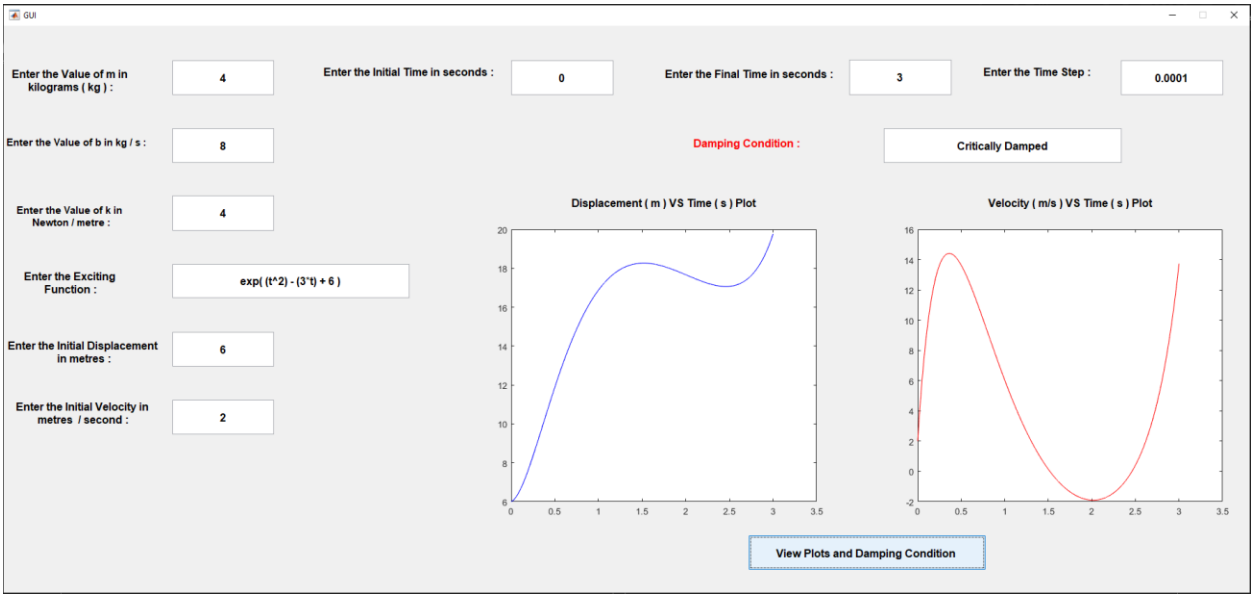
7. The system is critically damped. The exciting function is a sinusoidal expression.



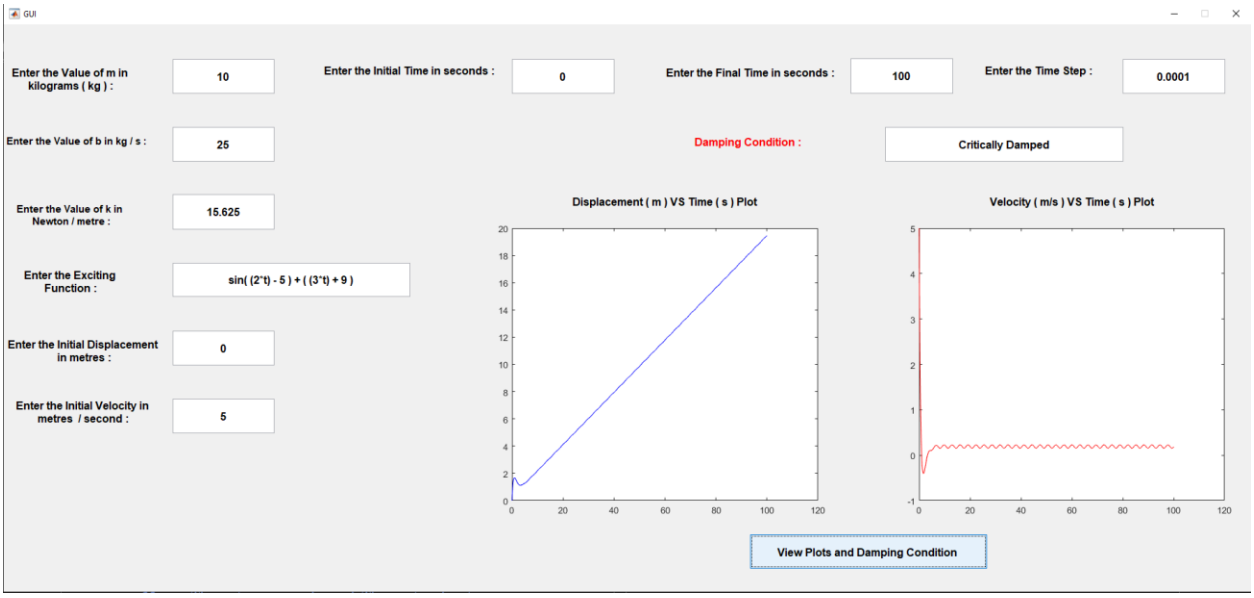
**8.** The system is critically damped. The exciting function is 0.



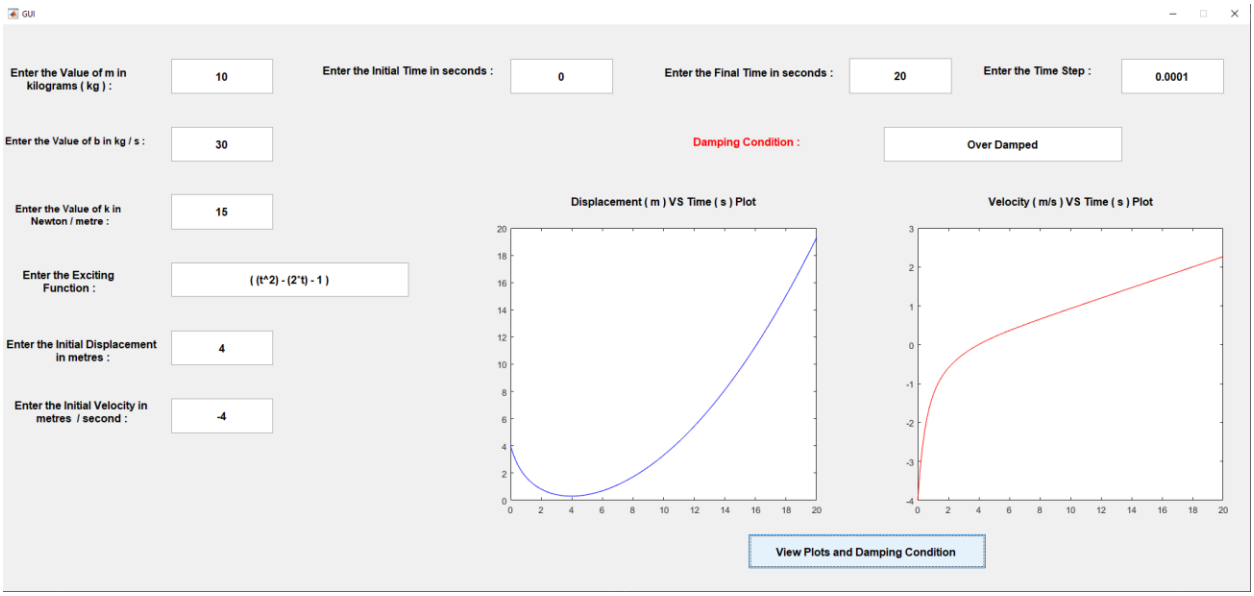
**9.** The system is critically damped. The exciting function is an exponential expression.



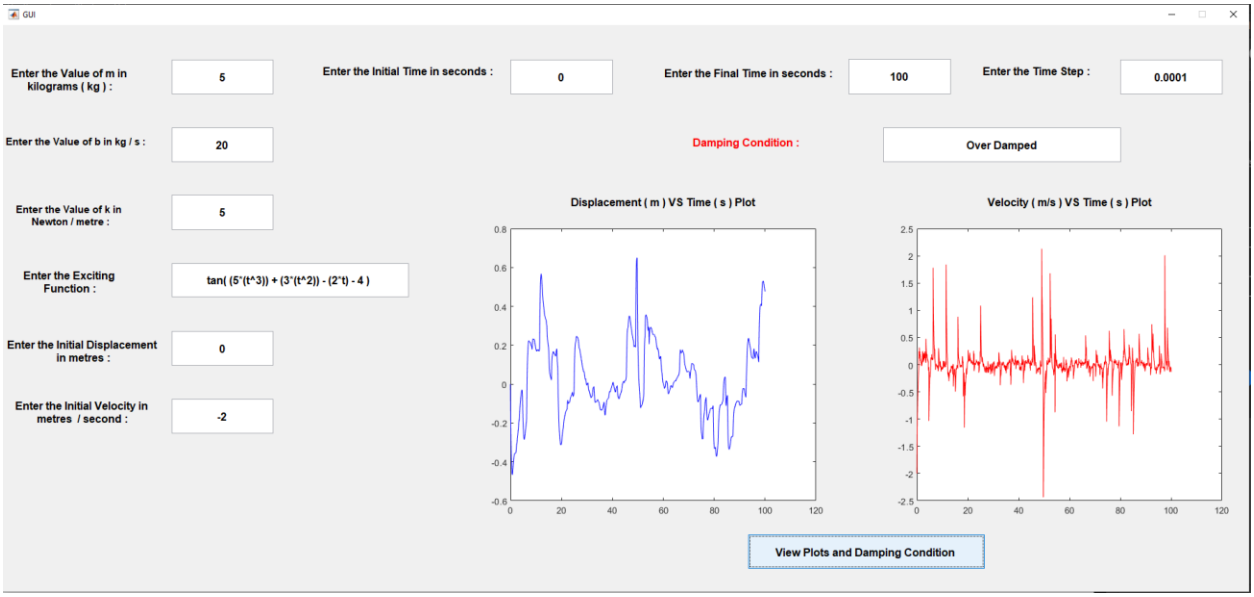
**10.** The system is critically damped. The exciting function contains a polynomial expression and a sinusoidal expression.



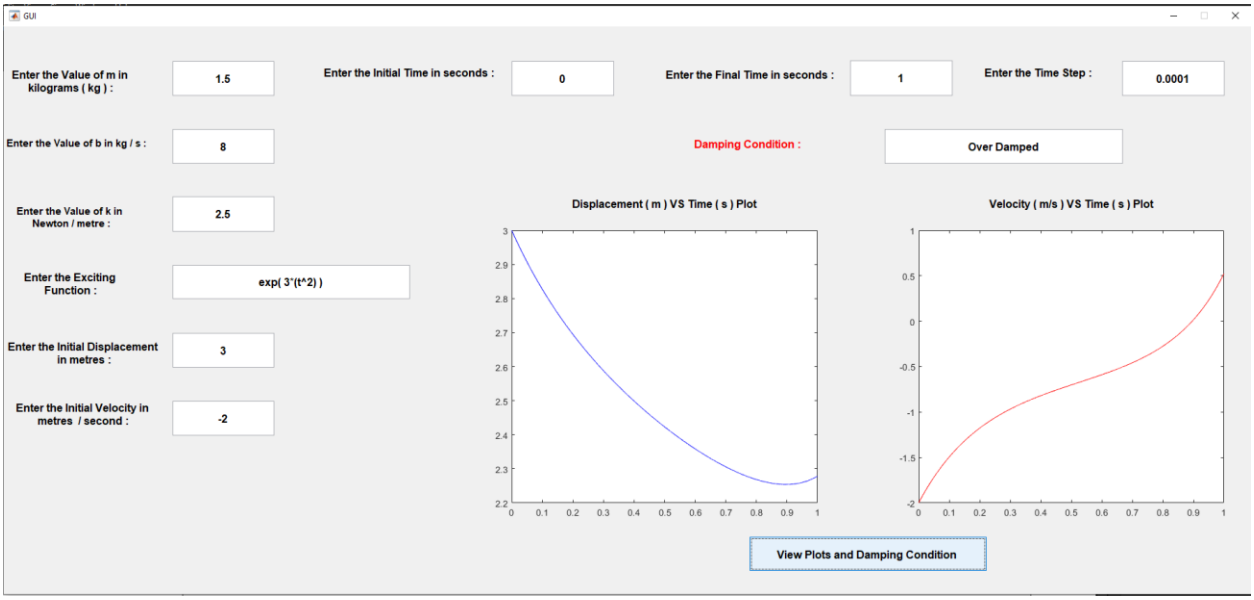
**11.** The system is over damped. The exciting function is a polynomial expression.



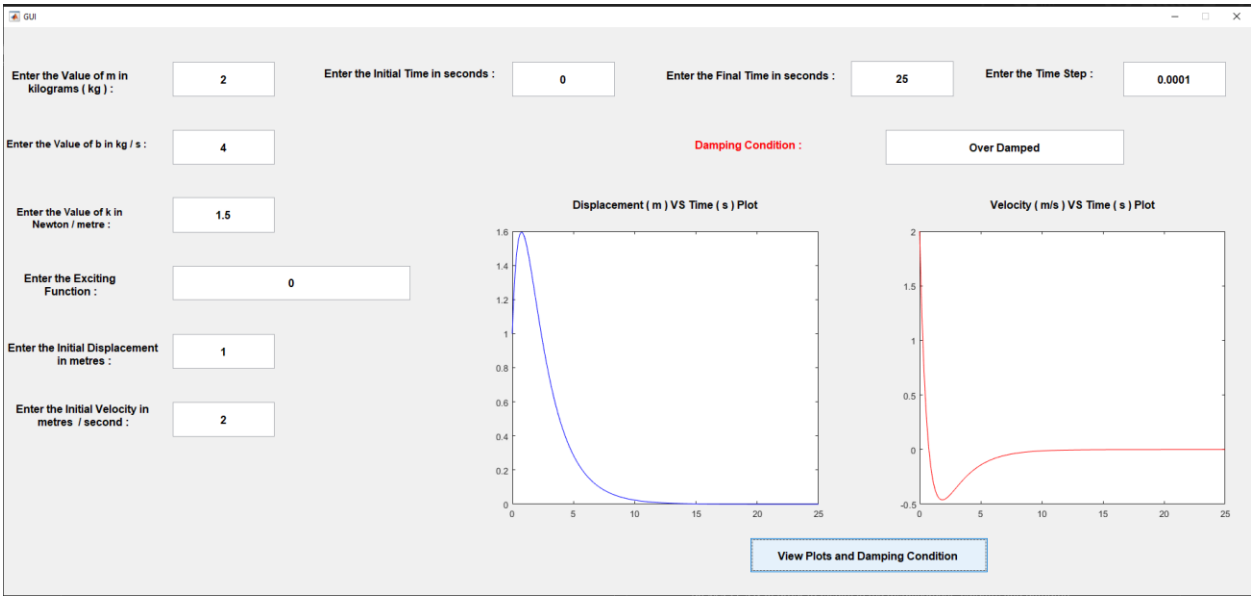
**12.** The system is over damped. The exciting function is a sinusoidal expression.



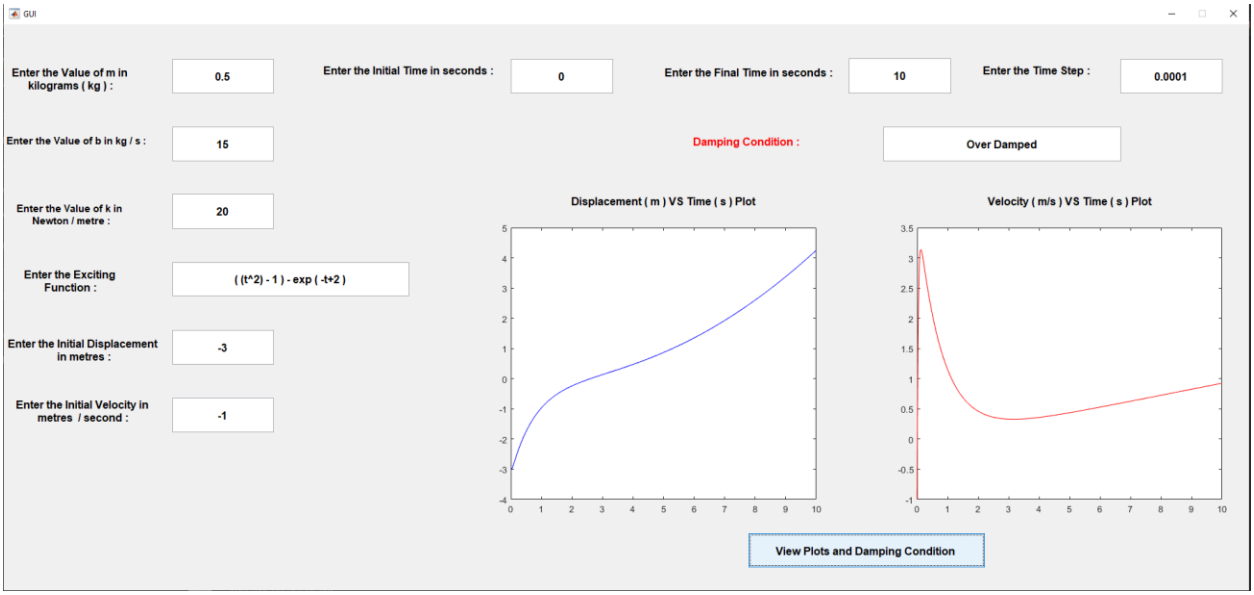
**13.** The system is over damped. The exciting function is an exponential expression.



**14.** The system is over damped. The exciting function is 0.



**15.** The system is over damped. The exciting function contains both a polynomial expression and an exponential expression.



## Conclusion

The goal of this project was to construct a user-friendly interface through coding on MATLAB in order to visualize the displacement, velocity and damping conditions of a spring mass system. With the view to materializing this goal I have used the GUIDE platform to build the GUI and the $2^{nd}$ order ODE that characterizes a spring-mass system mathematically has been solved numerically using Euler's method.

The GUI constructed may appear slow to respond due to the conditions imposed. The ODE has to be solved numerically and not using library functions built for solving ODEs such as the ODE45 function. Hence the loops add to the complexity of the code and the time needed for execution is augmented.

Despite the short-comings originating from a numerical approach to solving the given problem, the code was found to be executed without the occurrence of errors in the GUI. Hence it is safe to conclude that the users shall find it easy and effective to utilize the GUI in order to visualize the solution of any spring-mass system and the objectives of pursuing this project have been fulfilled successfully.