

# Control Systems Lab Manual

(Version 3.0: Spring 2015)



Revised and Updated by:  
Engr. Muniba Ashfaq (Lecturer, DCSE, UET  
Peshawar)

---

**Version 1.0**

**Designed and Supervised by:**

**Engr. G. Mubashar Hassan (Assistant Professor,  
DCSE, UET Peshawar)**

**Engr. Shah Mahmood (Lecturer, DCSE, UET Peshawar)**

Special thanks to the following students of Batch 5 who contributed a lot in the preparation of this Laboratory Manual for Control Systems in 2007:

1. Maqsood Muhammad Khan
2. Maqsood Ahmad
3. Shadman Khattak
4. Umbreen Gul
5. Hafeez Anwar
6. Osama Siddique
7. Muneem Shamsheer
8. Muhammad Bilal
9. Munazza Razzaq
10. Sajida Imran
11. Wajid Zaman

## Revision History

2007	Engr. G. Mubashar Hassan, Engr. Shah Mahmood	Version 1.0
2012	Engr. Farooq Kifayat Ullah	Version 2.0
2015	Engr. Muniba Ashfaq	Version 3.0

## List of Experiments

Lab 1: Introduction to MATLAB

Lab 2: System Response by different inputs in MATLAB and Simulink

Lab 3:

[A] Simulation of Differential Equations using MATLAB

[B] Simulation of Differential Equations using Simulink

Lab 4: Simulation of Higher Order Differential Equations using MATLAB and Simulink

Lab 5: Frequency Domain Modeling in MATLAB

Lab 6: Time Domain Modeling in MATLAB

Lab 7: System Stability in Simulink

Lab 8: System Interconnections in Simulink

Lab 9: Stability via Routh Hurwitz

Lab 10: Steady State Error Analysis using Simulink

Lab 11: Implementation of Root Locus using MATLAB

Lab 12: Implementation of Root Locus using Sisotool

Lab 13: System Design using Sisotool

Lab 14: Bode Plot using MATLAB

# Lab 1

## Introduction to MATLAB

### Introduction:

Matlab is commercial "MATrix LABoratory" package by Math works, which operates as an interactive programming language environment with graphical output.

In engineering Matlab is displaying popular programming languages, due to its interactive interface, reliable algorithmic, foundation, fully extensible environment and computational speed.

### 1. Matlab basic operations:

#### 1.1. Entering and running Matlab:

On system running Microsoft windows double click on Matlab icon to launch Matlab, a command window will appear with a prompt ">>" you are now in Matlab.

#### 1.2. Leaving Matlab:

A Matlab session is simply terminated by typing the following in Matlab prompt.

```
>> Quit or  
>> exit
```

#### 1.3. Online Help:

Online help is available from the Matlab prompt both general and scientific commands. We can type the following in command prompt for help.

```
>>help  
>>help demo
```

### 2. Desktop tools in Matlab:

#### 2.1. Command window:

In command window we type commands, view program variables; we can clear all the variables by typing Clear in the command Window.

## 2.2. Command History:

We can view the the past commands and save a whole session by using "diary".

## 3. Variables:

Matlab is a case sensitive that is 'a' is not the same as 'A'. Matlab has a built in variables like '**pi**', '**ans**' and '**eps**'. The variable '**ans**' will keep the track of the last output which was not assigned to another variable.

### 3.1. Variable assignment:

The equality sign is used to assign values to the variables. For example

```
>>x=3
>>y=x^2
```

Output can be suppressed by putting a semicolon to the command lines.

```
>>x=3;
>>y=x^2;
```

### 3.2. Active variables:

Who is the active variable in Matlab.

### 3.3. Removing variables:

Clear x and clear are used in matlab as removing the variables.

### 3.4. Saving and restoring variables:

Save filename, load file name are used for saving and restoring.

### 3.5. Variable arithmetic's:

Operators Precedence:

Let we have an equation like:

```
2+3*4^2
```

The following is the operator precedence in Matlab:

"^, \* or /, + or -."

So the ans of above equation will be:

```
>>ans=
    50
```

### 3.6. Command line editing:

In matlab the arrows keys allows "command line editing".

#### 4. Built in Matlab functions:

**Table 1-1** shows the basic built in matlab functions.

function	Meaning	Example
Sin	Sine	$\text{Sin}(\pi)=0.0$
Cos	Cosine	$\text{Cos}(\pi)=1.0$
Tang	Tangent	$\text{Tan}(\pi/4)=1.0$
Exp	Exponential	$\text{Exp}(1.0)=2.7183$
Log	Natural log	$\text{Log}(2.7183)=1.0$

**Table 1-1**

#### 5. Matrices:

The matrices can be created in Matlab by the following commands:

```
>>A=[1 2 3;3 4 5;3 4 7]; or
```

```
>>A=[1,2,3;4,5,6;3,4,7]
```

The matrix element located in the *i*th row and *j*th column of A can be access by usual way:

```
>>A (1, 2), A (2, 3)
```

Matrices can easily be modified by:

```
>>A (2, 3) =10;
```

##### 5.1. Built in matrix functions:

**Table 1-2** shows the built in matrix functions:

Function	Description
Diag	Return diagonal M.E as a vector
Eye	Identity matrix
Magic	Magic square
Ones	Matrix of ones
Rand	Randomly generated matrix

Zeros	Matrix of zeros
-------	-----------------

**Table 1-2**

## 5.2. Matrix operations:

+ 'Addition'  
 - 'Subtraction'  
 / 'Division'  
 , 'Transpose'  
 ^ 'Power'  
 \* 'Multiplication'  
 ./ 'element by element division.'  
 .^ 'element by element power.'

## 6. Round floating point number to integers:

Round, ceilling, floor, etc can easily done in matlab on Matrices. The function used for these are the following:  
 Round (f); ceil (f); floor (f)

## 7. Relational and logical operators:

**Table 1-3** shows relational and logical operators which are Supported by Matlab:

Operators	Description
==	Equal
~=	Not equal
<	Less then
<=	Less then or equal to
>	Greater then
>=	Greater then or equal to
&	And
	Or
~	Not

**Table 1-3**

## 8. Branching construct:

The following branching construct are supported by matlab:

### 8.1. If – end constructs:

If <condition>,



```
<Program>  
End
```

#### **8.2. If - else if end construct:**

```
If <condition1>  
    <program1>  
Else if <program2>  
    End
```

#### **8.3. If - else end construct:**

```
If <condition1>  
    <program1>  
Else  
    <program2>  
End
```

#### **8.4. Looping construct:**

We can use for loop, while loop and nested for loop in matlab just like c or c++.

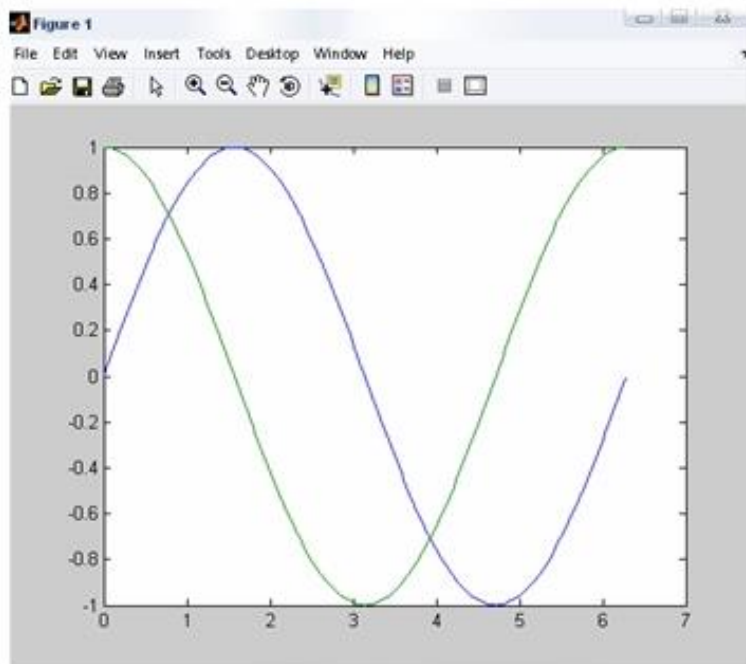
#### **9. Matlab M-file:**

Matlab commands can be run from one file without having to enter each command on command prompt. In order to use them in a program they are save first. For this purpose program should be written in the editor.

#### **10. Matlab graphics:**

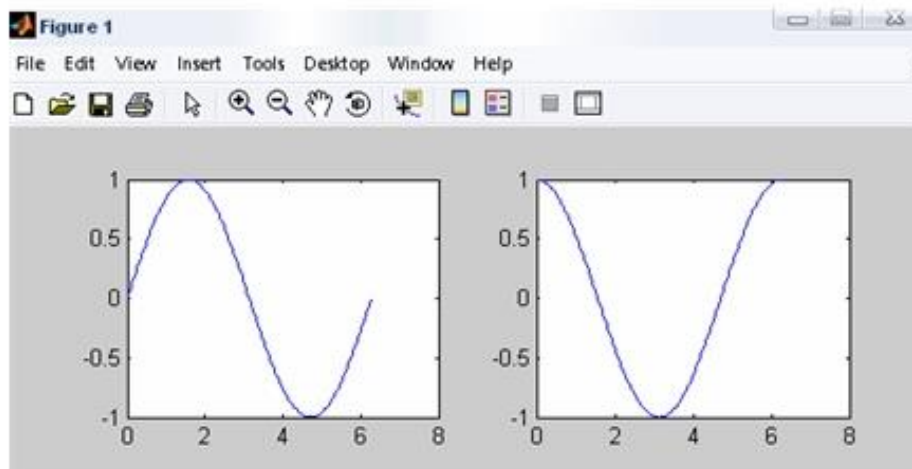
In matlab we can use multiple graphs, multiple plots, three dimensional plots etc.

The following **figure 1-1** shows multiple graphs:



**Figure 1-1**

**Figure 1-2** shows the multiple plots on a single graph.



**Figure 1-2**

**Figure 1-3** shows multiple graphs on a single graph

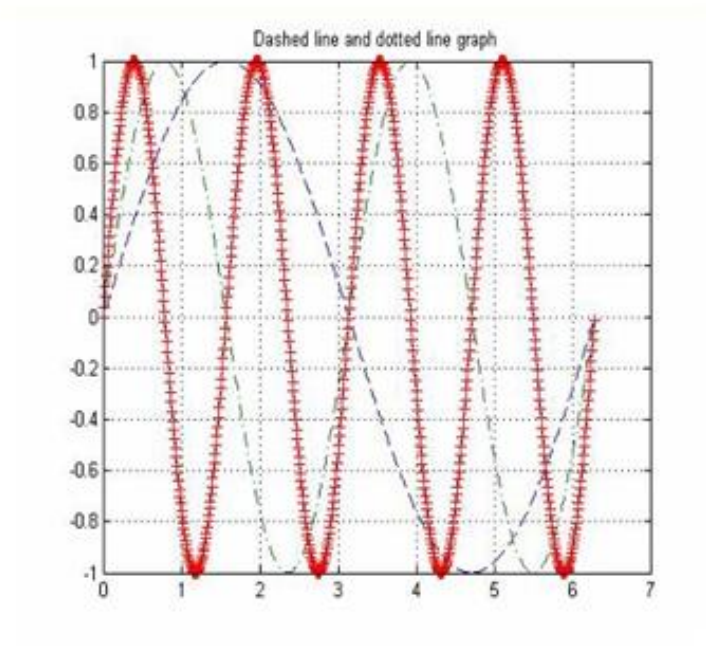


Figure 1-3

Figure 1-4 shows three dimensional graph

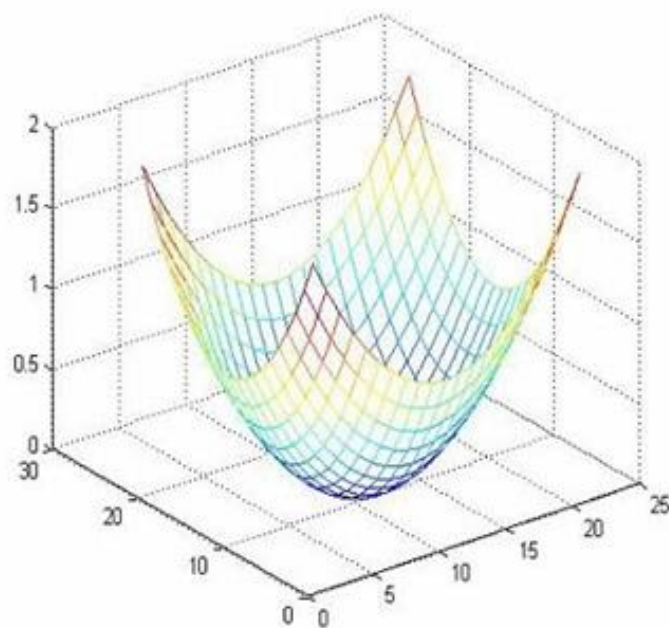


Figure 1-4

### 10.1. Graph functions:

**Table 1-4** describes different graph functions and there description.

functions	Description
Plot(x,y)	Linear plot
Plot(x,y1,x,y2)	Multiple plots on same graph
Mesh(z)	3-D graph
Stem(x)	Discrete plot
X-label('x axis label')	Add x-axis label
Y-label('y axis label')	Add y-axis label
Title('title of plot')	Title of the plot
Subplot(m,n,p)	Divide figure of window
Hold	Hold current graph in the figure
Zoom	Allow zoom in/out using mouse
Pause	Wait for user response

**Table 1-4**

### 11. Partial Fractions:

Partial fractions can easily be solved in matlab.

For example partial fraction for the **Equation (A)** can be found by the following way in the matlab:

$$(s+1)/(s^3+0.75s^2+0.125s)$$

**Equation (A)**

**Matlab Code:**

```
num = [1 1];  
denum= [1 0.75 0.125];  
[res,poles,rem]= residue (num,denum)
```

results are:

```
res = 3 -12 8  
poles = 8 -0.5 -0.25 0  
rem= []
```

Which becomes:

$$4/(s+0.5) - 12/(s+0.25) + 8/s$$

## 12. Impulse and step response:

Impulse and step response can be found in the matlab.

Let we are given **Equation (B)** to find its impulse and step response. The impulse and step response can be found in the matlab by the following Matlab code:

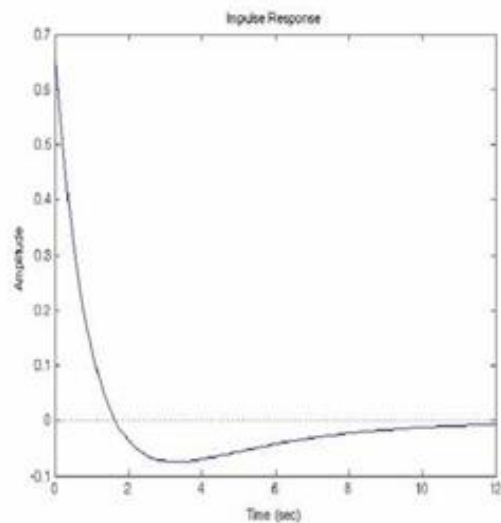
$$T(S) = \frac{2s}{3s^2 + 4s + 1}$$

**Equation (B)**

**Matlab code:**

```
Num= [2 0];  
Denum= [3 4 1];  
Sys= tf(num,denum);  
Impulse(sys)  
Step(sys)
```

**Figure 1-5** shows the impulse response of a system.



**Figure 1-5**

**Figure 1-6** shows the step response of a system

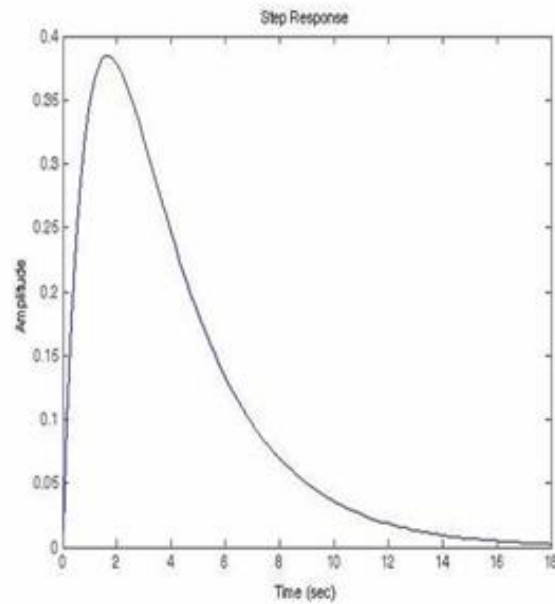


Figure 1-6

## Practice 1.0:

- 1) Use the "help" command of MATLAB to get familiar with the following functions
  - a. roots
  - b. poly
  - c. polyval
  - d. conv
  - e. residue
  - f. tf
  - g. pzmap
  - h. impulse
  - i. step
  - j. series
  - k. parallel
  - l. feedback
- 2) Write the description and give one example of each function
- 3) Show all the results in the report

## Lab 2

### System Response by different inputs in MATLAB and Simulink

- Find an impulse and step response of the following system by using Matlab. Use Simulink to find both responses and compare them with Matlab results. (Hint: impulse, step, Impulse is the derivative of the step)

$$G(s) = \frac{10}{s^2 + 2s + 20}$$

- Also apply the sinusoidal input to the above mentioned system in both Matlab and Simulink. Compare both the results too. Also plot output vs input in both.(Hint: lsim)
- Apply the following input to the system in both simulink and matlab.
  - $\sin(2\pi t) + u(t) + 2u(t-5)$
  - Square input with amplitude equal to 1 and time period equal to 10 seconds. Simulate the system for at least 40 seconds.
  - Combine both of above inputs.

#### Tools:

Matlab is required to perform all the experiments for this particular Lab..

1. Find an impulse and step response of the following system by using Matlab. Use simulink to find both responses and compare them with Matlab results.

$$G(s)=10/s^2+2s+20$$

### Introduction:

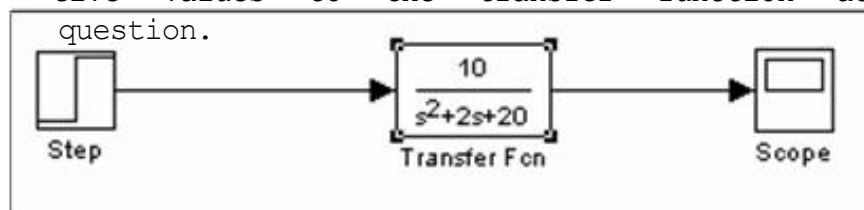
Design two systems in simulink, one for impulse response and one for step response. Then use the given equation to obtain the result using scope.

To find both the responses in Matlab, write the required code and finally compare both the results (of simulink and Matlab).

### Step Response Using Simulink:

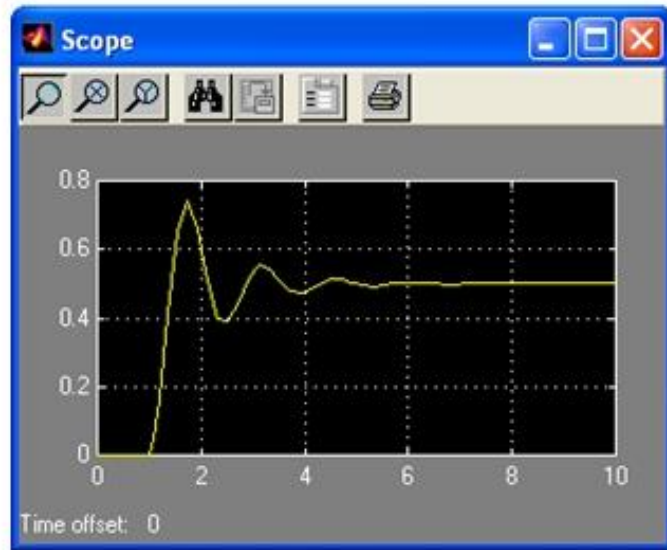
#### Procedure:

- Type "simulink" in Matlab command prompt.
- "Simulink Library Browser" window will appear.
- Create a new file by selecting "model" from the file option(file->New->Model)
- Drag the step function from the simulink library browser to the new file created followed by transfer function and scope. Scope is used to see the result(resulting waveform).
- Give values to the transfer function according to the question.



- After saving followed by simulation ,click on the scope in the figure to see the resulting waveform (graph).
- A window showing the step response will appear





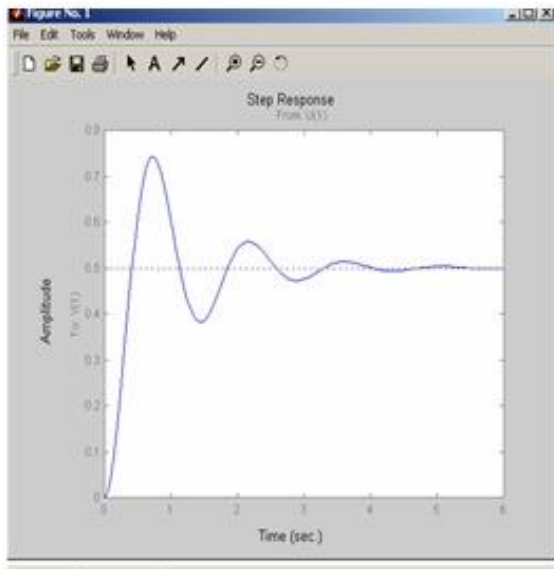
### Step Response Using Matlab:

#### Procedure:

- Create a new M-file(File->New->M-file)
  - Write the Matlab code given below
- ```
num= [10];  
deno= [1 2 20];  
t=tf (num, deno);  
step (t)
```

tf() is used to find the transfer function.

- Save and then run the program. A window showing the step response will appear which will be similar to the one obtained in Simulink as shown below.

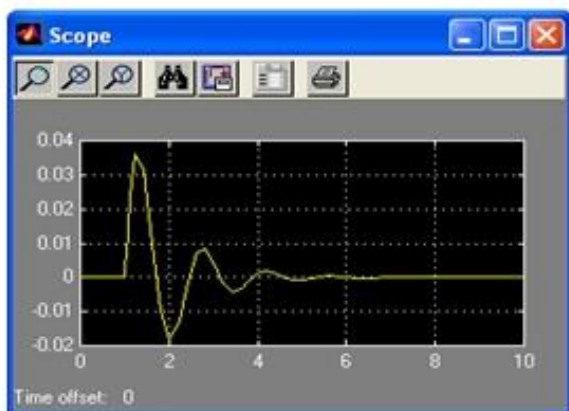


### Impulse Response Using Simulink

Now in order to find the impulse response repeat the same procedure as explained above but provide the output of the step function as input to the derivative as shown in figure:



The result obtained in this case will be



### Impulse Response In Matlab Procedure

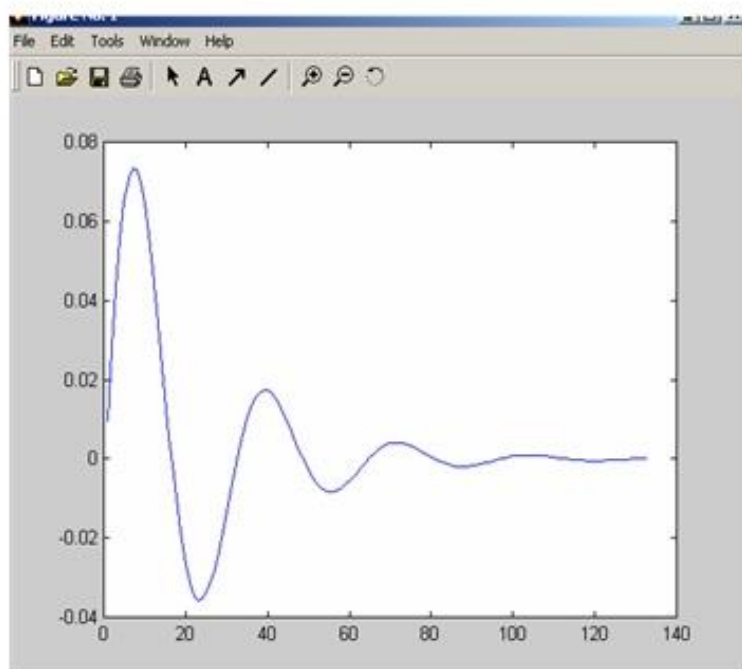
- Create a new M-file (File->New->M-file)

- Write the matlab code given below

```
num= [10];
deno= [1 2 20];
t=tf (num, deno);
step (t)
plot (diff(step(t)))
```

As derivative of step function is an impulse that's why `diff(step(t))` is used in the above code.

The figure obtained will be



### **Analysis:**

Compare the step and impulse responses of Matlab and Simulink.

2. Also apply the sinusoidal input to the above mentioned system in both Matlab and Simulink. Compare both the results too. Also plot output vs. input in both.

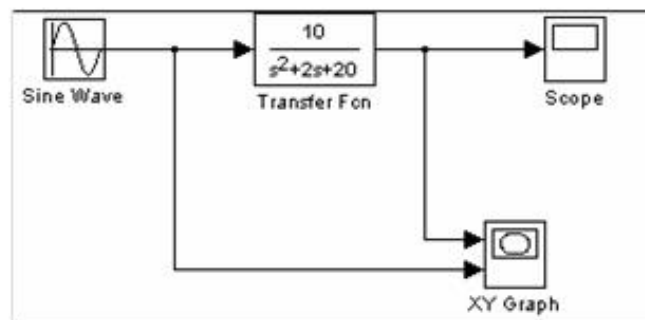
## Introduction:

Design a system in Simulink, using the given equation, and observe(see) the result on the scope. Then write the code for the same system in Matlab and compare its result with that of simulink.

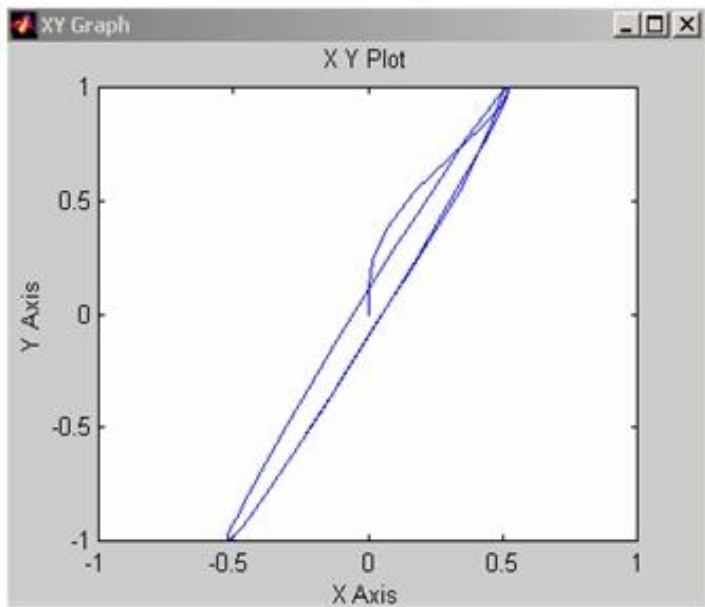
## Simulink:

### Procedure:

- Type "simulink" in Matlab command prompt.
- "Simulink Library Browser" window will appear.
- Create a new file by selecting "model" from the file option(file->New->Model)
- Drag the sine wave function from the simulink library browser to the newly created file followed by transfer function and scope. Scope is used to see the result.
- Give values to the transfer function according to the question.
- In order to plot the output vs. input also drag the "xy plot" from library browser.
- Connections of the components is shown below



- After giving values to the transfer function start simulation and observe the result using scope. The figure obtained will be

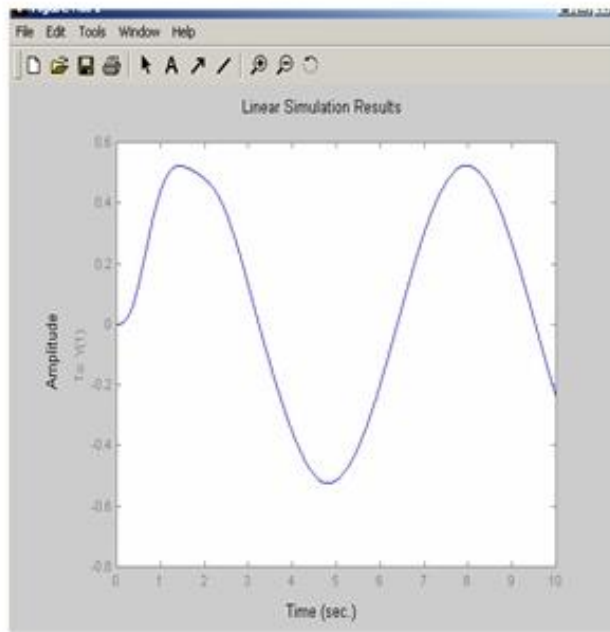


**Matlab:**

In order to find the result in Matlab, the code is

```
num= [10];  
deno= [1 2 20];  
t = 0:0.01:10;  
h=tf (num, deno);  
u=sin (t);  
lsim (h,u,t)
```

the figure obtained is:



### About Lsim:

LSIM Simulate time response of LTI models to arbitrary inputs.

LSIM(SYS,U,T) plots the time response of the LTI model SYS to the

input signal described by U and T. The time vector T consists of

regularly spaced time samples and U is a matrix with as many columns

as inputs and whose i-th row specifies the input value at time T(i).

For example,

```
t = 0:0.01:5;    u = sin(t);    lsim(sys,u,t)
```

simulates the response of a single-input model SYS to the input

$u(t)=\sin(t)$  during 5 seconds.

### Analysis:

Compare the Sinusoidal response of a sinusoidal input of Matlab and Simulink.

3. Apply the following input to the system in both Simulink and Matlab.

$$\sin(2\pi t) + u(t) + 2u(t-5)$$

#### Introduction:

Design a system in Simulink, using the given equation and obtain the result using scope.

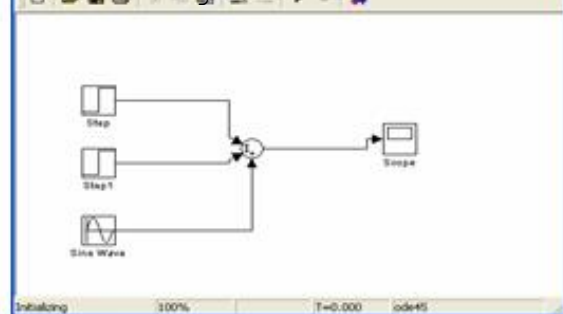
Then write Matlab code and compare its result with the one obtained in Simulink.

#### Simulink

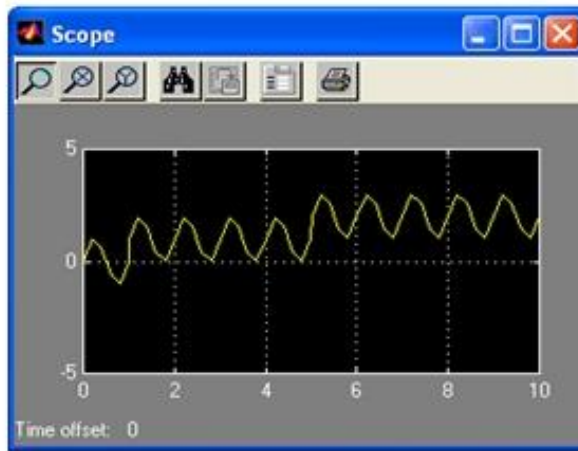
##### Procedure:

- Create a new file and drag the following components to the newly created file
  - o Step input 1; it gives the unit step input (ones and zeros) to the system
  - o Step input 2; it has the step time of 5 with gain of 2.
  - o Sine function with the frequency of  $2\pi$ .
  - o Sum; to add the three components
  - o Scope; it shows the output generated by the system.

All these components should be connected in the way as shown in figure



- Give the initial and final time values of 0 to 10 respectively to the scope.
- After simulation, waveform for the designed system will be



**Matlab:**

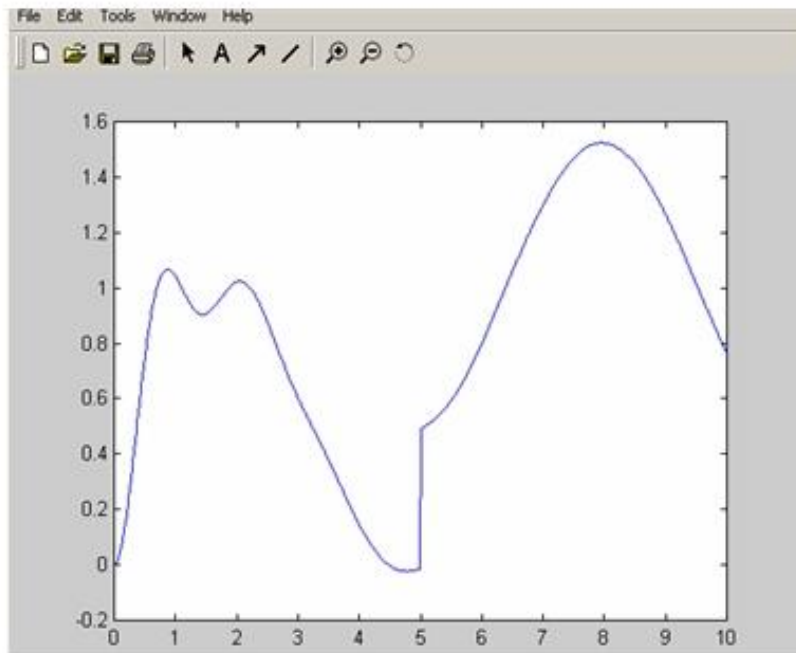
**Procedure:**

- Write the matlab code given below in a newly created M-file.

```
num= [10];  
deno=[1 2 20];  
t = 0:0.01:10;  
h=tf (num, deno);  
y1=lsim (h, u, t);  
y2=step (h, 0:0.01:10);  
y3=step (h, 5:0.01:10);  
Temp=zeros (500, 1);  
y3= [temp y3];  
y=y1+y2+y3;  
plot (t,y)
```

Save and run the program. The figure obtained will be





4. Square input with amplitude equal to 1 and time period equal to 10 seconds. Simulate the system for at least 40 seconds.

#### **Introduction:**

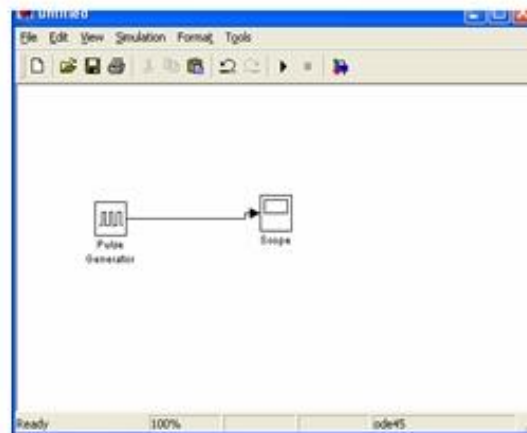
Design a system in Simulink, using the given equation and obtain the result using scope.

Then write Matlab code and compare its result with the one obtained in Simulink.

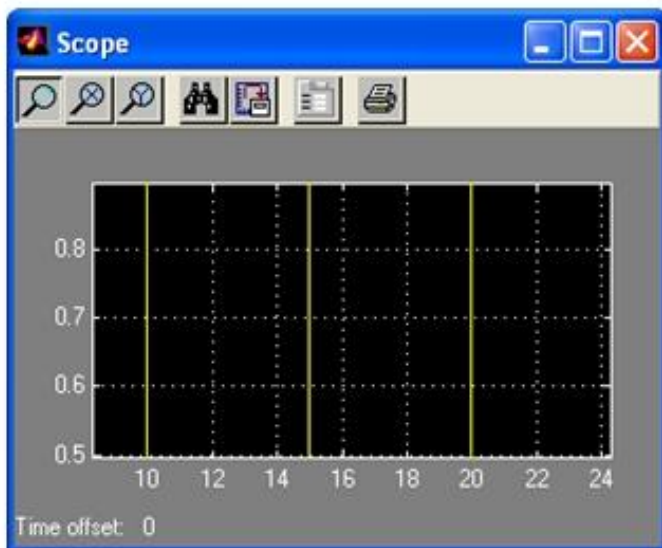
#### **Simulink:**

#### **Procedure:**

- create a new file and select the following components
  - Pulse generator; with the period of 10 and duty cycle of 50.
  - Scope; it shows the waveforms output generated by the system.
- set the value of parameter equal to 40 (as the simulation time is of 40 sec)



- save and start the simulation and double-click on the scope to view the result which is shown below



### Matlab:

#### Procedure:

Write the code given below to obtain the result

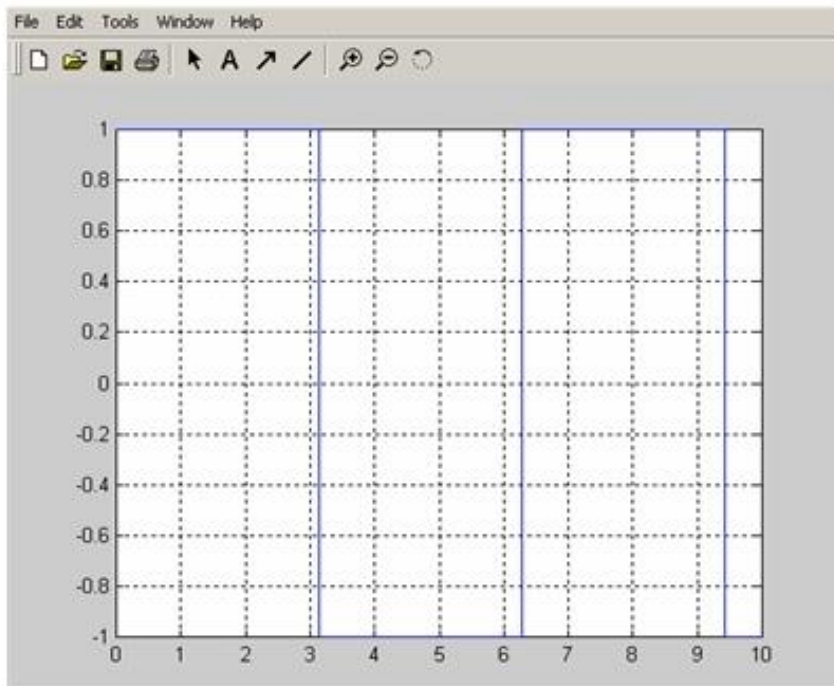
```
num= [10];
deno=[1 2 20];
t = 0:0.01:10;
u=sin (t)
h=tf (num, deno);
y1=lsim (h, u, t);
y2=step (h, 0:0.01:10);
y3=step (h, 5:0.01:10);
temp=zeros (500, 1);
```

```

y3= [temp y3];
y [4] =square(t);
y4';
plot (t,y4)
grid

```

save and run the program and the figure obtained will be



### Analysis:

Compare the response of square input in Matlab and Simulink.

**Q5. Combine both of the inputs of Q#3 and Q#4.**

### Introduction:

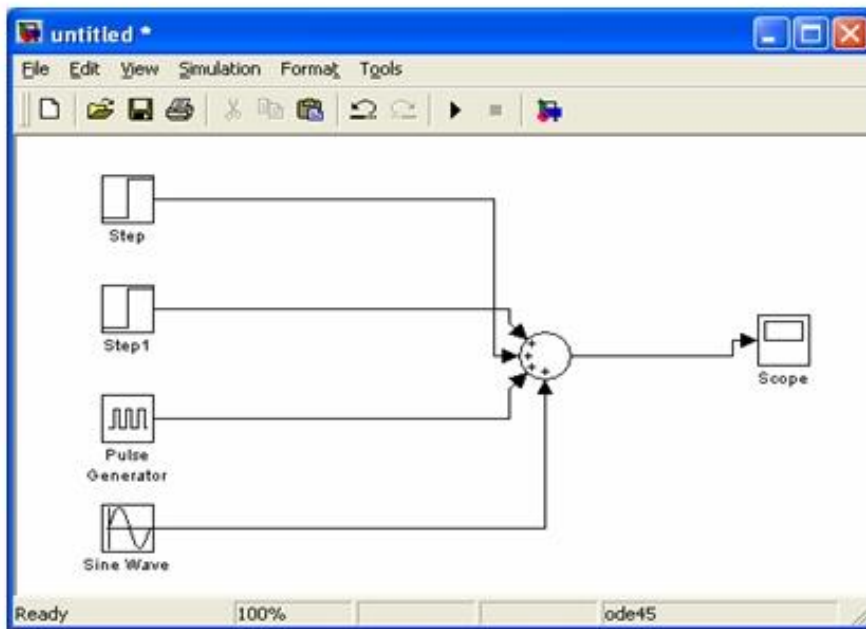
Design a system in Simulink, by combining the inputs of Q#4, 5. Obtain the result on the scope. Then write the code for the same system in Matlab. Result of both Matlab and Simulink is same.

### Simulink:

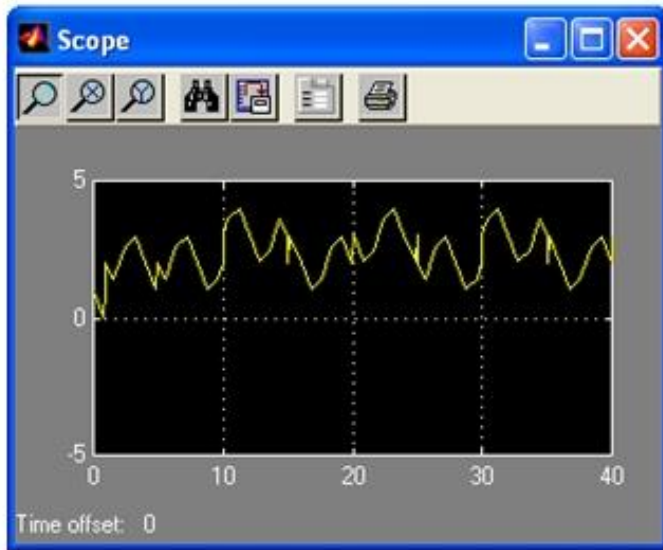
### Procedure:

- create a new file and select the following components

- o Step input 1; it gives the unit step input (ones and zeros) to the system
  - o Step input 2; it has the step time of 5 with gain of 2.
  - o Sine function with the frequency of  $2\pi$ .
  - o Scope; it shows the waveforms output generated by the system.
  - o Pulse generator; with the period of 10 and duty cycle of 50.
  - o Sum; it added the 4 functions.
  - o Scope; it shows the waveforms output generated by the system.
- Set the value of parameter equal to 40.
- Connect the components in the following manner:



- After simulation the output will be:



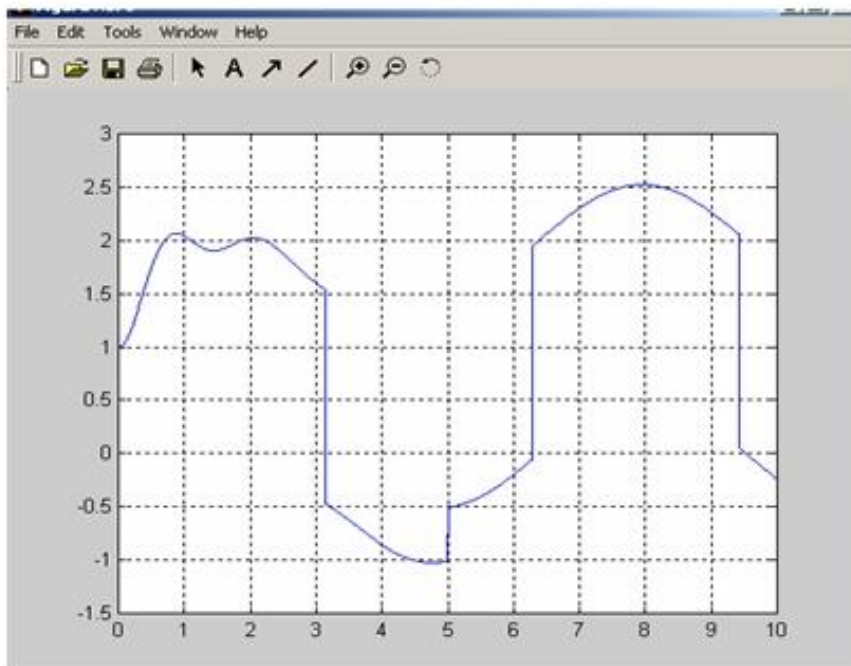
### Matlab:

#### Procedure:

- Write the following code to obtain the result

```
deno=[1 2 20];
t = 0:0.01:10;
u=sin (t)
h=tf (num, deno);
y1=lsim (h, u, t);
y2=step (h, 0:0.01:10);
y3=step (h, 5:0.01:10);
temp=zeros (500, 1);
y3= [temp y3];
y [4]=square(t);
y4';
y=y1+y2+y3+y4';
plot (t,y)
grid
```

after running the above code the figure obtained will be



**Analysis:**

- Compare the response of sinusoidal input, step input, delayed step input and a square wave in Matlab and Simulink.

# Lab 3A

## Simulation of Differential Equations using MATLAB

Simulate the below mentioned system which is represented by differential equations in Matlab. [Hint: ode23 or ode45]

$$\begin{aligned} y_1'(0) &= 0 & y_1' &= y_2 y_3 \\ y_2'(0) &= 1 & y_2' &= -y_2 y_3 \\ y_3'(0) &= 1 & y_3' &= -0.51 y_1 y_3 \end{aligned}$$

Simulate the below mentioned system which is represented by differential equations in Matlab. [Hint: ode23 or ode45]

$$y_1'' + y_{12} y_1 - y_1' + y_1 = 0$$

### INTRODUCTION:

This lab is about solving differential equation using Matlab. Two functions can be used for solving differential equations. One is the ODE23 and the other is ODE45. Either of the two can be used. A function is written in an M-file according to the given system. That function is called from command window using the ODE23 or ODE45 syntax. **[T, Y]**

**= ODE45 ('Func', TSPAN, Y0).** The function should return a column vector which is stored in Y. "Func" is the name of the function made in Matlab M-file. TSPAN = [T0 TFINAL] integrates the system of differential equations  $y' = F(t, y)$  from time T0 to TFINAL with initial conditions Y0. Each row in solution array Y corresponds to a time returned in column vector T.

### TASK 01:

Simulate the below mentioned system which is represented by differential equations in Matlab. [Hint: ode23 or ode45]

$$\begin{aligned} y_1'(0) &= 0 & y_1' &= y_2 y_3 \\ y_2'(0) &= 1 & y_2' &= -y_2 y_3 \\ y_3'(0) &= 1 & y_3' &= -0.51 y_1 y_3 \end{aligned}$$

**TOOL USED: MATLAB**

**PROCEDURE:**

- 1-1. Open Matlab command window. Go to the file menu and open Matlab M-file.
- 1-2. Code the function named "func" as given below.

**CODE:**

```
function dy = func(t,y);  
dy=zeros(3,1);  
dy(1)=y(2)*y(3);  
dy(2)=-y(3)*y(1);  
dy(3)=-0.51*y(2)*y(1);
```

- 1-3. Save the file with the of "func".
- 1-4. Go to command window and call the function using ODE23 as given below.
- 1-5. Plot y.

**IN COMMAN WINDOW:**

```
[t,y]= ode23('func',[0 40],[0 1 1])  
plot(t,y)
```

**FIGURE:**

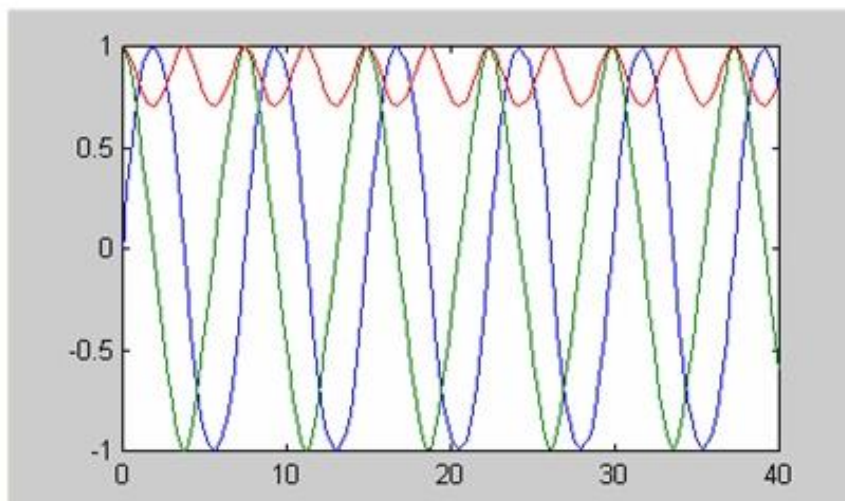


fig 1-1

**TASK 02:**



Simulate the below mentioned system which is represented by differential equations in Matlab. [Hint: ode23 or ode45]

$$y_1'' + y_1 y_2 - y_1' + y_1 = 0$$

**TOOL USED: MATLAB**

**PROCEDURE:**

**2-1.** Open Matlab command window. Go to the file menu and open Matlab M-file.

**2-2.** Code the function named "func1" as given below.

**CODE:**

```
function dy = func1(t,y);  
dy=zeros(2,1);  
dy(1)=y(1);  
dy(2)=-1*y(1)*y(1)*y(2)-y(1)+y(2);
```

**2-3.** Save the file with the of "func1".

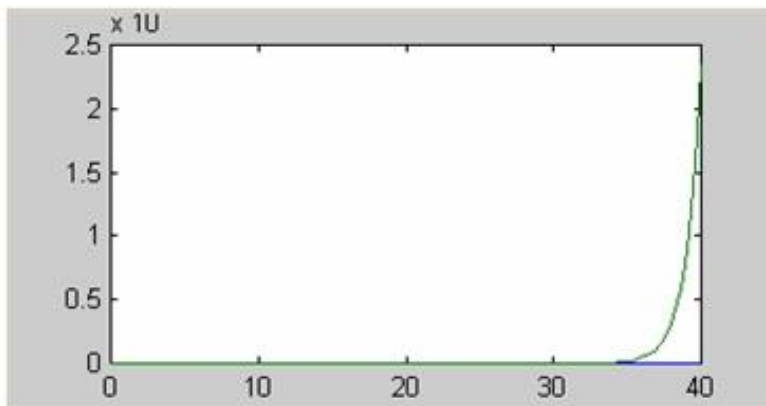
**2-4.** Go to command window and call the function using ODE23 as given below.

**2-5.** Plot y.

**IN COMMAND WINDOW:**

```
[t,y]= ode23('func1',[0 40],[0 1])  
plot(t,y)
```

**FIGURE:**



**fig 2-1**

## Lab 3B

# Simulation of Differential Equations using Simulink

Simulate the below mentioned system which is represented by differential equations in Simulink. [Hint: ode23 or ode45]

$$\begin{aligned}y_1'(0) &= 0 & y_1' &= y_2 y_3 \\ y_2'(0) &= 1 & y_2' &= -y_2 y_3 \\ y_3'(0) &= 1 & y_3' &= -0.51 y_1 y_3\end{aligned}$$

Simulate the below mentioned system which is represented by differential equations in Simulink. [Hint: ode23 or ode45]

$$y_1'' + y_1^2 y_1' - y_1' + y_1 = 0$$

### Introduction:

The following system in the form of ordinary differential equation is given.

$$y_1'' + y_1^2 y_1' - y_1' + y_1 = 0$$

This equation was simulated with Matlab and then with Simulink. The system response was generated using both the tools, and then studied separately. The results of both the tools were then compared for the verification of the process.

### Tools and technologies used:

For this experiment only the simulation was meant, therefore the most powerful simulation tool in engineering studies.

1. **Matlab** and the embedded software in Matlab
2. **Simulink** was used.

### Procedure 1:

- 1.1 The given ordinary differential equation was analyzed manually on a rough page to get two separate equations.
- 1.2 The “ode23” and “ode45” functions of Matlab were studied along with their behavior using the **Matlab help**.
- 1.3 The equations that were derived in the first step were coded in Matlab specific syntax in an M- file.
- 1.4 Another M- file was created and the function “ode23” was used in it and then the result was plotted.
- 1.5 The resulting Matlab code is following.

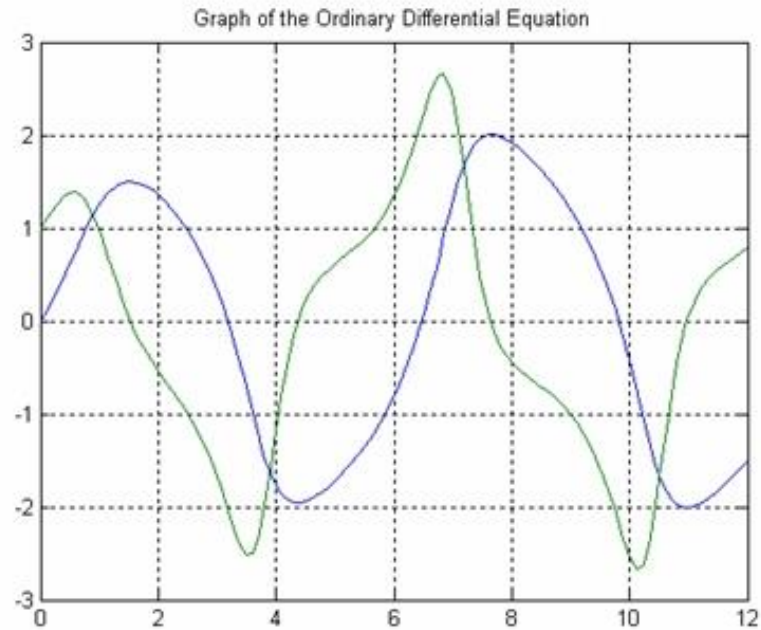
```
function dy = rigid(t,y)
dy = zeros(2, 1);
dy (1) = y(2);
dy (2) = y(2)*(1-(y(1)*y(1)))-y(1);
```

**Matlab code for the equations**

```
[t,y] = ode23('rigid',[0 12],[0 1]);
plot(t,y);
grid
```

**Matlab code for the “ode23” function**

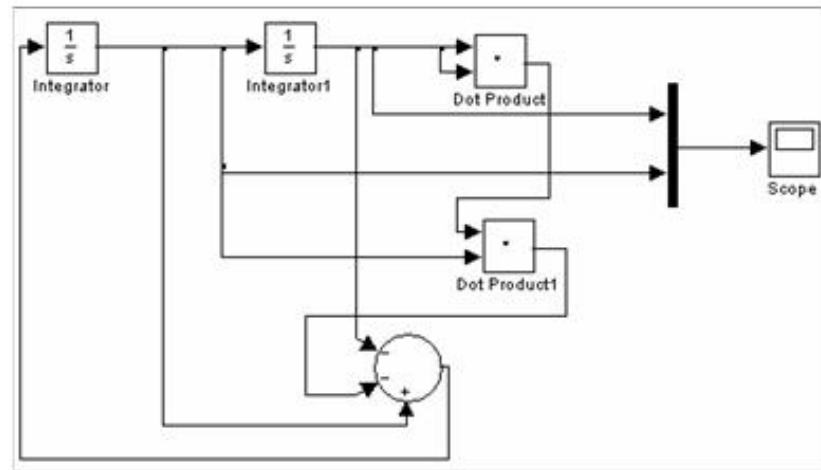
- 1.6 The above code, when simulated in Matlab gave the following results.



**Fig. no 1**  
**Matlab result of the simulated D.E**

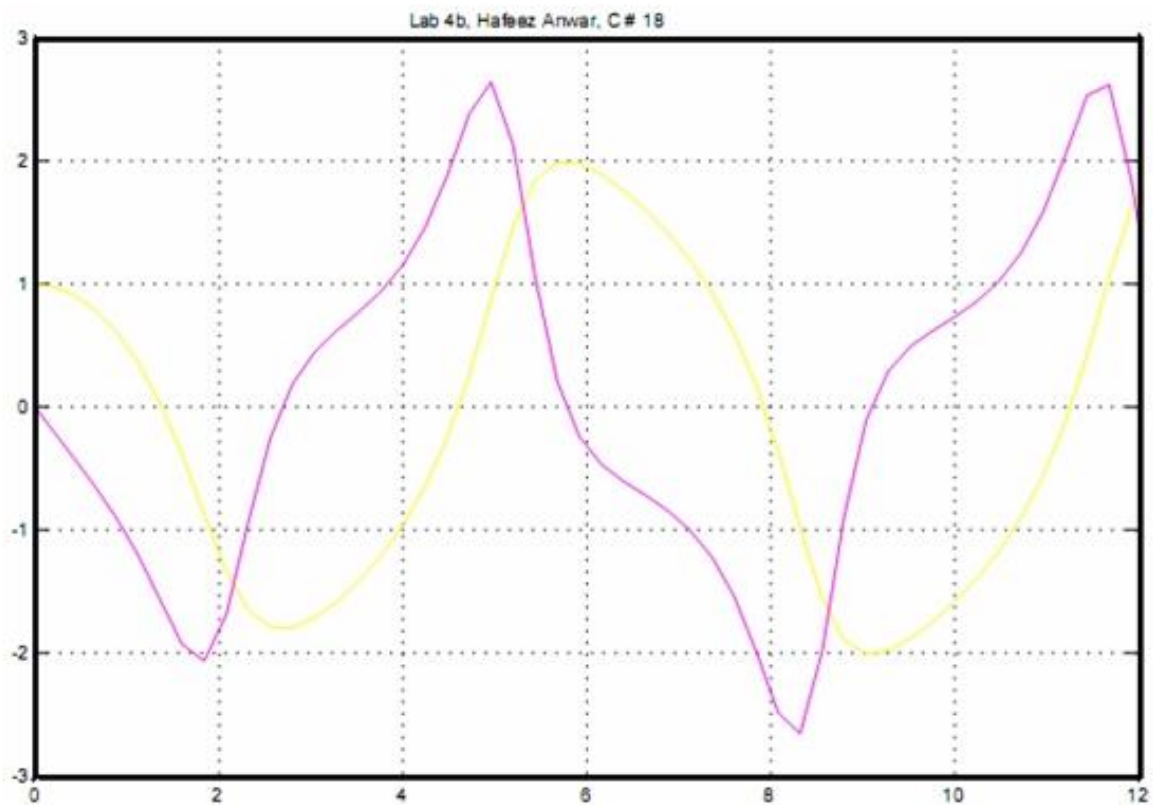
**Procedure 2:**

- 2.1 The given ordinary differential equation was analyzed and then solved manually on a rough to present it in integral form.
- 2.2 In Simulink a new file was created and necessary components were dragged and dropped from the library which are; Integrator, summation, dot product, scope and mux.
- 2.3 All of these components were interconnected to form the manually derived equation.
- 2.4 The model created was run and the result was studied using the scope.
- 2.5 The resulting model is following.



**Fig. no2**  
**Simulink model for the D.E**

2.6 The simulation resulted in the following graph on the scope.



**Fig. no 3**  
**Simulation result of Simulink**

**Analysis:**

1. The result obtained due to the Matlab code was observed.
2. The green curve is of the variable 'y' and the blue curve is of the derivative of 'y'.
3. It can be seen that the initially when 'y' is 1 its derivative is 0, and for the other values similarly the derivative is calculated.
4. The result obtained due to simulation in Simulink showed a similar kind of result.
5. The variable 'y' and its derivative showed the similar behavior.
6. At the end both the results were compared and were found similar to each other.

# Lab 4

## Simulation of Higher Order Differential Equations using MATLAB and Simulink

Simulate the below mentioned system which is represented by differential Equation using Matlab.

$$\frac{d^5 y}{dt^5} + 2 \frac{d^4 y}{dt^4} + 24 \frac{d^3 y}{dt^3} + 48 \frac{d^2 y}{dt^2} + 24 \frac{dy}{dt} + 20y + 10 = 0$$
$$y(0) = 2, y'(0) = 5, y''(0) = 10, y'''(0) = -4, y^{iv}(0) = -7$$

Also simulate it in Simulink and match the results.

### Intoduction:

Matlab have thousands of mathematical tools with the help of which we can solve mathematical problems and realize its output.

Same is the case here, in the above problem we are given a system represented by differential equation, to realize in matlab and simulink. Basically derivative is a change in a value with respect to another variable.

**Matlab command used:**

---

**[t,Y] = ode45(fun,tspan,y0)**

**ode:** Solve initial value problems for ordinary differential equations (ODEs) particularly here ode45 is used which solves higher order derivaties.

**Fun:** This function here used is user define, it means that own function can be built in matlab and then using [t,Y] = ode45(fun,tspan,y0) in command window can be called, But function name and saved file name should be same.

**tspan:** A vector specifying the interval of integration, [t0,tf]. The solver imposes the initial conditions at tspan(1), and integrates from tspan(1) to tspan(end). To obtain solutions at specific times (all increasing or all decreasing), use tspan = [t0,t1,...,tf]. For tspan vectors

with two elements  $[t_0 \ t_f]$ , the solver returns the solution evaluated at every integration step. For  $tspan$  vectors with more than two elements, the solver returns solutions evaluated at the given time points. The time values must be in order, either increasing or decreasing.

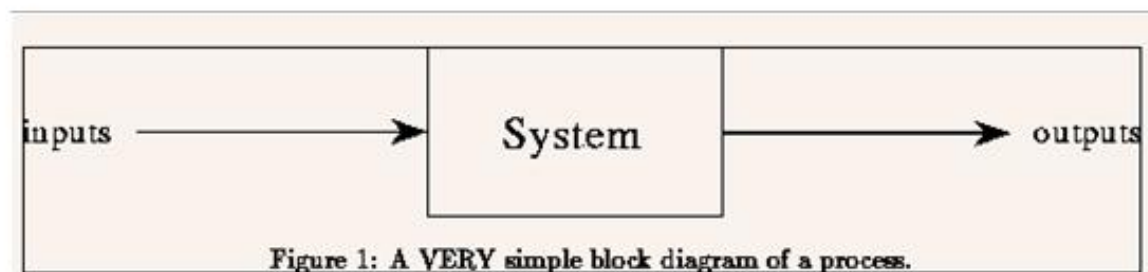
**y0:**

A vector of initial conditions.

**SIMULINK :**

is an icon-driven state of the art dynamic simulation package that allows the user to specify a block diagram representation of a dynamic process. Assorted sections of the block diagram are represented by icons which are available via various "windows" that the user opens (through double clicking on the icon). The block diagram is composed of icons representing different sections of the process (inputs, state-space models, transfer functions, outputs, etc.) and connections between the icons (which are made by "drawing" a line connecting the icons). Once the block diagram is "built", one has to specify the parameters in the various blocks, for example the gain of a transfer function. Once these parameters are specified, then the user has to set the integration method (of the dynamic equations), stepsize, start and end times of the integration, etc. in the simulation menu of the block diagram window.

A simple block diagram representation in simulink, for a single input and output is given below:



Now the solution of the given equation using the tools explained is given below:



### Part (a)

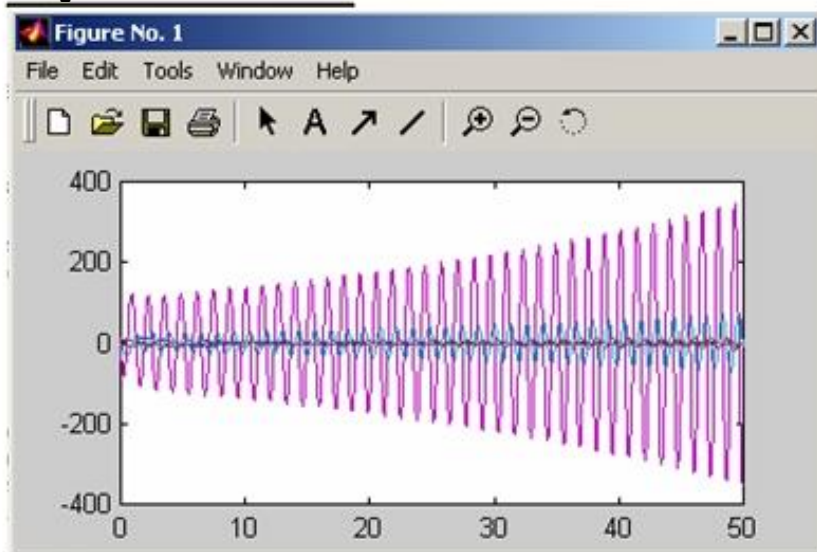
#### Matlab code

```
function dy1 =osama(t,y);  
dy1=zeros(5,1);  
dy1(1)=y(2);  
dy1(2)=y(3);  
dy1(3)=y(4);  
dy1(4)=y(5);  
dy1(5)=-2*y(5)-24*y(4)-48*y(3)-24*y(2)-20*y(1)-10;
```

Now save this file as explained earlier with function name osama.m, then write

`[t,y]=ode45('osama',[0 50],[2 5 10 -4 -7])` in command window in matlab, then write `plot(t,y)` which will show the result in a graph, as follows :

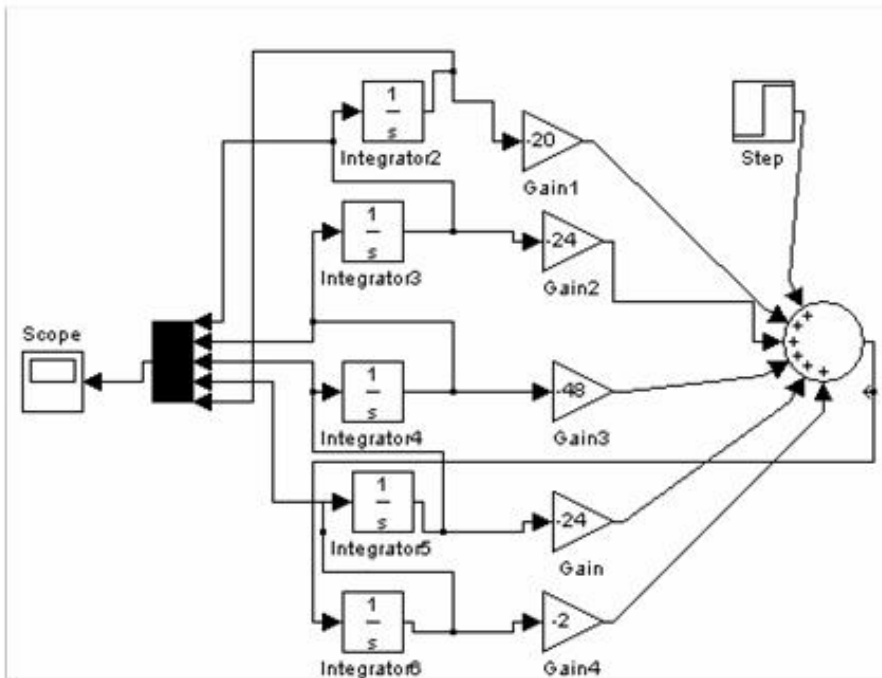
#### Output in Matlab :



As simulink is explained earlier the given equation can be solved in simulink as given :

### Part (b)

#### Simulink Diagram



As we can see a few common blocks r used :

**Integrator** : It simply integrates the value given for example if give  $Y'$  to the integrator it give us  $y$ .

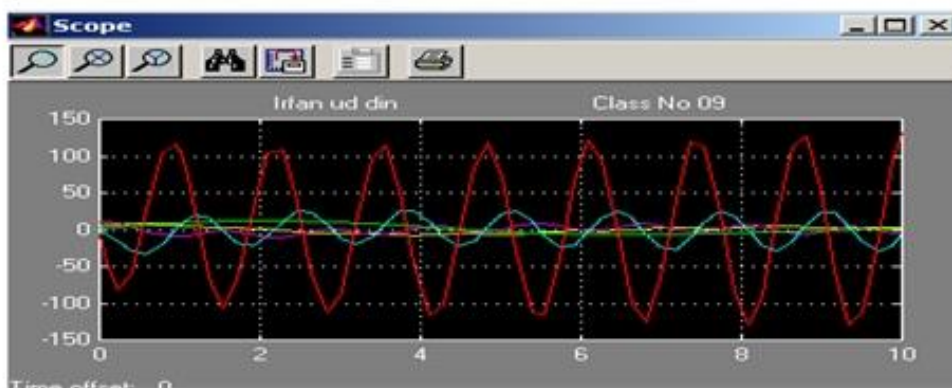
**Gain** : It is nothing but the constant value with an equation if it is 10 simply put 10 in gain block in simulink.

**Scope** : It shows us the graphical of the whole block diagram.

Summation :

As the equation given is divided into parts to integrate it, so reverse addition is done through summation.

### **Simulink Output result**



## Lab 5

### Frequency Domain Modeling in MATLAB

The objectives of this lab is to find the Laplace and Inverse Laplace transforms using MATLAB

**5.1 Use the MATLAB and Control System Toolbox to form a linear time invariant system transfer function**

$$F(s) = \frac{2}{(s+1)(s+2)^2}$$

**MATLAB Code:**

```
F=zpk([], [-1 -2 -2], 2)
```

**5.2 Use the MATLAB to get the equation**

$$f(t) = 2e^{-t} - 2te^{-2t} - 2e^{-2t}$$

**MATLAB Code:**

```
numf=2;  
denf=poly([-1 -2 -2]);  
[k,p,k]=residue...  
(numf,denf)
```

**5.3 Use the MATLAB and Control System Toolbox to form a linear time invariant system transfer function**

$$F(s) = \frac{3}{s(s^2 + 2s + 5)}$$

**MATLAB Code:**

```
F=tf([3], [1 2 5 0])
```

**5.4 Use the MATLAB to find the inverse laplace transform of the system transfer function**

$$F(s) = \frac{3}{s(s^2 + 2s + 5)}$$

**MATLAB Code:**

```
syms s
f=ilaplace...
(3/(s*(s^2+2*s+5)));
pretty(f)
```

5.5 Use MATLAB to get the following equation

$$F(s) = \frac{3/5}{s} - \frac{3}{20} \left( \frac{2+j1}{s+1+j2} + \frac{2-j1}{s+1-j2} \right)$$

**MATLAB Code:**

```
numf=3
denf=[1 2 5 0]
[k,p,k]=residue...
(numf,denf)
```

5.6 Use the MATLAB to find the inverse laplace transform of the system transfer function

$$C(s) = R(s)G(s) = \frac{1}{s(s+2)}$$

**MATLAB Code:**

```
syms s
C=1/(s*(s+2))
C=ilaplace(C)
```

5.7 Use MATLAB to plot the following function for t from 0 to 1 with the intervals of 0.01

$$c(t) = \frac{1}{2} - \frac{1}{2}e^{-2t}$$

**MATLAB Code:**

```
t=0:0.01:1;
plot...
(t,(1/2-1/2*exp(-2*t)))
```

5.8 Use MATLAB and Symbolic Math Toolbox to help you solve the following equation for currents.

$$\begin{aligned}
 &+(2s+2)I_1(s) - (2s+1)I_2(s) - I_3(s) = V(s) \\
 &-(2s+1)I_1(s) + (9s+1)I_2(s) - 4sI_3(s) = 0 \\
 &-I_1(s) - 4sI_2(s) + \left(4s+1+\frac{1}{s}\right)I_3(s) = 0
 \end{aligned}$$

**MATLAB Code:**

```

syms s I1 I2 I3 V
A=[(2*s+2) -(2*s+1)...
   -1
   -(2*s+1) (9*s+1)...
   -4*s
   -1 -4*s...
   (4*s+1+1/s)];
B=[I1;I2;I3];
C=[V;0;0];
B=inv(A)*C;
pretty(B)

```

## Lab 6

# Time Domain Modeling in MATLAB

The objective of this lab is to model the system in time domain. State Space representation is one of the unified method for modeling, analyzing and designing a wide range of systems.

6.1 Use the MATLAB code to form an LTI state space representation from the transfer function. The matrix A, B and C are shown below.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -24 & -26 & -9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} r$$

$$y = [b_0 \quad b_1 \quad b_2] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [2 \quad 7 \quad 1] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

**MATLAB Code:**

```
num=[1 7 2];
den=[1 9 26 24];
[A,B,C,D]=tf2ss...
(num,den);
P=[0 0 1;0 1 0;1 0 0];
A=inv(P)*A*P
B=inv(P)*B
C=C*P
```

6.2 Use the MATLAB to convert the state space representation to the transfer function for the following

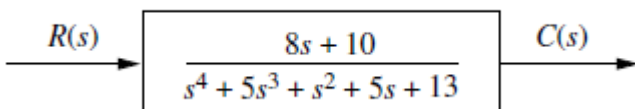
$$\dot{\mathbf{x}} = \begin{bmatrix} -4 & -1.5 \\ 4 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} u(t)$$

$$y = [1.5 \quad 0.625] \mathbf{x}$$

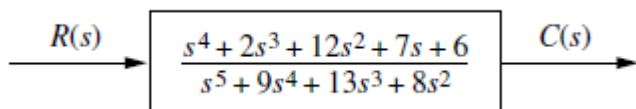
MATLAB Code:

```
A=[-4 -1.5; 4 0];
B=[2 0]';
C=[1.5 0.625];
D=0;
T=ss(A,B,C,D);
T=tf(T)
```

6.3 Write the MATLAB code for the conversion of transfer function to the state space representation of the following system



6.4 Write the MATLAB code for the conversion of transfer function to the state space representation of the following system



6.5 Write the MATLAB code for the conversion of state space representation to

the transfer function for the following

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 5 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -7 & -9 & -2 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 5 \\ 8 \\ 2 \end{bmatrix} r$$
$$y = [1 \quad 3 \quad 6 \quad 6] \mathbf{x}$$

6.6 Write the MATLAB code for the conversion of state space representation to the transfer function for the following

$$\dot{\mathbf{x}} = \begin{bmatrix} 3 & 1 & 0 & 4 & -2 \\ -3 & 5 & -5 & 2 & -1 \\ 0 & 1 & -1 & 2 & 8 \\ -7 & 6 & -3 & -4 & 0 \\ -6 & 0 & 4 & -3 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 2 \\ 7 \\ 8 \\ 5 \\ 4 \end{bmatrix} r$$
$$y = [1 \quad -2 \quad -9 \quad 7 \quad 6] \mathbf{x}$$



## Lab 7

# System Stability in Simulink

- Design a system in Simulink whose transfer function is as given below.

$$G(s) = \frac{100}{s^2 + 4s + 50}$$

The input to the system is Unit Step for 8 seconds and output should be shown graphically.

- Design 3 system with complex poles such that first is stable, second is unstable and third is marginally stable. Show the results graphically.

### Introduction:

This practical LAB work is quite straight forward, introducing the idea of how to design a system using the complex Laplace equations, and than by studying the graphical view of the system using Simulink, one can study different aspects and behaviors of a system. Basically a system has to be examined for its stability that becomes easier using graphical methods to construct a system, and to graphically show the output of the system. The singularity of the system can also be plotted using Matlab command window.

### Design:

The Simulink uses graphical method to deal with the complexities of a system. A simple design is to be constructed using the following Laplace equation:  
 $F(s) = 100 / (s^2 + 4s + 50)$

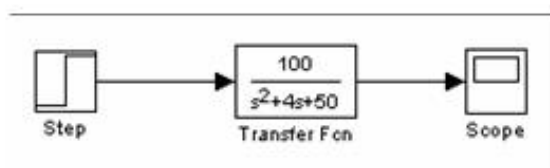
In order to design such a system which is already transformed to Laplace equation, a new file should be

created in Simulink, the file should then be saved with any given name. Now to show the equation's behavior graphically, the continuous option is to be selected from source library, which contains a number of tools that deals with the time dependent continuous equations. Now the transfer function block in continuous library is to be dragged to that file window. This block initially contains a transfer function which is predefined. The transfer function can be changed according to the user's requirement, by double clicking the block. A window will appear which contains numerator part. A value of 100 needs to be inserted here as given in the equation above. In the denominator part, a vector of three terms i.e. coefficient of square of 's' which is '1', coefficient of 's' which is '4', then the constant '50' as given above. After clicking 'ok' required transfer function block is ready for analysis.

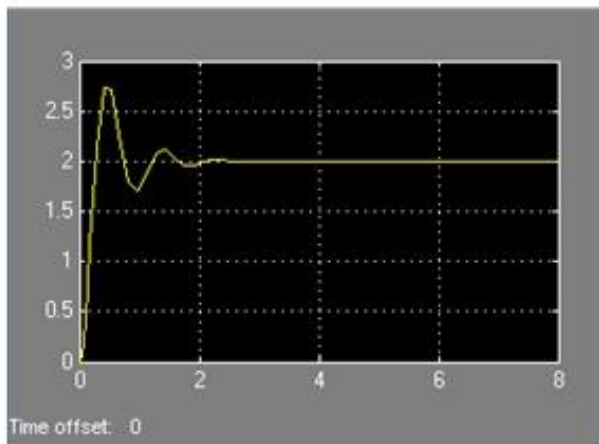
After the design of transfer function, an input should be given to this system which is suppose a unit step signal whose simulation duration should be kept to 8 seconds. Unit step block can be found in the sink library by just clicking the sink and then by dragging the 'step' block to the Simulink file. The simulation duration can be change within the Simulink file.

Now that the system is designed using transfer function and unit step input is applied to the system, it is now necessary to examine the system using the scope block, which can found in the 'source' library. By connecting the scope block to the system and running the simulation by clicking 'start' in the simulation menu, a graph appears shows the system response to the unit step input.

### Simulink Block Design:



### Scope Result:



## Design of three systems in Matlab and Simulink

Three systems stable, marginally stable, and unstable are to be implemented and then it has to be confirmed graphically. The complex poles of the system are figured out using the following Matlab command:

```
[z p k]=residue(num,denum)
```

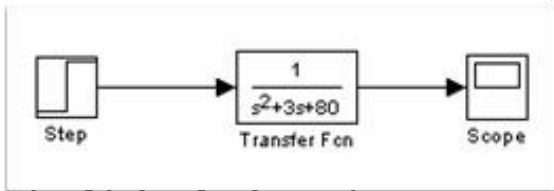
Where 'p' is/are the complex pole/poles of the system, 'z' is/are the complex zero/zeros of the system, and k is the constant generated while calculating the poles and zeros.

### Extra work:

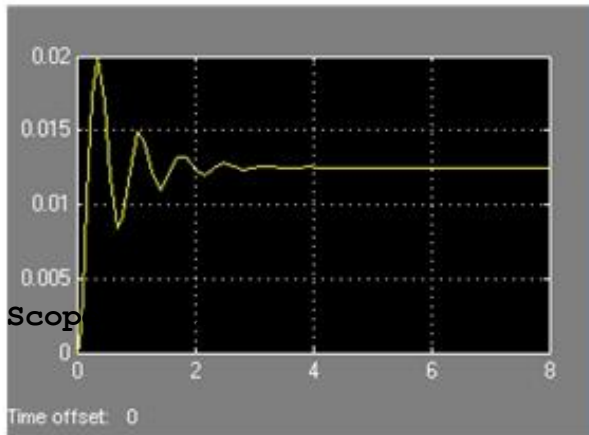
Beside this the poles and zeros of a system can also be shown graphically using the function `zplane(p)`, where p is/are the pole/poles of the system.

### Stable System:

$$100/(s^2+3s+80)$$



**Simulink Block Design:**



### **Matlab Coding:**

```

num=[1 1];
denum=[1 3 80];
[z p k]=tf2zp(num,denum)
zplane(z,p);

```

Also:

```

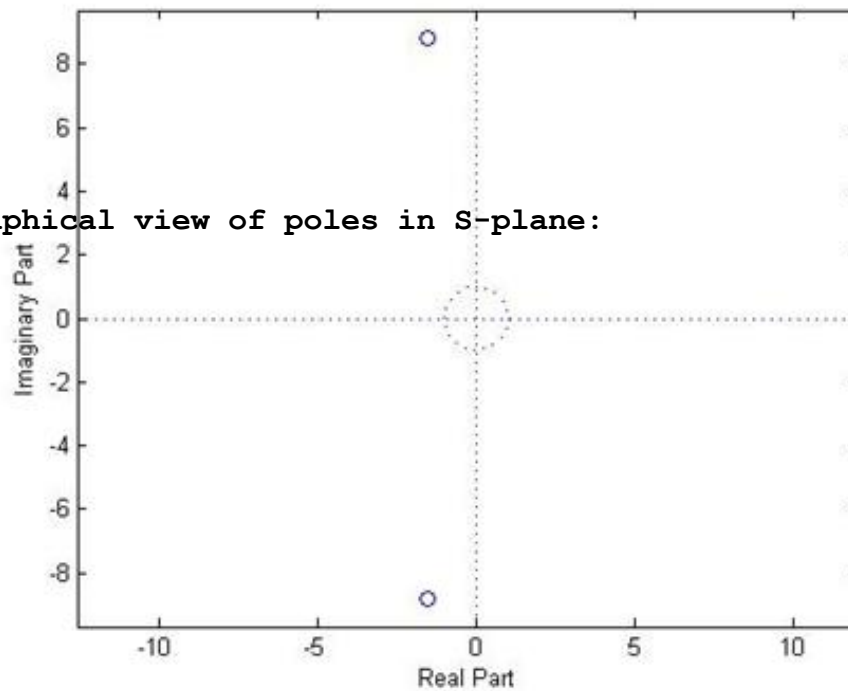
[A,B,C,D] = zp2tf(z,p,k)

```

### Code Result:

Poles:  $-1.5+8.817i$ ,  $-1.5+8.817i$   
Zero(s): At infinity  
K: 0

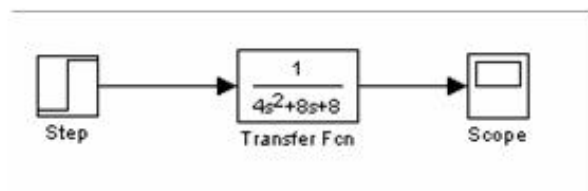
Graphical view of poles in S-plane:



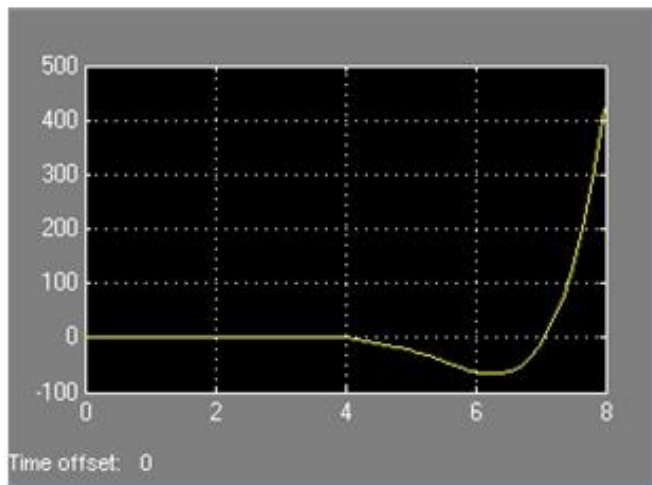
### Unstable System:

$$100/(3s^2-8s+8)$$

### Simulink Block Design:



### Scope Result:



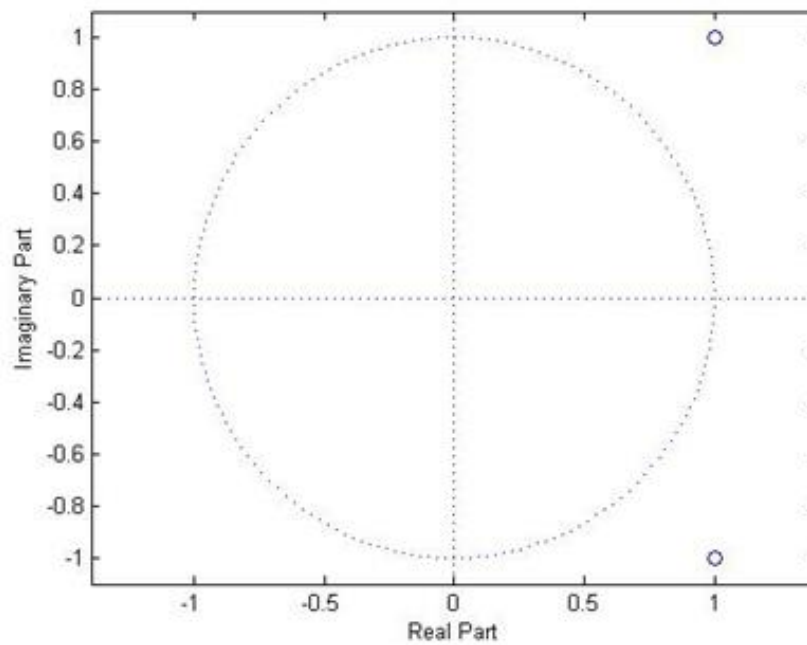
### Matlab Coding:

```
num=[1];  
denum=[4 -8 8];  
[z p k]=residue(num,denum)  
zplane(p);
```

### Code Result

Poles:  $1+i$ ,  $1-i$   
Zeros: At infinity  
K: 0

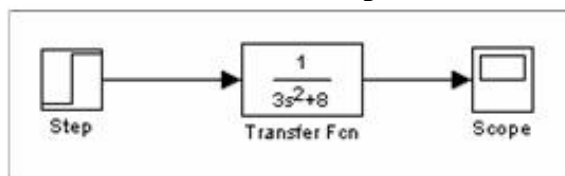
**Graphical view of poles in S-plane:**



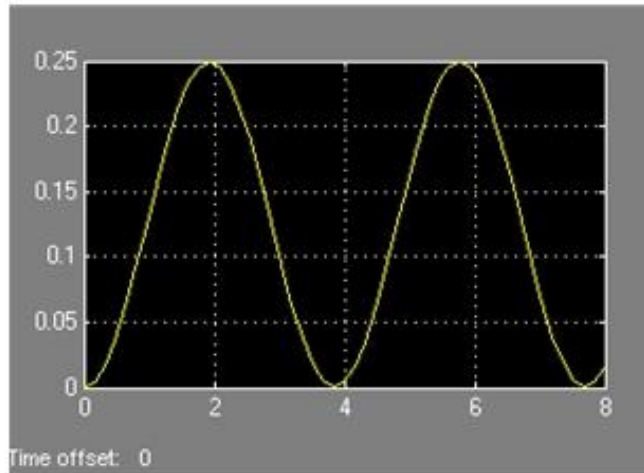
**Marginally Stable System:**

$$1/(3s^2+8)$$

**Simulink Block Design:**



### Scope Result:



### Matlab Coding:

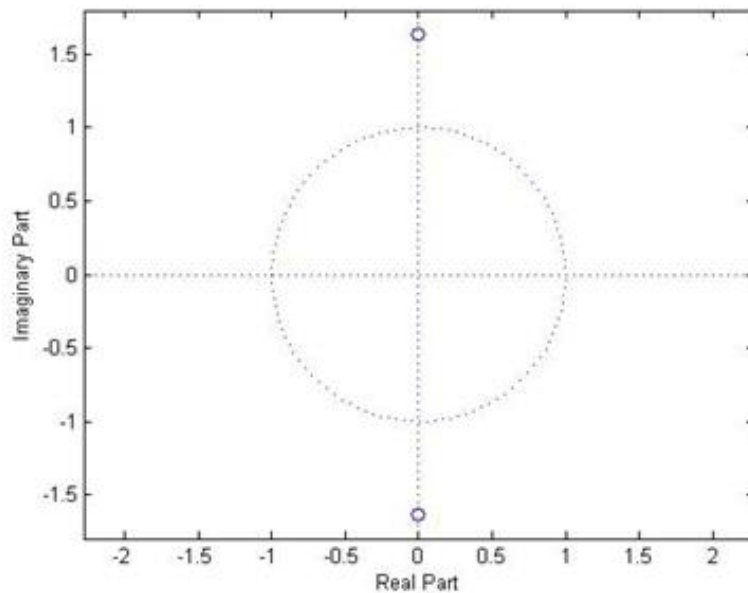
```
num=[1];  
denum=[3 0 8];  
[z p k]=residue(num,denum)  
zplane(p);
```

### Code Result:

```
Poles: 0+1.6330i, 0+1.6330i  
Zeros: At infinity  
K: 0
```



### Graphical view of poles:



### Conclusion:

The main behind this practical is that whenever a system is encountered whose denominator is a quadratic equation over 's' which is unfactorizable will have a pair of complex conjugate poles if:

- The coefficient of square of 's' and constant both are negative.
- The coefficient of 's' is negative or positive.

For the rest of the conditions the poles will be real.

Now the main thing is to find the stability of the system which is explained in the following points:

- A system is stable if the coefficient of 's' is positive.
  - If any of the term (either coefficients or constants) is negative, the system becomes unstable.
  - If there is no constant term, so one pole would always be at zero.
  - If there is no 's' term i.e. coefficient of 's' is zero, so it now depends on sign of square term and the constant.
- 
- The given system is marginally stable if the constant and square term both is either positive or both negative.
  - The system will have two poles equal in magnitude but opposite in sign if either constant or square term is negative.

## Lab 8

# System Interconnections in Simulink

### Series System Interconnection

Find the equivalent system of the following systems connected in series. Prove it using Simulink.

$$G1(s) = 1/(s+1);$$

$$G2(s) = 1/(s+4);$$

$$G3(s) = (s+3)/(s+5)$$

**Steps:**

**Input: Step function**

Step function is used as an input to the cascaded systems. Step function block is in sources library.



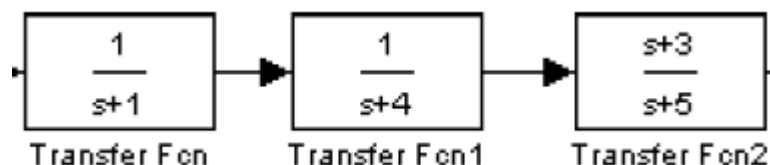
**Output: Scope**

Scope block is used to show the output. Scope block is in sinks library.



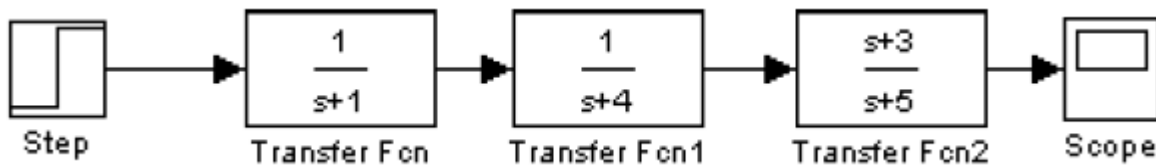
**Series system interconnection:**

For the system  $G1(s)$ ,  $G2(s)$  and  $G3(s)$  use transfer function block from the continuous library. Change the transfer function block parameters i.e., numerator and denominator according to the given transfer function.



Connect the systems such that the output of first system is the input of the second system. Similarly the output of second system is the input to the third system.

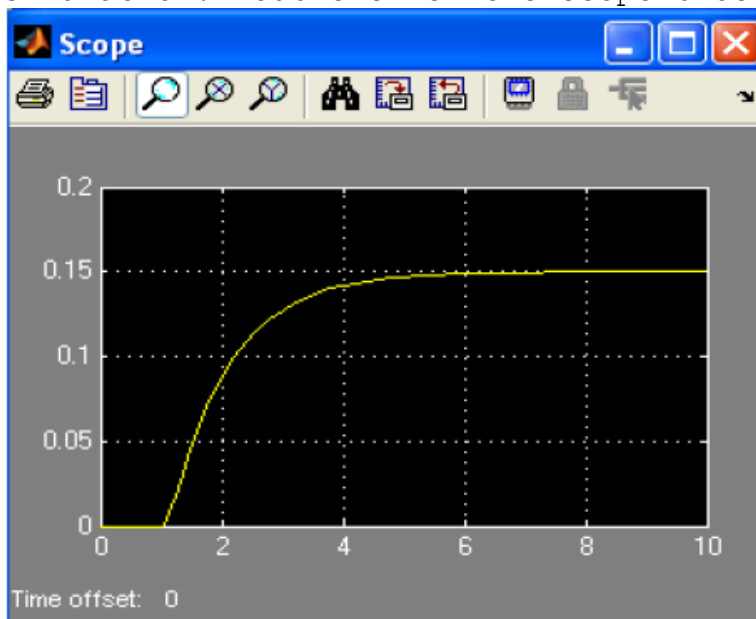
#### Overall interconnection:



The above figure shows the overall interconnection provided step function as an input. The output is shown by the scope.

#### Output:

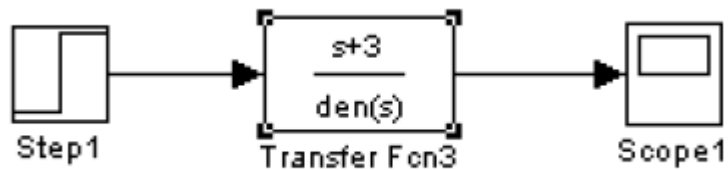
In order to show the output specify the simulation time. Start the simulation. Double click the scope block to see the plot.



### Equivalent system:

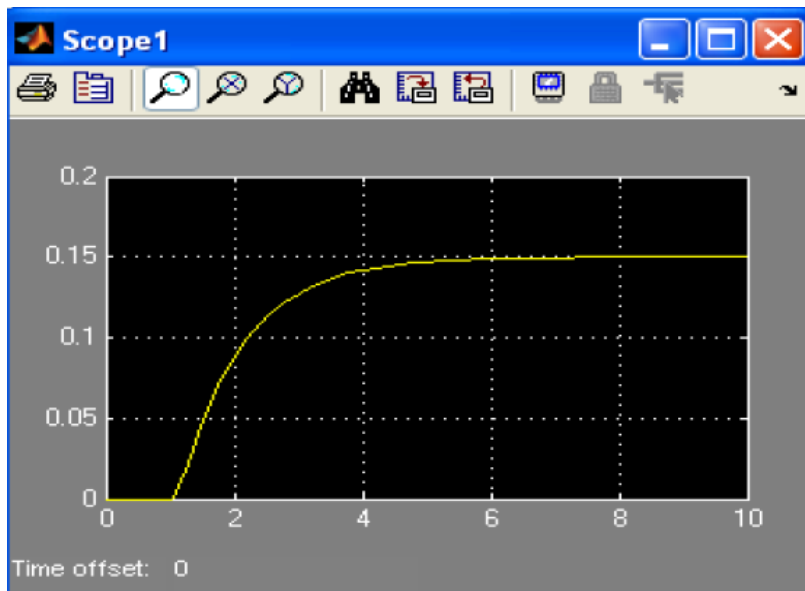
In order to find the equivalent system multiply the transfer functions. So the equivalent system would be as follows.

$$G(s) = (s+3) / (s^3 + 10s^2 + 29s + 20)$$



The above block diagram shows the equivalent system provided step function as an input. Output is shown on the scope.

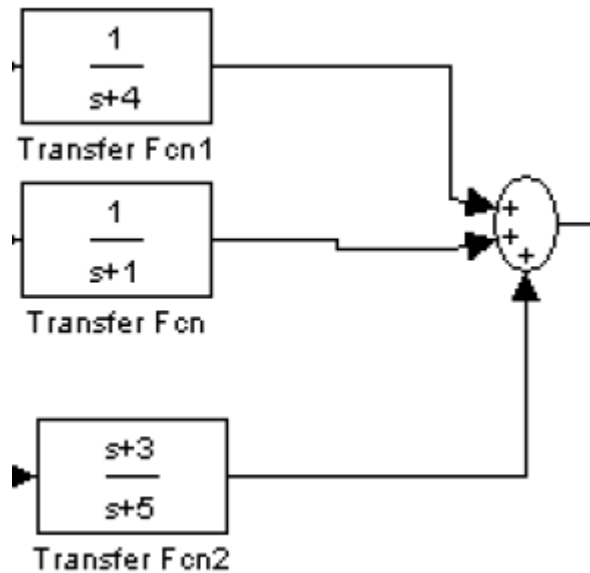
### Output:



Hence proved, both the outputs are same.

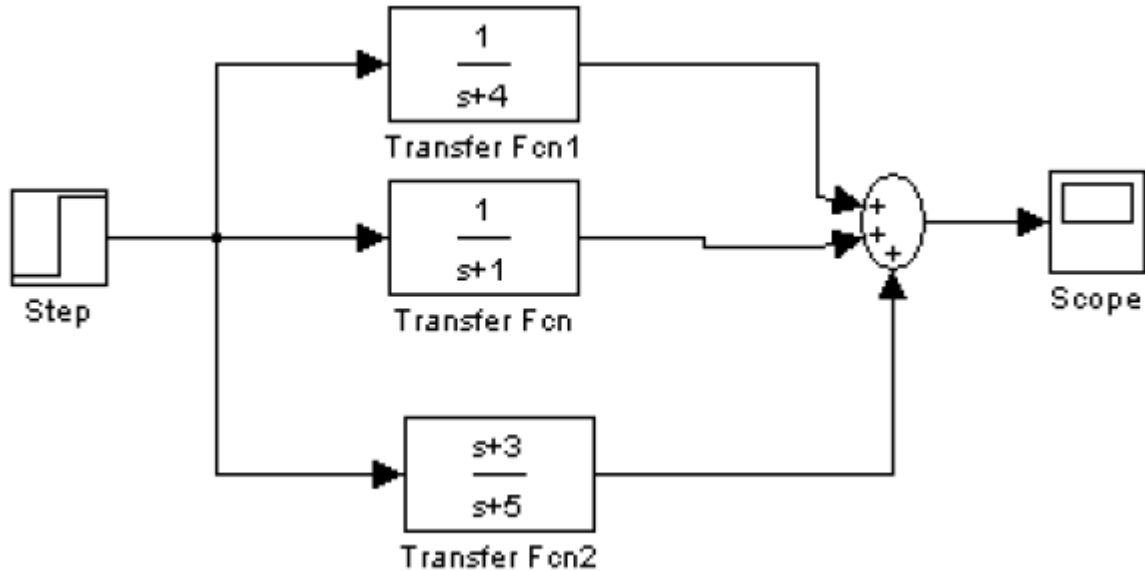
### Parallel system interconnection:

For the system  $G_1(s)$ ,  $G_2(s)$  and  $G_3(s)$  use transfer function block from the continuous library. Change the transfer function block parameters i.e., numerator and denominator according to the given transfer function.



Connect the systems such that the outputs of all the systems are summed up at the end.

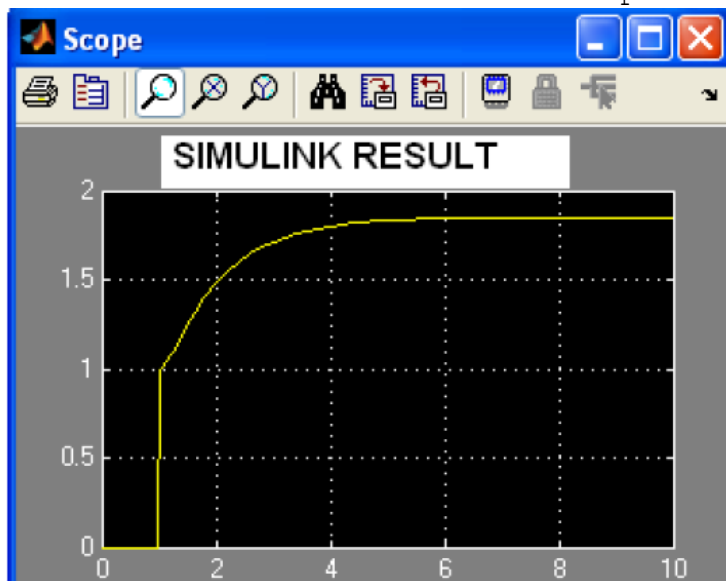
### Overall interconnection:



The above figure shows the overall interconnection provided step function as an input. The input is common to all the systems. The output is shown by the scope.

### Output:

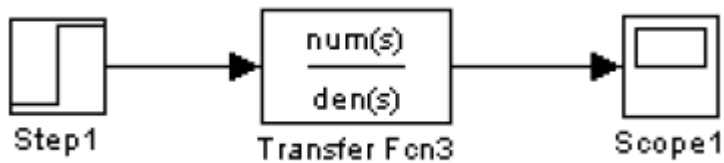
In order to show the output specify the simulation time. Start the simulation. Double click the scope block to see the plot.



### Equivalent system:

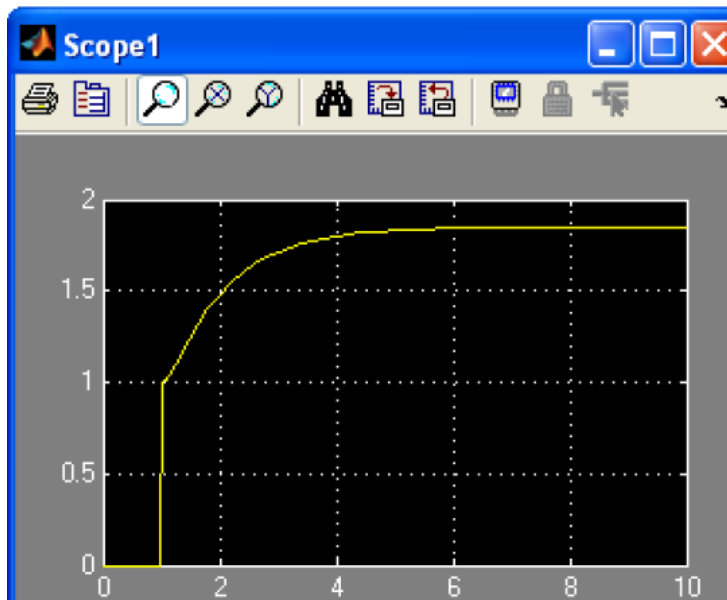
In order to find the equivalent system add the transfer functions. So the equivalent system would be as follows.

$$G(s) = (s^3 + 10s^2 + 34s + 37) / (s^3 + 10s^2 + 29s + 20)$$



The above block diagram shows the equivalent system provided step function as an input. Output is shown on the scope.

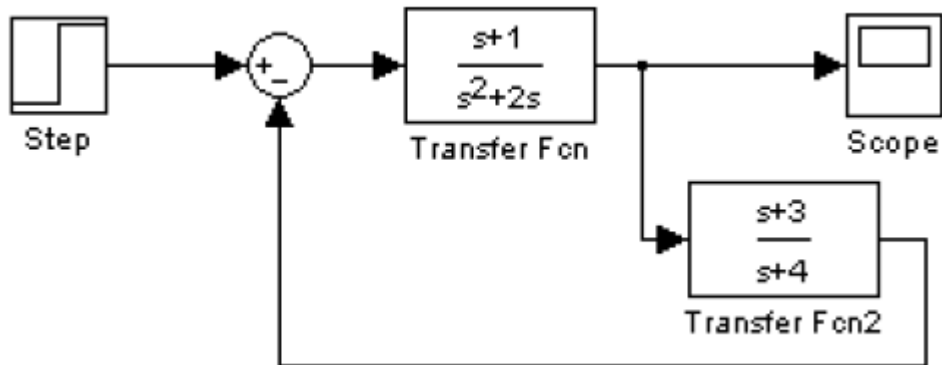
### Output:



Hence proved, both the outputs are same.



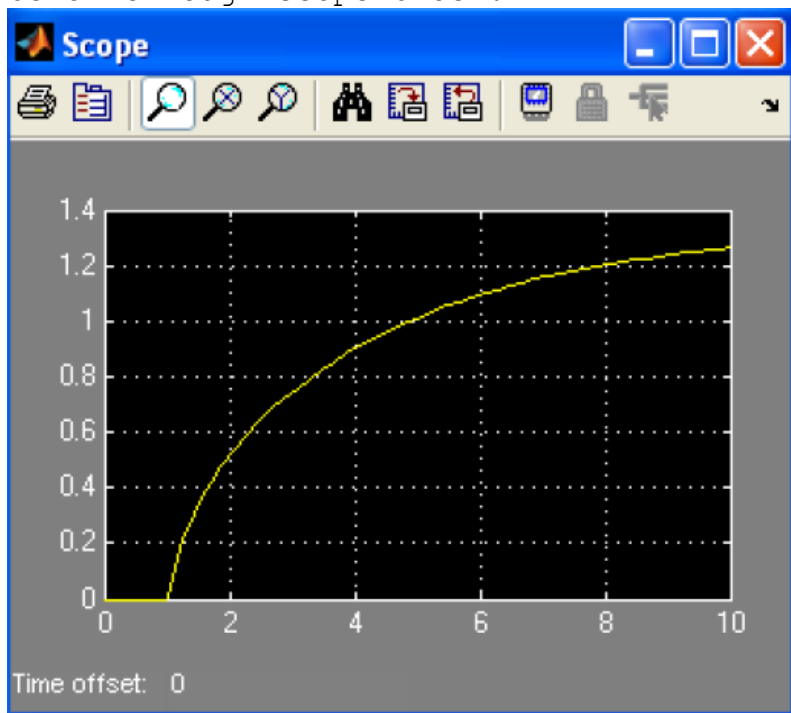
### Negative feedback system interconnection:



The above block diagram shows how the negative feedback system is interconnected. The step function is the input to the transfer function  $G(s)$  and the output of the  $G(s)$  is provided as an input to the  $H(s)$  and the output of  $H(s)$  is feedback to the  $G(s)$ .

### Output:

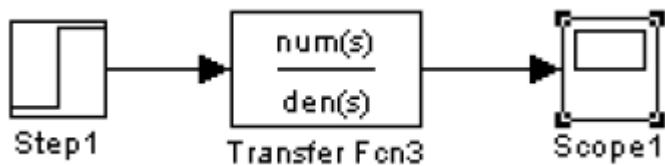
The output is calculated from the output of the  $G(s)$  and is shown as below through scope block.



### Equivalent System:

The equivalent system is found manually and is as follows.

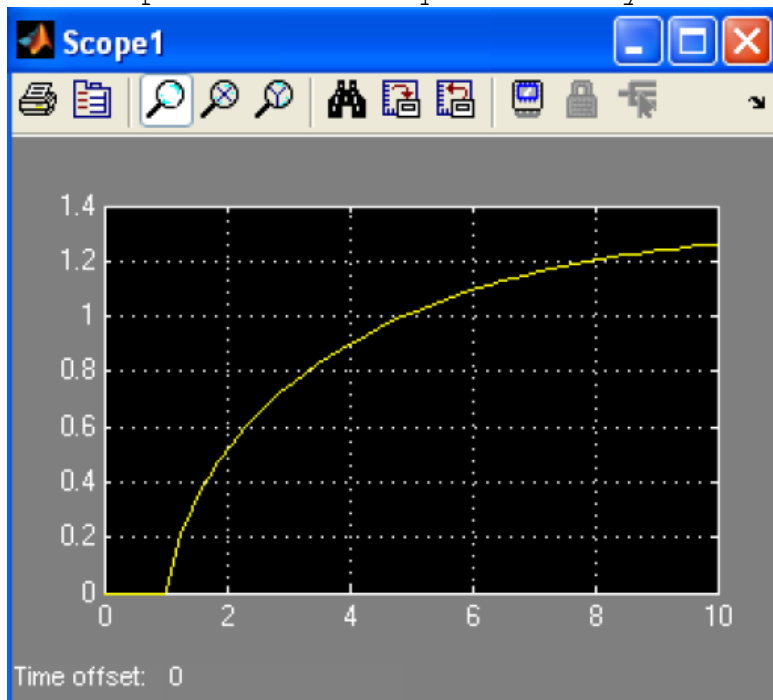
$$G(s) = (s^3 + 10s^2 + 34s + 37)/(s^3 + 10s^2 + 29s + 20)$$



The above block diagram shows how the equivalent system is provided by the step function as an input and the output is shown on the scope.

### Output:

The output from the equivalent system is shown below.



Hence proved the both inputs are same.

## Lab 9

### Stability via Routh Hurwitz

#### Lab Objectives:

- Design the system for determining the values of gain in negative feedback system for which the system is stable, unstable and marginally stable by using Routh table.
- Using Simulink show the step responses for different values of gain for each of the stable, unstable and marginally stable system.
- Show the poles locations for each of the stable, unstable and marginally stable system for varying gain values.

Routh Hurwitz Criterion

Routh Hurwitz Criterion is a tool to judge the stability of the closed loop system without solving for the poles of the closed loop system. It comprised of two steps.

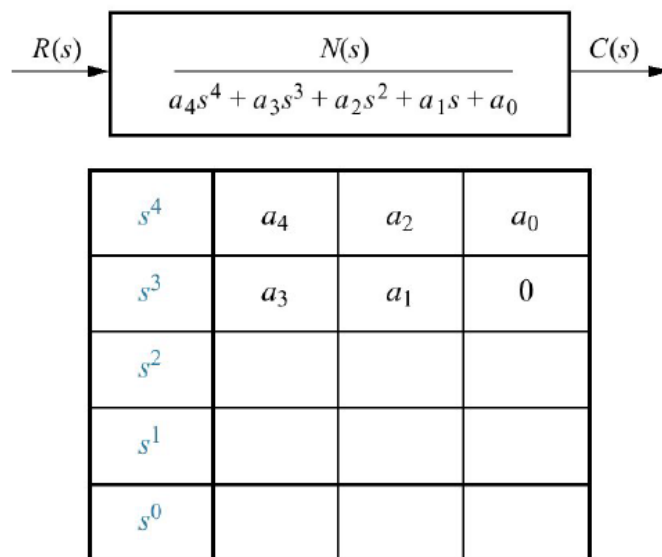
- 1) Generating the Routh Table
- 2) Interpreting the Routh Table

These two steps are explained in detail as follows.

#### 1) Generating the Routh Table

Routh table can be generated for the closed loop system as follows.

**Initial layout**



The remaining entries are filled up in the following manner.

|       |                                                                           |                                                                         |                                                                     |
|-------|---------------------------------------------------------------------------|-------------------------------------------------------------------------|---------------------------------------------------------------------|
| $s^4$ | $a_4$                                                                     | $a_2$                                                                   | $a_0$                                                               |
| $s^3$ | $a_3$                                                                     | $a_1$                                                                   | 0                                                                   |
| $s^2$ | $-\frac{\begin{vmatrix} a_4 & a_2 \\ a_3 & a_1 \end{vmatrix}}{a_3} = b_1$ | $-\frac{\begin{vmatrix} a_4 & a_0 \\ a_3 & 0 \end{vmatrix}}{a_3} = b_2$ | $-\frac{\begin{vmatrix} a_4 & 0 \\ a_3 & 0 \end{vmatrix}}{a_3} = 0$ |
| $s^1$ | $-\frac{\begin{vmatrix} a_3 & a_1 \\ b_1 & b_2 \end{vmatrix}}{b_1} = c_1$ | $-\frac{\begin{vmatrix} a_3 & 0 \\ b_1 & 0 \end{vmatrix}}{b_1} = 0$     | $-\frac{\begin{vmatrix} a_3 & 0 \\ b_1 & 0 \end{vmatrix}}{b_1} = 0$ |
| $s^0$ | $-\frac{\begin{vmatrix} b_1 & b_2 \\ c_1 & 0 \end{vmatrix}}{c_1} = d_1$   | $-\frac{\begin{vmatrix} b_1 & 0 \\ c_1 & 0 \end{vmatrix}}{c_1} = 0$     | $-\frac{\begin{vmatrix} b_1 & 0 \\ c_1 & 0 \end{vmatrix}}{c_1} = 0$ |

For the special cases where the first column contain any zero entry or entire row is zero use their corresponding methods to solve the Routh table

## 2) Interpreting the Routh Table

For the system to be stable, there should be no poles in the right half plane. The first column of the Routh table is used for the analysis if there is sign change; it means there are poles in the right half plane. The number of sign change corresponds to the number of poles locations. If the poles are on the imaginary axis with multiplicity 1 and on the left half plane then the system is marginally stable. If all the poles are in the left half plane the system is stable.

## Experiments

Do the following experiments

- 1) Find the equivalent negative feedback system shown in the figure 1

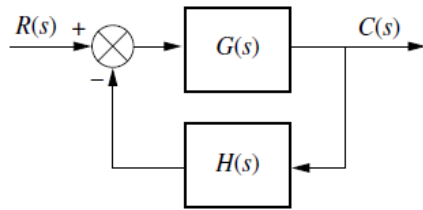


Figure 1: Negative feedback system

Here

$$G(s) = \frac{K}{s(s+2)^2} \text{ and } H(s) = 1$$

- 2) Generate the Routh table and find the ranges of K for the system to be stable, unstable and marginally stable
- 3) Using Simulink setup the negative feedback system for the system to be stable. Use five different values from the range of K for stable system and plot all step responses on one graph showing the response changes with respect to the varying the value of K.
- 4) Using Simulink setup the negative feedback system for the system to be unstable. Use five different values from the range of K for unstable system and plot all the step responses on one graph showing the response changes with respect to the varying the value of K.
- 5) Using Simulink setup the negative feedback system for the system to be marginally stable. Plot the step response for the value of K so that the system is marginally stable.
- 6) Using Simulink setup the negative feedback system and compare the step response of the system to be stable, unstable and marginally with only one selected value of K for each case. Plot the step responses on one graph.
- 7) Using MATLAB find the location of poles for each case of the system to be stable, unstable and marginally stable for all the values of K selected in each case.

**Note : Make a report which shows the Simulink block diagram, step responses and the poles location for each case.**

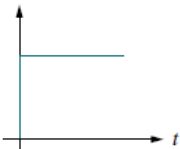
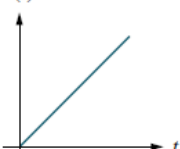
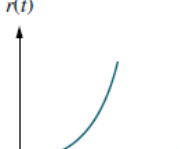
## Lab 10

# Steady State Error Analysis using Simulink

Steady state error is the difference between the input and the output for the prescribed test input as  $t$  (time) approaches to infinity.

The objective of this lab to analyze the steady state error analysis for different test signals in Simulink.

Following are the test signals that will be used for analysis of steady state error.

| Waveform                                                                            | Name     | Physical interpretation | Time function    | Laplace transform |
|-------------------------------------------------------------------------------------|----------|-------------------------|------------------|-------------------|
|   | Step     | Constant position       | 1                | $\frac{1}{s}$     |
|  | Ramp     | Constant velocity       | $t$              | $\frac{1}{s^2}$   |
|  | Parabola | Constant acceleration   | $\frac{1}{2}t^2$ | $\frac{1}{s^3}$   |

## Experiments

Setup negative feedback system for the following system

$$G(s) = K(s+6) / (s+4)(s+7)(s+9)(s+12)$$

$$H(s) = 1$$

- 1) Plot on one graph Plot on graph the error signal for the input of  $5u(t)$  with different values of  $k=50,500,1000,5000$ .
- 2) Plot on one graph Plot on graph the error signal for the input of  $5tu(t)$  with different values of  $k=50,500,1000,5000$ .
- 3) Plot on one graph Plot on graph the error signal for the input of  $5t^2u(t)$  with different values of  $k=50,500,1000,5000$ .

## Lab 11 A

### Implementation of Root Locus using MATLAB

- Find the root locus for the following systems using Matlab. [Hint: use *rlocus* command].

$$G(s)H(s) = \frac{K(s-2)}{(s+2)(s+2j)(s-2j)}$$

$$G(s)H(s) = \frac{K(s+2)}{s^2 + 2s + 3}$$

- Also find values of poles at  $K=0,1,2,3\dots10$

#### Introduction:

This lab is based on finding the root locus of system. Two basic commands can be used to get the root locus. `Sisotool(sys)` and `rootlocus(sys)` are used to find root locus but restriction for this lab is usage of `rlocus(sys)` command only.

The method adopted by W.R. Evans to find the roots of a characteristics equation is called root-locus method. Function for drawing a root locus in MATLAB is the `rlocus()` function. `Rlocus()` computes and plots the root locus of the single-input, single-output LTI model `sys`. `Rlocus(sys,k)` uses a user-specified vector `K` of gains.

#### Tools Used:

- MATLAB
- `rlocus(sys)`



### Procedure 1:

- a) Simplify the transfer function to generate num and denum.
- b) Using MATLAB write the code1, given below, to find the root locus.
- c) After executing code1 a graph (Figure 11.1a) will appear showing root locus of a system.
- d) Code2 is executed to give different values of gain (K) to the system and this time rlocus(sys,k) command is used to plot root locus with different value of gains.
- e) Graph (Figure 6.2a) is plotted only for range of K given in code2 where range of K is from 0 to 10.
- f) By changing the values of K, the value of poles of a system changes.
- g) In code2 for different values of K the value of poles are observed and are shown in Figure 6.3a.

### Simplification of Transfer function:

```
=(s-2)/(s+2)(s+2j)(s-2j)
=(s-2)/(s+2)(s^2+2s+3)
=(s-2)/(s^3+2s^2+4s+8)
So num= [1 -2] & denum= [1 2 4 8]
```

### MATLAB Code 1:

```
num=[1 -2]
denum=[1 2 4 8]
sys=tf(num,denum)
rlocus(sys)
title('Root locus of G(S)H(S)=K(s - 2)/(s^3 + 2 s^2 + 4 s + 8)')
```

### MATLAB Graph for Code1:

By executing above code one will get the graph given below.

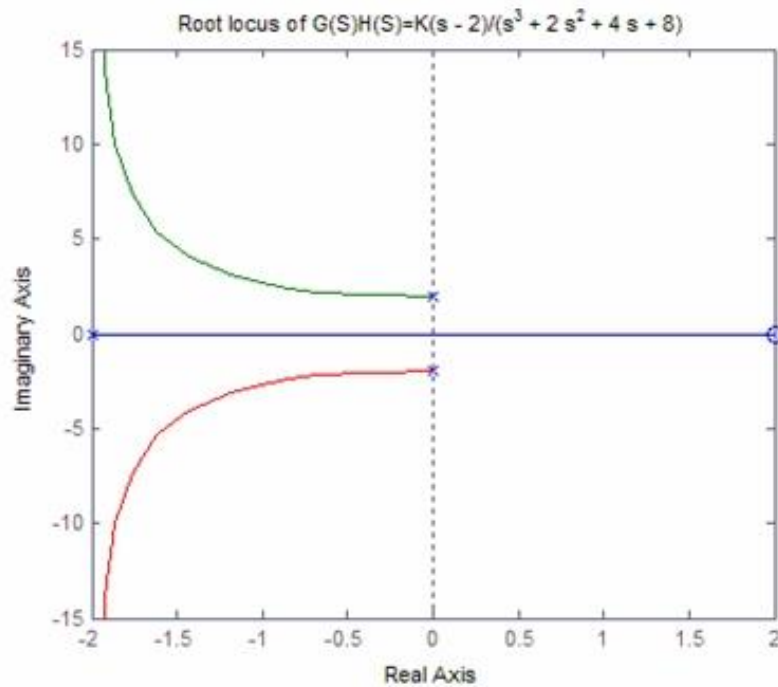


Figure 11.1a: Root locus of system for code1

**MATLAB Code2:**

```
num=[1 -2]
denum=[1 2 4 8]
sys=tf(num,denum)
k=[0:1:10]
rlocus(sys,k)
title('Root Locus Plot of  $G(s)H(s) = K(s-2)/(s+2)(s+2j)(s-2j)$ , for "K= 0 to 10"')
```

**MATLAB Graph for Code2:**

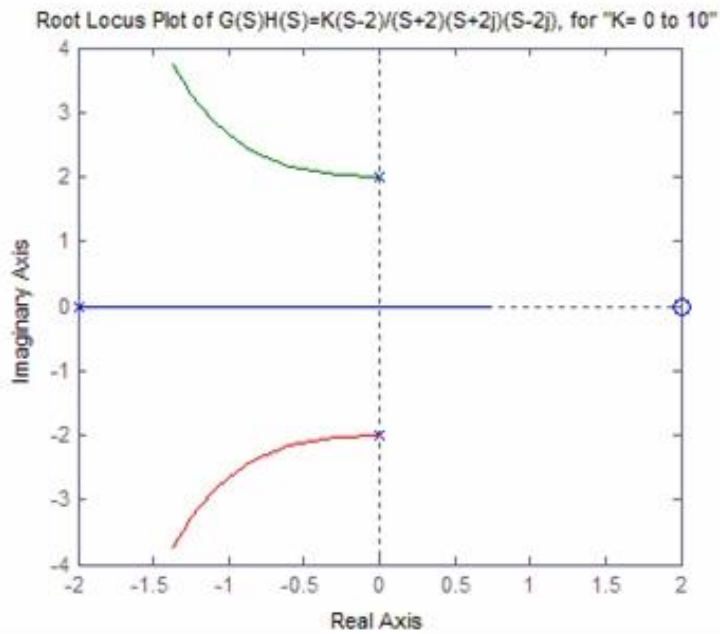


Figure 11.2a: Root locus of system for code2 for K= 0 to 10

**Observing Different Value of poles for different value of K in Figure 11.2a:**

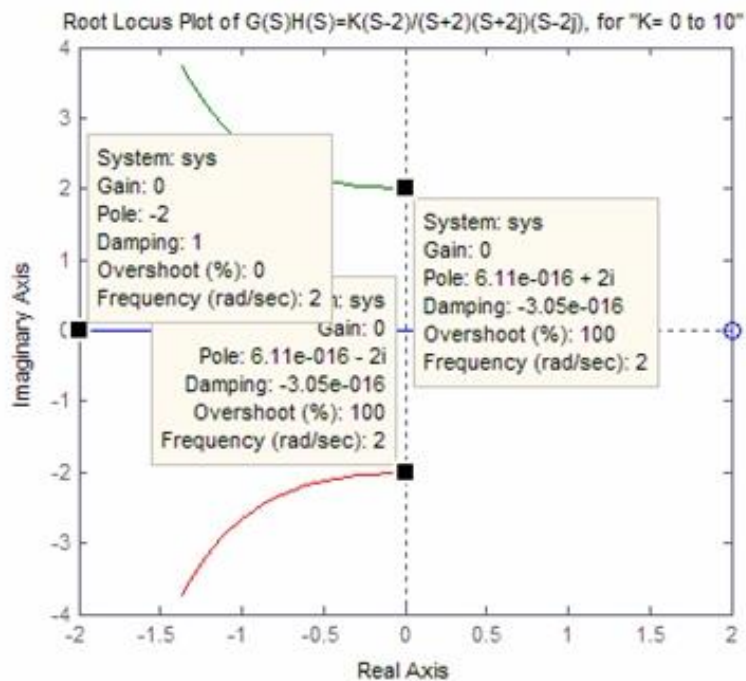


Figure 11.2a1 Value of poles for K=0

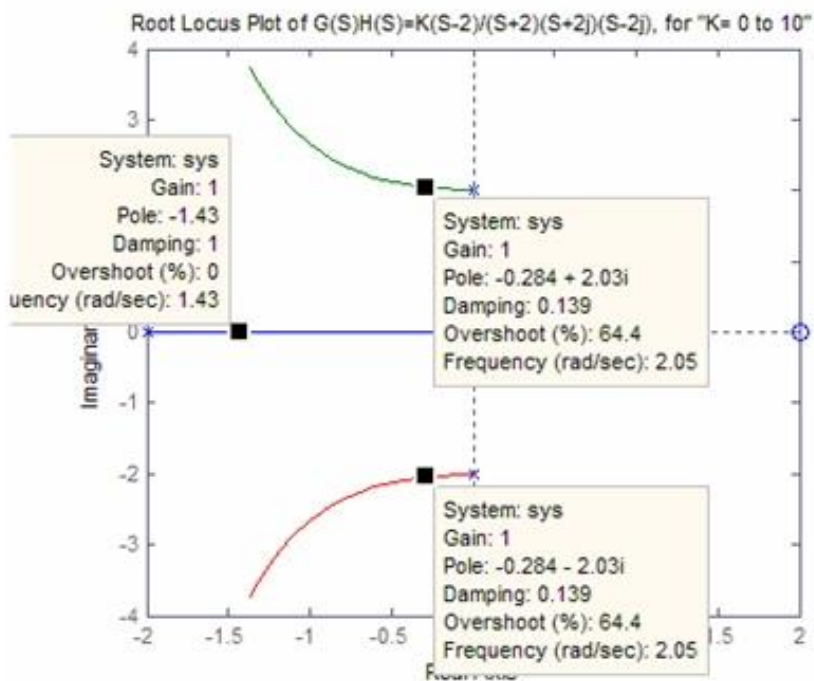


Figure 11.2a2 Value of poles for K=1

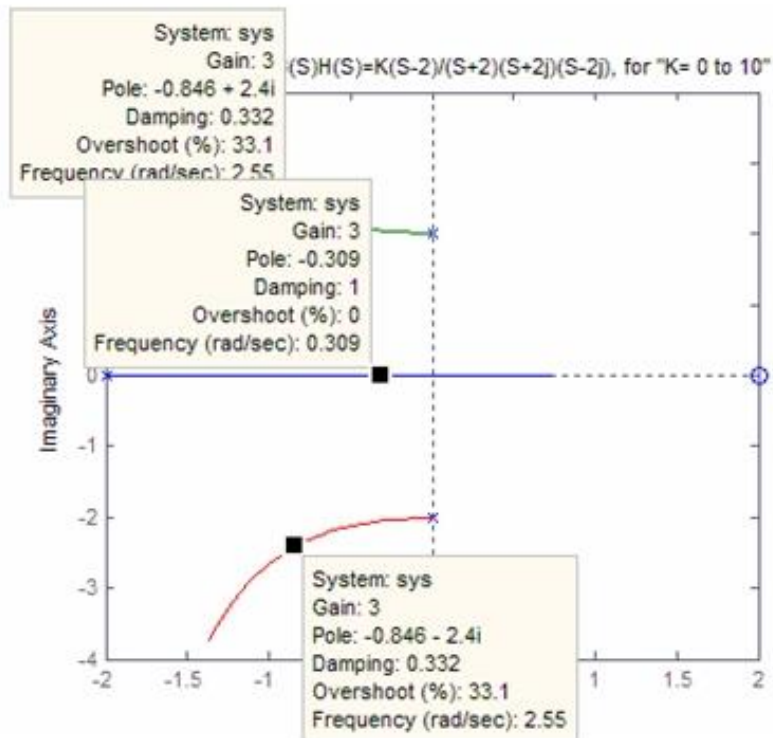


Figure 11.2a3 Value of poles for K=3

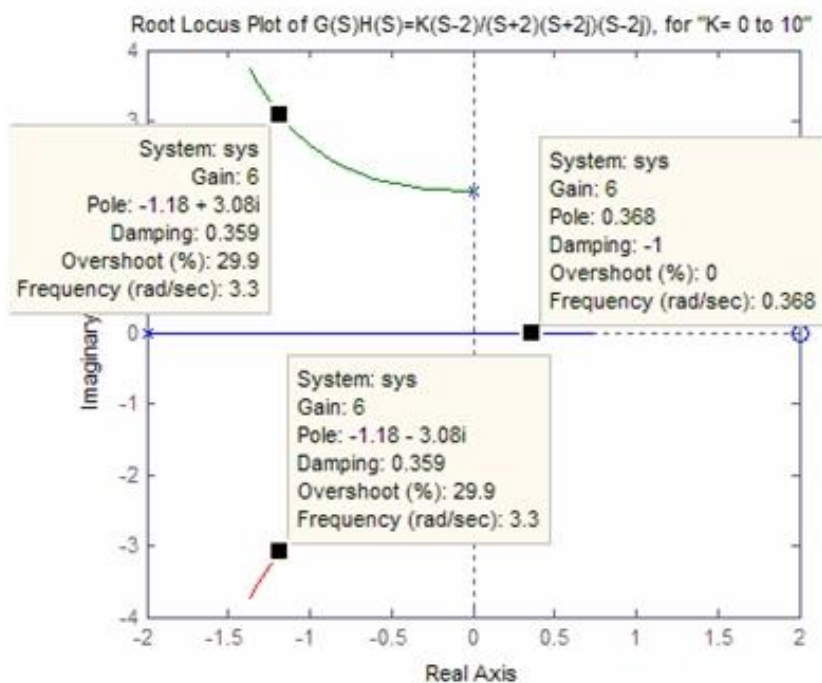


Figure 11.2a4 Value of poles for K=6

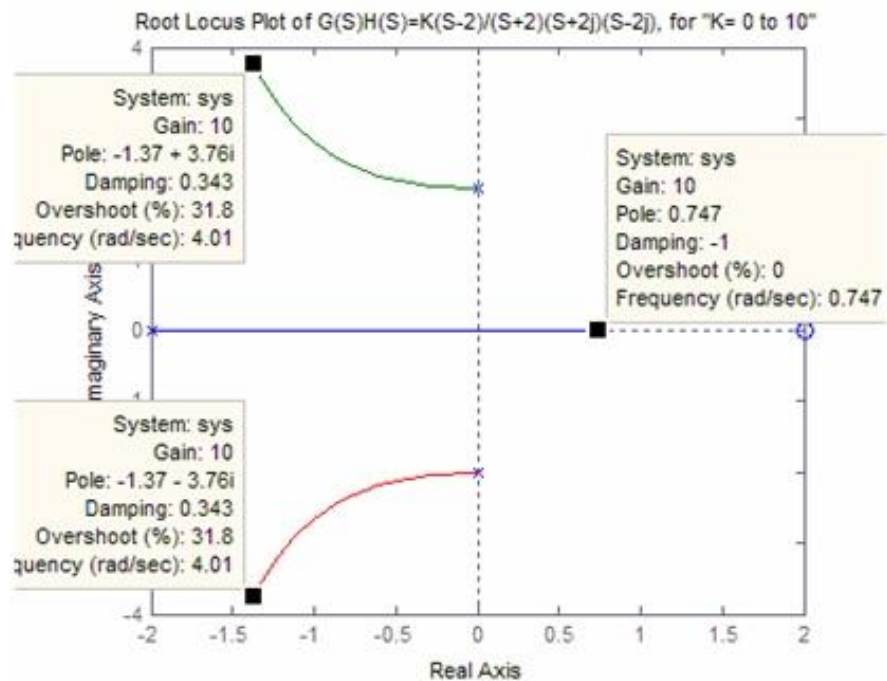


Figure 11.2a5 Value of poles for K=10

The above procedure is done for all the values of K (0 to 10) and different value of poles is found at different K.

**Table showing poles for different values of K:**

| Different Values of K | Value of poles at different K           |
|-----------------------|-----------------------------------------|
| 0                     | -2, $6.11e-16+2i$ , $6.11e-16-2i$       |
| 1                     | -1.43, $-0.284+2.03i$ , $-0.284-2.03i$  |
| 2                     | -0.793, $-0.603+2.16i$ , $-0.603-2.16i$ |
| 3                     | -0.309, $-0.846+2.4i$ , $-0.846-2.4i$   |
| 4                     | $8.88e-016$ , $-1+2.65i$ , $-1-2.65i$   |
| 5                     | 0.211, $-1.11+2.87i$ , $-1.11-2.87i$    |
| 6                     | 0.368, $-1.18+3.08i$ , $-1.18-3.08i$    |

|    |                                 |
|----|---------------------------------|
| 7  | 0.491, -1.25+3.27i, -1.25-3.27i |
| 8  | 0.591, -1.3+3.44i, -1.3-3.44i   |
| 9  | 0.675, -1.34+3.61i, -1.34+3.61i |
| 10 | 0.747, -1.37+3.76i, -1.37-3.76i |

### **Analysis:**

Loop gain or K start from 0 and approaches to 3.99 for stable system. As the values goes beyond 3.99 or 4 so the system become unstable. K is stable for 0 to 3.99 and unstable for 4 to 10.

### **Conclusion:**

Rlocus() is used to check the stability of a system. It's very hard to determine from the differential equation or the transfer function of a system whether the system is stable or not. MATLAB helps in finding many other properties or characteristics of a system.

## CONTROL SYSTEM LAB # 11B

### Introduction:

This lab is based on finding the root locus of system. Two basic commands can be used to get the result. `Sisotool(sys)` and `rootlocus(sys)` are used to find root locus but restriction for this lab is usage of `rlocus(sys)` command only.

The method adopted by W.R. Evans to find the roots of a characteristics equation is called root-locus method. Function for drawing a root locus in MATLAB is the `rlocus()` function. `Rlocus()` computes and plots the root locus of the single-input, single-output LTI model `sys`. `Rlocus(sys,k)` uses a user-specified vector `K` of gains.

### Tools Used:

- c) MATLAB
- d) `rlocus(sys)`

### Procedure 2:

- h) Get num and denum from given transfer function
- i) Using MATLAB write the code1, given below, to find the root locus.
- j) After executing code1 a graph (Figure 6.1b) will appear showing root locus of a system.
- k) Code2 is executed to give different values of gain (`K`) to the system and this time `rlocus(sys,k)` command is used to plot root locus with different value of gains.
- l) Graph (Figure 6.2b) is plotted only for range of `K` given in code2 where range of `K` is from 0 to 10.
- m) By changing the values of `K`, the value of poles of a system changes.
- n) In code2 for different values of `K` the value of poles are observed and are shown in Figure 6.3b.

### MATLAB Code1:

```
num=[1 2]
denum=[1 2 3]
sys=tf(num,denum)
rlocus(sys)
```



```
title('Root Locus Plot of  $G(S)H(S)=K(S+2)/(S^2+2S+3)$ , for  
"K= 0 to 10"')
```

### MATLAB Graph for Code1:

By executing above code one will get the graph given below.

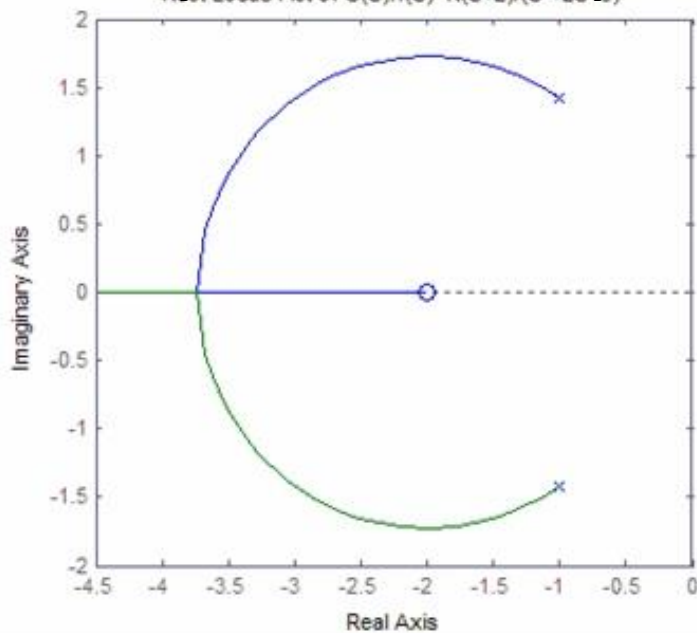


Figure 11.1b: Root locus of system for code1

### MATLAB Code2:

```
num=[1 2]
denum=[1 2 3]
sys=tf(num,denum)
k=[0:1:10];
rlocus(sys,k)
title('Root Locus Plot of  $G(S)H(S)=K(S+2)/(S^2+2S+3)$ , for  
"K= 0 to 10"')
```

**MATLAB Graph for Code2:**

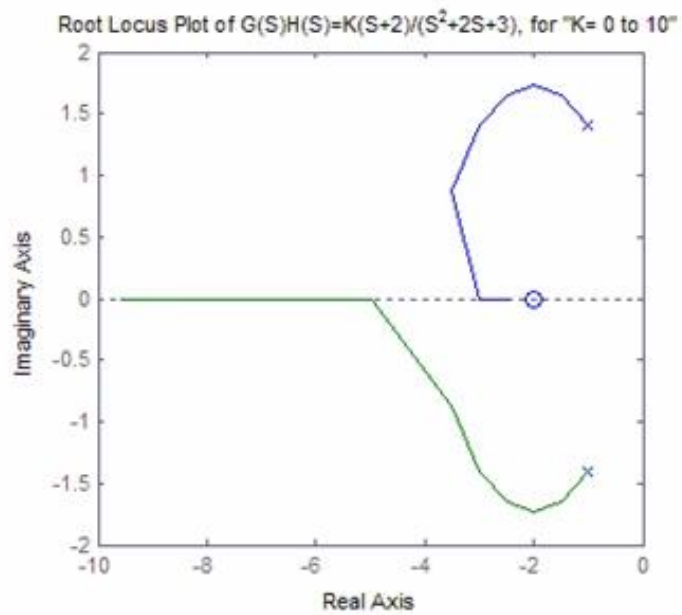


Figure 11.2b: Root locus of system for code2 for K= 0 to 10

**Observing Different Value of poles for different value of K in Figure 11.2b:**

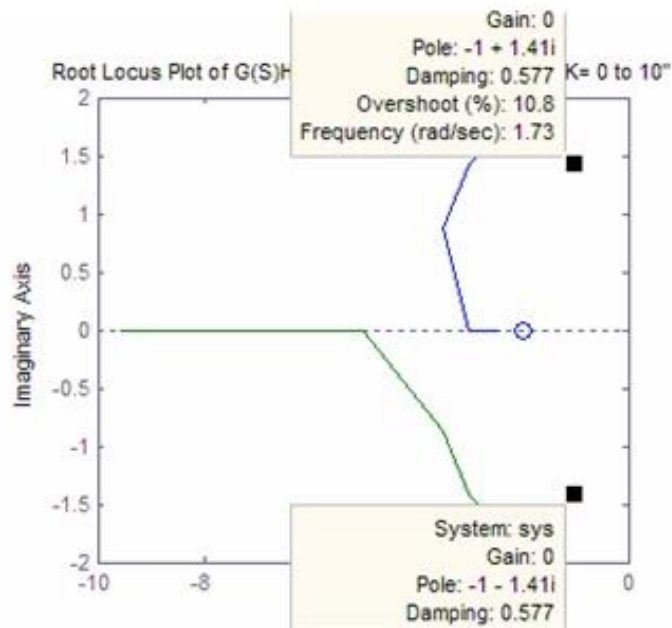


Figure 11.2b1 Value of poles for  $K=0$

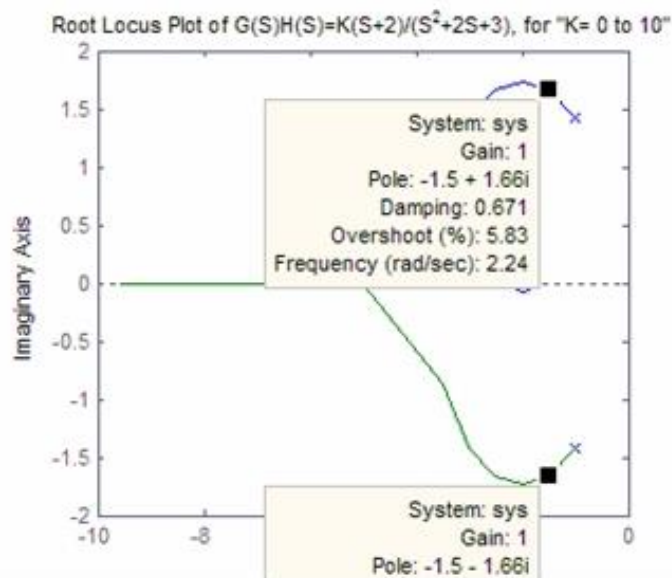


Figure 11.2b2 Value of poles for  $K=1$

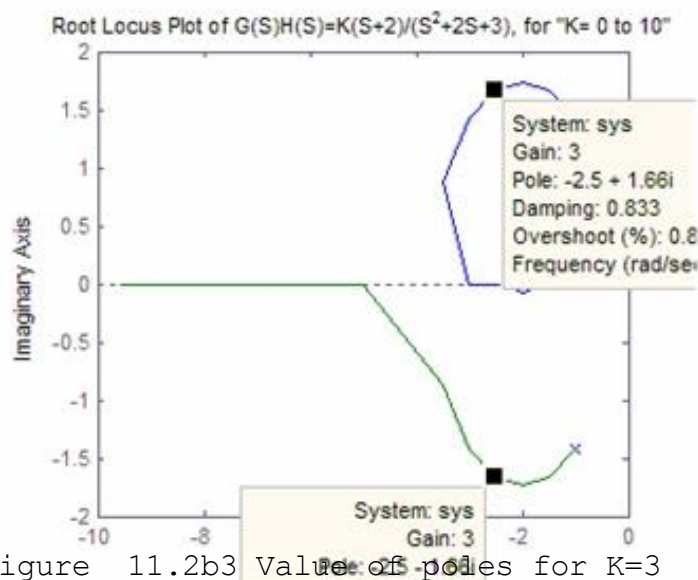


Figure 11.2b3 Value of poles for K=3

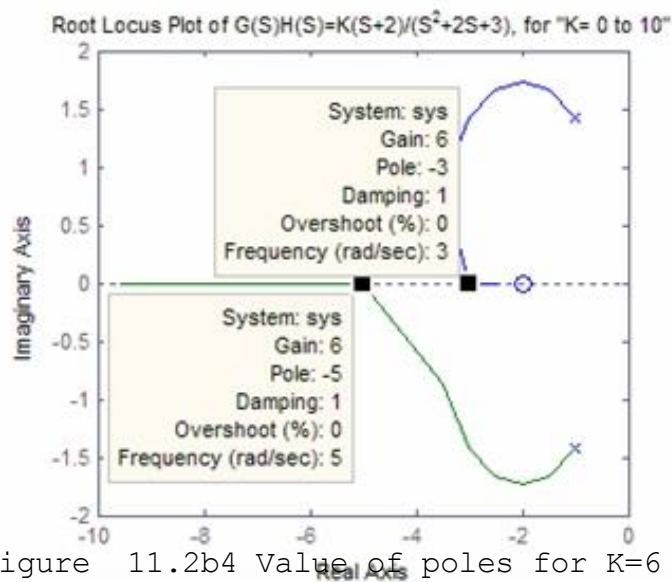


Figure 11.2b4 Value of poles for K=6

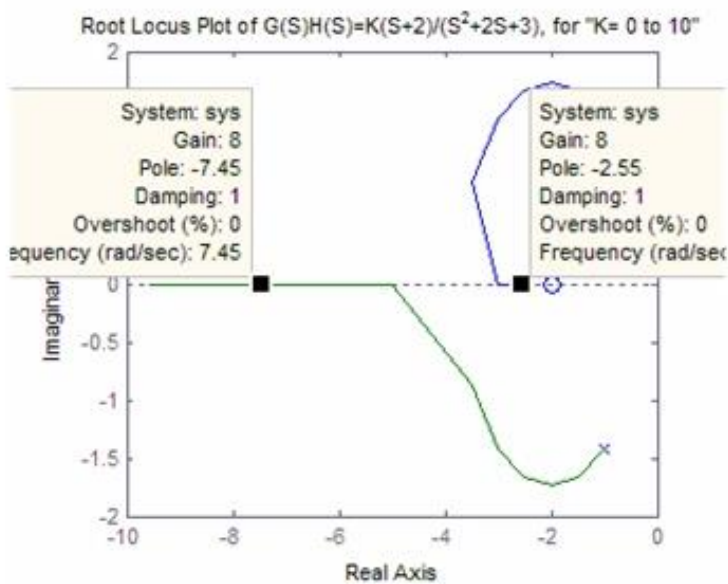


Figure 11.2b5 Value of poles for K=8

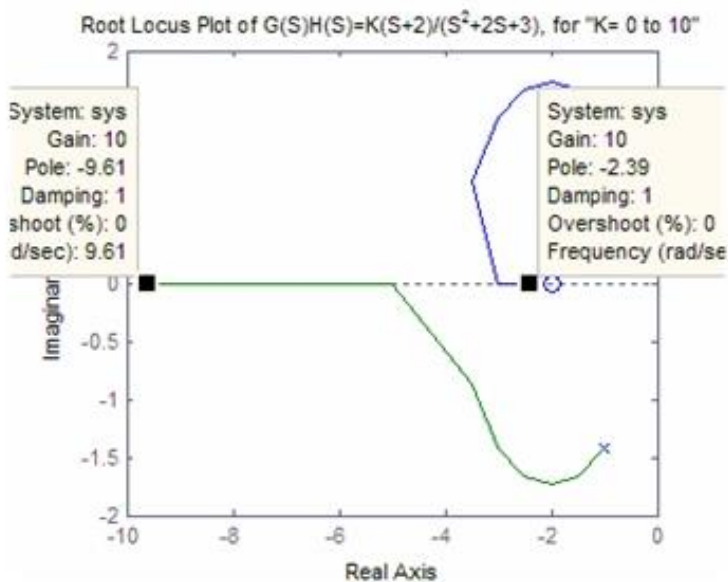


Figure 11.2b6 Value of poles for K=10

The above procedure is done for all the values of K (0 to 10) and different value of poles is found at different K.

**Table showing poles for different values of K:**

| Different Values of K | Value of poles at different K |
|-----------------------|-------------------------------|
| 0                     | -1+1.41i, -1-1.41i            |
| 1                     | -1.5+1.66i, -1.5-1.66i        |
| 2                     | -2+1.73i, -2-1.73i            |

|    |                            |
|----|----------------------------|
| 3  | $-2.5+1.66i, -2.5+1.66i$   |
| 4  | $-3+1.41i, -3+1.41i$       |
| 5  | $-3.5+0.866i, -3.5-0.866i$ |
| 6  | $-5, -3$                   |
| 7  | $-6.3, -2.7$               |
| 8  | $-7.54, -2.55$             |
| 9  | $-8.54, -2.46$             |
| 10 | $-9.61, -2.39$             |

### **Analysis:**

Loop gain or K start from 0 and approaches to 10. For all the values of K the poles of the given system are observed and the system is said to be stable.

### **Conclusion:**

Rlocus() is used to check the stability of a system. It's very hard to determine from the differential equation or the transfer function of a system whether the system is stable or not. MATLAB helps in finding many other properties or characteristics of a system.

## Lab 12

# Implementation of Root Locus using Sisotool in MATLAB

1. Using sisotool find the root locus design for the following system:  
$$G(S)H(S) = k(S+3)(S+4)/(S+5)(S+6);$$
2. Add a pair of complex poles to the system defines above.
3. Add a pair of complex zeros to the system defines above.
4. Add a real zero to the system.
5. Add a real pole to the system.

Also find the following for Questions 1 to 5. ⌚

The range of k for which the system is stable. ⌚

The break in/ breakout points if any.

⌚ The impulse response for k=2.

⌚ the step response for k=3

⌚ the rise time and peak response for k=3

### Introduction:-

SISOTOOL is Graphical User Interface allows us to design single-input/single-output (SISO) compensators by using the root locus, Bode, and Nichols plots of the open-loop system.

### Root Locus:-

rlocus computes the Evans root locus of a SISO open-loop model. The root locus gives the closed-loop pole trajectories as a function of the feedback gain k (assuming negative feedback). Root loci are used to study the effects of varying feedback gains on closed-loop pole locations. In turn, these locations provide indirect information on the time and frequency responses.

### Equipment:-

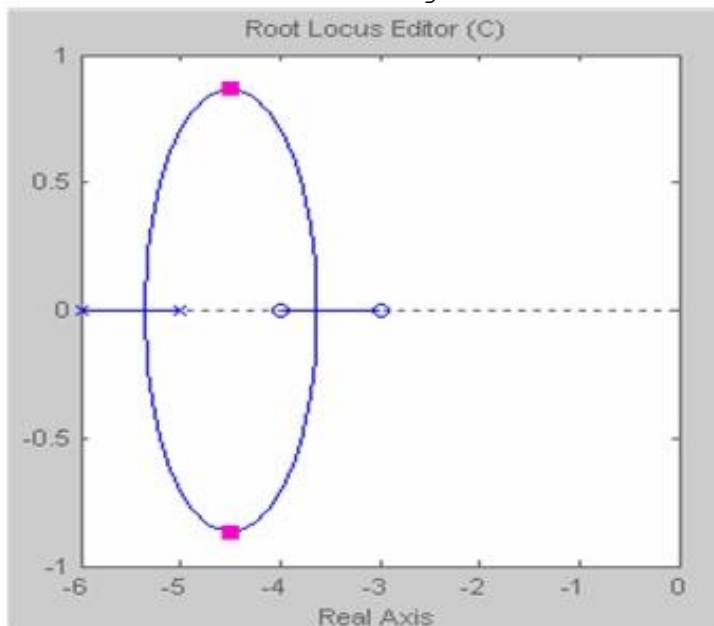
Using SISOTOOL in Matlab software.

**Q#01:-**

Matlab code for Q#01 is given below:

```
num =[1 7 12];  
den =[1 11 30];  
sys=tf(num,den);  
sisotool(sys);
```

Result of sisotool is given below.



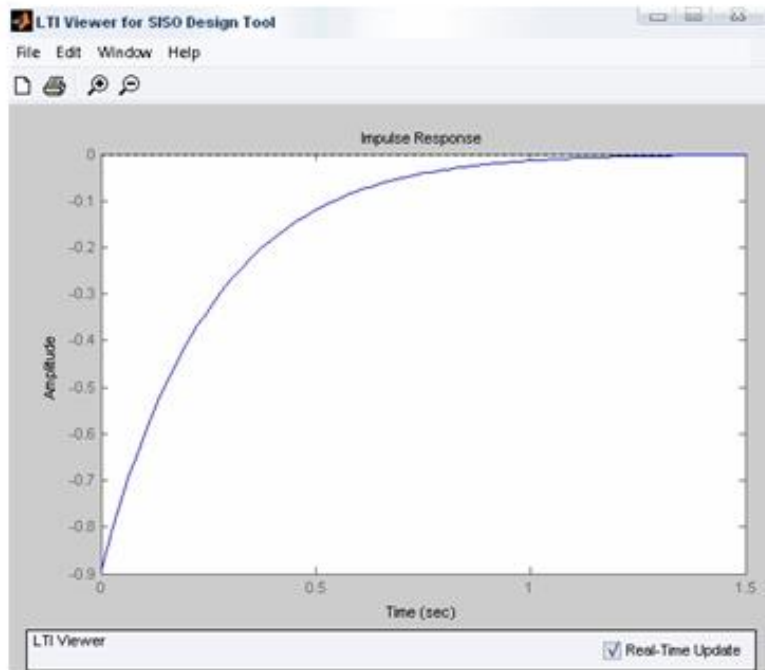
The range of K for which the system is stable.

The system is stable for  $k = -\infty$  to  $\infty$ .

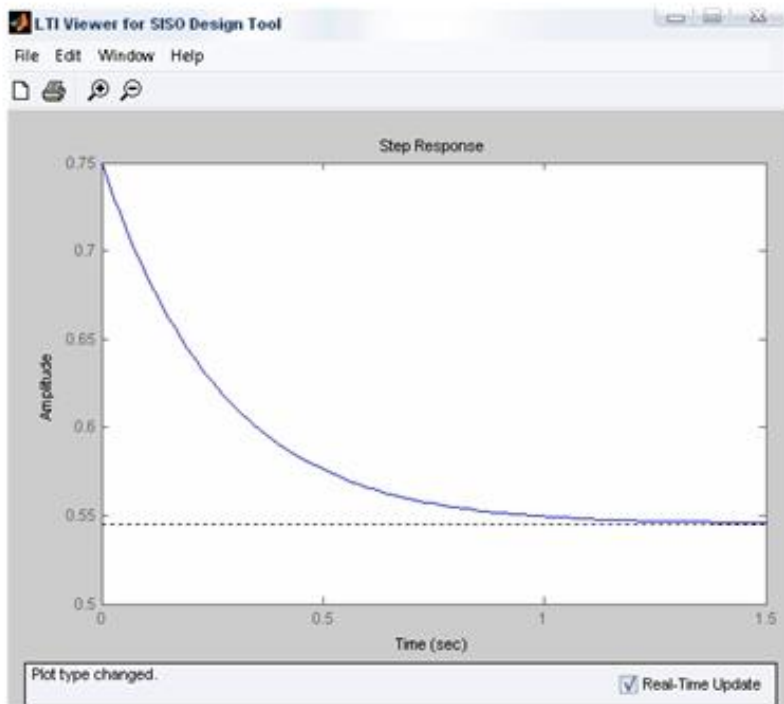
The breakaway point is 0.071



Impulse response for  $k=2$  is given below.



Step response for  $k=3$  is given below.

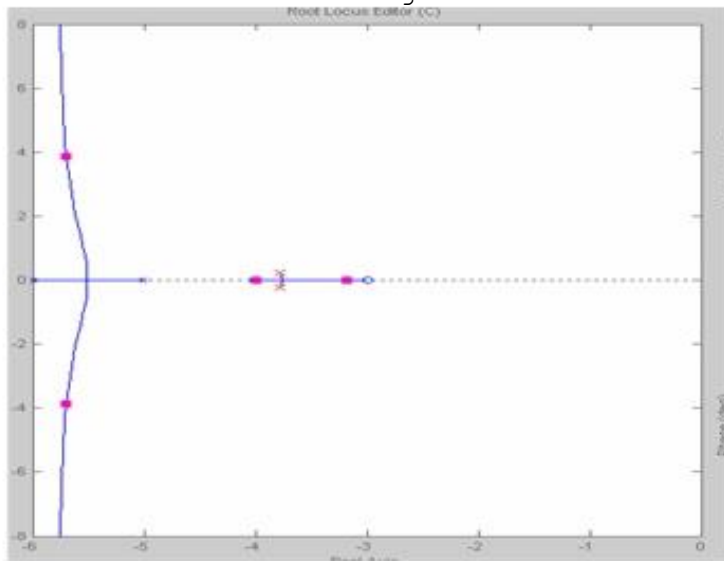


Rise time and peak response for  $k=3$

The rise time is 0.572 and peak response is 0.75.

**Q#02:-** Add a pair of complex poles to the system defines above.

A pair of complex poles to the above system is added to sisotool and result is given below.



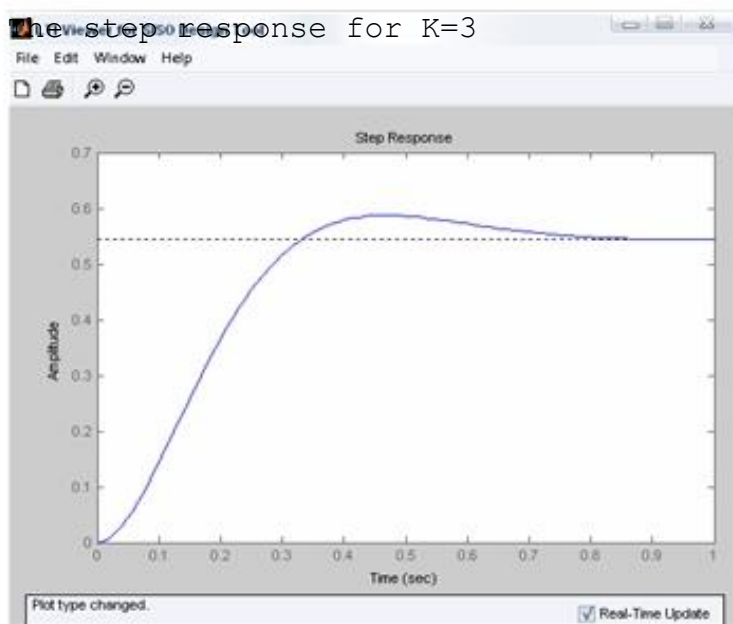
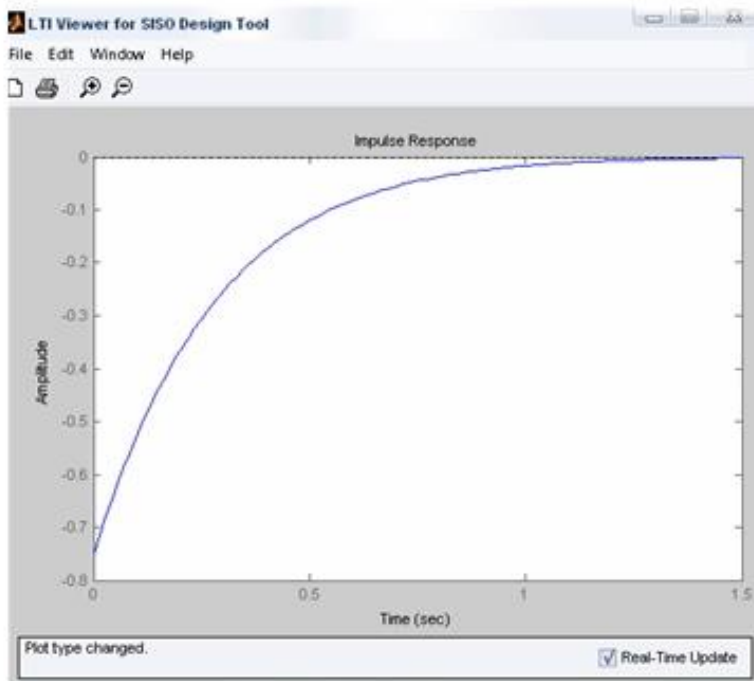
The range of  $K$  for which the system is stable.

The system is stable for  $K = -\infty$  to  $\infty$ .

The break in or break away points if any:

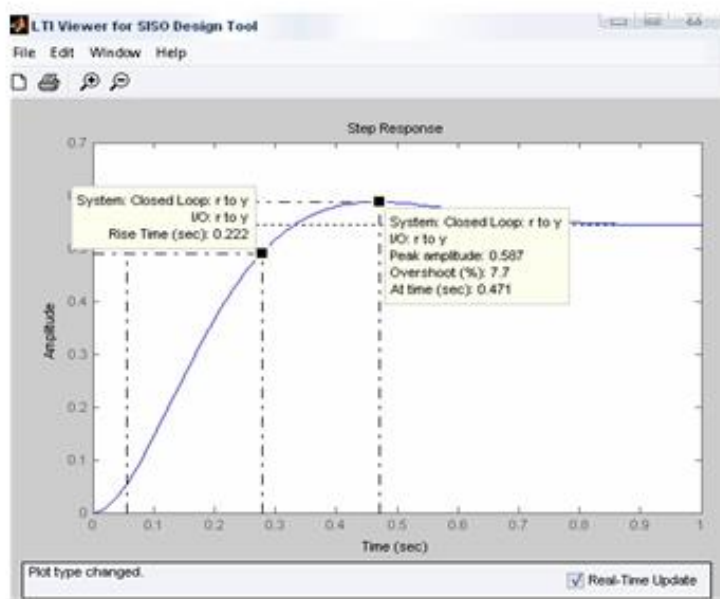
The break away point is 0.0139

The impulse response for  $K=2$

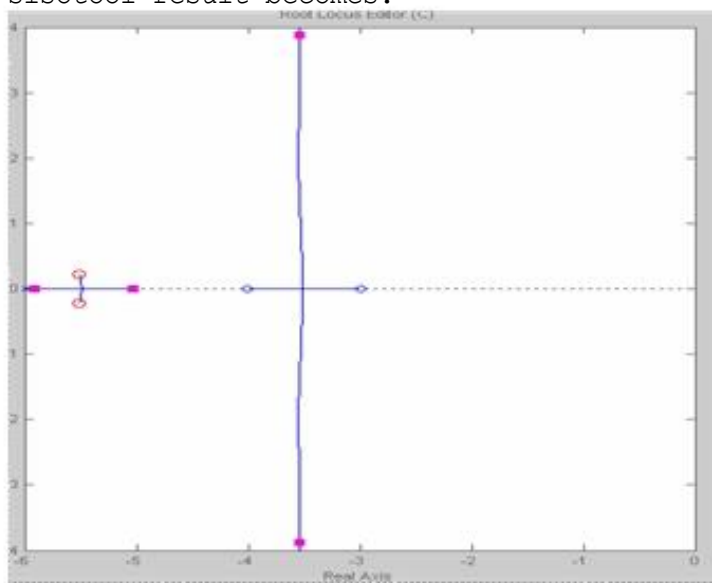


The rise time and peak response for K=3

Rise time=0.22      peak response=0.58

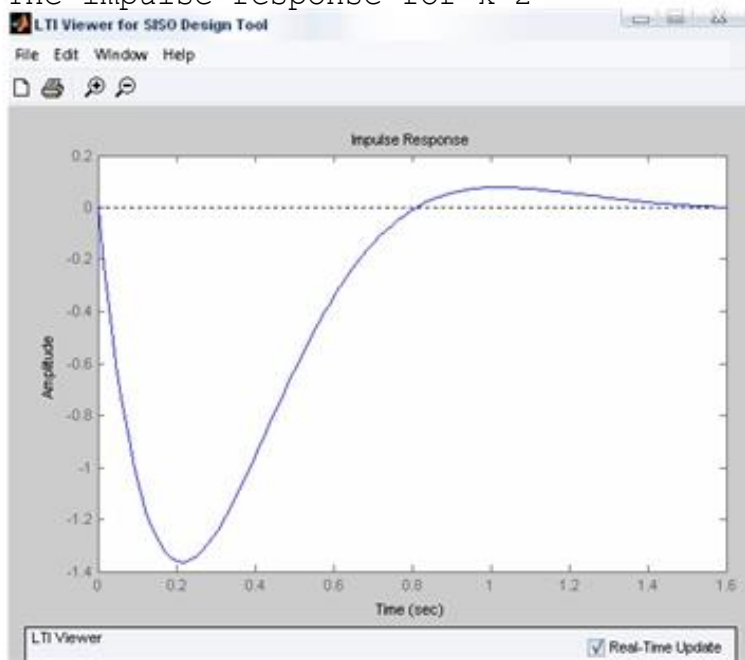


**Q#03:-** Add a pair of complex zeros to the system above.  
When the pair of complex zeros are added to the system then the sisotool result becomes:

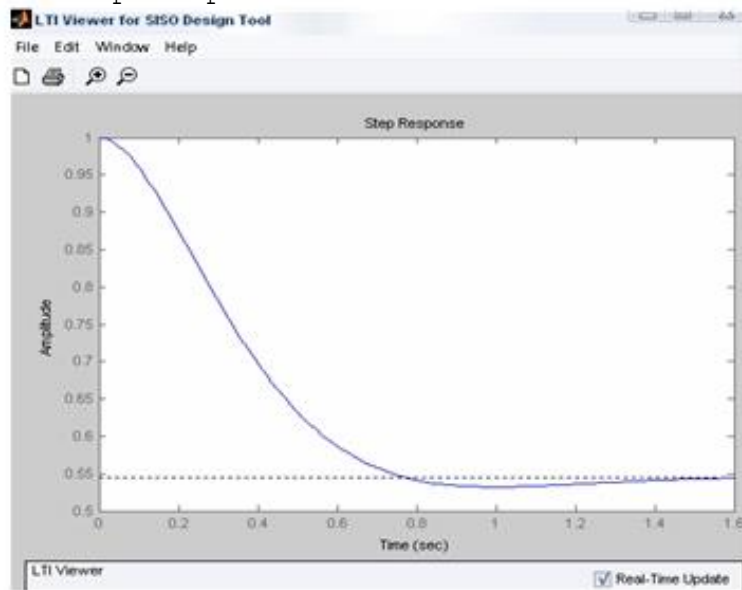


The range of  $K$  for which the system is stable:  
System is stable for  $K=0$  to  $\infty$   
The break in / break away points if any:  
The break away point is  $-2.56$

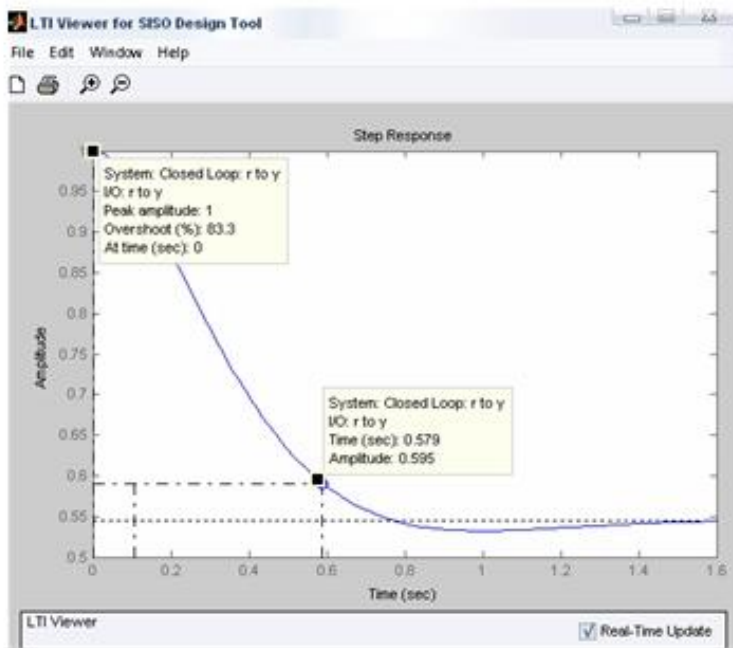
The impulse response for  $k=2$



The step response for  $k=3$

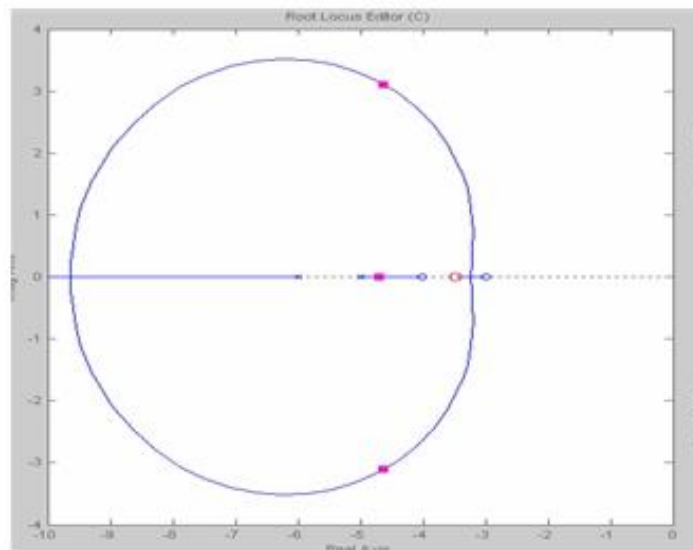


The rise time and peak response for  $k=3$   
Rise time is 0.418 and peak response is 1



**Q#04:-** Add real zeros to the system:

The real zeros are added to the system the sisotool result is given below.



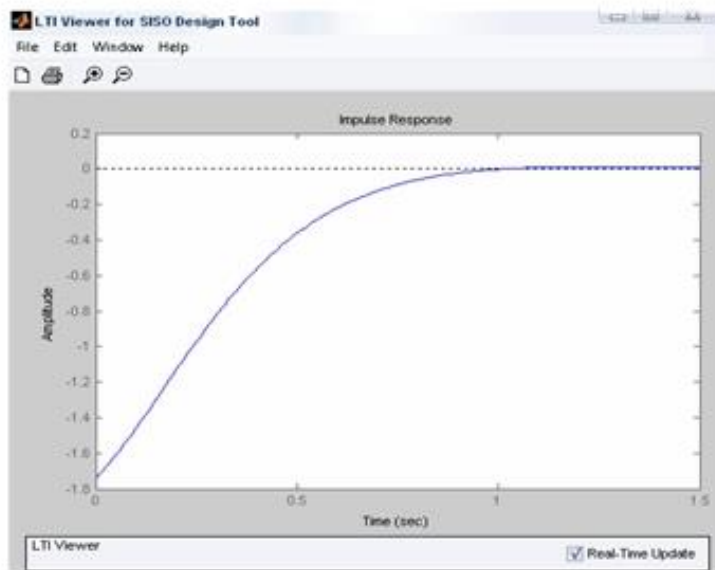
The range of  $k$  for which the system is stable:

System is stable for  $k=0$  to  $\infty$  and  $-3$  to  $-\infty$

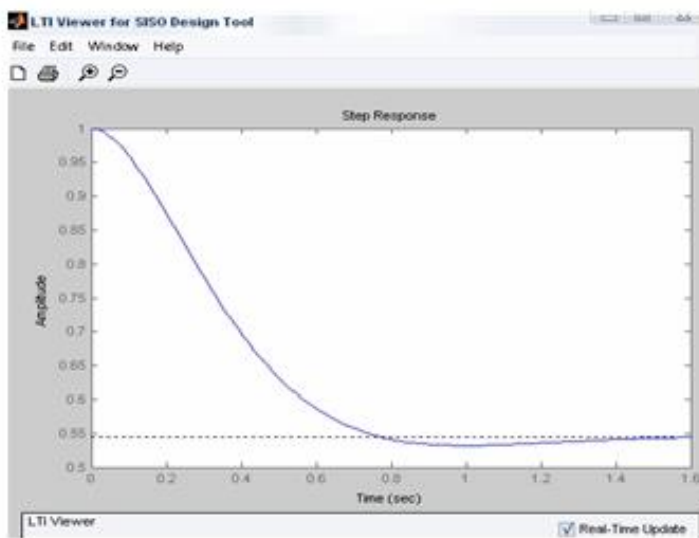
The break in or break away points if any:

Breakaway point is  $0.25$ , break in point is  $389$

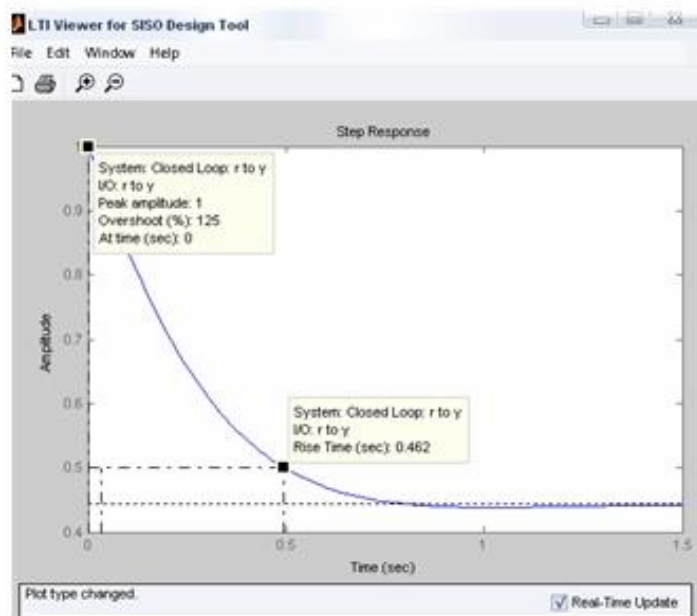
The impulse response for  $k=2$



The step response for  $k=3$

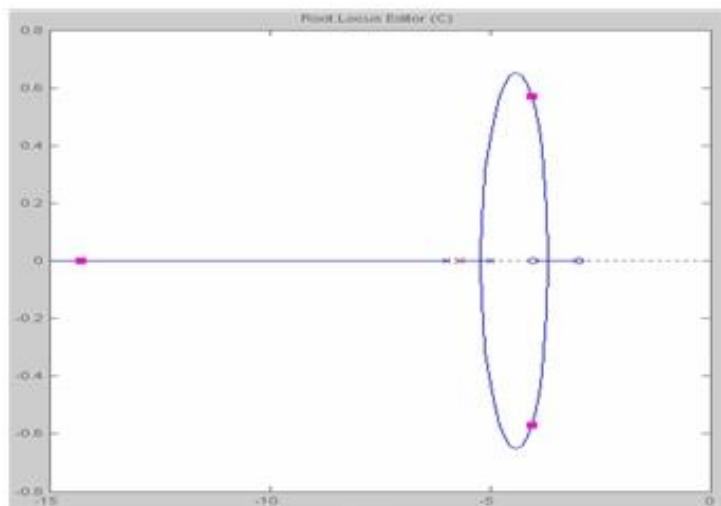


The rise time and peak response for  $k=3$   
 Rise time= 0.452      peak response=1



**Q#05:-**Add a real pole to the system to the above system.

When the real pole is added to the above system the sisotool result becomes:



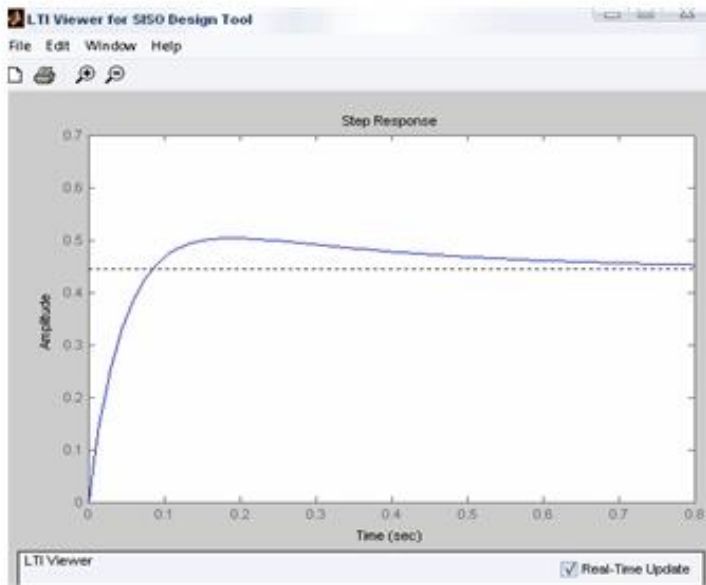
The range of  $k$  for which the system is stable.  
 The system is stable for  $k=0$  to  $\infty$

The break in or breakout points if any:

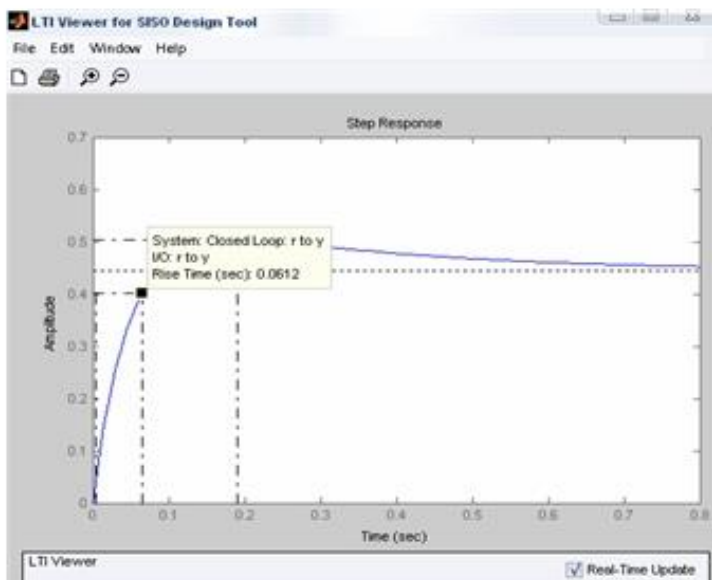
Break away point=0.00521 and break in= 4.97



The step response for  $k=3$



The rise time and peak response.  
Rise time= 0.0612 and peak response=0.1503



**Analysis:-**

Through root locus we can find the stability of a system. It also shows that at which points the system will be stable and which point the system is unstable.

Root Locus also shows rise time and peak time. It also inform about maximum overshoot.

# Lab 13

## System Design using Sisotool

**Task:** For  $K < 2$ , design a system (second order) with the following characteristics:

- Percent overshoot  $< 50$
- Damping ratio  $> 0.2$
- Stable system

**Introduction:**

**Percent overshoot:**

Systems may be stable system, unstable system and marginally stable system. A stable system may overshoot for some values at the start before coming to the stable level. Similarly in this lab a system is designed whose percent overshoot is  $< 50$ .

**Damping ratio:**

Damping ratio is a parameter that indicates that whether system is over damped ( $\zeta > 1$ ), under damped ( $\zeta < 1$ ) or critically stable ( $\zeta = 1$ ). In this lab a system is designed which must have damping ratio  $> 0.2$ .

**Stable system:**

Third condition which the system must satisfy is it must be stable for  $K < 2$ , also all the values (damping ratio and % overshoot) are set. It must be unstable for  $K \geq 2$ .

**Steps:**

1. First of all matlab code for the required system is written. Code is as follows:

**Code:**

```
num=[0 0 -1];
den=[1 3 2];
sys=tf(num,den)
sisotool(sys)
```

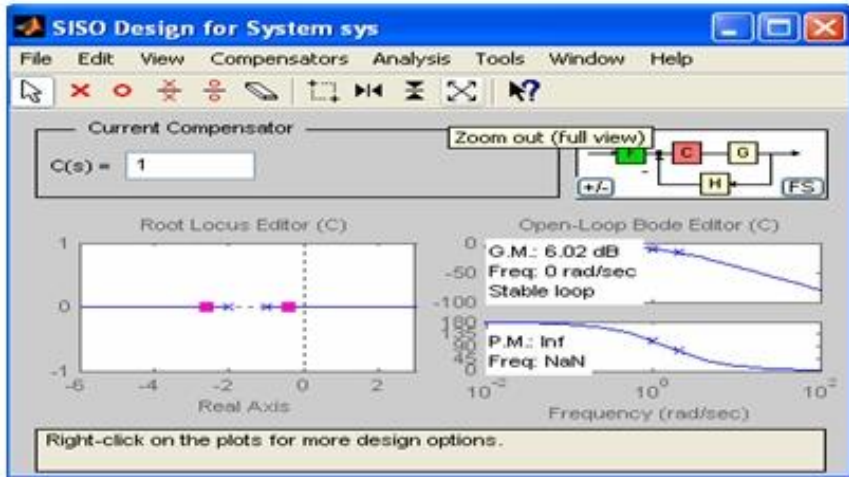
The system given by matlab as a result of writing above code is:

**Transfer function:**

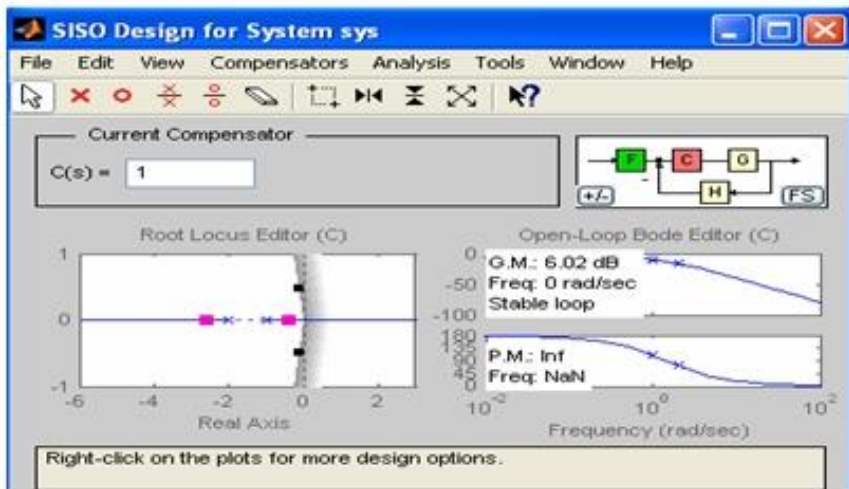
-1

$s^2 + 3s + 2$

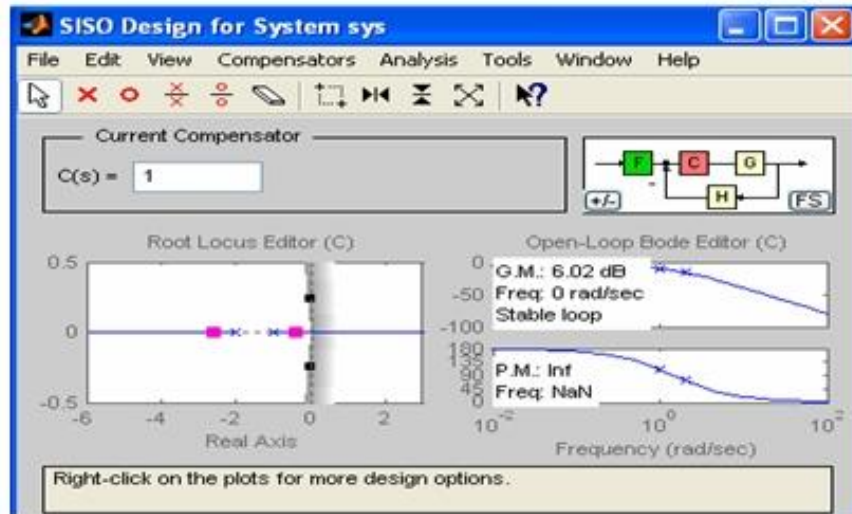
2. The program is run to see the system in sisotool which is



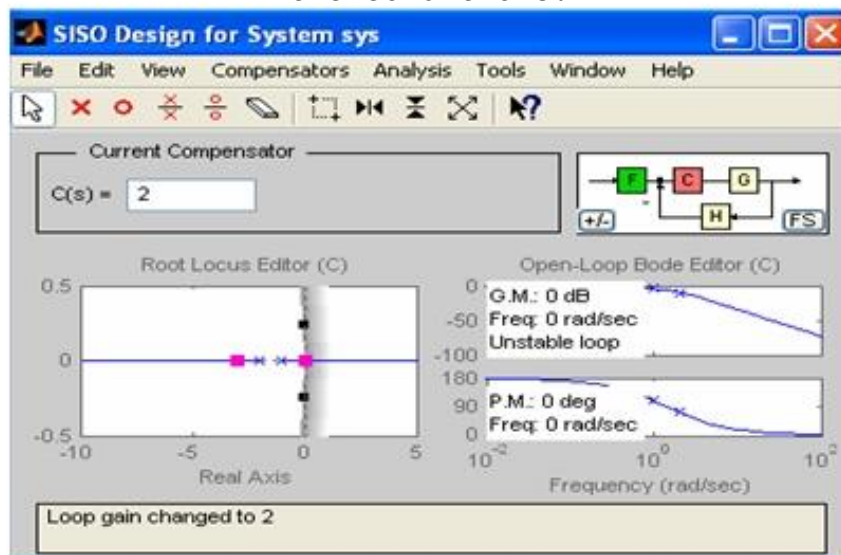
3. Value for % overshoot is set to less than 50. It is done by right click on the root locus window->Design constraints->New then plot appears as:



4. Value for damping ratio is set to greater than 0.2. It is done by right click on the root locus window->Design constraints->New then plot appears as:



5. After that stability of system is checked for different values of  $K$ . System is stable for all  $K < 2$ . It is unstable for  $K \geq 2$ . So this system fulfills all the conditions.



## Lab 14

### Bode Plot using MATLAB

1.  $G(s)H(s) = K / (s (s + 3) (s + 5))$

2.  $G(s)H(s) = K / ((s + 2) (s + 4) (s + 6))$

Using the transfer functions given above:

- A. Draw the Open loop Bode plots.
- B. Draw the Unity (-ve) Feedback Closed loop Bode Plots.
- C. Find the values of K for which the Open loop systems are stable.
- D. Find the values of K for which the Unity (-ve) Feedback Closed loop systems are stable.

#### Introduction:

A Bode plot, named for Hendrik Wade Bode, is usually a combination of a Bode magnitude plot and Bode phase plot: A Bode magnitude plot is a graph of log magnitude against log frequency often used in signal processing to show the transfer function or frequency response of an LTI system. It makes multiplication of magnitudes a simple matter of adding distances on the graph. Bode plot is a plot of gain versus frequency for a control loop, typically used to verify control loop stability, including phase margin.

#### Question # 01

$$G(s)H(s) = K/S(S+3)(S+5)$$

#### Procedure:

To show the bode plot of the given equation I wrote the Matlab code. I uses sisotool in the code for the plot.

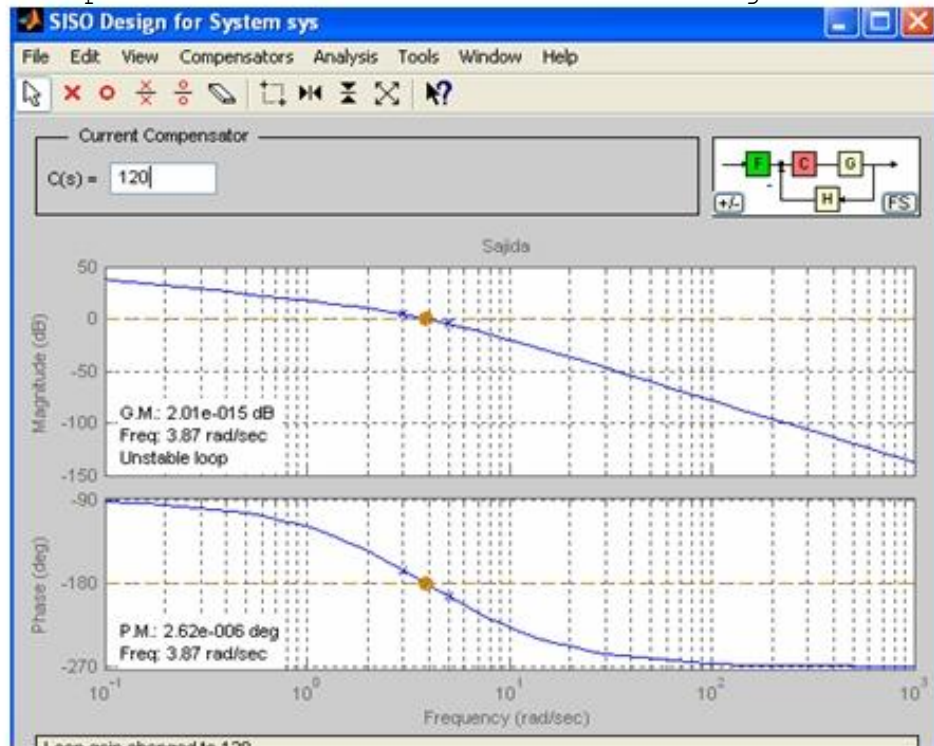
Running this program give me the bode plot of the given equation.

**Matlab code:**

```
Num = [ 1          ];  
Den = [          ];  
Sys=tf(num,den);  
Sisotool(sys)
```

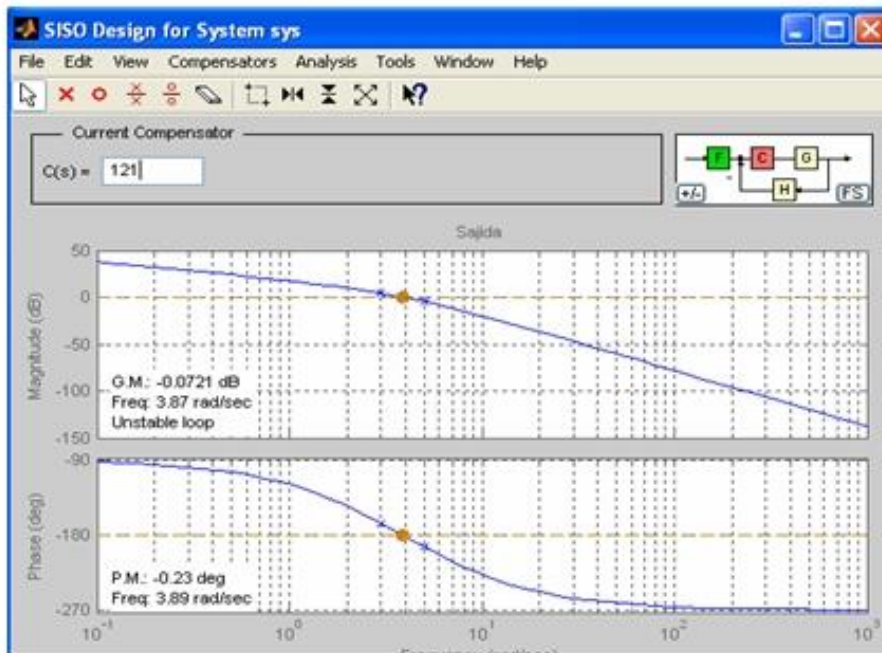
**System's stability:**

For  $K \leq 120$  the given system is stable. as its G.M and P.M are positive. as can be shown in the figure.



**System's instability:**

for  $K \geq 121$  the given system is unstable. as the G.M and P.M are negative.



### Question # 02

$$G(s)H(s) = K/(s+2)(s+4)(s+6)$$

#### Procedure:

To show the bode plot of the given equation I wrote the Matlab code. I used sisotool in the code for the plot. Running this program gives me the bode plot of the given equation.

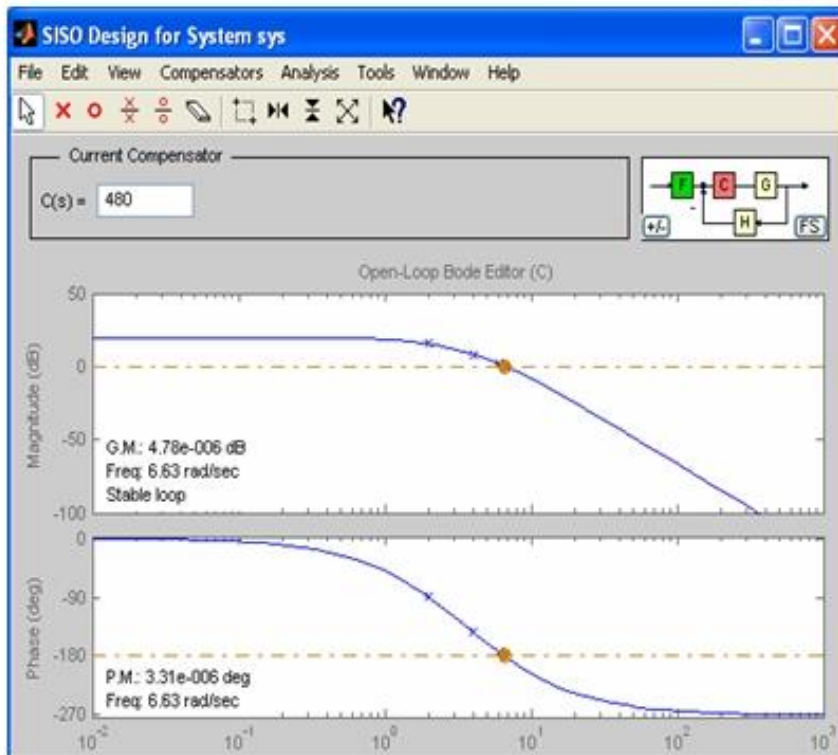
#### Matlab code:

```
Num = [          ];
Den = [          ];
Sys=tf(num,den);
sisotool(sys)
```

#### System's stability:

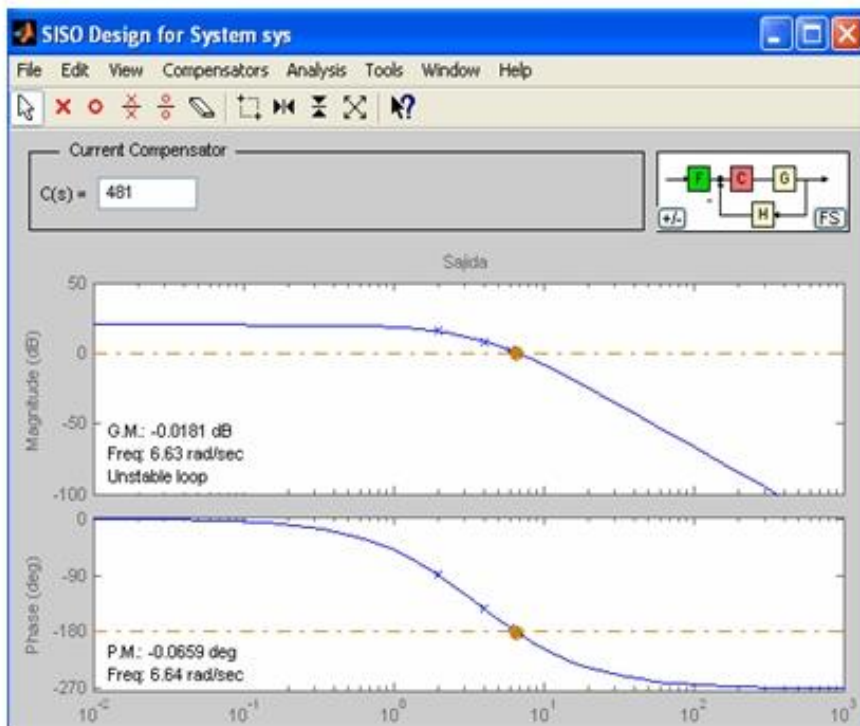
For  $K \leq 120$  the given system is stable, as its G.M and P.M are positive, as can be shown in the figure.





### System's instability:

for  $K \geq 121$  the given system is unstable. as the G.M and P.M are negative.



**Analysis:**

The Bode plot is an important tool for stability analysis of closed-loop systems. It is based on calculating the amplitude and phase angle for the transfer function. the given system becomes stable if the phase margin (P.M) and gain margin (G.m) is positive. and for negative it is vice versa. A commonly found statement about the Bode stability criterion is that it cannot be used if the frequency response of the open-loop system exhibits "nonmonotonic phase angles or amplitude ratios at frequencies higher than the first phase crossing of  $-180^\circ$ .

# Revision Version 3.0

(Revised by Engr. Muniba Ashfaq)

- 1) Lab 1: Practice 1.0 (MATLAB commands)
- 2) Lab 2 is swapped by Lab 3
- 3) Lab 3 (a) and (b) is the simulation of differential equations using MATLAB and Simulink respectively
- 4) Lab 4 is the simulation of higher order differential equation using MATLAB and Simulink
- 5) Lab 5 is the frequency domain modeling in MATLAB (A complete new lab is added to the manual)
- 6) Lab 6 is the Time domain modeling in MATLAB (A complete new lab is added to the manual)
- 7) Lab 7 is the Lab 3 of the previous lab manual and title changed to System Stability in MATLAB
- 8) Lab 8 is added to the manual for system interconnections in Simulink
- 9) Lab 9 is the new lab added to the manual for stability via Routh Hurwitz
- 10) Lab 10 is the new lab added to the manual for the analysis of steady state error using different test signals
- 11) Lab 11 is related to implementation of root locus using MATLAB
- 12) Lab 12 is related to implementation of root locus using sisotool
- 13) Lab 13 is related to the system design using sisotool
- 14) Lab 14 is related to the Bode Plot using MATLAB