Department of Electronics and Communication Engineering

PRACTICAL RECORD BOOK
21EC43
Signals and Control Systems
(Integrated subject, 3-0-2)

Name: ……

Semester: ……

USN: ……

VISION

The Electronics and Communication Engineering department shall impart quality technical education and entrepreneurship skills to develop creative individuals to face changing global scenario.

MISSION

To augment the national talent pool, with Electronics and Communication Engineers having all-encompassing technical knowledge, principled practices and nationalistic outlook.

Course Outcome (COs)

Learning Levels: Re (Remember) **Un** (Understand) **Ap** (Apply) **An** (Analysis) **Ev** (Evaluate) **Cr** (Create)				
At the end of the course, the student will be able to		**Learning Levels**	**PO**	**PSO**
1.	Distinctly *classify* different types of continuous and discrete time signals and systems as per their properties.	**Un, Ap**	1, 2, 3, 5	1, 2
2.	*Develop* transfer function of LTI systems by various methods and *identify* the type of system stability based on system responses.	**Un, Ap**	1, 2, 4, 5	1
3.	Draw *inference* about quality of different types of signals with the help of mathematical functions like correlation, ESD and PSD.	**Un, An**	1, 2, 3, 12	1, 3

Program Outcome of this course (POs)

Sl.no.		**PO No.**
1.	**Engineering Knowledge:** Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.	**1**
2.	**Problem Analysis:** Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.	**2**
3.	**Design/ Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.	**3**
4.	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.	**4**
5.	**Modern Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.	**5**
6.	**The Engineer and Society:** Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.	**6**
7.	**Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.	**9**
8.	**Life-long Learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.	**12**

LABORATORY CERTIFICATE

This is to certify that …… has satisfactorily completed the course of experiments in 21EC43-Signals and Control Systems prescribed by the department of Electronics and communication Engineering for the fourth semester of UG program during the year 2023-24.

Signature of Faculty-In-Charge
Date:

Signature of Head of the Department

Marks Obtained			
Max. Marks	**15 (Conduction)**	**25 (LAB Test)**	**40 (Total)**

INDEX SHEET

Expt. No.	Date	Title	CO	PO	Conduction (5 M)	Results (5M)	Viva (5M)	Sign
1		Introduction to control system functions and toolbox.	1	1,2, 3,5, 9				
2		Software based generation of standard test signals. Basic operation like delay, advance, fold, time scaling and amplitude scaling etc. on these basic signals.	1	1,2, 5,9, 12				
3		Mathematical modelling of 2nd order mechanical system and its equivalent FV and FI analogy. Determination of related Impulse and Step responses.	1	1,2, 3,5, 9,12				
4		Natural responses of 1st order RL and RC systems. Step and Ramp responses of RL, RC, and RLC systems. Determination of their time domain specifications.	1	1,2, 3,5, 9				
5		Determination of transfer function from system block diagram. Finding systems responses for various test signals.	2	1,2, 3,5, 9,12				
6		Computation of error coefficients and steady state errors for Type 0, Type I, Type II, and Type III systems with Step, Ramp and Parabolic inputs.	2	1,2, 3,5, 9				
7		System time domain response and stability analysis as per pole positions in 's-domain'. Lead, lag and lead-lag systems – their pole positions and time domain responses.	2	1,2, 3,5, 9				
8		Root locus plot for different systems with varying system gain.	2	1,2, 5,9, 12				
9		Bode plot-based determination of system stability for different physical systems. Systems with lead, lag and lead-lag compensation are also to be considered.	2	1,2, 5,9, 12				
10		Fourier Series based signal analysis and synthesis and generation of Fourier Spectra	3	1,2, 5,9				
				AVERAGE MARKS				

CIE Marks Distribution

S.No	Details	Max Marks	Marks Obtained	Total	Sign
1	Conduction and Journal	**5**			
2	Results, Graph, Conclusion and Outcome	**5**			
3	Viva Voce	**5**			

CO		PO	

Experiment No: 1 **Date:**

Title: Introduction to control system functions and toolbox.

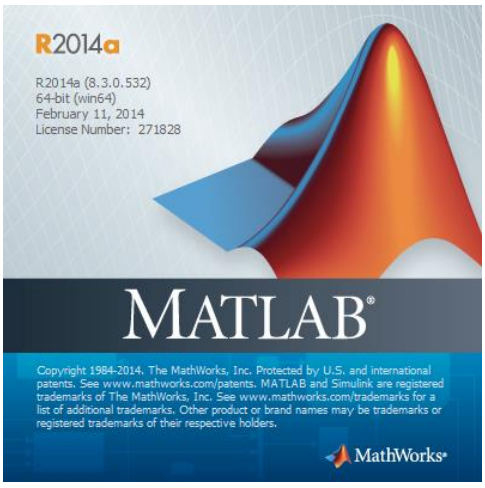
Objective: To explore MATLAB toolbox and programing.

Software Needed: MATLAB

Theory and Procedure:

Introduction to MATLAB Programming

This experiment gives introduction to MATLAB which is an abbreviation of **MAT**rix**LAB**oratory. It is registered trademark of computer software, version **R2021a (8.3.0.532)** developed by the Math Works Inc. The software is widely used in various science and engineering fields. It is an interactive program for numerical computation and data visualization. MATLAB is supported by Unix, Macintosh, and Windows environments. Contact thematchworks.cm for more information related to MATLAB. A windows version of MATLAB is assumed here. MATLAB gives flexible environment for technical and mathematical computing, data visualization etc. It helps in explore data, create algorithms and custom tools that provide early insights and competitive advantage. It is known for its highly optimized matrix and vector calculations. MATLAB offers an intuitive language for expressing problems and their solutions both mathematically and visually. It's typical uses include



- Numerical computation and algorithm development
- Symbolic computation (with built-in symbolic math functions)
- Modeling, simulation and prototyping
- Data analysis and signal processing (analog and digital signal processing)
- Engineering graphics and scientific visualization.

Procedure:

Important things to check in the first session of MATLAB based programming

1) **Display Window –** MATLAB has three display windows

i) **Command Window**–used to enter commands & data to display plots & graphs

ii) **Graphic Window** – used to display plots and graphs

iii) **Edit Window** – programming is done here and saved as M files. M files contain program or script of MATLAB command

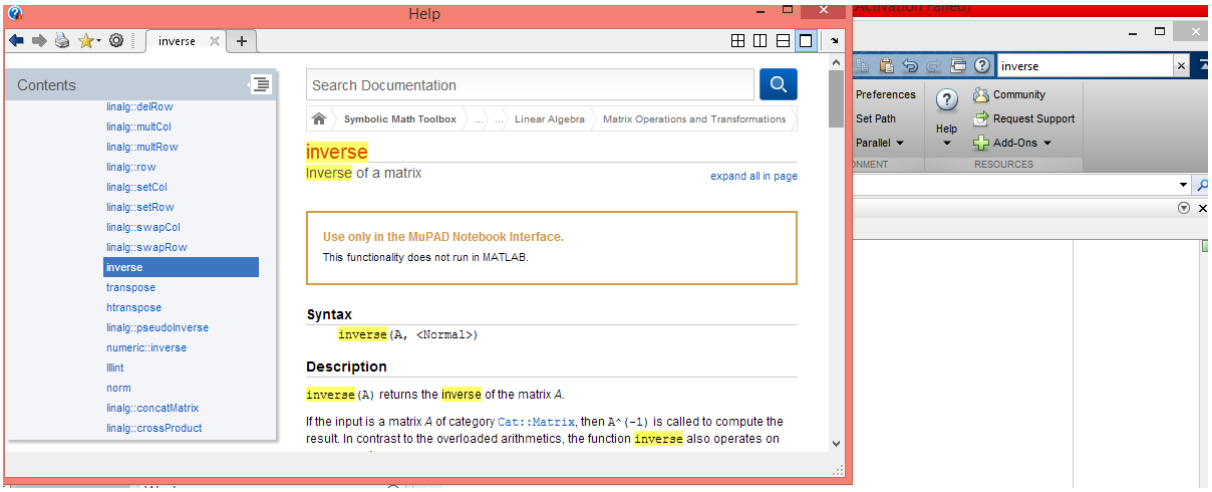
2) MATLAB commands are case sensitive and lower case words are used throughout. To execute M file, simply enter the name of the file without its extension. M file name should not be same as unction name, in that case the M file will not get stored.

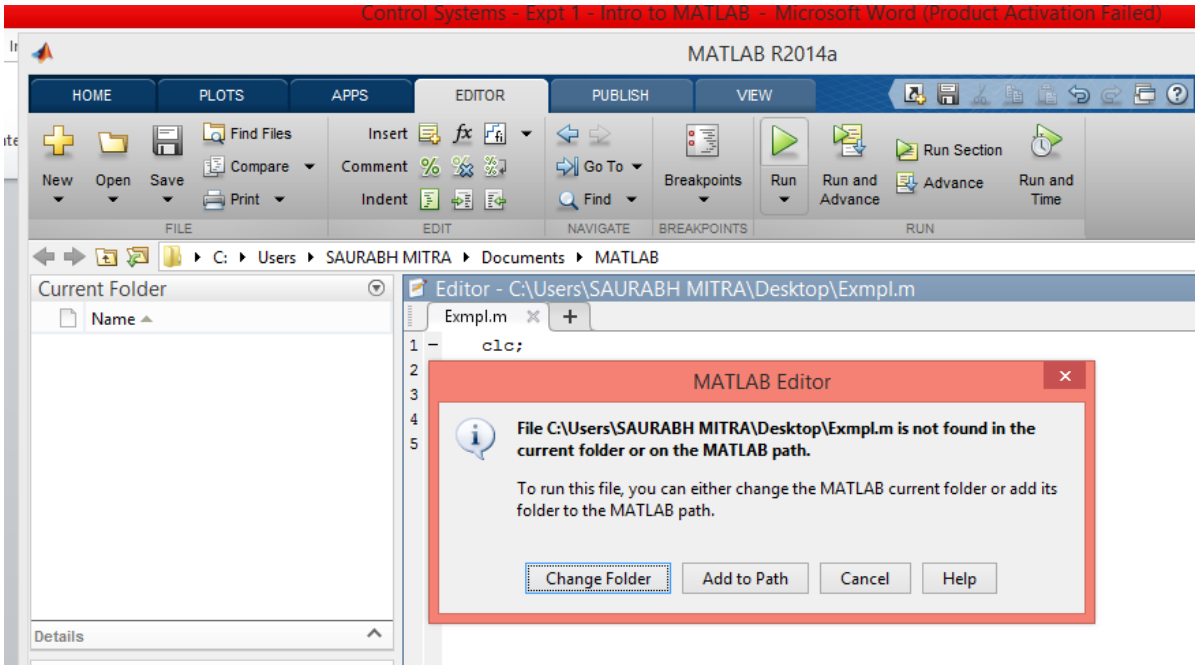
3) Semicolon (;) – If it is typed at the end of a command, then the output of the command is calculated, but not gets displayed.

4) % typed at the beginning of a line – this line will be designated as a comment. It will not be executed as a command.

5) clc command – 'clc' command given and 'enter' button pressed after that cleans the command window. Once clc is executed, a clear window is displayed.

6) Help command – MATLABhas a host of built-in functions. We do not have to remember the functions and its syntax while programming. Go to help menu and type which function is required. MATLABhelp will return with its syntax and related examples. Hence it is very much user friendly. Eg. For inverse of a matrix go to Help and type inverse, it will return the function 'inv' and its syntax and related examples.




Control Systems - Expt 1 - Intro to MATLAB - Microsoft Word (Product Activation Failed)
MATLAB R2014a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
New
Open
Save
Find Files
Compare
Print
Insert
Comment
Indent
Go To
Find
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
EDIT
NAVIGATE
BREAKPOINTS
RUN
C: ▸ Users ▸ SAURABH MITRA ▸ Documents ▸ MATLAB
Current Folder
Name
Details
Editor - C:\Users\SAURABH MITRA\Desktop\Exmpl.m
Exmpl.m
1 – clc;
MATLAB Editor
File C:\Users\SAURABH MITRA\Desktop\Exmpl.m is not found in the current folder or on the MATLAB path.
To run this file, you can either change the MATLAB current folder or add its folder to the MATLAB path.
Change Folder
Add to Path
Cancel
Help

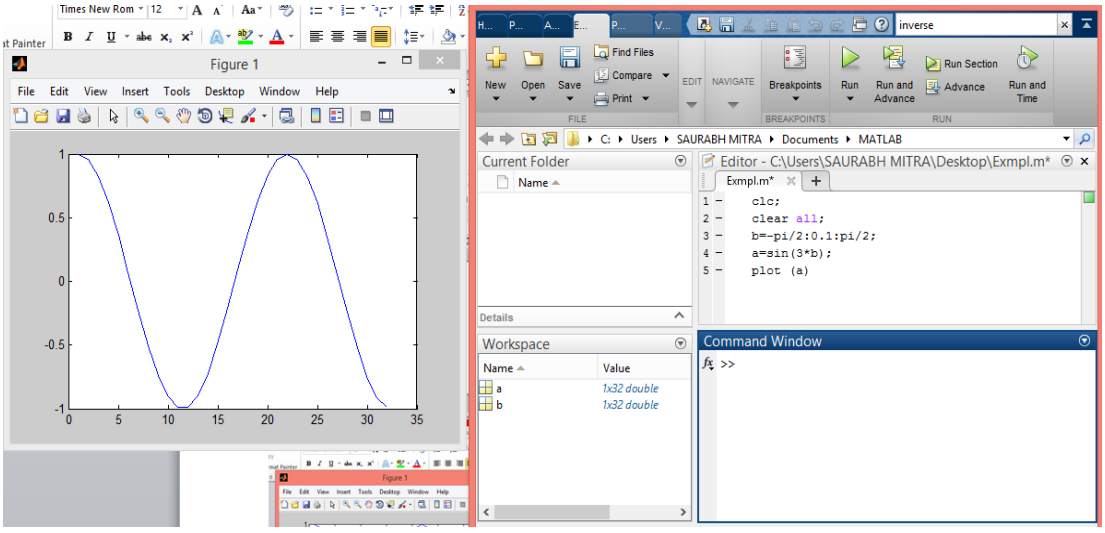
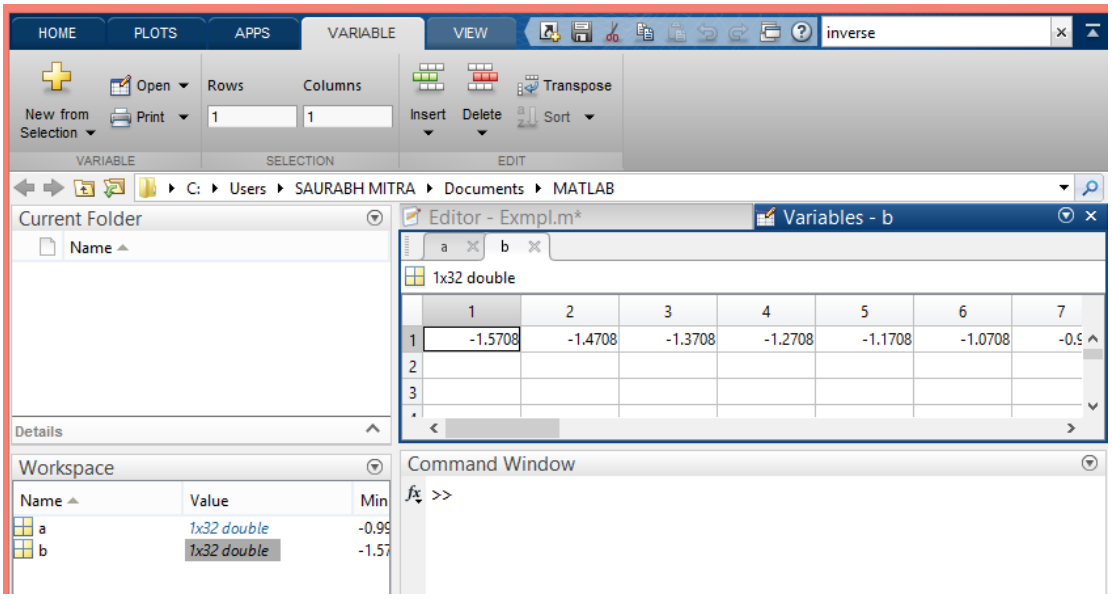
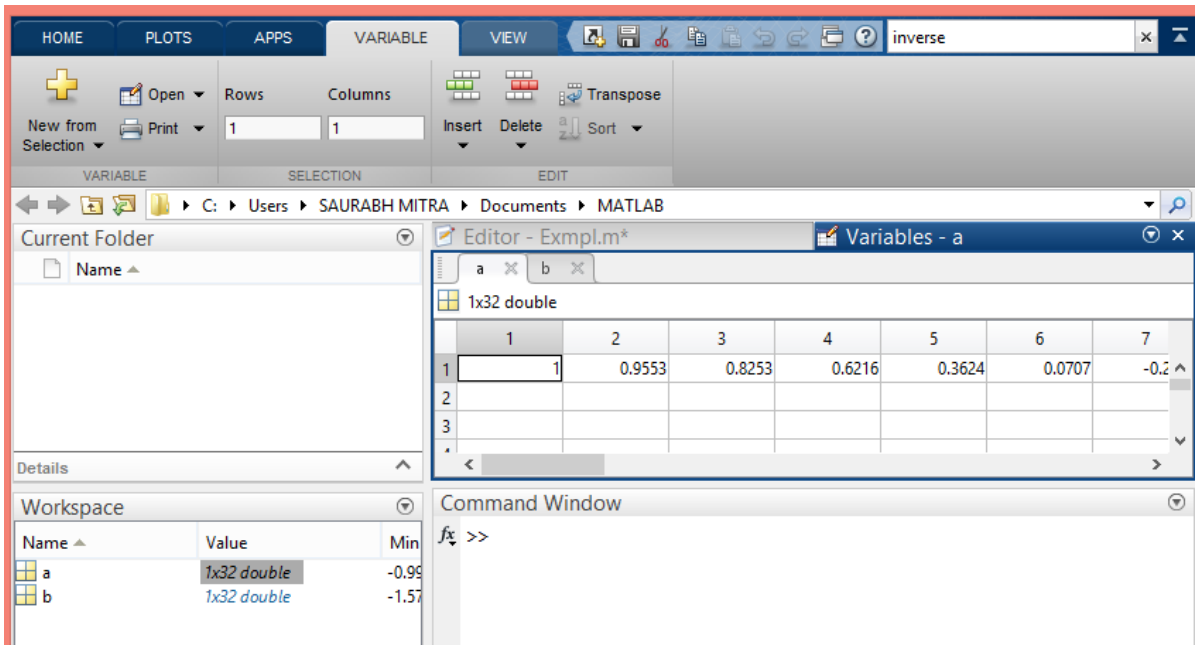


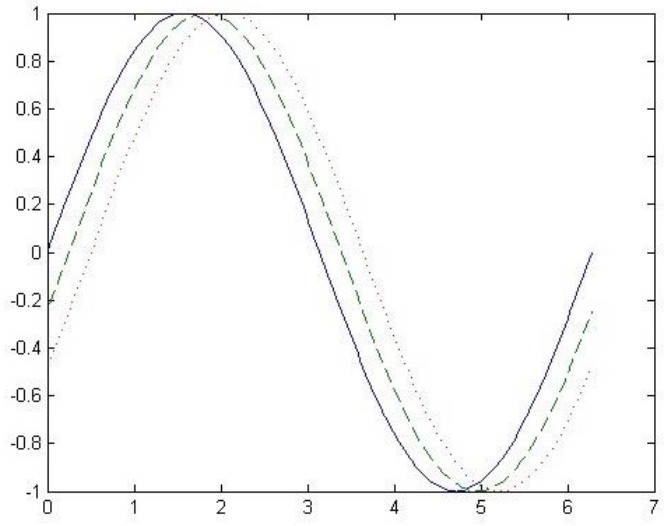
Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.5
0
-0.5
-1
0 5 10 15 20 25 30 35
Current Folder
Name
Editor - C:\Users\SAURABH MITRA\Desktop\Exmpl.m*
Exmpl.m*
1 – clc;
2 – clear all;
3 – b=-pi/2:0.1:pi/2;
4 – a=sin(3*b);
5 – plot (a)
Workspace
Name
Value
a 1x32 double
b 1x32 double
Command Window
>>


HOME
PLOTS
APPS
VARIABLE
VIEW
inverse
New from Selection
Open
Print
Rows
Columns
Insert
Delete
Transpose
Sort
VARIABLE
SELECTION
EDIT
C: ▸ Users ▸ SAURABH MITRA ▸ Documents ▸ MATLAB
Current Folder
Name
Editor - Exmpl.m*
Variables - b
a
b
1x32 double
1 2 3 4 5 6 7
1 -1.5708 -1.4708 -1.3708 -1.2708 -1.1708 -1.0708
Details
Workspace
Name Value Min
a 1x32 double
b 1x32 double
Command Window
>>


HOME
PLOTS
APPS
VARIABLE
VIEW
inverse
New from Selection
Open
Print
Rows
Columns
Insert
Delete
Transpose
Sort
VARIABLE
SELECTION
EDIT
C: ▸ Users ▸ SAURABH MITRA ▸ Documents ▸ MATLAB
Current Folder
Name
Editor - Exmpl.m*
Variables - a
a
b
1x32 double
1 2 3 4 5 6 7
1 1 0.9553 0.8253 0.6216 0.3624 0.0707
Details
Workspace
Name Value Min
a 1x32 double
b 1x32 double
Command Window
>>


```
x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);

figure
plot(x,y1,x,y2,'--',x,y3,':')
```



Create Line Plot From Matrix

Define `Y` as the 4-by-4 matrix returned by the `magic` function.

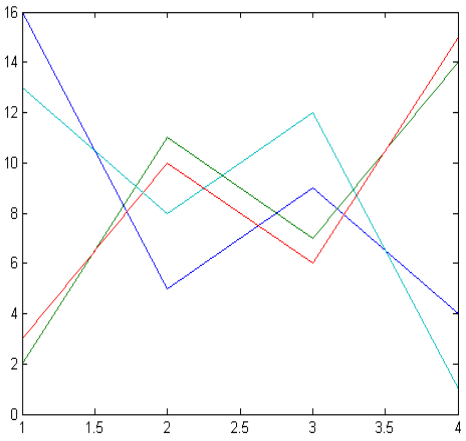
```
Y = magic(4)
```

```
Y =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

Create a 2-D line plot of `Y`. MATLAB® plots each matrix column as a separate line.

```
figure
plot(Y)
```



Specify Line Style, Color, and Marker

Plot three sine curves with a small phase shift between each line. Use a green line with no markers for the first sine curve. Use a blue dashed line with circle markers for the second sine curve. Use only cyan star markers for the third sine curve.

```
x = 0:pi/10:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);

figure
plot(x,y1,'g',x,y2,'b--o',x,y3,'c*')
```

Results and Graph:

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 2 **Date:**

Title: Software based generation of standard test signals. Basic operation like delay, advance, fold, time scaling and amplitude scaling etc. on these basic signals.

Objective: To learn and perform waveform generation in MATLAB.

Software Needed: MATLAB

Problem Statement:

Write the programs to generate the waveforms at different cases as given below.

Case 1 – change the axis for every wave and see different portion of wave with more resolution. One such example of axis change is given for impulse wave for the first plot. For remaining all 8 plots, axis may be change to see the wave under higher resolution (zoom).

```
%%%% code for all basic waveform generation
clc; clear all; close all;
%%% define time from -10 seconds to 10 seconds - non causal time frame
%%% define sampling time as 0.01 seconds
t=(-10:0.01:10)';
%%% impulse at t=0
impulse = t==0;
subplot(331);
plot(t,[impulse], 'red');
xlabel('Time');ylabel('Impulse Amplitude');
title('Impulse Waveform');
axis([-10 10 -0.2 1.2])
%%% zoom the impulse to see it more closely by uncommenting next axis
command
%%% comment the previous axis([-10 10 -0.2 1.2]) command now
% axis([-0.4 0.4 -0.2 1.2])
%%% step at t=0
unitstep=t>=0;
subplot(332); plot(t,[unitstep],'blue');
xlabel('Time');ylabel('Unitstep Amplitude');
title('Unitstep Waveform');
axis([-10 10 -0.3 1.3])
```

```
%%% ramp starting from t=0
ramp=t.*unitstep;
subplot(333); plot(t,ramp,'magenta');
xlabel('Time');ylabel('Ramp Amplitude');
title('Ramp Waveform');
%%axis([0 10  0 10])
%%% parabolic starting from t=0
parabolic=(t).^2.*unitstep;
subplot(334); plot(t,parabolic,'green');
xlabel('Time');ylabel('Parabolic Wave Amplitude');
title('Parabolic Waveform');
axis([-10 10  0 105])
%%% All 4 above waveforms displayed together in one plot
subplot(335); plot(t,[impulse unitstep ramp parabolic]);
title ('All four Waveforms in one plot');
xlabel('Time'); ylabel('Signal Amplitudes');
axis([-1 1 -0.1 1.1])
%%% exponential waveforms(increasing &decreasing both)
increasing_expo1=exp(-1.2*t).*unitstep;
decreasing_expo1=exp(0.2*t).*unitstep;
subplot(336); plot(t,increasing_expo1,'r',t,decreasing_expo1,'b');
xlabel('Time');ylabel('Exponential Amplitudes');
title('Exponential Waveforms');
axis([0 10 -1.3 4.3])
%%% square waveform of 7.5 volt_peak and 3 Hz frequency
f=3; sqwave = 7.5*square(2*pi*f*t);
subplot(337);plot(t,sqwave,'black');
xlabel('Time');ylabel('Square Wave Amplitude');
title('Square Waveform')
axis([0 1 -8 8])
%%%sinewave of 5 Hz and and cosine wave of 3 Hz
f1=5; sinewave = 4*sin(2*pi*f1*t);
f2=3; cosewave = 3.5*cos(2*pi*f2*t);
subplot(338); plot(t,sinewave,'c',t,cosewave,'m');
xlabel('Time');ylabel('Sine-Cosine Amplitudes');
title('Sine and Cosine Waves');
axis([0 1 -4.5 4.5])
%%%sinewave of 5 Hz and and cosine wave of 3 Hz
triangular_wave=triang(200)
subplot(339); plot(triangular_wave,'black');
xlabel('Time');ylabel('Triangular Amplitudes');
```

```
title('Triangular Wave');
axis([0 200 -0.2 1.1])
```

Case 2 – Baseband Sine wave and its second , third and fourth harmonics displayed together. For clearer distinguishment, legend command is used and for clearer visibility, four different colours are used for 4 different graphs.

```
%%% code for sine waves generation of different harmonics
clc; clear all; close all;
t=(0:0.01:2*pi)';

%%% Baseband signal is 1 radians/second sine wave
sine1=sin(t); plot(t,sine1,'red');
holdon;
%%% Second harmonics is of 2 radians/second
sine2=sin(2*t); plot(t,sine2,'cyan');
%%% Third harmonics is of 3 radians/second
sine3=sin(3*t); plot(t,sine3,'green');
%%% Fourth harmonics is of 4 radians/second
sine4=sin(4*t); plot(t,sine4,'black');
holdoff;
xlabel('Time');ylabel('Sinusoidal Amplitudes');
title('Baseband and 3 Harmonics Sinusoidal Waves ');
legend('sine1','sine2','sine3','sine4')
axis([0 2*pi -1.3 1.3])
gridon;
```

Case 3 – Baseband Cosine wave and its second , third and fourth harmonics displayed together. For clearer distinguishment, legend command is used and for clearer visibility, four different colours are used for 4 different graphs.

```
%%% code for Cosine waves generation of different harmonics
clc; clear all; close all;
t=(0:0.01:2*pi)';

%%% Baseband signal is 1 radians/second cosine wave
cosine1=cos(t); plot(t,cosine1,'red');
holdon;
%%% Second harmonics is of 2 radians/second
cosine2=cos(2*t); plot(t,cosine2,'cyan');
%%% Third harmonics is of 3 radians/second
cosine3=cos(3*t); plot(t,cosine3,'green');
%%% Fourth harmonics is of 4 radians/second
cosine4=cos(4*t); plot(t,cosine4,'black');
```

```
holdoff;
xlabel('Time');ylabel('Cosinewave Amplitudes');
title('Baseband and 3 Harmonics Cosine Waves ');
legend('cosine1','cosine2','cosine3','cosine4')
axis([0 2*pi -1.3 1.3])
gridon;
```

Case 4 – Basic Impulse, Delayed Impulse and Advanced Impulse Wave generation

```
%%%% code for shift operations on impulse signal
clc; clear all; close all;
%%% define time from -5 s to 5 seconds
%%% define sampling time as 0.01 seconds
t=(-5:0.01:5)';
%%% impulse at t=0
impulse = t==0;
plot(t,[impulse],'red');
holdon;
%%% delayed or right shifted impulse at t=2
delayed_impulse=(t-2)==0;
plot(t,[delayed_impulse],'blue');
%%% advanced or left shifted impulse at t=4
advanced_impulse=(t+4)==0
plot(t,[advanced_impulse],'green');
holdoff;
xlabel('Time');ylabel('Impulse Amplitude');
title('Basic and Time Shifted Impulse Waveforms'); axis([-5 5 -0.1 1.05]);
legend('impulse','delayedimpulse','advanced impulse')
```

Case 5 – Basic Unitstep, Delayed, advanced and Time Inverted Unitstep signals

```
%%%% code for shift operations on unit step signal
clc; clear all; close all;
%%% define time from -10 s to 10 seconds
%%% define sampling time as 0.01 seconds
t=(-10:0.01:10)';
%%% basic step at t=0
subplot(221);
unitstep=t>=0;
plot(t,[unitstep],'red');hold on;
xlabel('Time');ylabel('Unitstep Amplitude');
title('Basic Unitstep Waveform');
axis([-10 10 -0.3 1.3])
```

```
%%% delayed step by 6 units at t=-6
subplot(222);
delayed_step=t-6>=0;
plot(t,delayed_step,'blue')
xlabel('Time');ylabel('Delayed Unitstep Amplitude');
title('6 Second Delayed Unitstep Waveform');
axis([-10 10 -0.3 1.3])

%%% advanced step by 4 units at t=4
subplot(223);
advanced_step=t+4>=0;
plot(t,advanced_step,'magenta')
xlabel('Time');ylabel('Advanced Unitstep Amplitude');
title('4 second Advanced Unitstep Waveform');
axis([-10 10 -0.3 1.3])

%%% time inverted unitstep waveform
subplot(224);
inverted_step=-t>=0;
plot(t,inverted_step,'black')
xlabel('Time');ylabel('Time Inverted Unitstep Amplitude');
title('Time Inverted Unitstep Waveform');
axis([-10 10 -0.3 1.3])
```

Graph:

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 3 **Date:**

Title: Natural and step responses of 1st order RL and RC systems. Determination of their time domain specifications.

Objective: To learn Natural and step response of RL and RC systems

Software Needed: MATLAB

Problem Statement:

Write the programs to generate the Natural and Step response of 1st order RL and RC systems.

Case 1 –Natural Response of RC Circuit

```
clc; clear all; close all;
t=(0:0.01:16); V0=1;
C=1; R=1; Tou=R*C;
VC=V0*exp(-t/Tou);
IR=VC/R; iC=-IR;
subplot(211); plot(t,VC); grid on;
title ('Capacitive Voltage Graph')
legend ('Capacitive Voltage')
xlabel('Time');ylabel('Voltage Amplitude');
title('RC Circuit Natural Response - Capacitive Voltage');
subplot(212); plot(t,IR,'red',t,iC,'blue'); grid on;
xlabel('Time');ylabel('Current Amplitudes');
title ('Resistive Current and Capacitive Current Graphs')
legend ('Resistive Current', 'Capacitive Current')
```

***Natural Response RC Circuits for 3 different Time Constants**

```
clc; clear all; close all;
t=(0:0.01:30); V0=1;
C=1; R1=1; R2=5; R3=10;
Tou1=R1*C; v1=V0*exp(-t/Tou1);
Tou2=R2*C; v2=V0*exp(-t/Tou2);
Tou3=R3*C; v3=V0*exp(-t/Tou3);
plot(t,v1,'black', t,v2,'blue',t,v3,'red');
legend ('Smallest Tou - FASTEST', 'Moderate Tou', 'Highest Tou - SLOWEST')
```

```
xlabel('Time');ylabel('RC Natural Response Voltage Amplitudes');
title('RC Natural Responses with Different Time Constants & System Fastness');
gridon;
```

Case 2 –Step Response of RC Circuit

```
clc; clear all; close all;
t=(0:0.01:15); V0=0; C=1; R=1; Tou=R*C; Vs=1;

vC=Vs+((V0-Vs)*exp(-t/Tou));
iR=Vs/R*exp(-t/Tou);
subplot(211); plot(t,vC,'red', t, iR,'blue'); grid on;
xlabel('Time');ylabel('Capacitive Voltage &Capacitve Current');
title ('RC Step Response - Capacitive Voltage and Capacitive Current')
legend ('Capacitive Voltage','Capacitive Current')

vR=R*iR;
subplot(212); plot(t,vR,'black'); grid on;
xlabel('Time');ylabel('Resistive Voltage Amplitude');
title ('RC Step Response - Resistive Voltage')
legend ('Resistive Voltage')
```

RC Step Responses with Different Time Constants

```
t=(0:0.01:12); V0=0; Vs=1;
C=1; RA=0.5; RB=1; RC=2;
TA=RA*C; vA=Vs+((V0-Vs)*exp(-t/TA));
TB=RB*C; vB=Vs+((V0-Vs)*exp(-t/TB));
TC=RC*C; vC=Vs+((V0-Vs)*exp(-t/TC));

plot(t,vA,'black', t,vB,'blue',t,vC,'red');
legend ('Smallest Tou - FASTEST','ModerateTou', 'Highest Tou - SLOWEST')
xlabel('Time');ylabel('Voltage Output for Step I/P');
title('RC Step I/P - Voltage Responses with Different Time Constants & System
Fastness');
gridon;

%%% iA=I0*exp(-t/TA); iB=I0*exp(-t/TB); iC=I0*exp(-t/TC);
```

Case 3- Natural Response of RL Circuit

```
clc; clear all; close all;
t=(0:0.01:16); I0=1;
```

```
L=1; R=1; Tou=L/R;
iL=I0*exp(-t/Tou);
subplot(211); plot(t,iL); grid on;
title ('Inductive Current Graph')
xlabel('Time');ylabel('Current Amplitude');
title('RL Circuit Natural Response - Inductive Current');
legend ('Inductive Current')
vR=I0*R*exp(-t/Tou);
vL=-vR;
subplot(212); plot(t,vR,'red',t,vL,'blue'); grid on;
xlabel('Time');ylabel('Voltage Amplitudes');
title ('Resistive Voltage and Inductive Voltage Graph')
legend ('Resistive Voltage', 'Inductive Voltage')
```

Natural Response for RL Circuit with Different Time Constants

```
clc; clear all; close all;
t=(0:0.01:7); I0=1; L=1;
R1=1; Tou1=L/R1; i1=I0*exp(-t/Tou1); % Tou1 = 1
R2=2; Tou2=L/R2; i2=I0*exp(-t/Tou2); % Tou2 = 0.5
R3=5; Tou3=L/R3; i3=I0*exp(-t/Tou3); % Tou3 = 0.2
plot(t,i1,'black', t,i2,'blue',t,i3,'red');
legend ('Highest Tou - SLOWEST','ModerateTou', 'Smallest Tou - FASTEST')
xlabel('Time');ylabel('RL Circuit Natural Response Current Amplitudes');
title('RL Circuit Natural Responses with Different Time Constants & System
Fastness');
gridon;
```

Case 4-Step Response of RL Circuit

```
clc; clear all; close all;
t=(0:0.01:15); I0=0; L=1; R=1; Tou=L/R; Vs=1;

iL=Vs/R+((I0-(Vs/R))*exp(-t/Tou));
vL=Vs*exp(-t/Tou);
subplot(211); plot(t,iL,'red', t, vL,'blue');
legend ('Inductive Current','Inductive Voltage'); grid on;
xlabel('Time');ylabel('Inductive Current and Inducitve Voltage');
title ('Inductive Current and Inductive Voltage Graphs - RL Step Response')
vR=iL*R;
subplot(212); plot(t,vR,'black');
legend ('Resistive Voltage'); grid on;
xlabel('Time');ylabel('Resistive Voltage Amplitude');
```

```
title ('RL Step Response - Resistive Voltage Response')
```

RL Step Responses with Different Time Constants

```
t=(0:0.01:12)'; I0=0; Vs=1;
L=1; RA=0.5; RB=1; RC=2;
TA=L/RA; iA=Vs/RA+((I0-Vs/RA)*exp(-t/TA));
TB=L/RB; iB=Vs/RB+((I0-Vs/RB)*exp(-t/TB));
TC=L/RC; iC=Vs/RC+((I0-Vs/RC)*exp(-t/TC));
plot(t,iA,'black', t,iB,'blue',t,iC,'red');
legend ('Highest Tou - SLOWEST','ModerateTou', 'Smallest Tou - FASTEST')
xlabel('Time'); ylabel('RL Step Response Current Amplitudes');
title('RL Step Responses with Different Time Constants & System Fastness');
gridon;
```

Graph:

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 4 Date:

Title: Second order time domain step response for different damping conditions, for electrical and mechanical system.

Objective: To learn and perform Second order time domain step response for different damping conditions, for electrical and mechanical system.

Problem Statement:

CASE1: For series RLC circuit with unit step voltage input, write code to generate output response, show all the detailed calculation, generate output graph and write inference (understanding and conclusion) for the cases given below. Assume output is obtained across the capacitor i)zeta=0 ii)zeta=0.3 iii)zeta=0.7 iv) zeta = 1 v) zeta = 5. For underdamped cases show calculation for all time and amplitude parameters. Show the specifications in output graph. Compare the theoretically calculated values with the graphically obtained values.

Undamped Condition

i) Zeta =0

```
clc; close all; clear all; t=0:0.01:60;
L=1; C=1;
wn=1/sqrt(L*C) R=0;
zeta_undamped=(R/2)*sqrt(C/L) Pole1_undamp=j*wn
Pole2_undamp=-j*wn
ct_undamp=1-cos(wn*t);
plot(t,ct_undamp)
xlabel('time') ylabel('ct_undamp')
```

Output:

zeta_undamped = 0 wn=1;

Pole1_undamp = 0.0000 + 1.0000i

Pole2_undamp = 0.0000 - 1.0000i

Undamped Response Graph:

Inference: The system is fastest, with highest gain and energy, most unstable with sustained oscillation and the poles lie on the complex axis (i.e. poles are imaginary).

Underdamped Condition

ii) Zeta =0.3

```
clc; close all; clear all; t=0:0.01:60;
L=1; C=1;
wn=1/sqrt(L*C) R=0.6;
zeta=(R/2)*sqrt(C/L) x=wn*sqrt(1-zeta^2);
Pole1_underdamp=-zeta*wn-j*wn*x Pole2_underdamp=-zeta*wn+j*wn*x
ct_underdamp=1-exp(-zeta*wn*t)/x.*(x*cos(wn*x.*t)+zeta*sin(wn*x.*t));
td=(1+0.7*zeta)/wn
a=tan((sqrt(1-zeta^2))/zeta) tr=(pi - a)/x
tp=pi/x
per_Mp=exp((-pi*zeta)/sqrt(1-zeta^2)*100) ts_2=4/(zeta*wn)
ts_5=3/(zeta*wn) plot(t,ct_underdamp) xlabel('time') ylabel('ct-underdamp')
```

Output:

```
wn = 1
zeta = 0.3000
Pole1_underdamp = -0.3000 - 0.9539i Pole2_underdamp = -0.3000 + 0.9539i td = 1.2100
a = 0.0382
```

tr = 3.2532
tp = 3.2933
Mp = 0.3723
ts_2 = 13.3333
ts_5 = 10

Underdamped Graph

Inference: The system is fast with high gain and energy, the poles lie towards the left of the Laplace plane (real part is negative) and is slightly more stable than case 1 system.

iii) Zeta =0.7

```
clc; close all; clear all; t=0:0.01:60;
L=1; C=1;
wn=1/sqrt(L*C) R=1.4;
zeta=(R/2)*sqrt(C/L) x=wn*sqrt(1-zeta^2);
Pole1_underdamp=-zeta*wn-j*wn*x Pole2_underdamp=-zeta*wn+j*wn*x
ct_underdamp=1-exp(-zeta*wn*t)/x.*(x*cos(wn*x.*t)+zeta*sin(wn*x.*t));
td=(1+0.7*zeta)/wn
a=atan((sqrt(1-zeta^2))/zeta) tr=(pi - a)/x
tp=pi/x
per_Mp=exp((-pi*zeta)/sqrt(1-zeta^2)*100)
ts_2=4/(zeta*wn)
ts_5=3/(zeta*wn) plot(t,ct_underdamp) xlabel('time') ylabel('ct-underdamp')
```

Output:

```
wn = 1
zeta = 0.7000
Pole1_underdamp = -0.7000 - 0.7141i Pole2_underdamp = -0.7000 + 0.7141i td = 1.4900
a =0.7954
tr = 3.2853
tp = 4.3991
Mp = 0.0460
ts_2 =5.7143
ts_5 =4.2857
```

Underdamped Graph

Inference: The system is slightly slower than the previous system (zeta=0.3) with a slightly less gain and energy, the poles lie more to the left of the plane and the system is more stable than when zeta=0.3 i.e., it is more damped when compared to the previous case.

iv) Critically damped Condition:

Zeta = 1

```
clc; close all; clear all; t=0:0.01:60;
L=1; C=1;
wn=1/sqrt(L*C) R=2;
zeta=(R/2)*sqrt(C/L) x=wn*sqrt(1-zeta^2);
Pole1_underdamp=-zeta*wn-j*wn*x Pole2_underdamp=-zeta*wn+j*wn*x
zeta_criticaldamp=(R/2)*sqrt(C/L) x=sqrt(1-zeta_criticaldamp^2);
Pole1_criticaldamp=-zeta_criticaldamp*wn-wn*x
Pole2_criticaldamp=-zeta_criticaldamp*wn+wn*x
ct_criticaldamp=1-exp(-wn*t).*(1+wn*t);
plot(t,ct_criticaldamp)
xlabel('time')
ylabel('ct-critically damped')
```

Critically damped Graph

Inference: The system is slow and this is the optimal damping condition for any system, the system is marginally stable with lesser gain and energy when compared to the underdamped and undamped cases. The poles of this system are real and equal i.e. they lie on the same position.

V) Over damped condition:

Zeta = 5

```
clc; close all; clear all; t=0:0.01:60;
R=10;
L=1; C=1; wn=1/sqrt(L*C);
zeta_overdamp=(R/2)*sqrt(C/L) y=sqrt((zeta_overdamp^2)-1);
Pole1_overdamp=-zeta_overdamp*wn-wn*y
Pole2_overdamp=-zeta_overdamp*wn+wn*y
s1=Pole1_overdamp;
s2=Pole2_overdamp;
ct_overdamp=1+wn/(2*y)*(exp(s1*t)/-s1-exp(s2*t)/-s2);
plot(t,ct_overdamp)
xlabel('time')
ylabel('ct-over damped')
```

Over damped Graph

Inference: The system is very slow to reach the steady state but is the most stable system of them all. The system has less gain and energy and has real and distinct poles.

Case 2 - For parallel MBK system with unit step force input, write code to generate output response, generate output graph and write inference

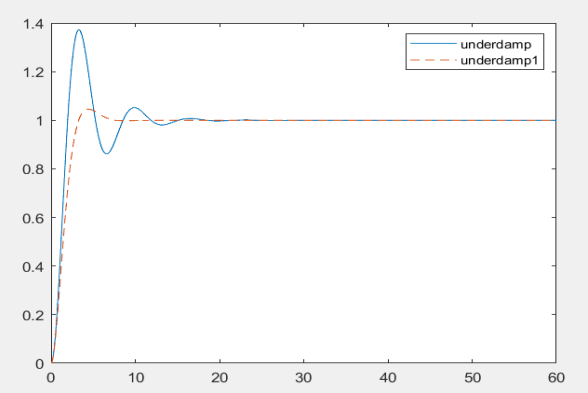
vi) Zeta =0.5

```
clc; close all; clear all; t=0:0.01:60;
k=1; M=1;
wn=1/sqrt((1/k)*M) B=1;
zeta=((1/B)/2)*sqrt(M*k) x=wn*sqrt(1-zeta^2);
Pole1_underdamp=-zeta*wn-j*wn*x
Pole2_underdamp=-zeta*wn+j*wn*x
ct_underdamp=1-exp(-zeta*wn*t)/x.*(x*cos(wn*x.*t)+zeta*sin(wn*x.*t));
plot(t,ct_underdamp) xlabel('time') ylabel('ct-underdamp')
```

Output Graph:

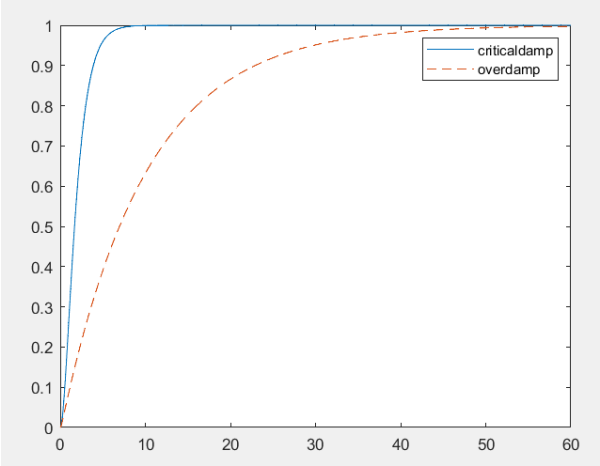
Inference: The above system is unstable fast and has high energy. It is an underdamped system.

a) Comparing cases 2 and 3(zeta=0.3 andzeta=0.7):



We can see from the above response comparison that even if both the cases are underdamped cases, the one with higher zeta value (0.7) is more stable and slower than the one with the lower zeta value (0.3). So higher the zeta value, the better the stability of the system.
Also the poles of the system with the higher zeta value lie more towards the left i.e. the real part is a higher negative value.

b) Comparing cases 4 and 5(zeta=1 and zeta=5):



We can see from the above response comparison that an over damped system is much slower to reach its final value compared to the critically damped case. Both the systems have real poles but the critically damped case has both its poles equal. Also the over damped case is more stable than the other but critical damping case is the most optimal of them.

vi) Zeta =3

```
clc; close all; clear all; t=0:0.01:60;
k=1; M=1;
wn=1/sqrt((1/k)*M) B=1/6;
zeta=((1/B)/2)*sqrt(M*k) x=wn*sqrt(1-zeta^2);
Pole1_underdamp=-zeta*wn-j*wn*x
Pole2_underdamp=-zeta*wn+j*wn*x
ct_underdamp=1-exp(-zeta*wn*t)/x.*(x*cos(wn*x.*t)+zeta*sin(wn*x.*t));
plot(t,ct_underdamp)
xlabel('time')
ylabel('ct-overdamped')
```

Output Graph:

Inference: The above system is slow and stable. It is an over damped system. It has less energy compared to the previous system.

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 5 **Date:**

Title: Determination of transfer function from system block diagram. Finding systems responses for various test signals.

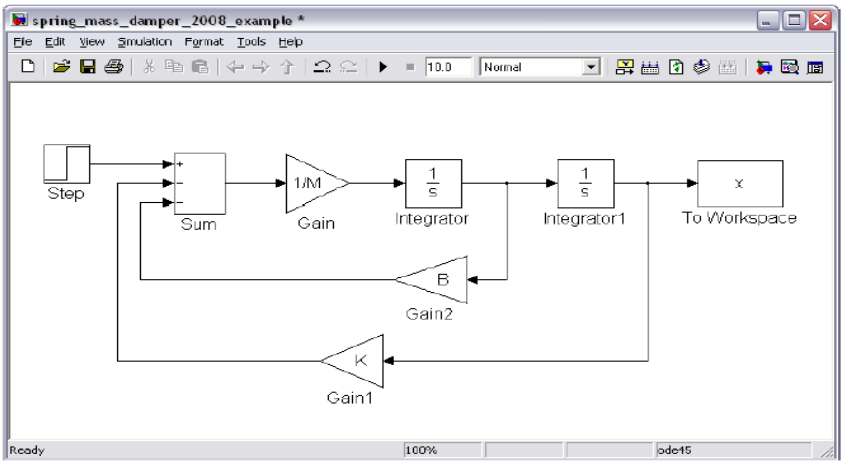
Objective: To learn Natural and step response of RL and RC systems

Software Needed: MATLAB (SIMULINK)

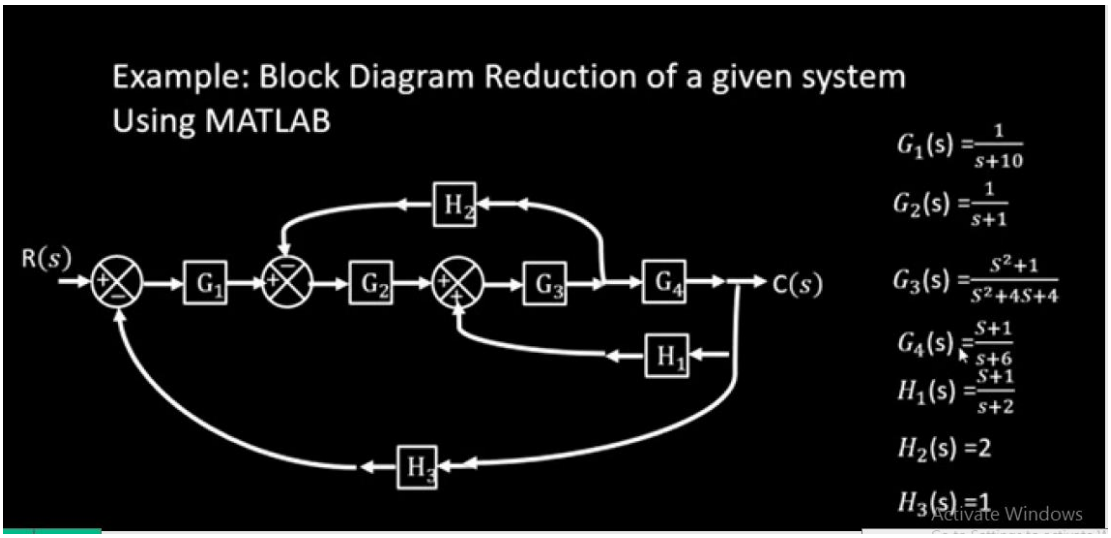
Problem Statement:
Draw the Block diagram in Simulink and study the response of the system

CASE 1:

$$a = \frac{1}{M}(-Bv - Kx + F(t))$$



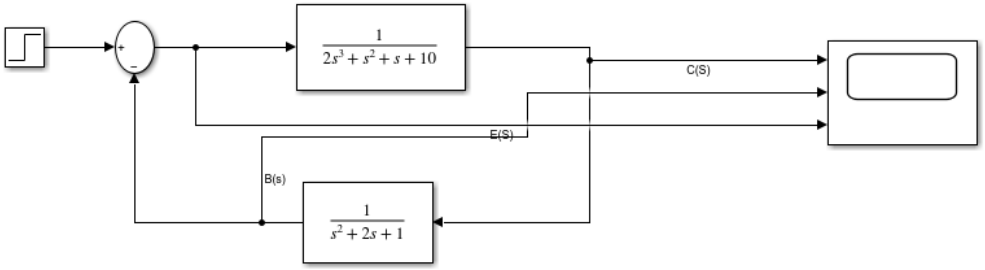
CASE 2:



Output Graph

CASE 3

Draw the Given Block Diagram on SIMULINK and study the response



Output Graph

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 6 Date:

Title: Computation of error coefficients and steady state errors for Type 0, Type I, Type II, and Type III systems with Step, Ramp and Parabolic input.

Objective: To learn Steady State Error Analysis for Various 'System Types' with Standard Inputs

Software Needed: MATLAB

Problem Statement:

Write the programs to generate the response of system for various System Types.

CASE 1:

For type 0 system (OLTF), find the steady state error of the CLTF with –ve feedback with unit step, unit ramp and unit parabolic inputs.

Solution: The problem is solved with 2 codes. The Code 1 gives the inverse Laplace transform of output C(s) and we obtain the time domain responses c(t) for all three inputs. Then Code 2 is used for plotting the c(t) or responses for all the three inputs. The two codes are to be run in two different m files. Get the output of Code 1 first. Plug in the 'responses c(t) obtained from Code 1' into Code 2 and plot the responses. In Code 2, for all the three graphs, the input and corresponding responses are plotted in one graph. **Steady state error may be found from the difference in the BLUE (input) graph and the Red (response) graph**. OLTF G(s) of type 0 is to be defined in Code 1. H(s) in Code 1 is assumed as 1. The set of Code 1 and 2 are used for finding steady state error for Type 0, system with unit step [u(t)], unit ramp [t. u(t)] & unit parabolic [t^2/2.u(t)] inputs. G(s)=2/(s+5) taken.

```
% CODE 1
% inverse Laplace Transform to find steady state system error
% Steady State Error Finding for the Given G(s)=Gs here. assumed H(s)=1
% Ts= CLTF, 'Cs_Step' is Laplace domain step response
% 'Step_Response_Time' is time domain step response
% same notations applied for Ramp and Parabolic responses as well
```

```
clc; close all; clear all; syms s;
t=(0:0.01:10);
% declare the OLTF G(s) = Gs, assume H(a)=1
Gs= 2/(s+5);
Ts=Gs/(1+Gs);
Cs_Step=Ts/s; Cs_Ramp=Ts/s^2; Cs_Parabolic=Ts/s^3;

% Time domain responses for all three inputs
Step_Response_Time=ilaplace(Cs_Step)
Ramp_Response_Time=ilaplace(Cs_Ramp)
Parabolic_Response_Time=ilaplace(Cs_Parabolic)

% copy all the results from the command window and paste in CODE 2
----------------------------------------------------------------------
% CODE - 2
% system response in time domain for different inputs
clc; close all; clear all;
t=(0:0.01:10);

Step_Response_Time = 2/7 - (2*exp(-7*t))/7;
Ramp_Response_Time = (2*t)/7 + (2*exp(-7*t))/49 - 2/49;
Parabolic_Response_Time = t.^2/7 - (2*exp(-7*t))/343 - (2*t)/49 + 2/343;

subplot(311);
plot(t,(t./t),'b',t,Step_Response_Time,'r');
axis([0 10  0   1.2]);
title('Input-Blue, Output-Red - Type 0 STEP Response');

subplot(312);
plot(t,t,'b',t,Ramp_Response_Time,'r');
%axis([0 10  0   1.2]);
title('Input-Blue, Output-Red - Type 0 RAMP Response');

subplot(313);
plot(t,(t.^2)/2,'b',t,Parabolic_Response_Time,'r')
%axis([0 10  0   1.2])
title('Input-Blue, Output-Red - - Type 0 PARABOLIC Response')
```

Outputs Graphs obtained – **You have to label these graphs with amplitude levels**

In MATLAB graph of **type 0 system with unit step input**, calculate the finite $e_{ss}(t)$ value from the first graph of type 0 system response. For **unit ramp and unit parabolic responses of the type 0 system** (the middle and the lowest graphs respectively), find the difference between input and output at 2 different times **i)** at t = 5 seconds and **ii)** at t = 8 seconds. Show that the error is increasing with time for both ramp and parabolic responses of type 0 system. Draw conclusion that the difference goes toward infinity between input and output for these two responses as time tends to infinity. **This makes steady state error $e_{ss}(t)$ of type 0 system infinity for unit ramp and unit parabolic inputs.**

Case 2 – steady state error finding for Type I system

Q 2 – For type I system (OLTF), find the steady state error of the CLTF with –ve feedback with unit step, unit ramp and unit parabolic inputs.

Ans – Same, two MATLAB code process as explained for Type 0 system is to be followed here as well. Assumed here type I system G(s)=4/(s*(s+5)), and H(s)=1.

```
% CODE 1
% inverse Laplace Transform to find steady state system error
% Steady State Error Finding for the Given G(s)=Gs here. assumed H(s)=1
% Ts= CLTF, 'Cs_Step' is Laplace domain step response
% 'Step_Response_Time' is time domain step response
% same notations applied for Ramp and Parabolic responses as well

clc; close all; clear all; symss;
t=(0:0.01:10);
% declare the OLTF G(s) = Gs, assume H(a)=1
Gs= 4/(s*(s+5));
Ts=Gs/(1+Gs);
Cs_Step=Ts/s; Cs_Ramp=Ts/s^2; Cs_Parabolic=Ts/s^3;
% Time domain responses for all three inputs
Step_Response_Time=ilaplace(Cs_Step)
Ramp_Response_Time=ilaplace(Cs_Ramp)
Parabolic_Response_Time=ilaplace(Cs_Parabolic)

% copy all the results from the command window and paste in CODE 2
----------------------------------------------------------------------
% CODE - 2
% system response in time domain for different inputs
clc; close all; clear all;
t=(0:0.01:10);

Step_Response_Time = exp(-4*t)/3 - (4*exp(-t))/3 + 1;
Ramp_Response_Time = t + (4*exp(-t))/3 - exp(-4*t)/12 - 5/4;
Parabolic_Response_Time = exp(-4*t)/48 - (4*exp(-t))/3 - (5*t)/4 + t.^2/2 + 21/16;

subplot(311);
plot(t,(t./t),'b',t,Step_Response_Time,'g');
axis([0 10  0   1.2]);
title('Input-Blue, Output-Green - Type I STEP Response');

subplot(312);
plot(t,t,'b',t,Ramp_Response_Time,'g');
%axis([0 10  0   1.2]);
```

```
title('Input-Blue, Output-Green - Type I RAMP Response');
subplot(313);
plot(t,(t.^2)/2,'b',t,Parabolic_Response_Time,'g')
%axis([0 10  0   1.2])
title('Input-Blue, Output-Green - Type I PARABOLIC Response')
```

Outputs Graphs obtained – **You have to label these graphs with amplitude levels**

In MATLAB based **unit step response graphs (top graph) fortype I system,** it is found that the output reaches to the input level (amplitude 1) just after 4^{th} second. Hence, for unit step input, at steady state, no level difference or difference in amplitude exists between the input and output as t increases. **So, for the type I system with unit step input, the steady state error $e_{ss}(t) = 0$.** This fact can be easily seen from the first graph among the three shown above.

For **type I system response with unit ramp input**, it is seen that both the input and the response goes to infinite amplitude as time tends to infinity. **But the input and output always keeps a finite distance (as calculated from the middle graph = 1.25 units) between them.** Hence, it may be concluded that type I system with unit ramp input has a finite $e_{ss}(t)$ steady state error [value to be calculated from middle graph & graph is to be labelled accordingly].

For **type I system response with unit parabolic input**, find the difference between input and output at 2 different times i) at t = 6 seconds and ii) at t = 10 seconds. Show that the error (difference between input and response levels) is increasing with time for parabolic input to type I system. Draw conclusion that the difference between input

and output goes toward infinity for parabolic input as time tends to infinity. **This makes steady state error $e_{ss}(t)$ of type I system infinity for unit parabolic input.**

CASE 3– steady state error finding for Type II system.

For type II system (OLTF), find the steady state error of the CLTF with –ve feedback with unit step, unit ramp and unit parabolic inputs.

Solution: Three different codes are used for finding the unit step, unit ramp and unit parabolic response. Try to understand the code by decoding the functions used in each line of the code. If you get any doubt, in any of the line – ask me.

Assumed is G(s)=((s+1)*(s+4))/((s^2*(s+2)*(s+6)), and H(s)=1.

```
%%% Code for Unit Step Response of Type 2 System
s = tf('s');
G = ((s+1)*(s+4))/(s^2*(s+2)*(s+6));
sys_cl = feedback(G,1);
[y,t] = step(sys_cl);
u = ones(size(t));
plot(t,y,'b',t,u,'r')
% axis([0 20 0 1.6])
xlabel('Time(secs)')
ylabel('Amplitude')
title('Input-red, Output-blue - Type II STEP Response')
```

Output Graph

Type II system response with Unit Step input, it gives Zero steady state error, $e_{ss}(t) = 0$

```
%%% Code for Unit Ramp Response of Type 2 System
s = tf('s');
G = ((s+1)*(s+4))/(s^2*(s+2)*(s+6));
sys_cl = feedback(G,1);
t = 0:0.1:50;
u = t;
[y,t,x] = lsim(sys_cl,u,t);
plot(t,y,'b',t,u,'r')
% change xmax value of 20 in axis command to 50, to get longer time response
% also change Ymax value of 30 in axis command to 50
axis([0,20,0,30])
xlabel('Time(secs)')
ylabel('Amplitude')
title('Input-red, Output-blue - Type II STEP Response')
```

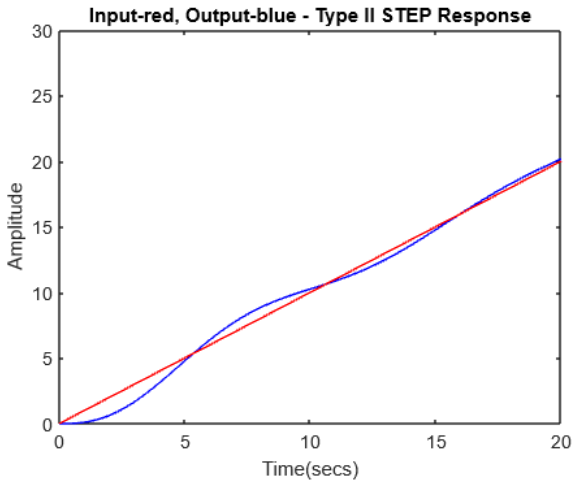


Figure (i)

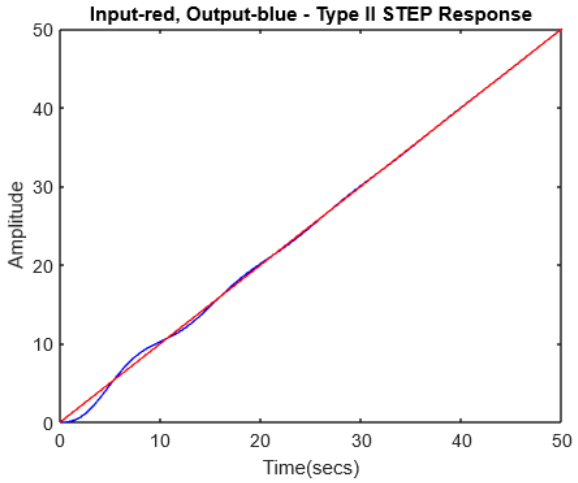


Figure (ii)

Type II system response with Unit Ramp input, for **Figure i)** x axis till 20 and **Figure ii)** x axis till 50

Zero steady state error, $e_{ss}(t) = 0$

```
%%% Code for Unit Parabolic Response of Type 2 System
s = tf('s');
G = ((s+1)*(s+4))/(s^2*(s+2)*(s+6));
sys_cl = feedback(G,1);
t = 0:0.1:30;
u = 0.5*t.*t;
[y,x] = lsim(sys_cl,u,t);
plot(t,y,'b',t,u,'r')
% axis([0 20 0 300])  % axis([15 20 100 200])    % axis([2 10 0 100])
xlabel('Time(secs)')
ylabel('Amplitude')
title('Input-red, Output-blue - Type II STEP Response')
```

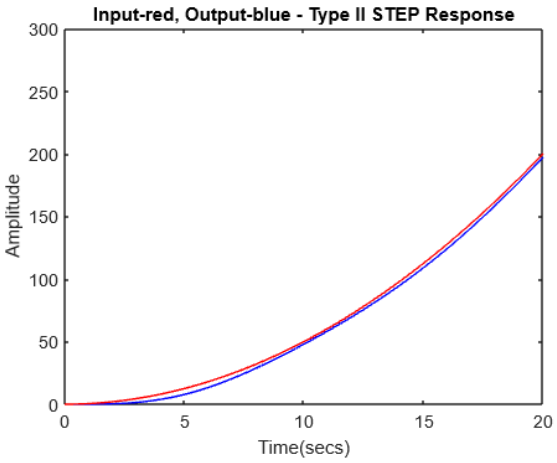


Figure (i) axis([0 20 0 300])

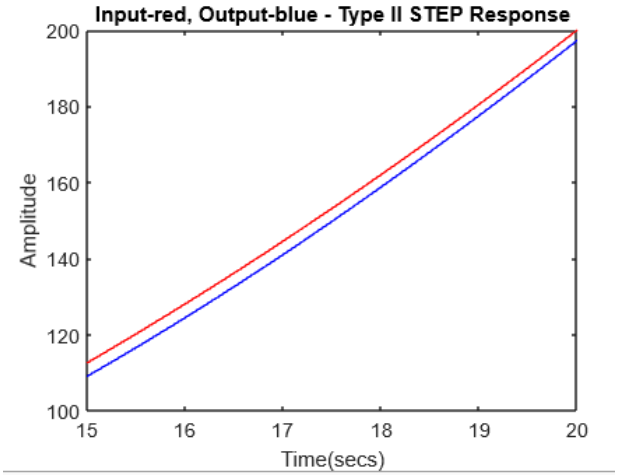


Figure (ii) axis([15 20 100 200])

Type II system response with Unit Parabolic input, different time ranges shown Finite steady state error (**Figure ii**), though responses will be of infinite value as t tends to infinity

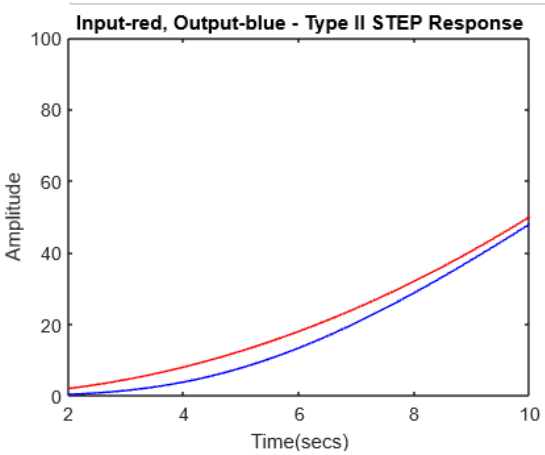


Figure (iii) axis([2 10 0 100])

Explain observations in these three figures. Steady state error is finite.

Final Task –

At the end of the experiment, prepare the table with details for all the 9 cases of systems that you define and the input that are mentioned in the question (A=2). Mention the G(s) that you assume in the left side column, and fill up the 6 corresponding column in the right side for the inputs mentioned above.

	STEP INPUT = A		RAMP INPUT = At		PARABOLIC I/P = $\frac{At^2}{2}$	
	k_p	e_{ss}	k_v	e_{ss}	k_a	e_{ss}
TYPE 0 SYSTEM	k_p	$\frac{A}{1+k_p}$	0	∞	0	∞
TYPE 1 SYSTEM	∞	0	k_v	$\frac{A}{k_v}$	0	∞
TYPE 2 SYSTEM	∞	0	∞	0	k_a	$\frac{A}{k_a}$

PROVE ALL 3 POINTS FROM THIS TABLE

Write all the values that are obtained from the experimentation carried out by you.

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 7 Date:

Title- System Response with standard test inputs with respect for various 'system pole positions'.

Generate system output c(t) for unit impulse, unit step and unit ramp inputs for the various system pole positions given below.

Generate c(t) by using 'ilaplace' command in MATLAB as the Laplace domain pole positions of characteristics equations are given. Write code, generate output graph, and write inference for all the given cases. Inference (Conclusion or understanding) should be written for system stability (stable, unstable, relatively more or less stable), system gain (high or low), system energy (high or low), fastness or slowness of system time domain response, damping conditions etc... And other features as you wish to add.

Case1: Single real pole case

i) s=4

```
clc; close all; clear all;
syms s;
c_t=1/(s-4);
ilaplace(c_t)
```

```
clc; clear all; close all;
t=(0:0.01:50);
c=exp(4*t);plot(t,c);xlabel('time');ylabel('c(t)')
title('Timedomain responseofsingle polesystem(s=4')
```

Output Graph

ii) s=-5

```
clc; close all; clear all;
syms s;
c_t=1/(s+5);
ilaplace(c_t)

clc; clear all; close all;
t=(0:0.01:50);
c=exp(-5*t);
plot(t,c)
plot(t,c)xlabel('time')ylabel('c(t)')
title('Time domain response of single pole system (s=-5)')
```

Output Graph

Case2: Double real pole case

1)s=-2, and-4

```
clc; close all; clear all;
syms s;
c_t=1/((s+1)*(s+4));
ilaplace(c_t)

clc; close all; clear all;
t=(0:0.01:50);
c=exp(-t)/3 - exp(-4*t)/3;
plot(t,c)
title('Time domain response of double pole system (s=-2,-4)');
xlabel('time');ylabel('c_t');
```

Output Graph

2) s=0,and -2

```
clc; close all; clear all;
syms s;c_t=1/((s)*(s+2));
ilaplace(c_t)

clc; close all; clear all;
t=(0:0.01:50);
c= 1/2 - exp(-2*t)/2plot(t,c)
title('Time domain response of double pole system (s=0,-2)');
xlabel('time');
ylabel('c_t');
```

Output Graph

3)s=-5, and-5

```
clc; close all; clear all;
syms s;
c_t=1/((s+5)*(s+5));
ilaplace(c_t)

t=(0:0.01:50);
c=t.*exp(-5*t)
plot(t,c)
title('Time domain response of double pole system (s=-5,-5)');
xlabel('time');
ylabel('c_t');
```

Output Graph

4)s=-2, and 3

```
clc; close all; clear all;
syms s;
c_t=1/((s+2)*(s-3));
ilaplace(c_t)
t=(0:0.01:50);
c=exp(3*t)/5 -exp(-2*t)/5
plot(t,c)
title('Time domain response of double pole system (s=-2,3)');
xlabel('time');
ylabel('c_t');
```

Output Graph

Case3: Triple real pole case

1) s=-2,-4 ,-6

```
clc; clear all; close all;
syms s;
c_t=1/((s+2)*(s+4)*(s+6));
ilaplace(c_t)

t=(0:0.01:50);
c=exp(-2*t)/8 - exp(-4*t)/4 + exp(-6*t)/8;
plot(t,c)
title('Time domain response of triple pole system (s=-2,-4,-6)');
xlabel('time');
ylabel('c_t');
```

Output Graph

2)s=-2,-6, 0

```
clc; clear all; close all;
syms s;
c_t=1/((s+2)*(s+6)*(s));
ilaplace(c_t)

t=(0:0.01:50);
c=exp(-6*t)/24 - exp(-2*t)/8 + 1/12;plot(t,c)
title('Time domain response of triple pole system (s=-2,-6,0)');
xlabel('time');ylabel('c_t');
```

Output Graph

3)s =-4,-2 , 3

```
clc; clear all; close all;
syms s;c_t=1/((s+4)*(s+2)*(s-3));
ilaplace(c_t)

t=(0:0.01:50);
c=exp(3*t)/35 - exp(-2*t)/10 + exp(-4*t)/14;
plot(t,c)
title('Time domain response of triple pole system (s=-4,-2,3)');
xlabel('time');ylabel('c_t');
```

Output Graph

Case4:s1 =+j3,s2=-j3

```
clc; clear all; close all;
t=(0:0.01:50);
symss;
c_t=1/(s^2+9);
ilaplace(c_t)
c=sin(3*t)/3;
plot(t,c)
title('Time domain response of double pole system (s=3i,-3i)');
xlabel('time');
ylabel('c_t');
```

Output Graph

Case5: s1 =-2+j4, s2 =-2 – j4

```
clc;clear all;closeall;
t=(0:0.01:50);
symss;
c_t=1/(s^2+(4*s)+20);
ilaplace(c_t)
c=(sin(4*t).*exp(-2*t))/4;
plot(t,c)
title('Time domain response of double pole system (s=-2+4i,-2-4i)');
xlabel('time');ylabel('c_t');
```

Output Graph

Case 6: s1 = 3 + j5, s2 = 3 - j5

```
clc;clear all; closeall;
t=(0:0.01:50);
syms s;
c_t=1/(s^2-(6*s)+34);
ilaplace(c_t)

c=(sin(5*t).*exp(3*t))/5;plot(t,c)
title('Time domain response of double pole system (s=3+5i,3-5i)');
xlabel('time');
ylabel('c_t');
```

Output Graph

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 8 **Date:**

Title: Root locus plot for different systems with varying system gain.

Objective: To learn how to define stability of the system using ROOT LOCUS plot

Software Needed: MATLAB

Problem Statement:

Write the programs to generate the Root Locus plot for different open loop transfer function for the system with varying system gain.

CASE 1

```
%root locus plot
clc;
close all;
clear all;
num=[2 5 1];
den=[1 2 3]
%g=1/((s^2+8))
tf1=tf(num,den)
%g=(s+1)/s*(s+2)*(s+5)
%g= 1 /(s*(s+1))
rlocus(tf1)
```

OR (alternate way of writing the program)

```
s=tf('s')
clc;closeall;clear all;
g=(s+1)/s*(s+2)*(s+5)
rlocus(g)
```

CASE 2:

```
clc; close all; clear all;
s=tf('s');
g=(s^2+8*s+17)/(s^2+4*s+13)
%g=1/((s+1)*(s+2)*(s+6)*(s-6));
rlocus(g)
%g=1/(s*(s+4)*(s^2+4*s+20))
% g is the G(s)H(s) for finding the root locus
rlocus(g)
```

Output Graph:

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 9 **Date:**

Title: Bode plot-based determination of system stability for different physical systems. Systems with lead, lag and lead-lag compensation are also to be considered.

Objective: To learn system stability for different systems using BODE PLOT.

Software Needed: MATLAB

Problem Statement:

Write the programs to generate the Bode Plot for the different system and analyse the system stability

```
clc;
clear all;
close all;
num1= [ 10  2 ];
den1 = [ 1 30 5 ];
disp ('Transfer function :-  ');
TF1= tf(num1 , den1)
bode (TF1)
```

Graph:

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge

CO		PO	

Experiment No: 10 **Date:**

Title: Fourier series based signal analysis and synthesis and generation of Fourier Spectra

Objective: To learn the response for response for Fourier Spectra using Fourier series based signal analysis

Software Needed: MATLAB

Problem Statement:

Write the programs to generate the response for Fourier Spectra using Fourier series based signal analysis

Graph:

Observations:

Inference:

Outcome:

Evaluation:

Conduction(5)	
Results/Outcome(5)	
Viva(5)	
Total(15)	

Signature of Faculty-In-Charge