QMP7.1 D/F

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belagavi & Approved by AICTE, New Delhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH206(B.H.Road),Gubbi,Tumkur–572216.Karnataka.

Department of Electronics & Communication Engineering

CONTROL SYSYEMS LAB (IPCC) (BEC403)

(NEP, Outcome Based Education (OBE) and Choice Based Credit System (CBCS))

B.E - IV Semester

Lab Manual 2023-24

Name:____

USN:____

Batch:____________________Section:_____________________

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belagavi & Approved by AICTE, New Delhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH206(B.H.Road),Gubbi,Tumkur–572216.Karnataka.

Department of Electronics & Communication Engineering

CONTROL SYSYEMS LAB (IPCC) (BEC403)

Manual

Prepared by:
1. Mr. Manohara T N
Assistant Professor

2 Mr. Sreenivasa T V
Assistant Professor

Reviewed by:
Mrs. Noor Afza
Assistant Professor

Approved by:

Dr. Thejaswini R

HOD, Dept. Of ECE

.

INSTITUTE VISION

- To create centers of excellence in education and to serve the society by enhancing the quality of life through value based professional leadership.

INSTITUTE MISSION

- To provide high quality technical and professionally relevant education in a diverse learning environment.
- To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society.
- To prepare next generation of skilled professionals to successfully compete in the diverse global market.
- To promote campus environment that welcomes and honors women and men of all races, creeds and cultures, values and intellectual curiosity, pursuit of knowledge and academic integrity and freedom.
- To offer wide variety of off-campus education and training programmes to individuals and groups.
- To stimulate collaborative efforts with Industry, Universities, Government and Professional Societies.
- To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.

Quality Policy

Our Organization delights customers (Student, Parents and Society) by providing value added quality education to meet the National and International requirements. We also provide necessary steps to train the students for placement and continue to improve our methods of education to the students through effective. Quality Management System, Quality Policy and Quality Objectives.

Department of Electronics & Communication Engineering

VISION OF THE DEPARTMENT

"To create globally competent Electronics and Communication Engineering professionals with ethical and moral values for the betterment of the society"

MISSION OF THE DEPARTMENT

- To nurture the technical/professional/engineering and entrepreneurial skills for overall self and societal upliftment through co-curricular and extra-curricular events.
- To orient the Faculty/Student community towards the higher education, research and development activities.
- To create the Centres of Excellence in the field of electronics and communication in collaboration with industries/Universities by training the faculty through latest technologies.
- To impart quality technical education in the field of electronics and communication engineering to meet over the current/future global industry requirements.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

PEO1: Apply Mathematical, Scientific and Engineering skills for solving problems in the areas of Electronics and Communication Engineering.

PEO2: Expose to Emerging Technologies and excel in Industries/higher studies/research.

PEO3: Apply analytical skills in the areas of Electronics and Communication Engineering to become competent and Employable.

PEO4: Inculcate Professional ethics, human values, team work for solving Engineering problems and contribute to societal needs.

PROGRAM SPECIFIC OUTCOMES

PSO1: Build Analog and Digital Electronic systems for Multimedia Applications, VLSI and Embedded Systems in Interdisciplinary Research / Development.

PSO2: Design and Develop Communication Systems as per Real Time Application and Current Trends.

COURSE OBJECTIVES

1. Understand basics of control systems and design mathematical models using block diagram reduction, SFG, etc.
2. Understand Time domain and Frequency domain analysis.
3. Analyze the stability of a system from the transfer function
4. Familiarize with the State Space Model of the system.

COURSE OUTCOMES

CO1: Deduce transfer function of a given physical system, from differential equation Representation or Block Diagram representation and SFG representation.

CO2: Calculate time response specifications and analyse the stability of the system.

CO3: Draw and analyse the effect of gain on system behaviour using root loci.

CO4: Perform frequency response Analysis and find the stability of the system.

CO4: Represent State model of the system and find the time response of the system.

PROGRAM OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs.
4. **Conduct investigations of complex problems:** An ability to design and conduct scientific and engineering experiments, as well as to analyze and interpret data to provide valid conclusions
5. **Modern tool usage:** Ability to apply appropriate techniques, modern engineering and IT tools, to engineering problems.
6. **The engineer and society:** An ability to apply reasoning to assess societal, safety, health and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** An ability to understand the impact of professional engineering solutions in societal and environmental contexts
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Ability to function effectively as an individual, and as a member or leader in a team, and in multidisciplinary tasks.
10. **Communication:** Ability to communicate effectively on engineering activities with the engineering community such as, being able to comprehend and write effective reports and design documentation, make effective presentations.
11. **Project management and finance:** An ability to apply knowledge, skills, tools, and techniques to project activities to meet the project requirements with the aim of managing project resources properly and achieving the project's objectives.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Department of Electronics & Communication Engineering

Syllabus

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI
B.E: Electronics & Communication Engineering
NEP, Outcome Based Education (OBE) and Choice Based Credit System (CBCS)
SEMESTER – IV

CONTROL SYSTEMS LABORATORY (IPCC) (BEC403)

Course Code	**BEC403**	**CIE Marks**	**50**
Teaching Hours/Week (L: T: P)	**(3:0:2)**	**SEE Marks**	**50**
Total Hours of Pedagogy	**40 hours Theory + 12 Lab slots**	**Total Marks**	**100**
Credits	**04**	**Exam Hours**	**03**

PRACTICAL COMPONENT OF IPCC

Using suitable simulation software (P-Spice/ MATLAB / Python / Scilab / OCTAVE / LabVIEW)

SL. No	**Experiments**
1	Implement Block diagram reduction technique to obtain transfer function a control system.
2	Implement Signal Flow graph to obtain transfer function a control system.
3	Simulation of poles and zeros of a transfer function.
4	Implement time response specification of a second order Under damped System, for different damping factors.
5	Implement frequency response of a second order System.
6	Implement frequency response of a lead lag compensator.
7	Analyse the stability of the given system using Routh stability criterion.
8	Analyse the stability of the given system using Root locus.
9	Analyse the stability of the given system using Bode plots.
10	Analyse the stability of the given system using Nyquist plot.
11	Obtain the time response from state model of a system.
12	Implement PI and PD Controllers.
13	Implement a PID Controller and hence realize an Error Detector
14	Demonstrate the effect of PI, PD and PID Controller on the system response

Suggested Learning Resources

Text Books	1. Control Systems Engineering, I J Nagrath, M. Gopal, New age international Publishers, Fifth edition.
Web links and Video Lectures (e-Resources):	https://nptel.ac.in/courses/108106098

COURSE ASSESSMENT AND EVALUATION						
Direct Assessment Methods						
What		**To whom**	**When/Where (Frequency in the course)**	**Max. Marks**	**Evidence Collected**	**Contributing to Course outcomes**
CIE	Record & Observation	Students	Every lab session (Avg. of all experiment marks)	15	Observation book written at each lab + Record submitted at each lab + Viva	CO1 – CO5
	IA Test		one	05	Blue Books	CO1 – CO5

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES																
		PO												**PSO**		
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CO	**1**	2	2	2	1									1		
	2	2	2	2	1									1		
	3	2	2	2	1									1		
	4	2	2	2	1									1		
	5	2	2	2	1									1		
	Sum	10	10	10	5									5		
	Avg	2	2	2	1									1		

3: High correlation, 2: Medium correlation, 1: Low correlation

Instructions to the Candidates

General Lab Guidelines:

• Conduct yourself in a responsible manner at all times in the laboratory. Intentional misconduct will lead to the exclusion from the lab.

• Do not wander around, or distract other students, or interfere with the laboratory experiments of other students.

• Read the handout and procedures before starting the experiments. Follow all written and verbal instructions carefully. If you do not understand the procedures, ask the instructor or teaching assistant.

• Attendance in all the labs is mandatory, absence permitted only with prior permission from Class teacher.

• The workplace has to be tidy before, during and after the experiment.

• Do not eat food, drink beverages or chew gum in the laboratory.

• Every student should know the location and operating procedures of all Safety equipment including First Aid Kit and Fire extinguisher.

DO'S:-

- Uniform and ID card are must.
- Strictly follow the procedures for conduction of experiments.
- Records have to be submitted every week for evaluation.
- Chairs and stools should be kept under the workbenches when not in use.
- After the lab session, switch off every supply, disconnect and disintegrate the experiments and return the components.
- Keep your belongings in designated area.
- Never use damaged instruments, wires or connectors. Hand these parts to the instructor/ teaching assistant.
- Sign the log book when you enter/leave the laboratory.

DONT'S:-

- Don't touch open wires unless you are sure that there is no voltage. Always disconnect the plug by pulling on the connector body not by the cable. Switch off the supply while you make changes to the experiment.
- Don't leave the experiment table unattended when the experimental setup supply is on.
- Students are not allowed to work in laboratory alone or without presence of the teaching staff/ instructor.
- No additional material should be carried by the students during regular labs.
- Avoid stepping on electrical wires or any other computer cables

Evaluation of Control system Laboratory	**University weightage Marks**
Lab internal: Conduction for 50 marks **1. Write up – 10M** **Discrete Experiments:** (Aim– 1M, Circuit – 3M, Design – 3M, Waveforms – 3M) **Programming Experiments:** (Aim– 1M, Flowchart – 3M,Program –6M) **2. Program Execution – 30M** (Program/Code Output – 25M, Comments – 3M, Optimization– 2M, (Partial output – 25M, No Output – 00M) **3. Results & Viva – 10M** (Identifying & Showing the inputs and outputs – 2M and/or theoretical calculations – 2M, Output Verification – 1M	**05**
Record and Observation: **Conduction for 30 marks (10M record and 20 M Observation)** (Aim & Apparatus, Algorithm/flowchart (each experiment should have at least one flowchart, Calculations, Input/Output observations & Result (10+20=30M)	**15**
Total Marks	**20**

INTRODUCTION TO MATLAB

MATLAB stands for Matrix Laboratory. It is a high-performance language that is used for technical computing. It was developed by Cleve Molar of the company MathWorks. INC in the year 1984.It is written in C, C++, Java. It allows matrix manipulations, plotting of functions, implementation of algorithms and creation of user interfaces. It is both a programming language as well as a programming environment. It allows the computation of statements in the command window itself.

The built-in functions of MATLAB offer top-notch resources for performing calculations, including optimization, linear algebra, numerical solution of ordinary differential equations (ODEs), data analysis, quadrate, signal processing, and many other scientific tasks. Modern algorithms are used for the majority of these functions. There are many of these for both animations and 2-D and 3-D graphics. MATLAB also supports an external interface.

The user can create their own functions in the MATLAB language. Thus, they are not restricted to using only the built-in functions. Additional toolboxes are provided by MATLAB. These toolboxes were created for common uses such as neural networks, symbolic computations, image processing, control system design, and statistics.

The various uses of MATLAB are: Developing algorithms, performing linear algebra that is linear, Graph plotting for larger data sets, Data visualization and analysis, Numerical Matrix Computation

MATLAB is generally used for these types of tasks:

- Signal processing
- Optimization of functions
- Control system design
- Image and Audio processing
- Machine learning and Deep learning

Features of MATLAB

- **MATLAB is a high-level language:** MATLAB Supports Object oriented programming. It also supports different types of programming constructs like Control flow statements (IF-ELSE, FOR, WHILE). MATLAB also supports structures like in C programming, Functional programming (writing functions to contain commonly used code and later calling them). It also contains Input / Output statements like disp() and input().

- **Interactive graphics:** MATLAB has inbuilt graphics to enhance user experience. We can actually visualize whatever data is there in forms of plots and figures. It also supports processing of image and displaying them in 2D or 3D formats. We can visualize and manipulate our data across any of the three dimensions (1D, 2D, and 3D). We can plot the functions and customize them also according to our needs like changing bullet points, line color and displaying/not displaying grid.

- **A large library of Mathematical functions:** MATLAB has a huge inbuilt library of functions required for mathematical analysis of any data. It has common math functions like sqrt. factorial etc. It has functions required for statistical analysis like median, mode and std (to find standard deviation), and much more. MATLAB also has functions for signal processing like filter, butter(Butterworth filter design) audio read, Conv, xcorr, fft, fftshift etc. It also supports image processing and some common functions required for image processing in MATLAB are rgb2gray, rgb2hsv, adaptthresh etc.

- **Data access and processing:** MATLAB allows accessing of data from external sources like image files (.jpg, .PNG), audio files (.mp), and real-time data from JDBC/ ODBC. We can easily read data from external sources using the inbuilt MATLAB functions like audioread for reading audio files and imread for reading external images.

- **Interactive environment:** MATLAB offers interactive environment by providing a GUI (Graphical user interface) and different types of tools like signal analyses and tuners. MATLAB also has tools for debugging and the development of any software. Importing and exporting files becomes easy in MATLAB through the GUI. We can view the workspace data as we progress in the development of our software and modify it according to our needs.

- **MATLAB can interface with different languages:** We can write a set of codes (libraries) in languages like PERL and JAVA, and we can call those libraries from within the MATLAB itself. MATLAB also supports ActiveX and .NET libraries.

- **MATLAB and Simulink :** MATLAB has an inbuilt feature of Simulink wherein we can model the control systems and see their real-time behavior. We can design any system either using code or building blocks and see their real-time working through various inbuilt tools. It has lucid examples of basic control systems and their working.

- **MATLAB's Application programming interface (API):** MATLAB consists of an extensive API. Through this API, we can link our C/C++ programs directly to MATLAB. Some options available in MATLAB API are calling MATLAB programs, read and write M-files, and using MATLAB as an interface to run applications. MATLAB can be used both as a computation and analysis tool.
- **Machine Learning, Deep Learning, and Computer vision:** The most demanding technologies like Machine learning, Deep learning, and Computer vision can be done in MATLAB. We can create and

interconnect layers of a deep neural network, We can build custom training loops and training layers with automatic differentiation. For machine learning, we can use the DBSCAN algorithm to discover clusters and noise in DATA. For computer vision, we can do object tracking, object recognition, gesture recognition, and processing 3D point clouds.

- **Computational Biology toolbox:** This toolbox provides a great way for biologists and researchers to create and analyze new algorithms and patterns for development in biological and biochemical domains. We can build biological models and analyze them using this toolbox. Moreover, for students, this toolbox can be very much educational if they want to explore the biological domain.

Writing a MATLAB Program

1. **Using Command Window:** Only one statement can be typed and executed at a time. It executes the statement when the enter key is pressed. This is mostly used for simple calculations. Note: ans is a default variable created by MATLAB that stores the output of the given computation.

2. **Using Editor:** Multiple lines of code can be written here and only after pressing the run button (or F5) will the code be executed. It is always a good practice to write clc, clear and close all in the beginning of the program.Note: Statements ending with a semicolon will not be displayed in the command window, however, their values will be displayed in the workspace. Any statement followed by % in MATLAB is considered as a comment

3. **Vector Operations:** Operations such as addition, subtraction, multiplication and division can be done using a single command instead of multiple loops

Basic Functions in MATLAB

Function	Description
disp()	The values or the text printed within single quotes is displayed on the output screen
clear	To clear all variables
close all	To close all graphics window

Function	Description
clc	To clear the command window
exp(x)	To compute the exponential value of x to the base e
abs(x)	To compute the absolute value of x
sqrt(x)	To compute the square root of x
log(x)	To compute the logarithmic value of x to the base e
log10(x)	To compute the logarithmic value of x to the base 10
rem(x, y)	To compute the remainder of x/y
sin(x)	To compute the sine of x
cos(x)	To compute the cosine of x
tan(x)	To compute the tangent of x

Advantages of MATLAB

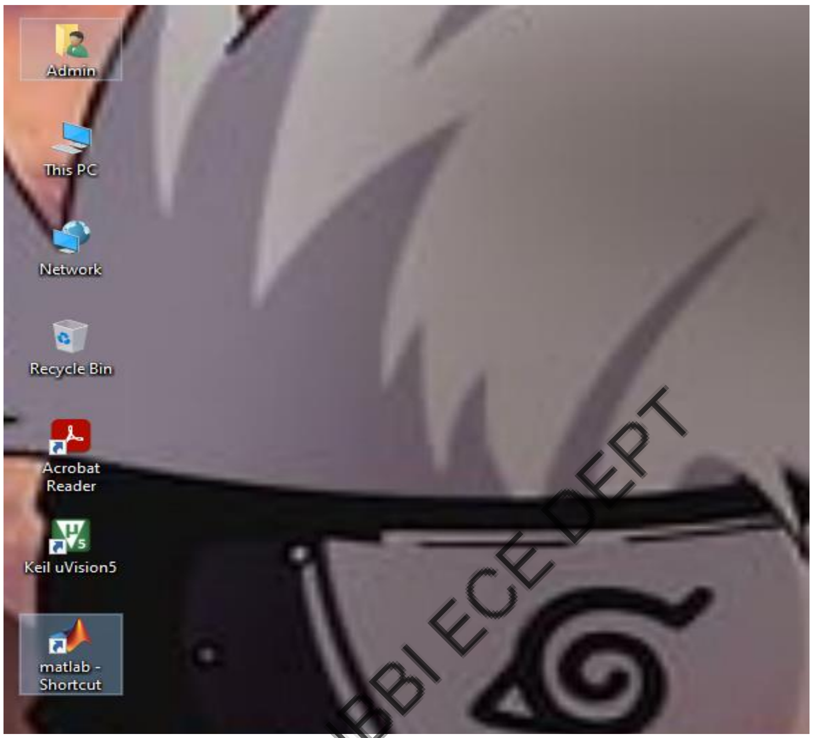
- **Easy to use interface:** A user-friendly interface with features you want to use is one click away.
- **A large inbuilt database of algorithms:** MATLAB has numerous important algorithms you want to use already built-in, and you just have to call them in your code.
- **Extensive data visualization and processing:** We can process a large amount of data in MATLAB and visualize them using plots and figures.
- **Debugging of codes easy:** There are many inbuilt tools like analyser and debugger for analysis and debugging of codes written in MATLAB.
- **Easy symbolic manipulation:** We can perform symbolic math operations in MATLAB using the symbolic manipulation algorithms and tools in MATLAB

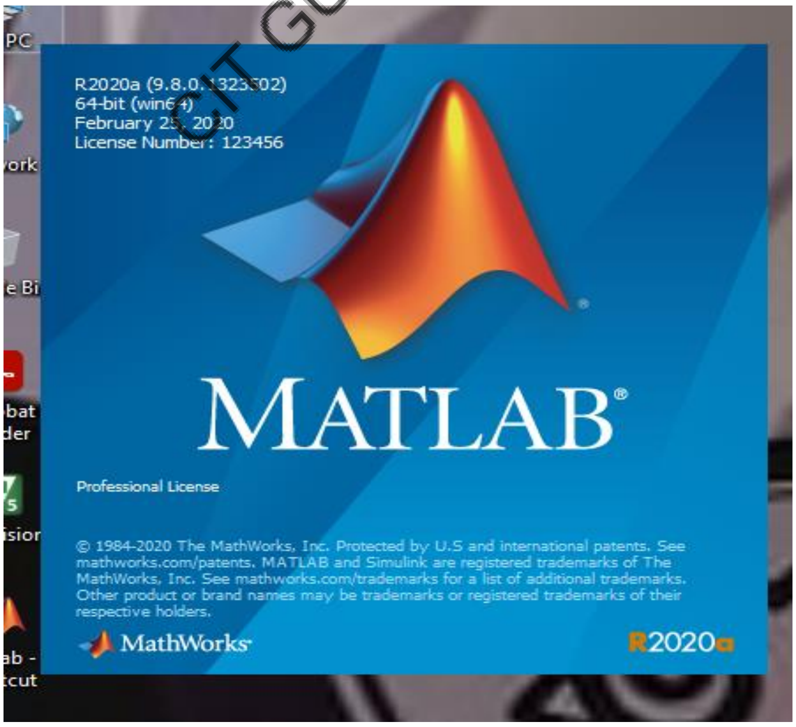
Disadvantages of MATLAB

- MATLAB is slow since it is an interpreted language that is MATLAB programs are not converted into Machine language but are run by external software, so it can sometimes be slow.
- We cannot create the OUTPUT file in MATLAB.
- One cannot use graphics in MATLAB with -nojvm option, on doing so, we will get a runtime error.
- We cannot make functions in one single .m file as we have in the case of other programming languages. We have to create different files for different functions.
- Sometimes, the error messages are not much informative, so you have to figure out the error yourself.

PROCEDURE TO EXECUTE MATLAB PROGRAMS

Step1: Double Click on MATLAB Icon on the Desktop





R2020a (9.8.0.1323502)
64-bit (win64)
February 25, 2020
License Number: 123456

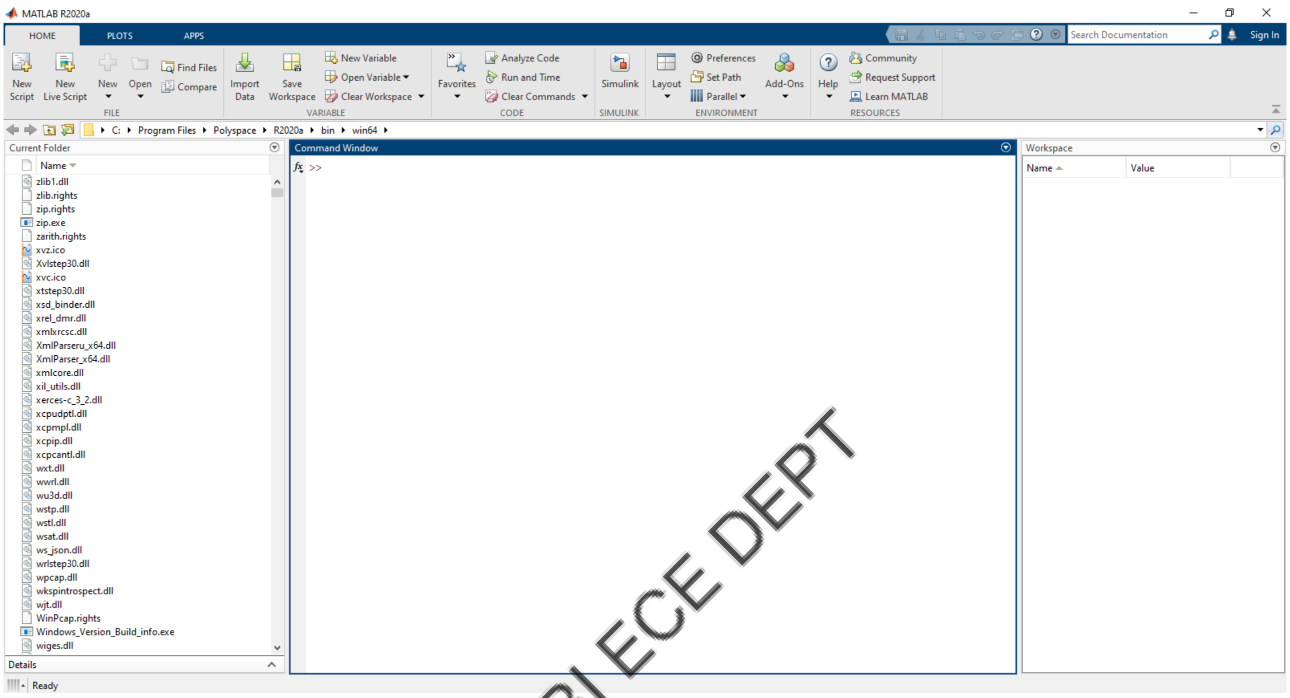
Professional License

© 1984-2020 The MathWorks, Inc. Protected by U.S and international patents. See mathworks.com/patents. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

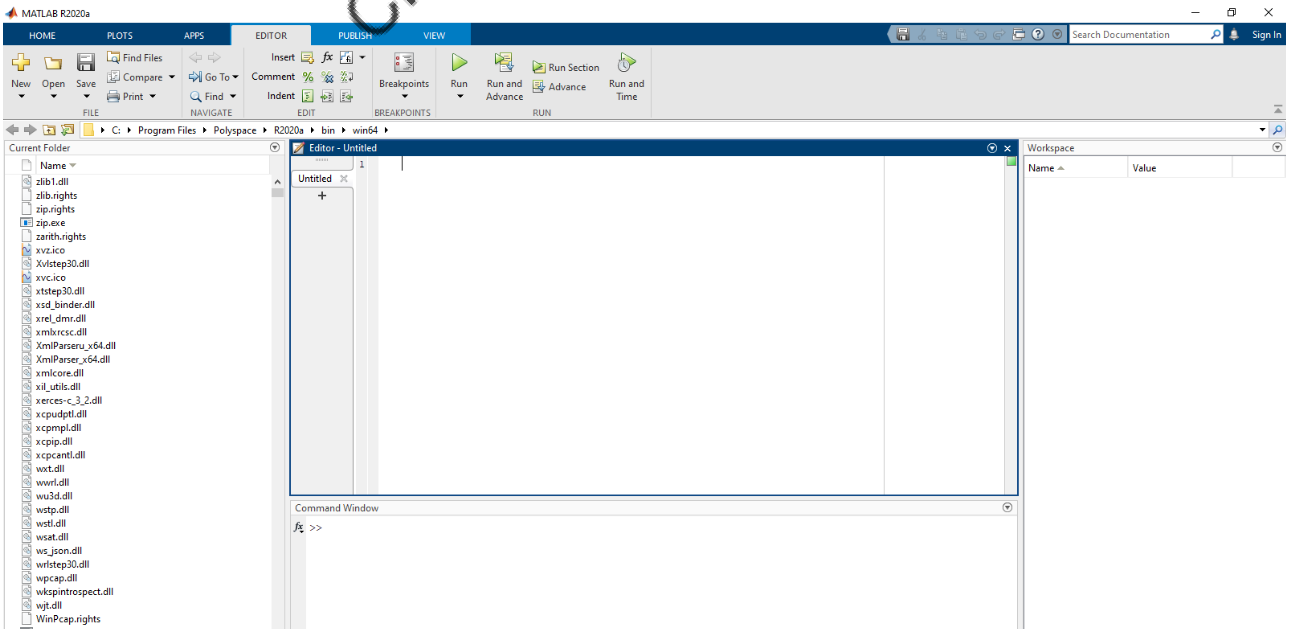
MATLAB Windows is open as follows:

it Contains three windows

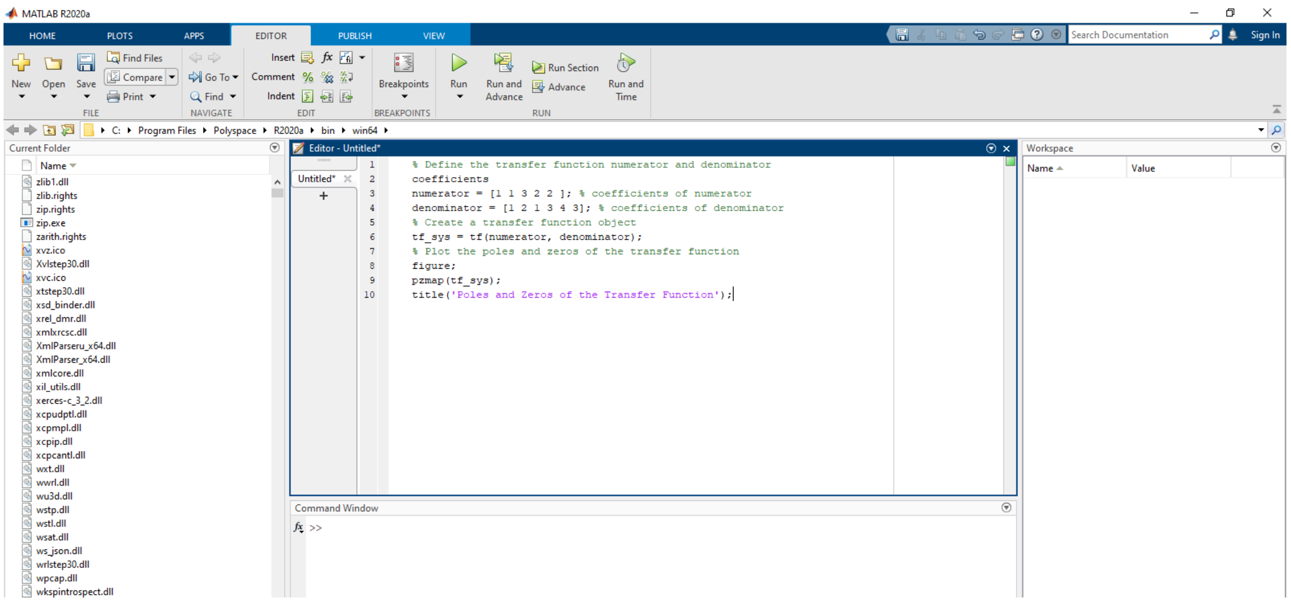
1. Current Folder Window
2. Command Window
3. Work Space



Step2: Click on New button and select M-File, An Editor window will open as follows

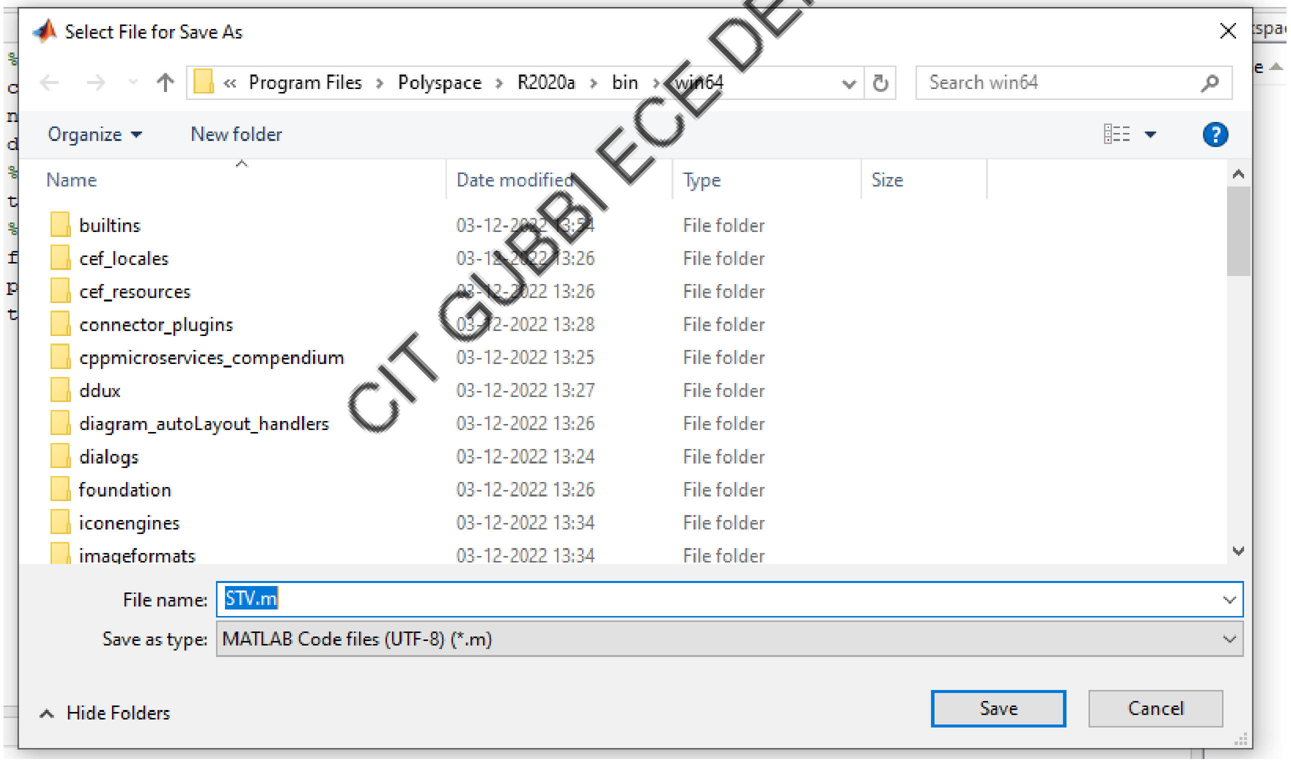


Step3: Type MATLAB Program in editor window



```
% Define the transfer function numerator and denominator
coefficients
numerator = [1 1 3 2 2 ]; % coefficients of numerator
denominator = [1 2 1 3 4 3]; % coefficients of denominator
% Create a transfer function object
tf_sys = tf(numerator, denominator);
% Plot the poles and zeros of the transfer function
figure;
pzmap(tf_sys);
title('Poles and Zeros of the Transfer Function');
```

Step4: Save the MATLAB Program in a designated folder with .m Extension.



Select File for Save As

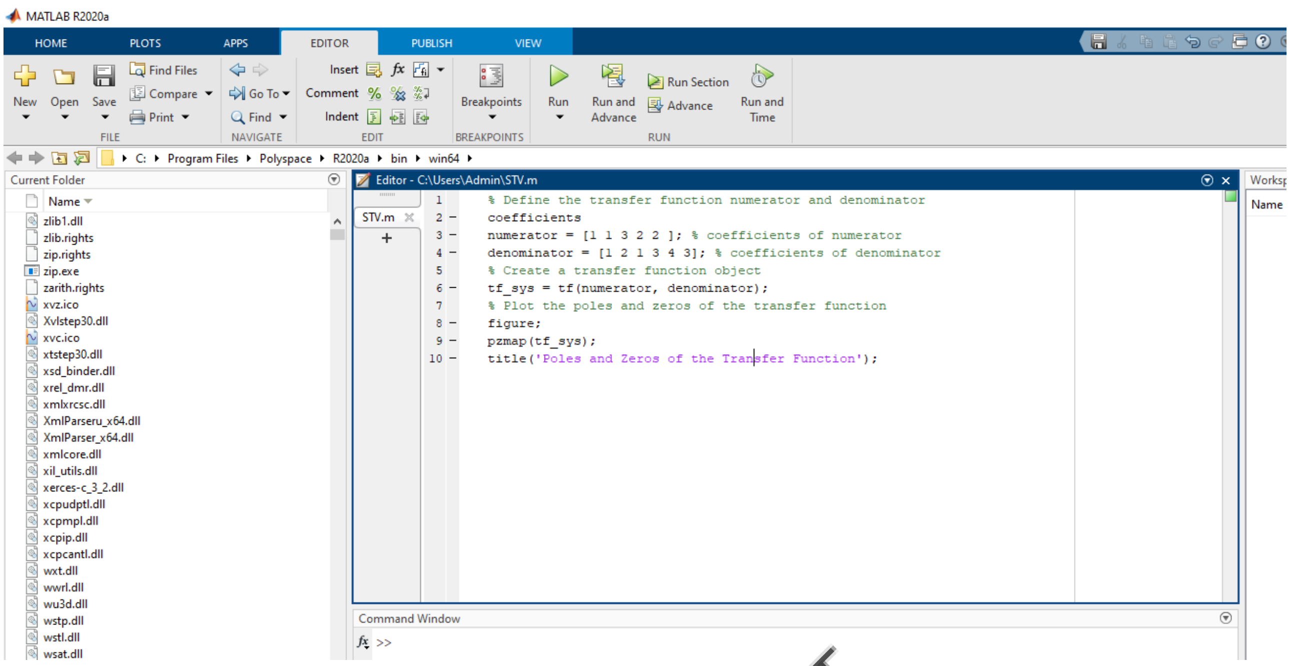
« Program Files › Polyspace › R2020a › bin › win64

Name	Date modified	Type	Size
builtins	03-12-2022 13:54	File folder	
cef_locales	03-12-2022 13:26	File folder	
cef_resources	03-12-2022 13:26	File folder	
connector_plugins	03-12-2022 13:28	File folder	
cppmicroservices_compendium	03-12-2022 13:25	File folder	
ddux	03-12-2022 13:27	File folder	
diagram_autoLayout_handlers	03-12-2022 13:26	File folder	
dialogs	03-12-2022 13:24	File folder	
foundation	03-12-2022 13:26	File folder	
iconengines	03-12-2022 13:34	File folder	
imageformats	03-12-2022 13:34	File folder	

File name: STV.m

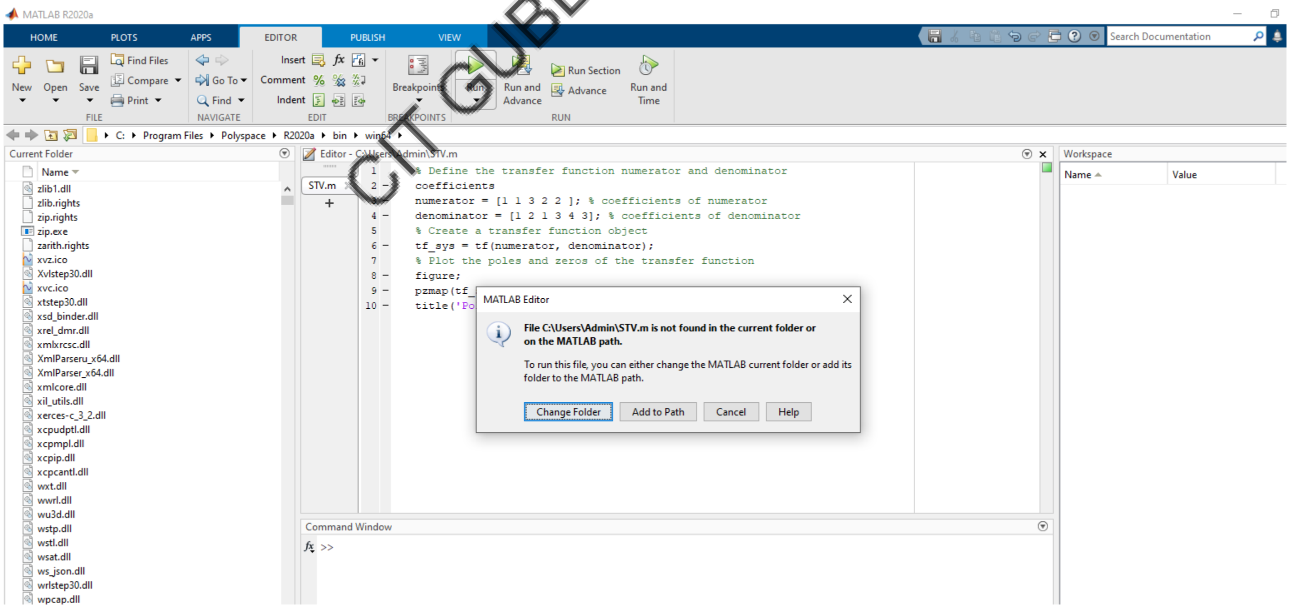
Save as type: MATLAB Code files (UTF-8) (*.m)

After Saving MATLAB Program Editor Window will look as follows.


Editor - C:\Users\Admin\STV.m

```
% Define the transfer function numerator and denominator
coefficients
numerator = [1 1 3 2 2 ]; % coefficients of numerator
denominator = [1 2 1 3 4 3]; % coefficients of denominator
% Create a transfer function object
tf_sys = tf(numerator, denominator);
% Plot the poles and zeros of the transfer function
figure;
pzmap(tf_sys);
title('Poles and Zeros of the Transfer Function');
```

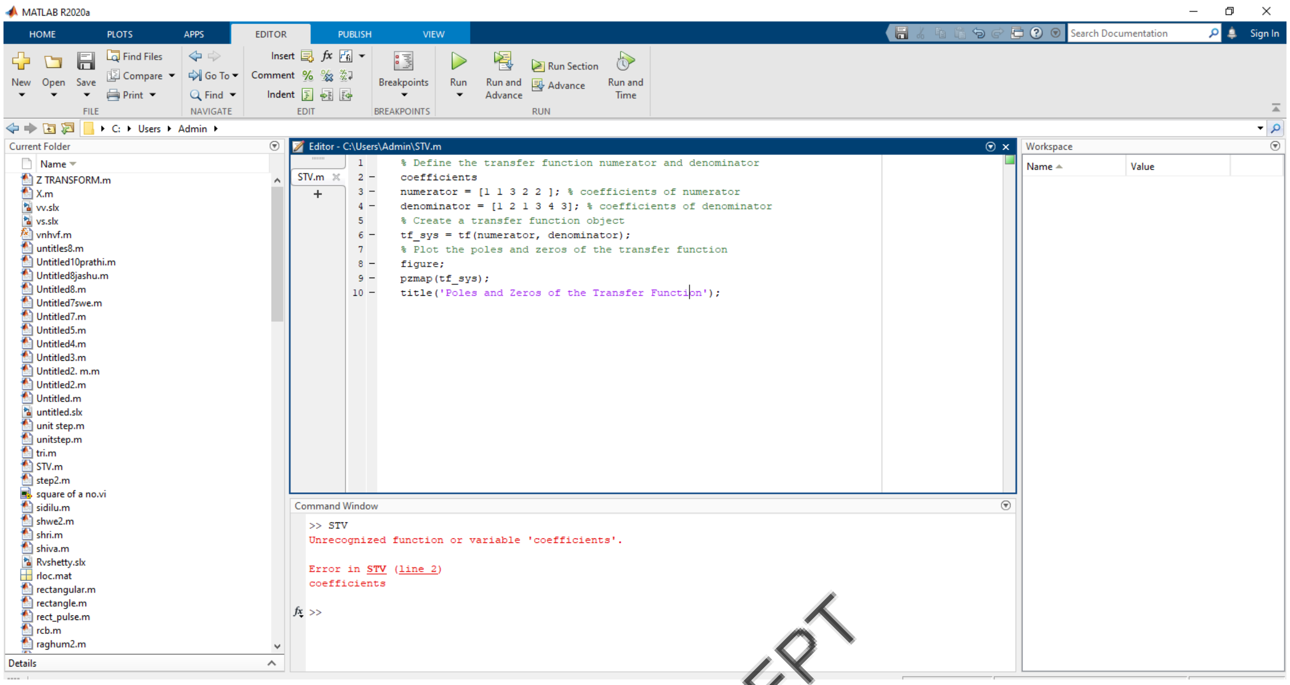
Step5: By Selecting Editor Button Click on RUN Button to Execute MATLAB Program Click on Change folder


MATLAB Editor

File C:\Users\Admin\STV.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.

Step6: If any error is present in MATLAB Program it will be displayed in Command window as follows. Correct the respective error.



```
% Define the transfer function numerator and denominator
coefficients
numerator = [1 1 3 2 2 ]; % coefficients of numerator
denominator = [1 2 1 3 4 3]; % coefficients of denominator
% Create a transfer function object
tf_sys = tf(numerator, denominator);
% Plot the poles and zeros of the transfer function
figure;
pzmap(tf_sys);
title('Poles and Zeros of the Transfer Function');
```

```
>> STV
Unrecognized function or variable 'coefficients'.

Error in STV (line 2)
coefficients

>>
```

Step7: After Correcting the error click on the RUN button again, MATLAB program will be Executed and result will be displayed in command window and respective result will be displayed.

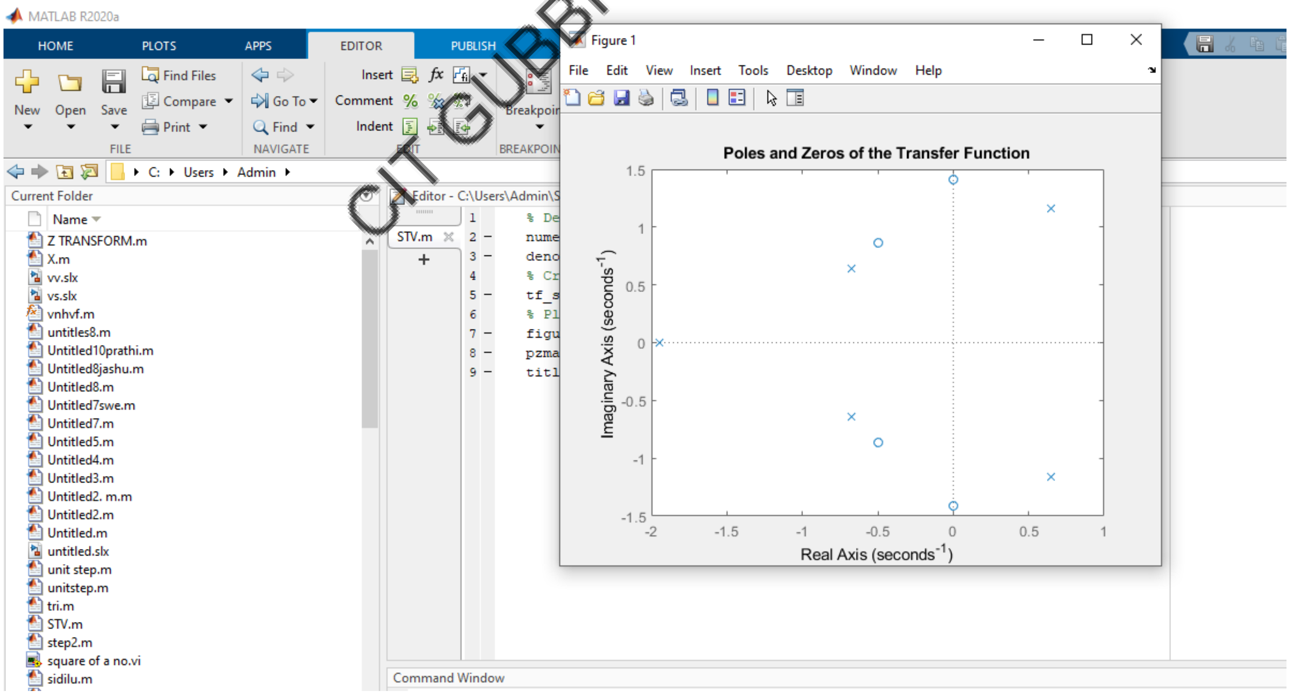


Figure 1
Poles and Zeros of the Transfer Function
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

EXPERIMENT NO 1

IMPLEMENT BLOCK DIAGRAM REDUCTION TECHNIQUE TO OBTAIN TRANSFER FUNCTION A CONTROL SYSTEM

Program:

```
% Define transfer function G1(s)
numerator_G1 = [1];
denominator_G1 = [1, 2];
G1 = tf(numerator_G1, denominator_G1);

% Define transfer function G2(s)
numerator_G2 = [1];
denominator_G2 = [1, 1];
G2 = tf(numerator_G2, denominator_G2);

% Define transfer function G3(s)
numerator_G3 = [1];
denominator_G3 = [1, 3];
G3 = tf(numerator_G3, denominator_G3);

% Define transfer function H1(s)
numerator_H1 = [1];
denominator_H1 = [1, 1];
H1 = tf(numerator_H1, denominator_H1);

% Define transfer function H2(s)
numerator_H2 = [1];
denominator_H2 = [1, 4];
H2 = tf(numerator_H2, denominator_H2);

% Perform block diagram reduction
sys = feedback(series(G1, G2), H1, -1);
sys = feedback(sys, G3, -1);
sys = series(sys, H2);

% Display the transfer function
disp('Transfer Function of the System after Block Diagram
Reduction:');
disp(sys);
```

Result:

```
Transfer Function of the System after Block Diagram Reduction:
tf with properties:
Numerator: {[0 0 0 1 4 3]}
Denominator: {[1 11 45 87 86 40]}
```

EXPERIMENT NO 2

IMPLEMENT SIGNAL FLOW GRAPH TO OBTAIN TRANSFER FUNCTION A CONTROL SYSTEM.

Program:

```
% Define the Signal Flow Graph (SFG) represented as an adjacency
matrix
sfg = [
    1, 2, 1;  % Node 1 to Node 2 with gain 1
    2, 3, 1;  % Node 2 to Node 3 with gain 1
    3, 4, 1;  % Node 3 to Node 4 with gain 1
    4, 2, -1; % Node 4 to Node 2 with gain -1 (negative feedback)
];

% Create the transfer function from the SFG
[numerator, denominator] = sfg2tf(sfg);

% Display the transfer function
sys_tf = tf(numerator, denominator);
disp('Transfer Function of the System:');
disp(sys_tf);
```

Result:

Transfer Function of the System:

Numerator: {[1 3]}
Denominator: {[1 1 5 8 4]}

EXPERIMENT NO 3

SIMULATION OF POLES AND ZEROS OF A TRANSFER FUNCTION.

Program:

```
% Define the transfer function numerator and denominator
coefficients
numerator = [1 1 3 2 2 ]; % coefficients of numerator
denominator = [1 2 1 3 4 3]; % coefficients of denominator

% Create a transfer function object
tf_sys = tf(numerator, denominator);

% Plot the poles and zeros of the transfer function
figure;
pzmap(tf_sys);
title('Poles and Zeros of the Transfer Function');
```

Result:

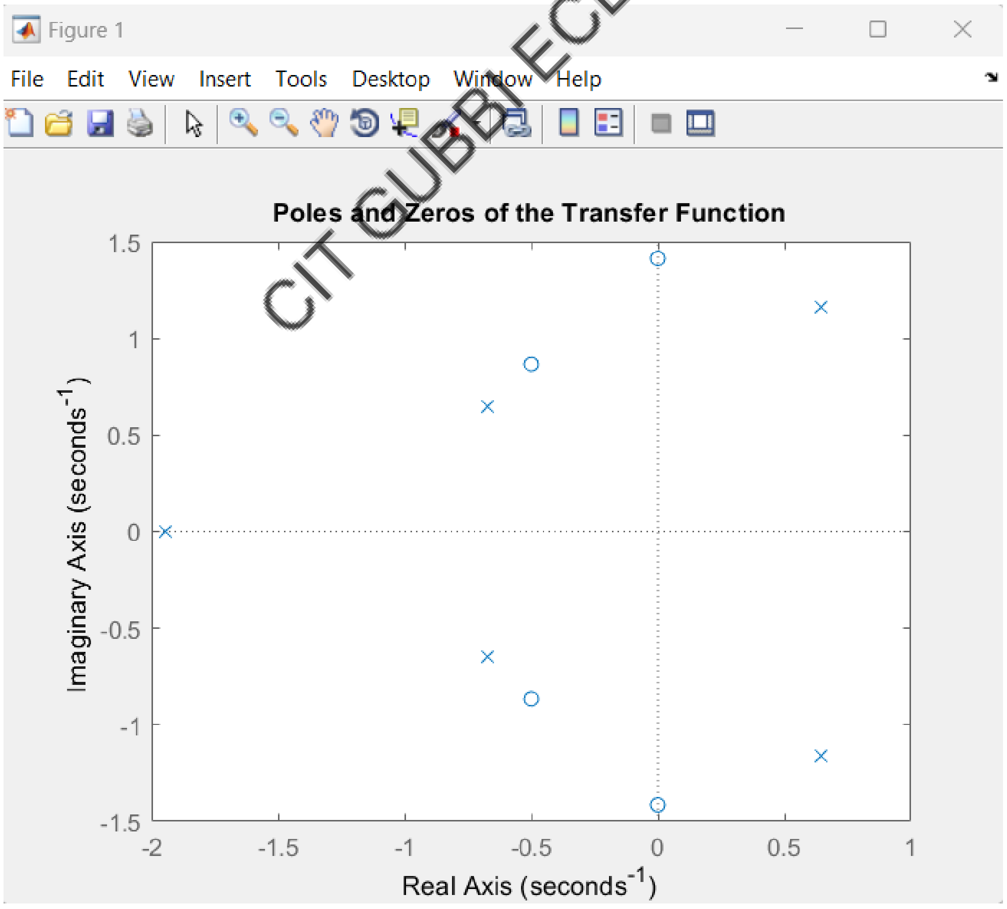


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Poles and Zeros of the Transfer Function
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

EXPERIMENT NO 4

IMPLEMENT TIME RESPONSE SPECIFICATION OF A SECOND ORDER UNDER DAMPED SYSTEM, FOR DIFFERENT DAMPING FACTORS.

Program:

```
e1=0.2;
wn=5;
n1=[wn^2];
d1=[1 2*e1*wn wn^2];
c1=tf(n1,d1);
t=0:0.01:5;
subplot(2,2,1);
step(c1,t);
grid;
e2=0.5;
wn=5;
n2=[wn^2];
d2=[1 2*e2*wn wn^2];
c2=tf(n2,d2);
t=0:0.01:5;
subplot(2,2,2);
step(c2,t);
grid;
e3=0.7;
wn=5;
n3=[wn^2];
d3=[1 2*e3*wn wn^2];
c3=tf(n3,d3);
t=0:0.01:5;
subplot(2,2,3);
step(c3,t);
grid;
e4=0.9;
wn=5;
n4=[wn^2];
d4=[1 2*e3*wn wn^2];
c4=tf(n4,d4);
t=0:0.01:5;
subplot(2,2,4);
step(c4,t);
grid;
figure(2)
step(c1,c2,c3,c4,'r--');
title('comparision of all the under damped responses');
legend({'e1=0.2','e2=0.5','e3=0.7','e4=0.9'});
legend('boxoff');
grid;
```

Result:

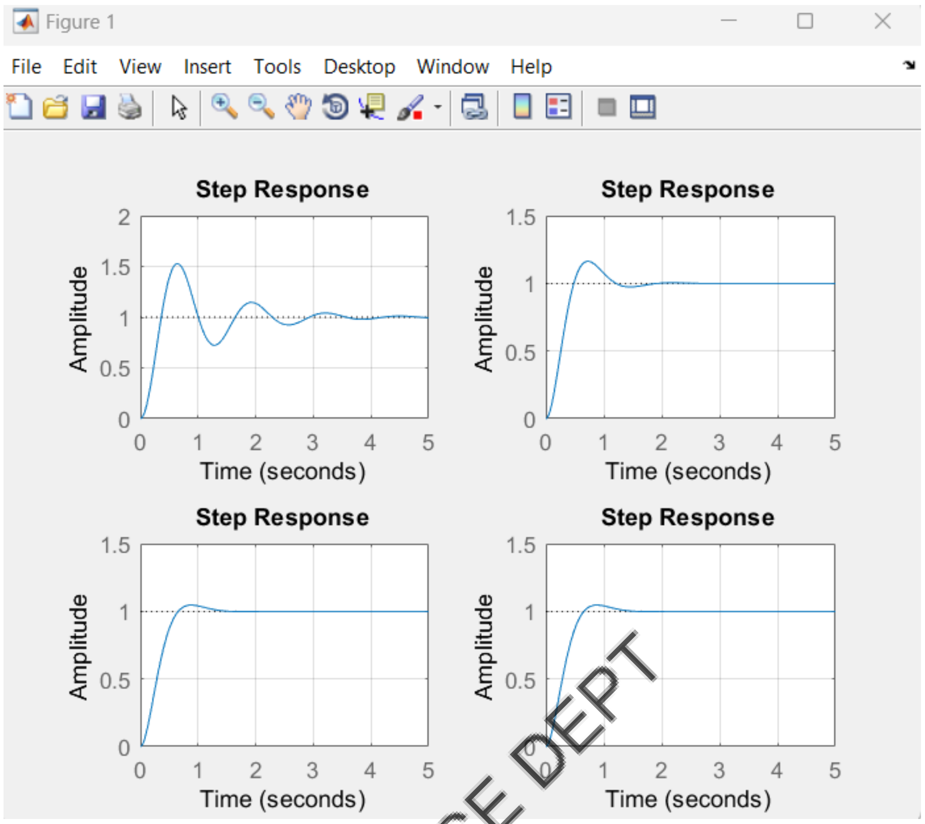


Figure 1
File Edit View Insert Tools Desktop Window Help
Step Response
Amplitude
Time (seconds)

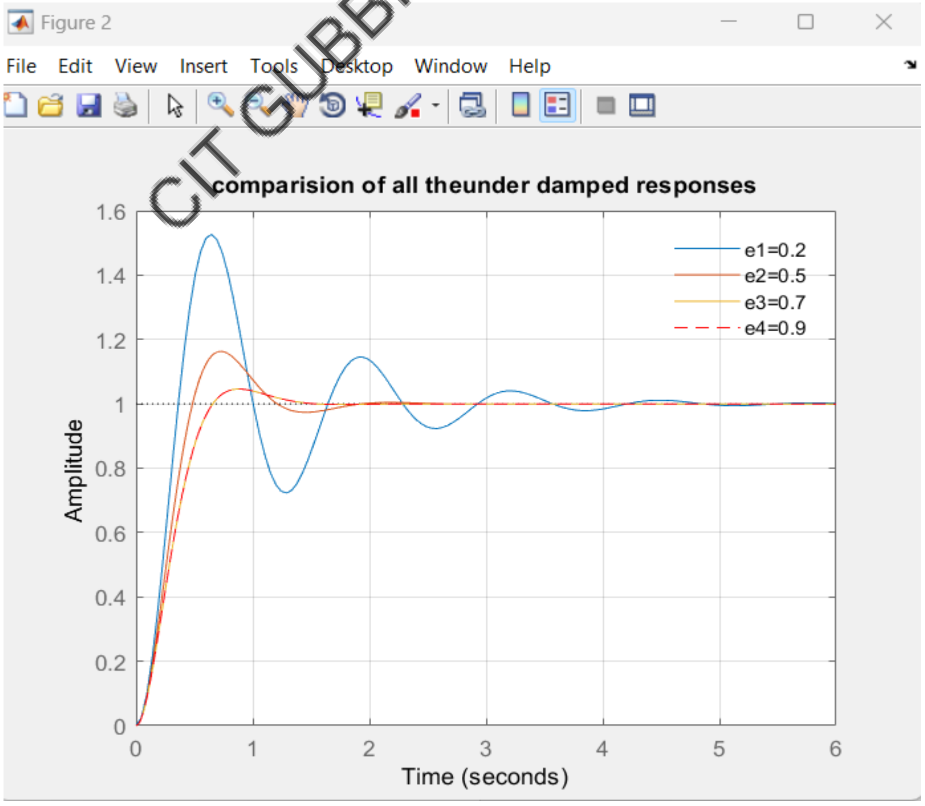


Figure 2
File Edit View Insert Tools Desktop Window Help
comparision of all theunder damped responses
e1=0.2
e2=0.5
e3=0.7
e4=0.9
Amplitude
Time (seconds)

EXPERIMENT NO 5

IMPLEMENT FREQUENCY RESPONSE OF A SECOND ORDER SYSTEM.

Program:

```
% Define the parameters of the second-order system
zeta = 0.1; % damping ratio
omega_n = 10; % natural frequency

% Define the range of frequencies (logarithmically spaced)
omega = logspace(-1, 2, 100);

% Calculate the frequency response of the second-order system
H = (omega_n^2) ./ (omega.^2 - 2*zeta*omega_n*1i*omega +
omega_n^2);

% Calculate the magnitude and phase response
magnitude = abs(H);
phase = angle(H);

% Plot the magnitude response
subplot(2,1,1);
loglog(omega, magnitude);
xlabel('Frequency (rad/s)');
ylabel('Magnitude');
title('Frequency Response - Magnitude');
grid on;

% Plot the phase response
subplot(2,1,2);
semilogx(omega, rad2deg(phase));
xlabel('Frequency (rad/s)');
ylabel('Phase (degrees)');
title('Frequency Response - Phase');
grid on;
```

Result:

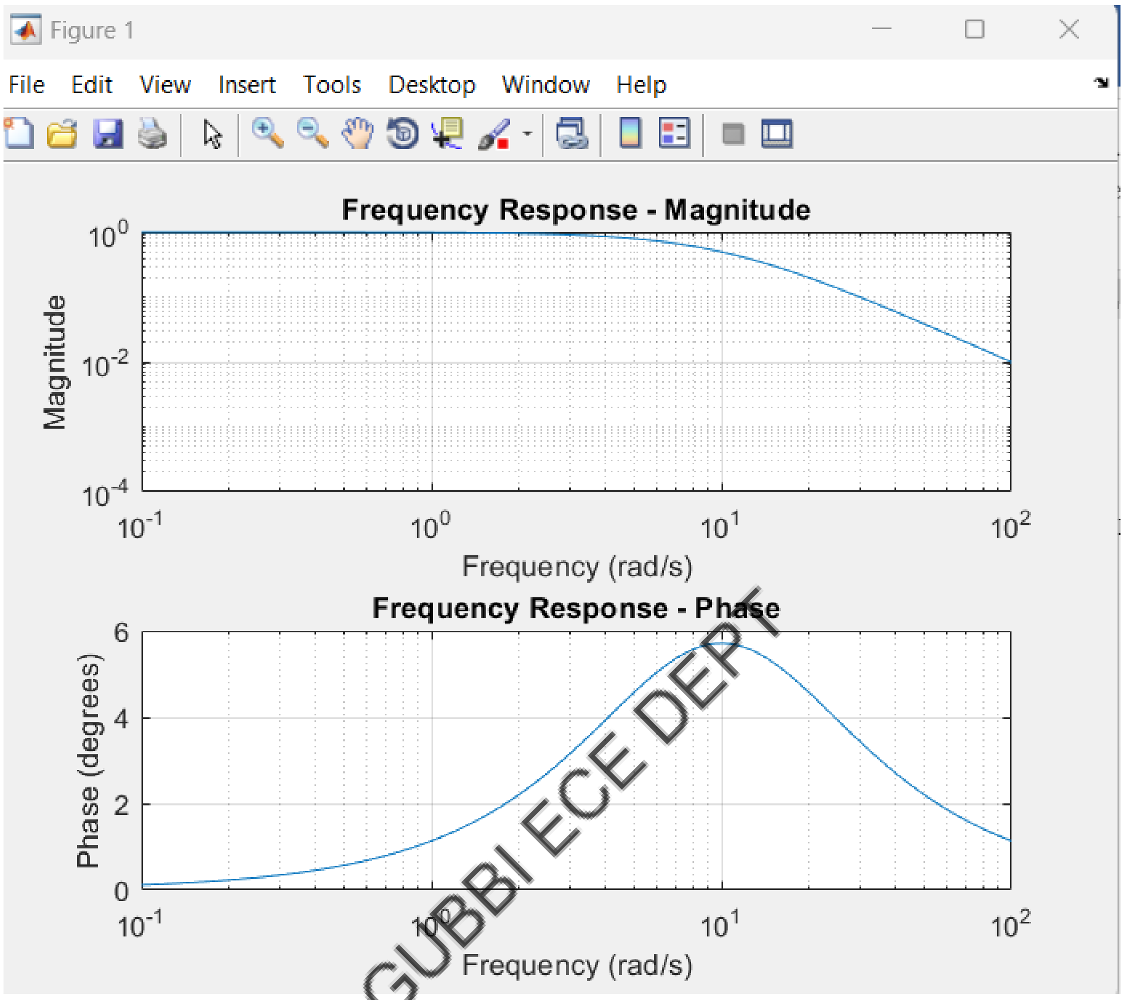


Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency Response - Magnitude
Magnitude
10^0
10^{-2}
10^{-4}
10^{-1}
10^0
10^1
10^2
Frequency (rad/s)
Frequency Response - Phase
Phase (degrees)
6
4
2
0
10^{-1}
10^0
10^1
10^2
Frequency (rad/s)

EXPERIMENT NO 6

IMPLEMENT FREQUENCY RESPONSE OF A LEAD LAG COMPENSATOR.

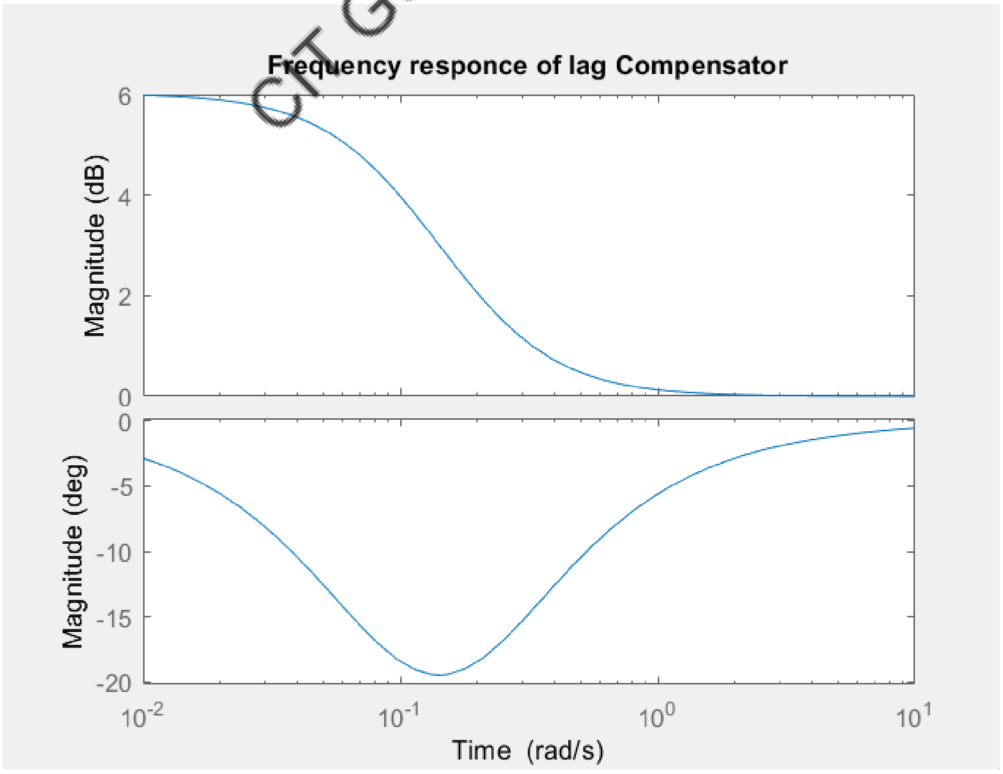
a) Frequency response of a lag compensator.

Program:

```
T=input('enter the value of time constant');
a=input('enter the value of the constant');
n=[1 1/T];
d=[1 1/(a*T)];
c=tf(n,d);
step(c);
bode(c);
xlabel('Time');
ylabel('Magnitude');
title('Frequency responce of lag Compensator');
```

Result:

```
enter the value of time constant 5
enter the value of the constant 2
```



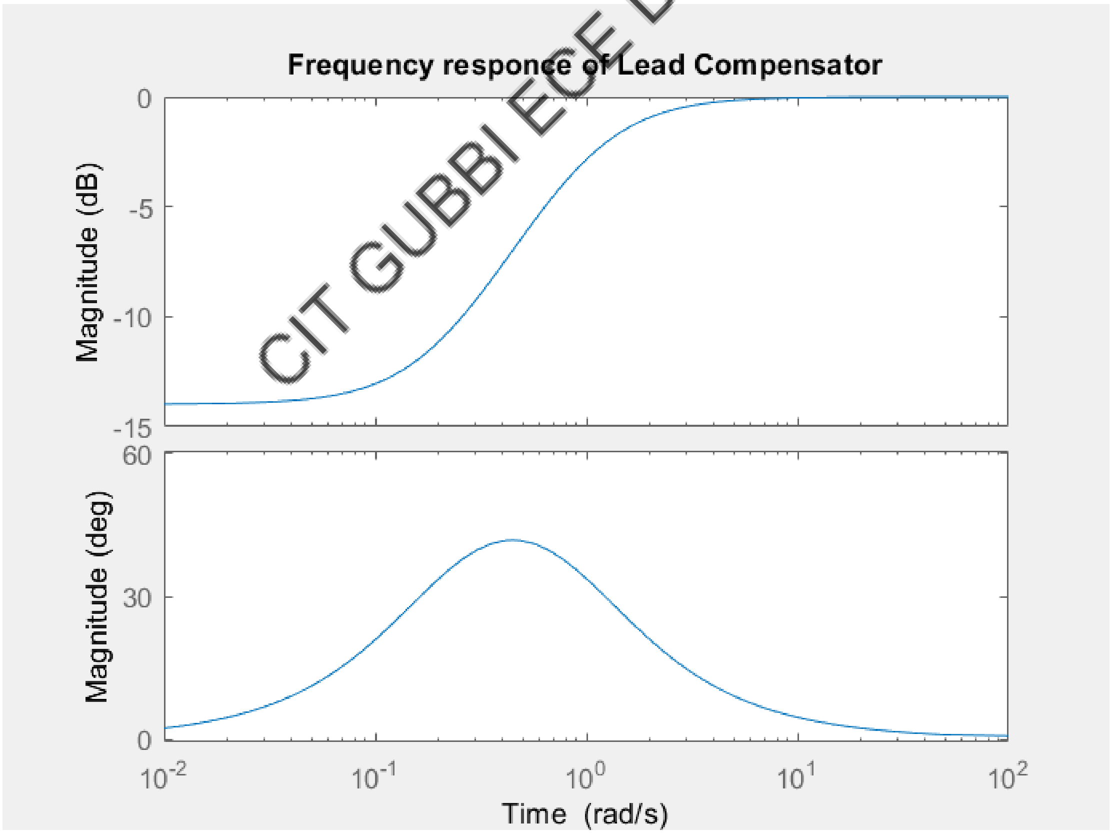
b) Frequency response of a Lead compensator.

Program:

```
T=input('enter the value of time constant');
b=input('enter the value of the constant');
n=[1 1/T];
d=[1 1/(b*T)];
c=tf(n,d);
step(c);
bode(c);
xlabel('Time');
ylabel('Magnitude');
title('Frequency responce of Lead Compensator');
```

Result:

```
enter the value of time constant 5
enter the value of the constant 0.2
```



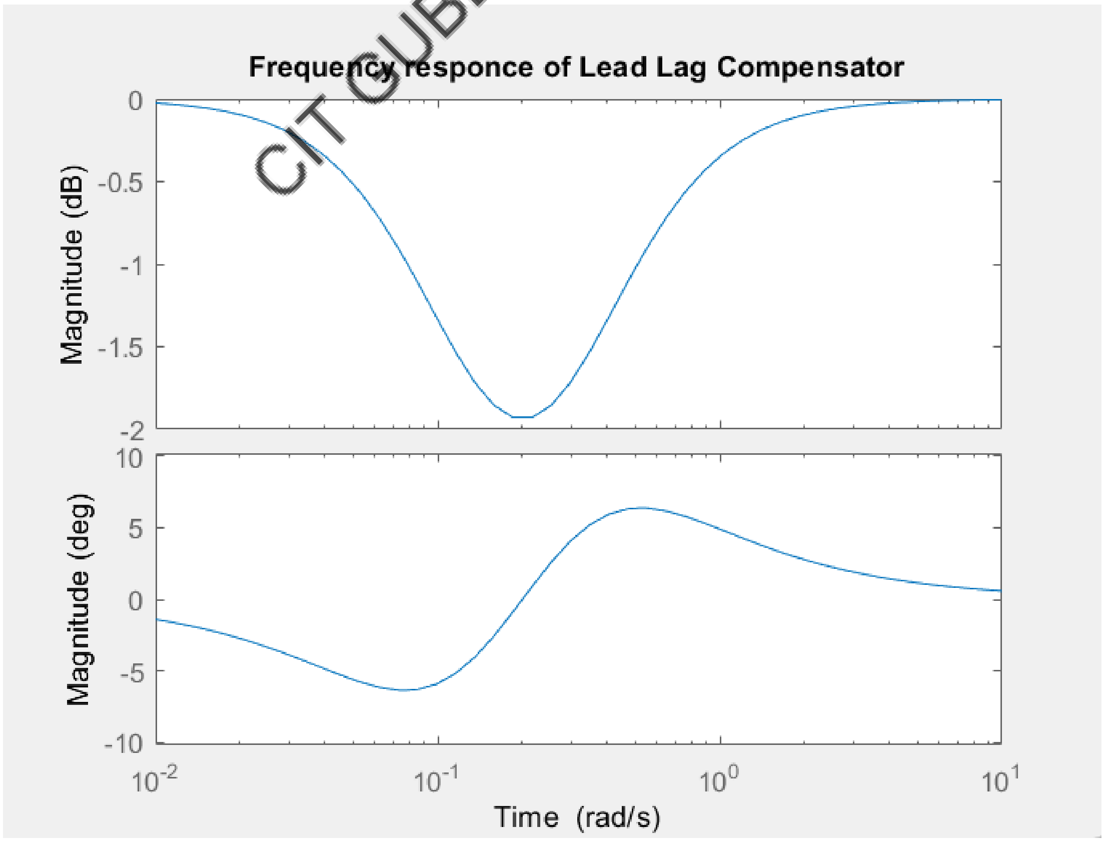
b) Frequency response of a Lead Lag compensator.

Program:

```
T1=input('enter the value of time constant T1:');
T2=input('enter the value of time constant T2:');
b=input('enter the value of the constant b:');
a=input('enter the value of the constant a:');
n1=[1 1/T1];
n2=[1 1/T2];
num=conv(n1,n2);
d1=[1 1/(b*T)];
d2=[1 1/(a*T)];
den=conv(d1,d2);
c=tf(num,den);
step(c);
bode(c);
xlabel('Time');
ylabel('Magnitude');
title('Frequency responce of Lead Lag Compensator');
```

Result:

```
enter the value of time constant T1:5
enter the value of time constant T2:5
enter the value of the constant b:2
enter the value of the constant a:0.5
```



EXPERIMENT NO 7

ANALYSE THE STABILITY OF THE GIVEN SYSTEM USING ROUTH STABILITY CRITERION.

Program:

```
%%%Routh Hurwitz Automatically
clear
clc
close all

%%%%%%INPUT COEFFICIENTS OF CHARACTERISTIC POLYNOMIAL HERE%%%%%
%coeff = [1,-6,-7,-52];
%coeff = [1,2,1];
%coeff = [1,6,11,7,200];
%coeff = [1,10,31,1030]
syms K % - hey if you have the symbolic toolbox try this
%coeff = [1,18,77,K];
%or this
%coeff = [1,9,(K-10),2]
coeff=[1,9,8,K]

%%%%Create the First Row of the Routh Table
routh_table = [];
first_row = [];
for idx = 1:2:length(coeff)
    first_row = [first_row,coeff(idx)];
end
while length(first_row) <= 2
    first_row = [first_row,0];
end
disp('First Row')
routh_table = [routh_table;first_row]
%%%%Create the second row of the Routh Table
second_row = [];
for idx = 2:2:length(coeff)
    second_row = [second_row,coeff(idx)];
end
while length(second_row) < length(first_row)
    second_row = [second_row,0];
end
disp('Second Row')
routh_table = [routh_table;second_row]
routh_table_width = length(first_row);

%%%Now create the next rows
required_rows_to_compute = length(coeff)-2;

%%%Check and see if this is first order or smaller
if required_rows_to_compute < 0
```

```
        disp('Trivial Solution')
        return
    end

    for loop_row = 1:required_rows_to_compute
        row = [];
        %disp(['Computing Row ',num2str(loop_row+2)])
        disp('Divide All Determinants by this var = ')
        divisor = routh_table(loop_row+1,1)
        %%THe left part of the determinant for a given row is constant
        %Thanks Sumit Godara for pointing that out
        %disp('Left Half Determinat')
        left_half_det = routh_table(loop_row:loop_row+1,1);
        for col = 1:routh_table_width
            %disp(['Computing Column ',num2str(col)])
            if col == routh_table_width
                right_half_det = [0;0];
            else
                right_half_det =
    routh_table(loop_row:loop_row+1,col+1);
            end
            disp('Determinant to be computed')
            both_det = [left_half_det,right_half_det]
            value = -det(both_det)/divisor;
            row = [row,value];
        end
        disp('Next Row of Routh Table')
        routh_table = [routh_table;row]
    end

    %%%Check for stability
    %%%Grab the first column
    routh_table
    first_column = routh_table(:,1)
    s = sign(first_column(1));
    unstable = 0;
    for idx = 2:length(first_column)
        value = first_column(idx);
        if s ~= sign(value)
            disp('Unstable ')
            unstable = 1;
            break
        end
    end
    if ~unstable
        disp('System is Stable')
    end
```

Result:
Unstable

EXPERIMENT NO 8

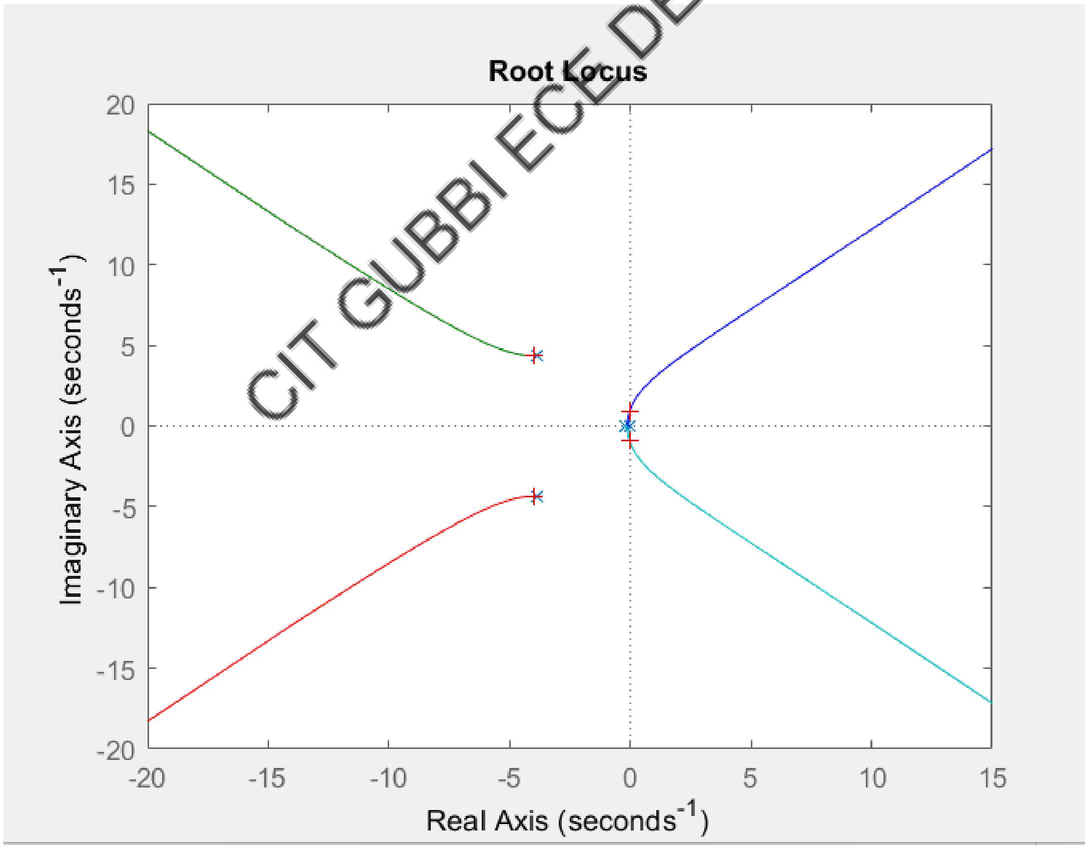
ANALYSE THE STABILITY OF THE GIVEN SYSTEM USING ROOT LOCUS.

Program:

```
num=input('enter the value of Numerator');
den=input('enter the value of Denominator');
sys=tf(num,den);
rlocus(sys);
[K,poles]=rlocfind(sys);
```

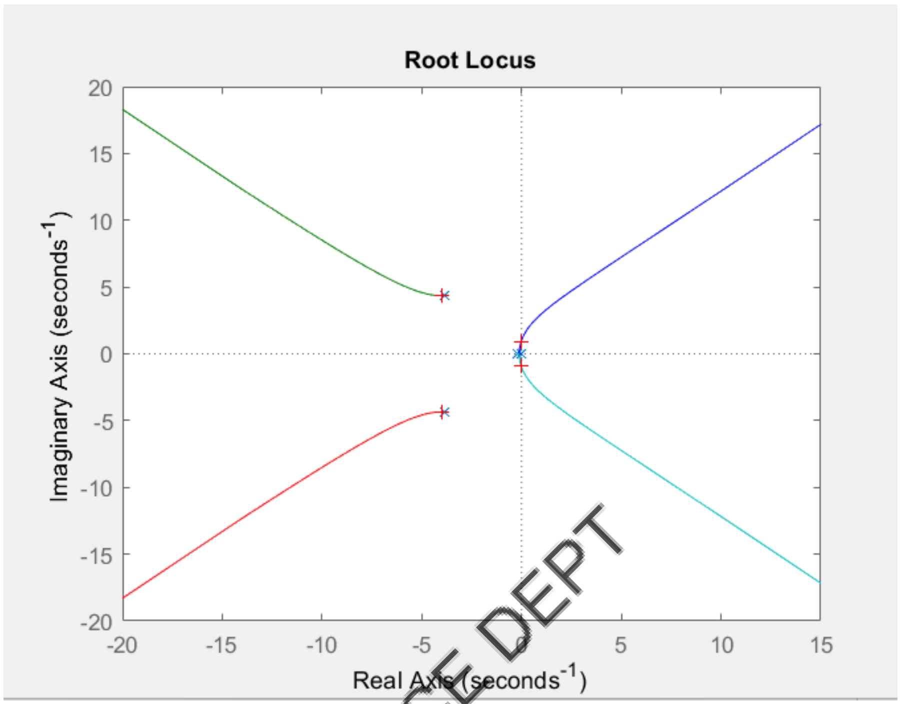
Result 1:

```
enter the value of Numerator1
enter the value of Denominator [1 8 36 8 0]
```



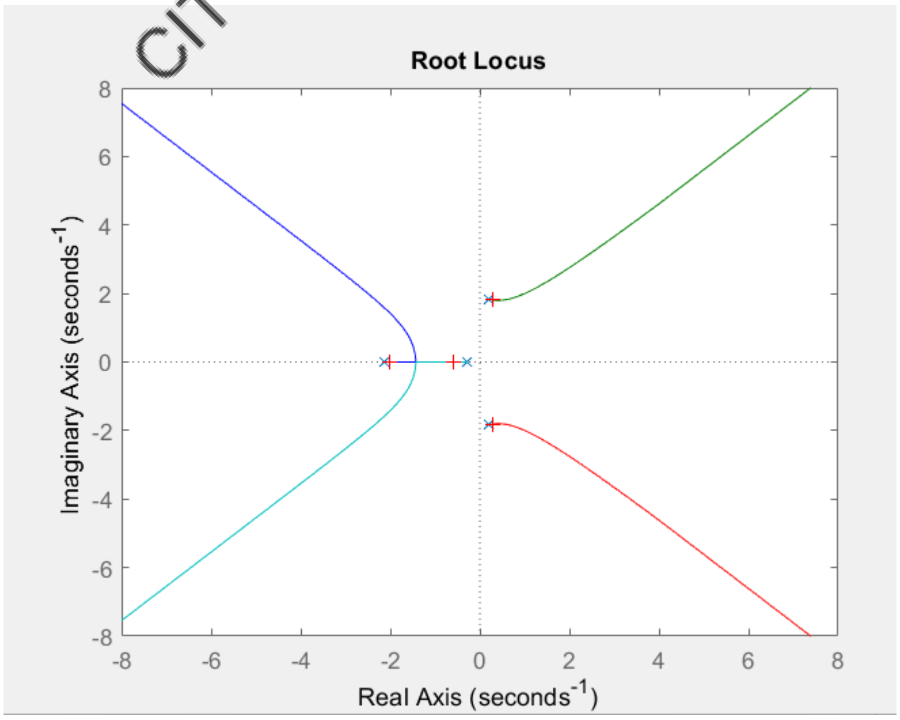
Result 2:

```
enter the value of Numerator1
enter the value of Denominator [1 2 3 8 2]
```



Result 3:

```
enter the value of Numerator1
enter the value of Denominator [1 2 8 12 20 16 16]
```



EXPERIMENT NO 9

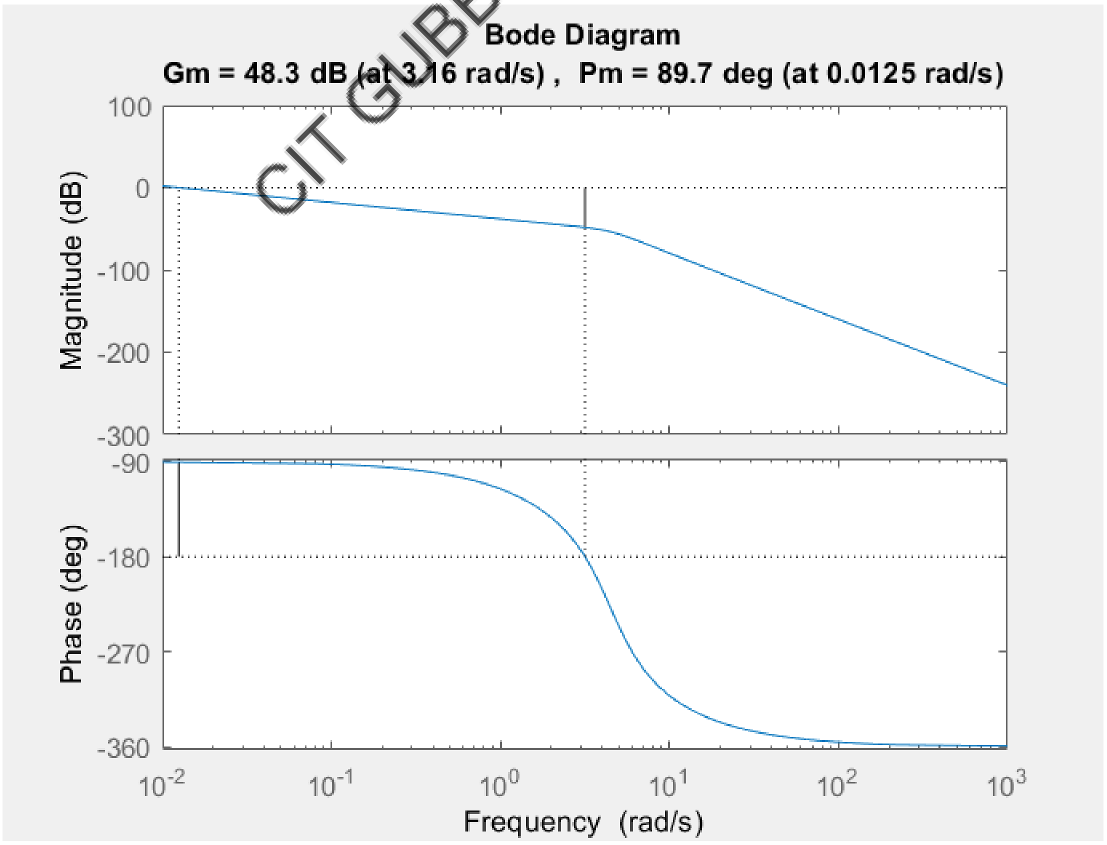
ANALYSE THE STABILITY OF THE GIVEN SYSTEM USING BODE PLOTS.

Program:

```
num=input('enter the value of Numerator');
den=input('enter the value of Denominator');
sys=tf(num,den);
bode(sys);
[gm, pm, wpc, wgc]=margin(sys);
margin(sys);
if(wpc>wgc)
disp('system is stable')
else
disp('system is unstable')
end
```

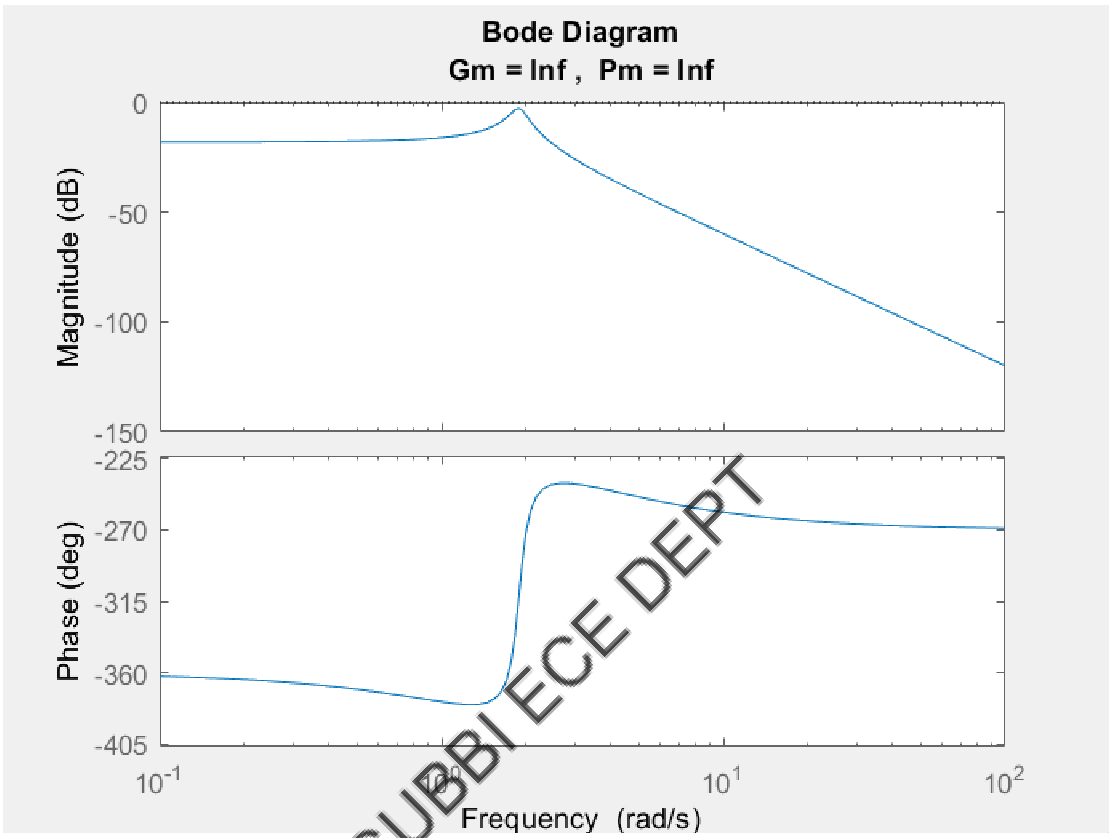
Result 1:

```
enter the value of Numerator1
enter the value of Denominator [1 8 36 80 0]
system is stable
```



Result 2:

```
enter the value of Numerator1
enter the value of Denominator [1 2 3 8]
system is unstable
```



EXPERIMENT NO 10

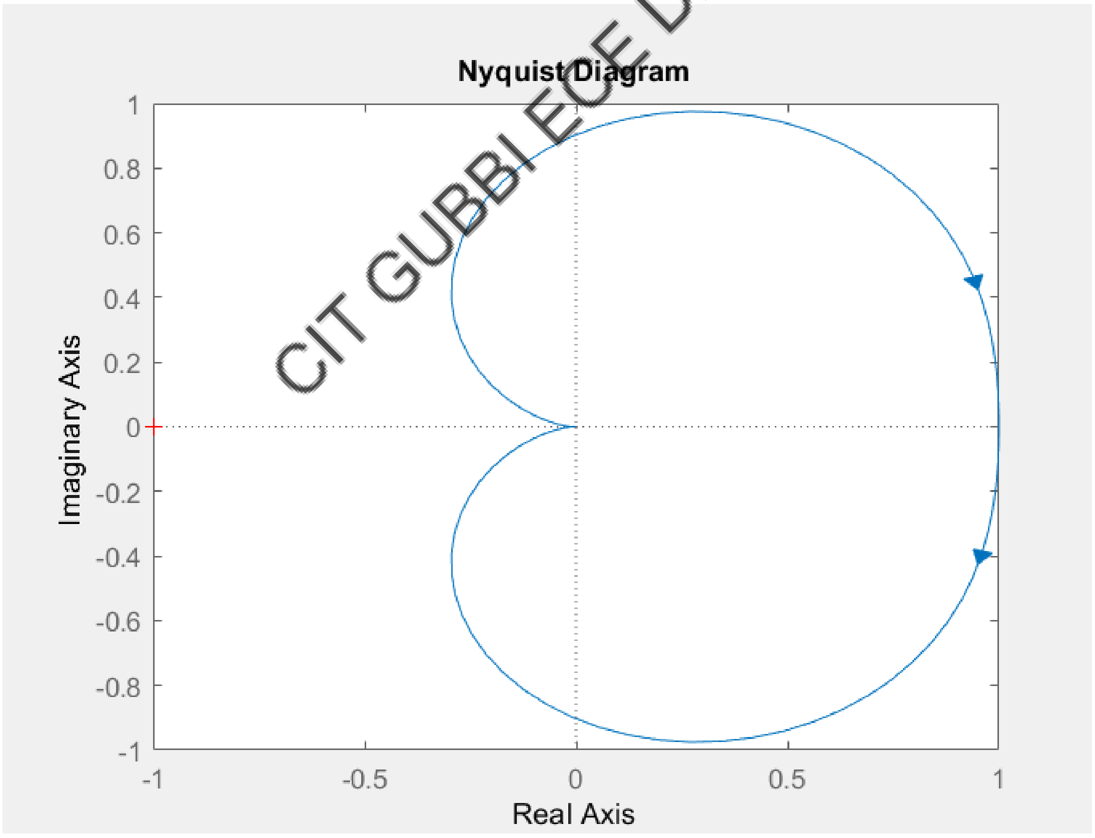
ANALYSE THE STABILITY OF THE GIVEN SYSTEM USING NYQUIST PLOT.

Program:

```
num=input('enter the value of Numerator');
den=input('enter the value of Denominator');
sys=tf(num,den);
nyquist(sys);
```

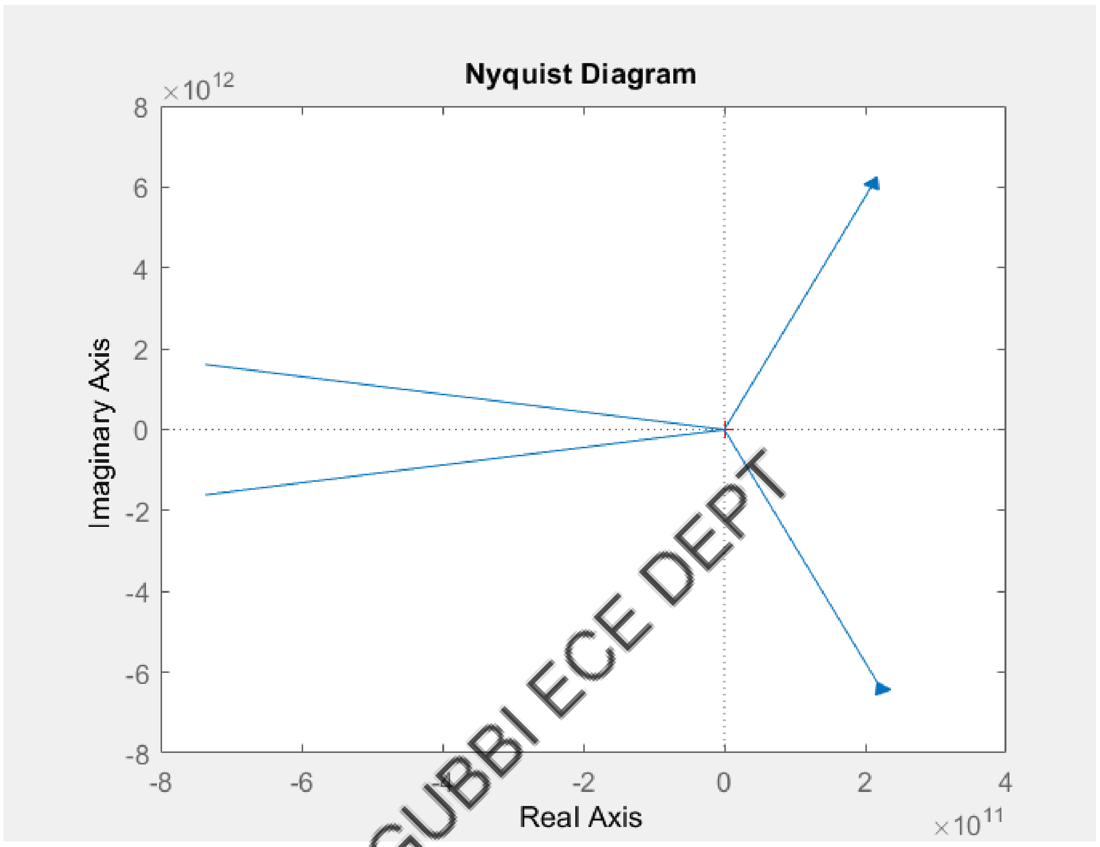
Result 1:

```
enter the value of Numerator1
enter the value of Denominator [1 64 320 20 1]
system is stable
```



Result 2:

```
enter the value of Numerator1
enter the value of Denominator [1 2 8 12 20 16 16]
system is unstable
```



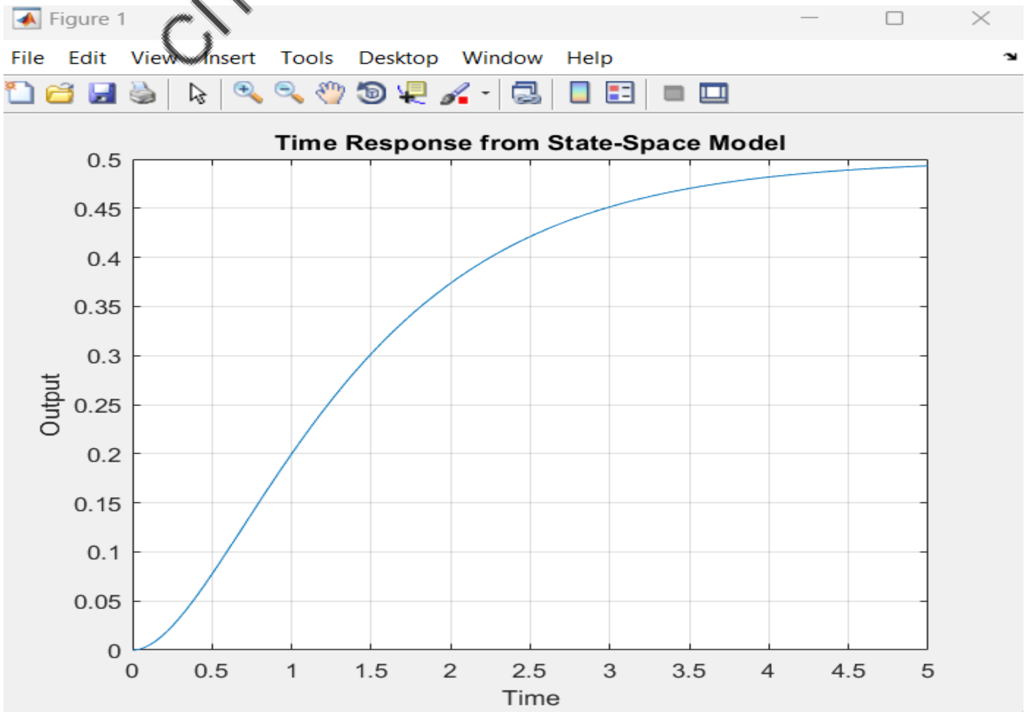
EXPERIMENT NO 11

OBTAIN THE TIME RESPONSE FROM STATE MODEL OF A SYSTEM.

Program:

```
% Define the state-space model
A = [0 1; -2 -3];
B = [0; 1];
C = [1 0];
D = 0;

% Define time vector
t = 0:0.01:5;

% Define input vector (optional)
u = ones(size(t));  % For example, a step input

% Initial condition (optional)
x0 = [0; 0];  % Initial state

% Obtain the time response using lsim
[y, t, x] = lsim(ss(A, B, C, D), u, t, x0);

% Plot the response
plot(t, y);
xlabel('Time');
ylabel('Output');
title('Time Response from State-Space Model');
grid on;
```

Result :



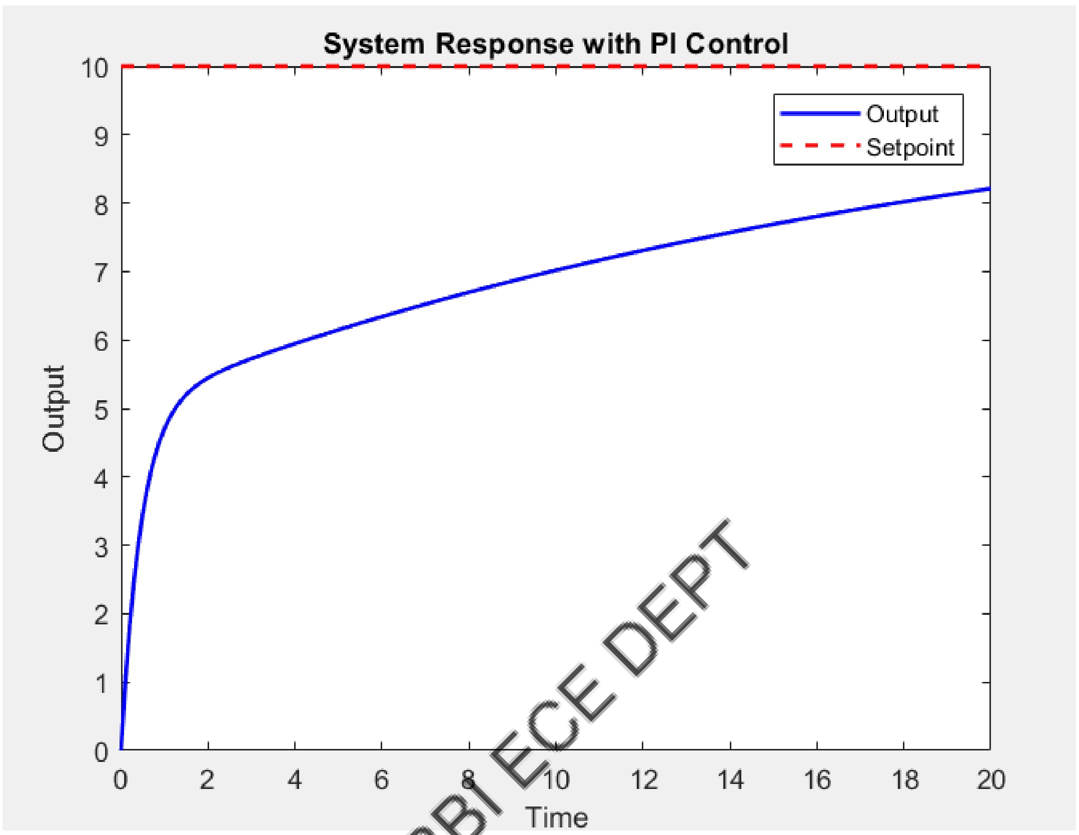
EXPERIMENT NO 12

IMPLEMENT PI AND PD CONTROLLERS.

Program: PI CONTROLLERS

```
% Define system parameters
setpoint = 10; % Setpoint
Kp = 1; % Proportional gain
Ki = 0.1; % Integral gain
dt = 0.1; % Time step
t_end = 20; % Simulation end time
initial_value = 0; % Initial value of the system

% Initialize variables
time = 0:dt:t_end;
error = zeros(size(time));
output = zeros(size(time));
integral = 0;

% Simulate system with PI control
for i = 1:length(time)
    % Calculate error
    error(i) = setpoint - output(i);

    % Calculate proportional term
    P = Kp * error(i);

    % Calculate integral term
    integral = integral + error(i) * dt;
    I = Ki * integral;

    % Calculate control signal
    control_signal = P + I;

    % Update system output (for simplicity, assume a first-order
system)
    output(i+1) = output(i) + dt * (control_signal - output(i));
end

% Plot results
figure;
subplot(1,1,1);
plot(time, output(1:end-1), 'b', 'LineWidth', 1.5);
hold on;
plot(time, setpoint*ones(size(time)), 'r--', 'LineWidth', 1.5);
xlabel('Time');
ylabel('Output');
legend('Output', 'Setpoint');
title('System Response with PI Control');
```

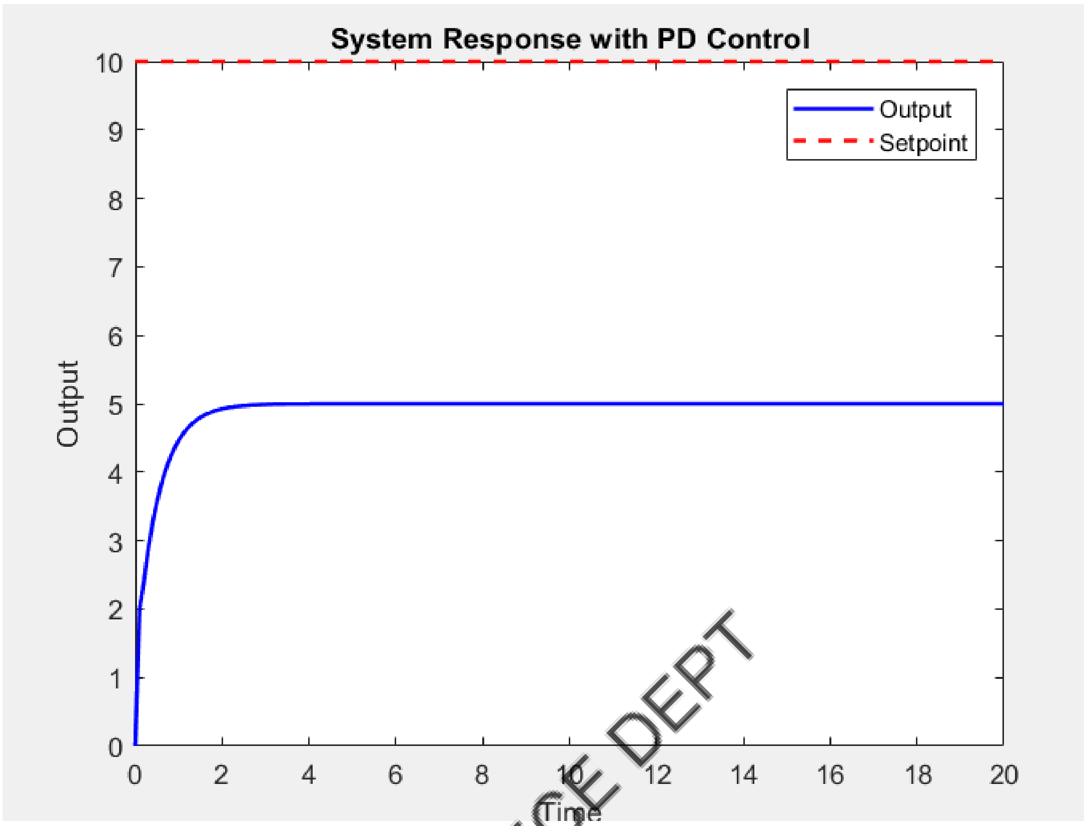
Result:


System Response with PI Control
Output
Setpoint
Output
Time
0
2
4
6
8
10
12
14
16
18
20
0
1
2
3
4
5
6
7
8
9
10

Program: PD CONTROLLERS

```
% Define system parameters
setpoint = 10; % Setpoint
Kp = 1; % Proportional gain
Kd = 0.1; % Derivative gain
dt = 0.1; % Time step
t_end = 20; % Simulation end time
initial_value = 0; % Initial value of the system

% Initialize variables
time = 0:dt:t_end;
error = zeros(size(time));
output = zeros(size(time));
prev_error = 0;

% Simulate system with PD control
for i = 1:length(time)
    % Calculate error
    error(i) = setpoint - output(i);

    % Calculate proportional term
    P = Kp * error(i);

    % Calculate derivative term
    D = Kd * (error(i) - prev_error) / dt;

    % Calculate control signal
    control_signal = P + D;

    % Update system output (for simplicity, assume a first-order
system)
    output(i+1) = output(i) + dt * (control_signal - output(i));

    % Update previous error
    prev_error = error(i);
end

% Plot results
figure;
subplot(1,1,1);
plot(time, output(1:end-1), 'b', 'LineWidth', 1.5);
hold on;
plot(time, setpoint*ones(size(time)), 'r--', 'LineWidth', 1.5);
xlabel('Time');
ylabel('Output');
legend('Output', 'Setpoint');
title('System Response with PD Control');
```

Result:


System Response with PD Control
Output
Setpoint
Output
Time
0
1
2
3
4
5
6
7
8
9
10
0
2
4
6
8
10
12
14
16
18
20

EXPERIMENT NO 13

IMPLEMENT A PID CONTROLLER AND HENCE REALIZE AN ERROR DETECTOR

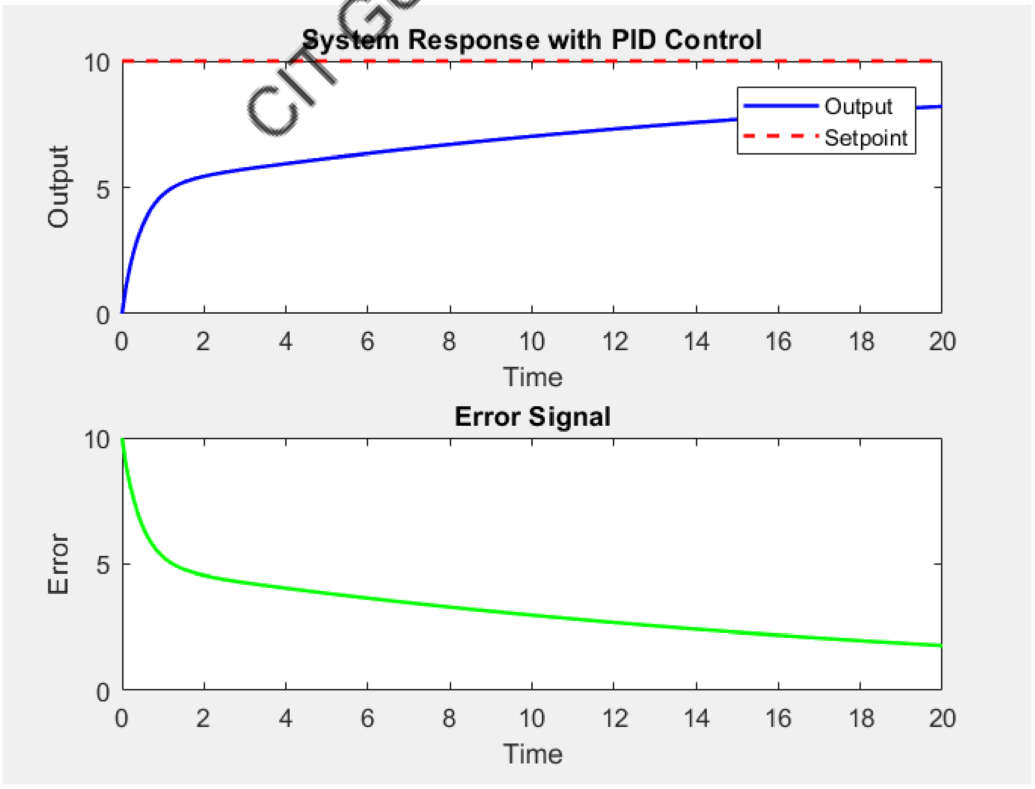
Program:

```
% Define system parameters
setpoint = 10; % Setpoint
Kp = 1; % Proportional gain
Ki = 0.1; % Integral gain
Kd = 0.01; % Derivative gain
dt = 0.1; % Time step
t_end = 20; % Simulation end time
initial_value = 0; % Initial value of the system

% Initialize variables
time = 0:dt:t_end;
error = zeros(size(time));
output = zeros(size(time));
integral = 0;
derivative = 0;
prev_error = 0;

% Simulate system with PID control
for i = 1:length(time)
    % Calculate error
    error(i) = setpoint - output(i);

    % Calculate proportional term
    P = Kp * error(i);

    % Calculate integral term
    integral = integral + error(i) * dt;
    I = Ki * integral;

    % Calculate derivative term
    derivative = (error(i) - prev_error) / dt;
    D = Kd * derivative;

    % Calculate control signal
    control_signal = P + I + D;

    % Update system output (for simplicity, assume a first-order
system)
    output(i+1) = output(i) + dt * (control_signal - output(i));

    % Update previous error
    prev_error = error(i);
end
```

```
% Plot results
figure;
subplot(2,1,1);
plot(time, output(1:end-1), 'b', 'LineWidth', 1.5);
hold on;
plot(time, setpoint*ones(size(time)), 'r--', 'LineWidth', 1.5);
xlabel('Time');
ylabel('Output');
legend('Output', 'Setpoint');
title('System Response with PID Control');

subplot(2,1,2);
plot(time, error, 'g', 'LineWidth', 1.5);
xlabel('Time');
ylabel('Error');
title('Error Signal');

% Error detection
acceptable_error = 0.3; % Define acceptable error threshold
if max(abs(error)) <= acceptable_error
    disp('System is within acceptable error bounds.');
else
    disp('Error detected: System is outside acceptable error
bounds.');
end
```

Result:

Error detected: System is outside acceptable error bounds.



EXPERIMENT NO 14

DEMONSTRATE THE EFFECT OF PI, PD AND PID CONTROLLER ON THE SYSTEM RESPONSE

Program:

```
n=[1];
d=[1 10 20];

%PD Controller
kp=500;
kd=10;
C=pid(kp,0,kd);
P=tf(n,d);
T1=feedback(C*P,1);
t=0:0.01:2;
subplot(3,1,1);
step(T1,t);
grid;
%PI Controller
kp=30;
ki=70;
C=pid(kp,ki);
P=tf(n,d);
T2=feedback(C*P,1);
t=0:0.01:2;
subplot(3,1,2);
step(T2,t);
grid;
%PID Controller
kp=500;
ki=400;
kd=50;
C=pid(kp,ki,kd);
P=tf(n,d);
T3=feedback(C*P,1);
t=0:0.01:2;
subplot(3,1,3);
step(T3,t);
grid;
figure(2)
step(k,T1,T2,T3,'r--');
title('comparision of all the responses');
legend({'pd_controller','pi_controller','pid_controller'});
legend('boxoff');
grid;
```

Result :

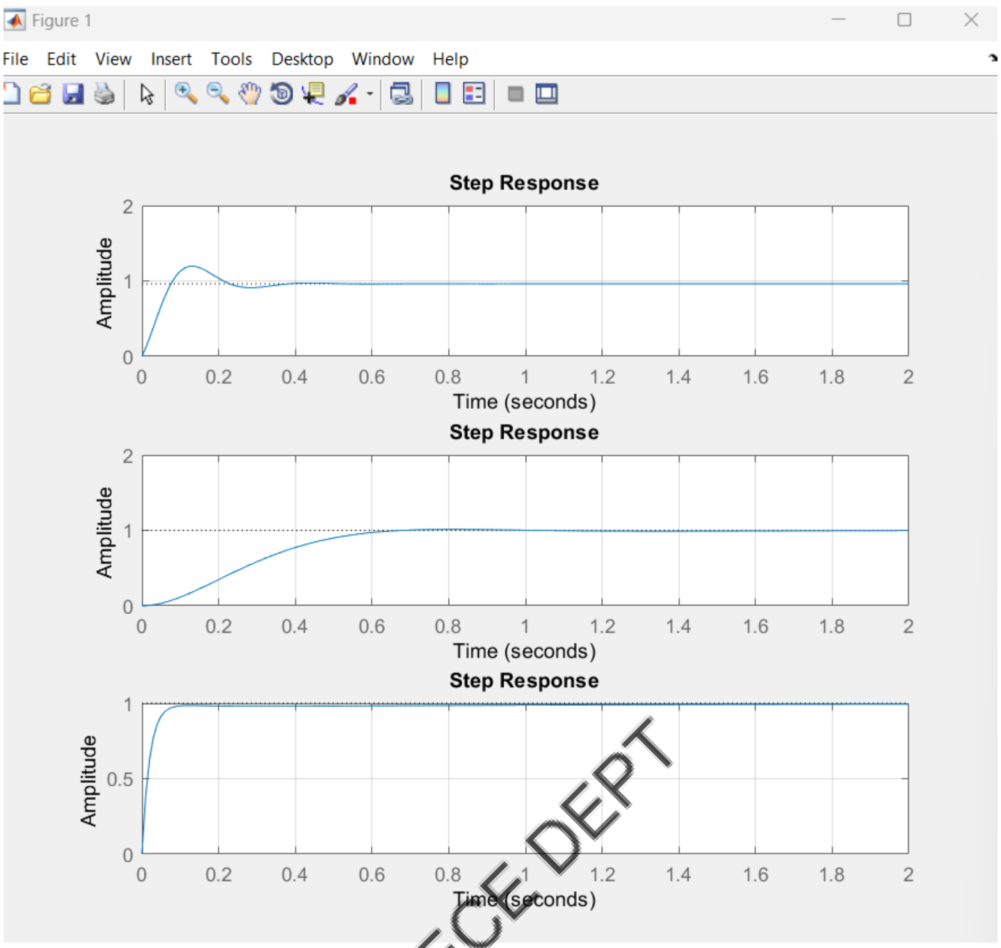
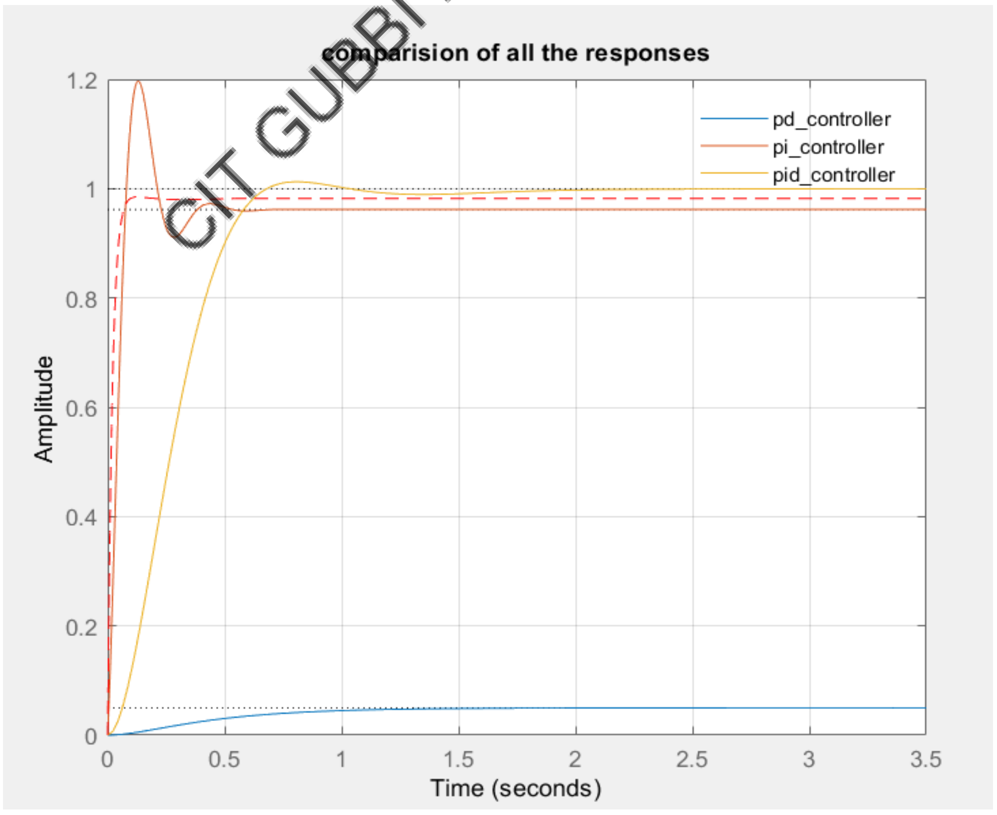


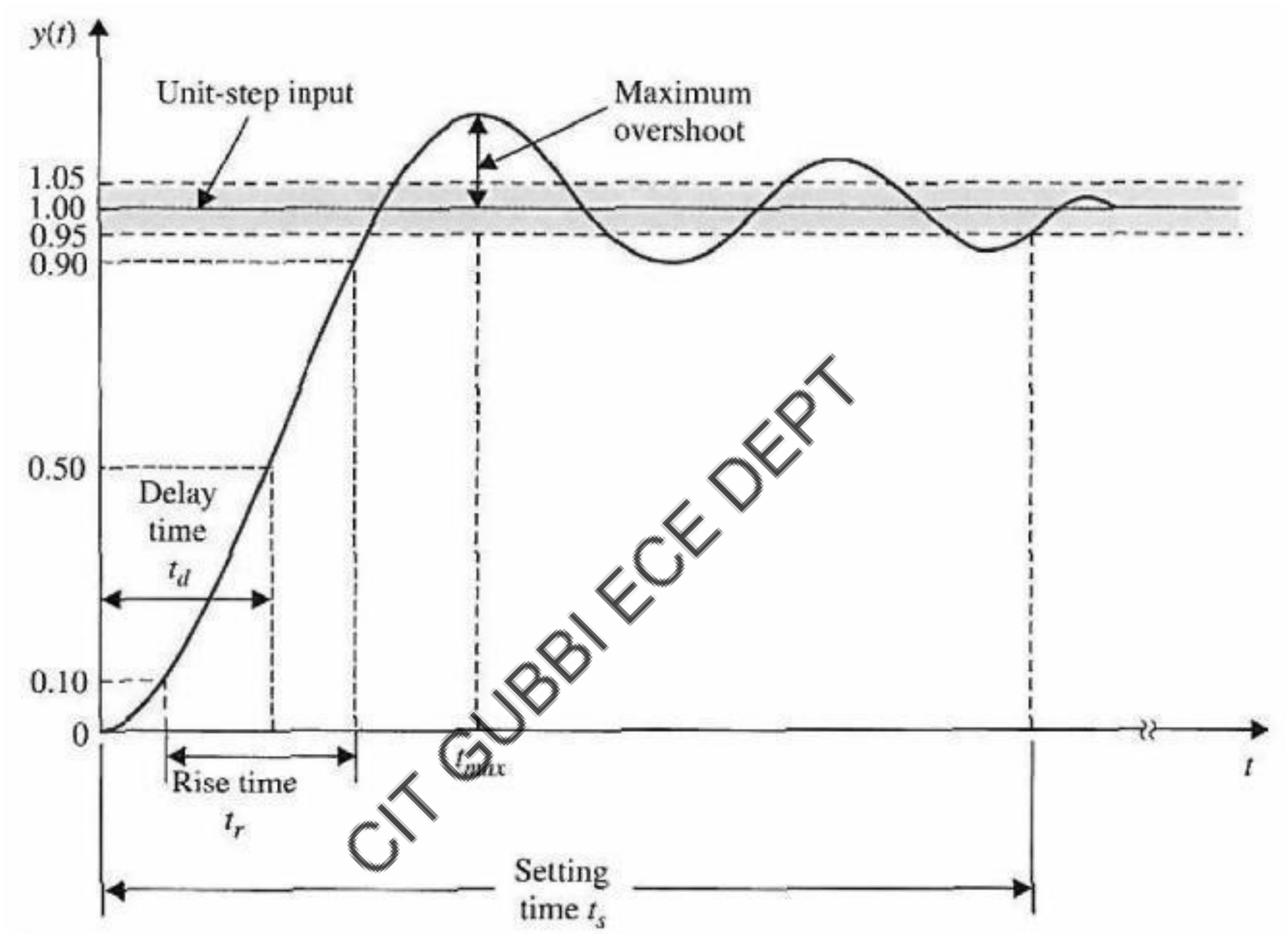
Figure 1
File Edit View Insert Tools Desktop Window Help
Step Response
Amplitude
Time (seconds)
Step Response
Amplitude
Time (seconds)
Step Response
Amplitude
Time (seconds)


comparision of all the responses
pd_controller
pi_controller
pid_controller
Amplitude
Time (seconds)

Beyond Syllabus Experiments

EXPERIMENT A

PLOT UNIT STEP RESPONSE OF GIVEN TRANSFER FUNCTION AND FINDS DELAY TIME, RISE TIME, PEAK TIME AND PEAK OVERSHOOT



Program:

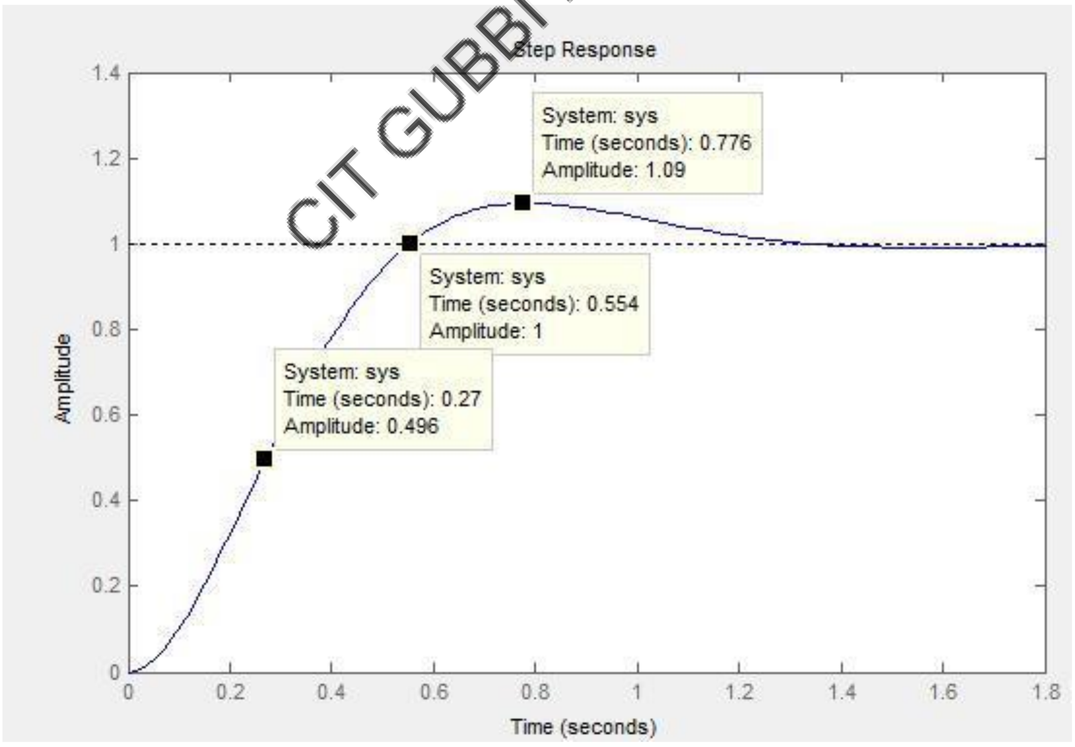
```
% Plot unit step response of given transfer function and find peak
overshoot, peak time, rise time and delay time using MATLAB.

% Transfer Function = 25/(s^2+6s+25)
clear all
clc
syms s t
n=[25]
d=[1 6 25]
sys = tf(n,d)
R=roots(d)
S = stepinfo(sys,'RiseTimeLimits',[0.00,1.00])
ltiview(sys)
```

Result:

```
n =
25
d =
1 6 25
```

Transfer function:

```
25
--------------
s^2 + 6 s + 25
R =
-3.0000 + 4.0000i
-3.0000 - 4.0000i
S =
RiseTime: 0.5538
SettlingTime: 1.1886
SettlingMin: 0.9910
SettlingMax: 1.0948
Overshoot: 9.4778
Undershoot: 0
Peak: 1.0948
PeakTime: 0.7869
```



EXPERIMENT B

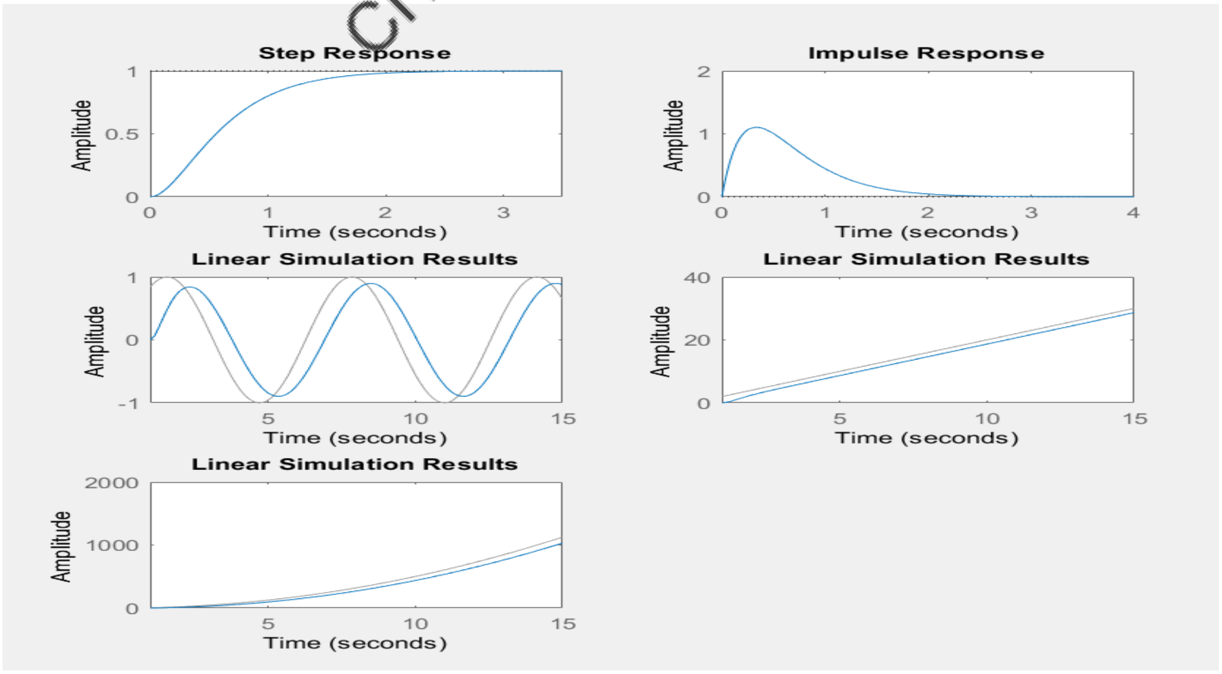
DETERMINE THE TIME RESPONSE OF THE GIVEN SYSTEM SUBJECTED TO ANY ARBITRARY INPUT.

Program:

```
close all;
h1=tf([9],[1 6 9])
t=linspace(1,15); %STEP RESPONSE%
subplot(3,2,1);
step(h1); %IMPULSE RESPONSE%
subplot(3,2,2);
impulse(h1);
r=sin(t);
s=2*t;
q=5*(t.^2); %SINE RESPONSE%
subplot(3,2,3);
lsim(h1,r,t); %RAMP RESPONSE%
subplot(3,2,4);
lsim(h1,s,t); %PARABOLIC RESPONSE%
subplot(3,2,5);
lsim(h1,q,t);
```

Result:

```
h1 =

      9
  -------------
  s^2 + 6 s + 9

Continuous-time transfer function.
```


Step Response
Amplitude
Time (seconds)
Impulse Response
Linear Simulation Results
Linear Simulation Results
Linear Simulation Results