Bharati Vidyapeeth's College of Engineering
New Delhi

LAB MANUAL

Department	**Electronics and Communication Engineering**
Academic Year	**2024-25**
Semester	**5th**
Subject Name	**Introduction to Control Systems Lab**
Subject Code	**EEC-355**
Faculty Name	**Mr. Bhawanand Jha**

VISION OF INSTITUTE

To be an institute of excellence that provides quality technical education and research to create competent graduates for serving industry and society.

MISSION OF INSTITUTE

M1: To impart quality technical education through dynamic teaching-learning environment

M2: To promote research and innovation activities which gives opportunities for life-long learning in context of academic and industry.

M3: To build up links with industry-institute through partnerships and collaborative developmental works.

M4: To inculcate work ethics and commitment in graduates for their future endeavors to serve the society.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF DEPARTMENT

The department aspires to be an advanced center of learning by synergizing teaching, learning, and research to produce competent Electronics and Communication Engineers serving society.

MISSION OF DEPARTMENT

M1: To prepare graduates with strong foundation of technical knowledge and motivate them to explore emerging areas of research.

M2: To create environment for the development of Research & Innovation activities in the related field.

M3: To build strong relationship with industry through collaborative partnerships, research & product development, and student internships.

M4: To instill ethical and professional values among graduates with societal and environmental concern.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

PEO1: To produce graduates with in-depth knowledge in Electronics and Communication Engineering who can provide professional engineering solutions in societal and environmental context.

PEO2: To provide graduates having self-learning abilities and effective communication skills for working as an efficient team member.

PEO3: To provide graduates committed to professional ethics, responsibilities, and standards of Engineering.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Analyse and Design of circuits for analog and digital systems.

PSO2: Identify the role of interfacing devices in communication systems and create a prototype to meet the required functionality.

PROGRAM OUTCOMES (PO)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze 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 with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
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 the information to provide valid conclusions for complex problems.
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.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. 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

TABLE OF CONTENTS

1. COURSE DETAILS

1.1 COURSE OBJECTIVE

The objective of the course is to make students aware with the new concepts of control system and its applications. This course also helps the students in controlling the real-world problems and analysing the behaviour of systems.

1.2 COURSE OUTCOMES

At the end of the course student will be able to:		PO/ PSO	Bloom Level
EEC355.1	Ability to define, understand various terms related to control system and evaluation of transfer function	PO1	L1-Remembering, L2-Understanding,
EEC355.2	Ability to apply knowledge of various types of signals in time response of systems	PO1, PO2, PO4, PO5	L3- Apply
EEC355.3	Ability to analyse frequency response of systems	PO2, PO4, PO5	L4-Analyzing
EEC355.4	Ability to design compensators and controllers	PO3, PO4, PO5	L5-Evaluate

1.3 MAPPING COURSE OUTCOMES (CO) AND PROGRAM OUTCOMES (PO)/ PROGRAM SPECIFIC OUTCOME (PSO)

CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O 1	PS O 2
CO 1	3	3	2	1	1	1	1	-	1	3	-	3		
CO 2	3	2	1	3	2	1	1	-	1	3	-	1		
CO 3	3	2	1	2	3	1	1	-	1	3	-	3	2	
CO 4	3	3	2	1	1	1	1	-	1	3	-	3	3	

1=Slightly, 2=moderately, 3=substantially

1.4 EVALUATION SCHEME

	Laboratory	
Components	**Internal**	**External**
Marks	40	60
Total Marks	100	

1.5 GUIDELINES FOR CONTINUOUS ASSESSMENT FOR EACH EXPERIMENT

- **Attendance and performance in minimum eight experiments – 30 marks for all semesters**
  - Each Experiment will carry a weight of 20 marks
  - Experiment performance [5 Marks]
  - File [10 Marks]
  - Attendance [5 Marks]
- **2 innovative experiments (Content Beyond syllabus) 10 marks for 1st & 2nd Semester**
- **2 innovative experiments (Content Beyond syllabus) 5 marks for 3rd , 4th ,5th , 6th ,7th 8th Semester**
- **Viva 5 marks for 3rd , 4th ,5th , 6th ,7th 8th Semester**

1.6 The Rubrics for Experiment execution and Lab file+ viva voce is given below:

Experiment Marks details:

Status	*Completed and Executed perfectly*	*Completed but partially Executing*	*Logically Incorrect Program or errors*	*Unacceptable efforts/Absent*
Marks	4-5	2-3	1	0

File Marks Details:

Status	*File Contents & Checked Timely*	*File Contents & Checked not Timely (after one week)*	*File Contents & Checked After two weeks*
Marks	4-5	2-3	0-1

Viva-Voce Marks details:

Status	*Viva (Good)*	*Viva (Average)*	*Viva (Unsatisfactory)*
Marks	4-5	1-3	0

Note: Viva Voce Questions for each experiment should be related to Course Outcomes.

1.6 Safety Guidelines/Rules for laboratory(AS PER SUBJECT/LABORATORY)

DO'S:

1. All students must wear uniform compulsory.
2. Must follow the schedule time, late comers will not be permitted.
3. Personal belongings should be placed in the specified place
4. Silence & tidiness should be maintained in the Lab.
5. Cycle of experiments should be followed.
6. Students are expected to come prepared for experiments &VIVA.
7. Handle all the equipments with care & strictly follow the instructions.
8. Check the circuit connections properly & get it checked, verified by staff in-charge before switch it ON.
9. Equipments should be switched OFF and chairs should be placed back in position before leaving the lab.
10. Separate Lab observation book should be maintained. Details regarding observation & relevant information about the experiments should be maintained.
11. Get the observation book signed from the staff-in-charge before leaving the lab.
12. Switch OFF & remove all connections, return instruments before leaving the lab.

13. Practical records should be submitted regularly with complete information (circuit diagram, theory etc).

DON'TS:

1. Don't come late to the lab.
2. Don't enter into the lab with golden rings, bracelets and bangles.
3. Don't make or remove the connections with power ON.
4. Don't switch ON the supply without verifying by the staff member.
5. Don't switch OFF the machine with load.
6. Don't leave the Lab without the permission of the staff in-charge.

1.7 Format for students while writing Experiment in Lab file.

Experiment No: 1

Aim:

Course Outcome:

Software/Hardware used:

Theory:

Flowchart/Algorithm/Code:

Results:

Expected Outcome attained: YES/NO

2. LIST OF EXPERIMENTS AS PER GGSIPU

Sr. No.	Title of Lab Experiments	CO
1	Determination of step & impulse response for a second-order unity feedback system.	(CO2)
2	To study the speed-torque characteristics of SERVO MOTOR.	(CO1)
3	Experiment to draw synchro pair characteristics.	(CO1)
4	To determine the Transfer Function of the DC Machine.	(CO1)
5	Plot unit step response of the given transfer function and finds delay time, rise time, and peak overshoot.	(CO2)
6	Plot the pole-zero configuration in the s-plane for the given transfer function.	(CO3)
7	To determine the characteristics of Magnetic Amplifiers.	(CO1)
8	Linear System Analysis (Time Domain Analysis, Error Analysis) Using MATLAB.	(CO2)
9	To observe the effect of P, PI, PID, and PD Controller for open loop and closed loop of second order system.	(CO4)
10	To analyze the frequency response of a system by plotting Root locus, Bode plot, and Nyquist plot using MATLAB software.	(CO3)
11	Experiment to draw the frequency response characteristics of the lag–lead compensator network and determination of its transfer function.	(CO4)
12	Temperature Controller Using PID Controller.	(CO4)
13	Study of operation of a stepper motor interface with a microprocessor.	(CO1)

CONTENT BEYOND SYLLABUS

S.No	Name of Experiment
1.	To draw polar plot of the control system using MATLAB
2	To study the stability of any higher order control system.

3. EXPERIMENTAL SETUP DETAILS FOR THE COURSE

Software Requirements:
MATLAB

Minimum Hardware Requirements

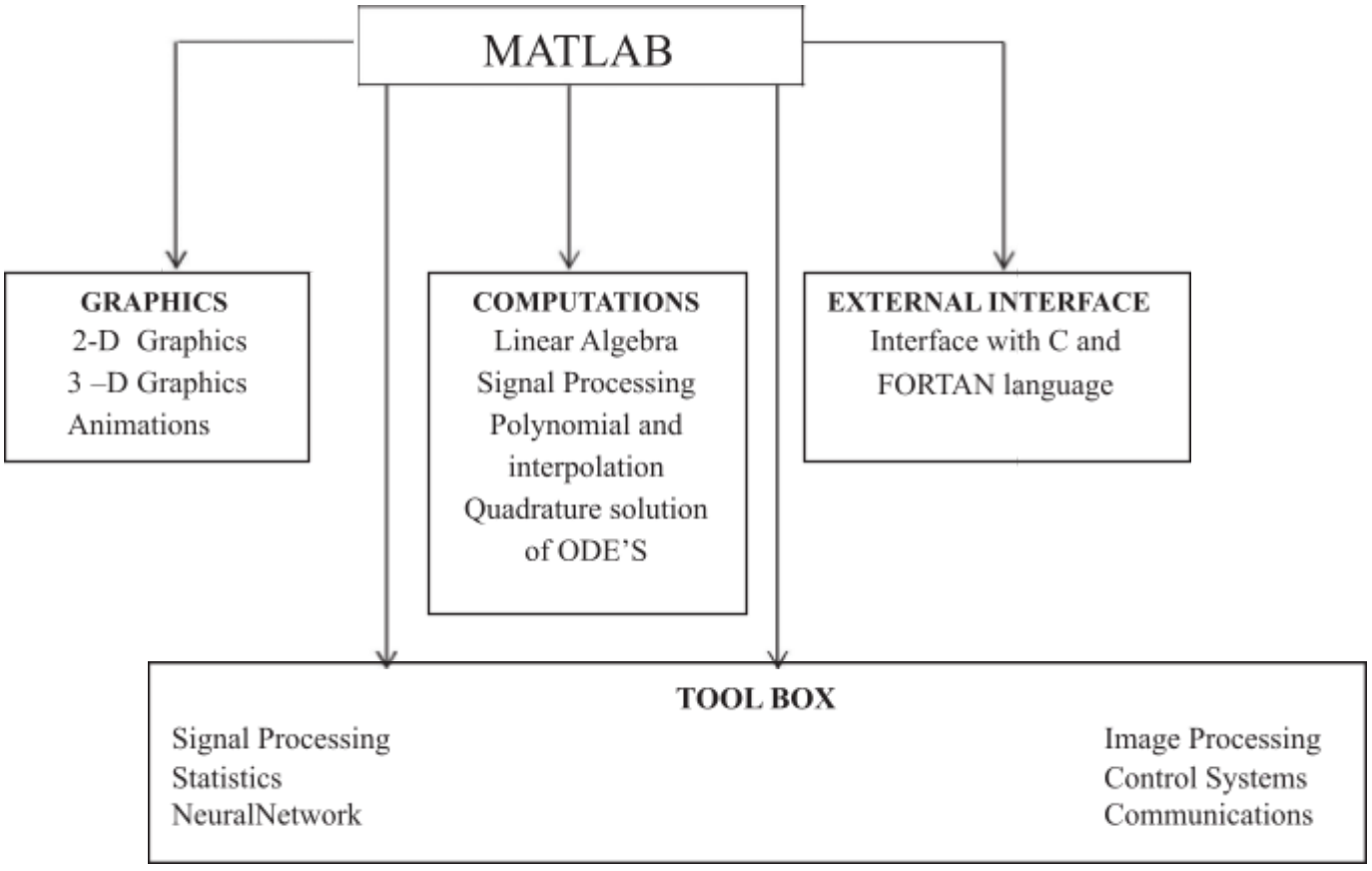
PID control KIT
DC motor open loop /close loop Control KIT
AC servomotor
DC servomotor
Linear systems

1. Introduction

Experimental set up and Introduction to Software

Control system using MATLAB

MATLAB: MATLAB is a software package for high performance numerical computation and visualization provides an interactive environment with hundreds of built in functions for technical computation, graphics and animation. The MATLAB name stands for MATRIX Laboratory The diagram shows the main features and capabilities of MATLAB.



As its core ,MATLAB is essentially a set (a "toolbox") of routines (called "m files" or "mex files") that sit on your computer and a window that allows you to create new variables with names (e.g. voltage and time) and process those variables with any of those routines (e.g. plot voltage against time, find the largest voltage, etc).

It also allows you to put a list of your processing requests together in a file and save that combined list with a name so that you can run all of those commands in the same order at some later time. Furthermore, it allows you to run such lists of commands such that you pass in data and/or get data back out (i.e. the list of commands is like a function in most programming languages). Once you save a function, it becomes part of your toolbox (i.e. it now looks to you as if it were part of the basic toolbox that you started with).

For those with computer programming backgrounds: Note that MATLAB runs as an interpretive language (like the old BASIC). That is, it does not need to be compiled. It simply reads through each line of the function, executes it, and then goes on to the next line. (In practice, a form of compilation occurs when you first run a function, so that it can run faster the next time you run it.

MATLAB Windows:

MATLAB works with through three basic windows

Command Window : This is the main window .it is characterized by MATLAB command prompt >> when you launch the application program MATLAB puts you in this window all commands including those for user-written programs ,are typed in this window at the MATLAB prompt

Graphics window: the output of all graphics commands typed in the command window are flushed to the graphics or figure window, a separate gray window with white background color the user can create as many windows as the system memory will allow

Edit window: This is where you write edit, create and save your own programs in filescalled M files.

INPUT-OUTPUT:

MATLAB supports interactive computation taking the input from the screen and flushing, the output to the screen. In addition it can read input files and write output files

Data Type: the fundamental data –type in MATLAB is the array. It encompasses several distinct data objects- integers, real numbers, matrices, character strings, structures and cells. There is no need to declare variables as real or complex, MATLAB automatically

sets the variable to be real.

Dimensioning: Dimensioning is automatic in MATLAB. No dimension statements are required for vectors or arrays .we can find the dimensions of an existing matrix or a vector with the size and length commands.

BASIC INSTRUCTIONS IN MAT LAB

1. **T = 0: 1:10**

This instruction indicates a vector T which as initial value 0 and final value 10 with an increment of 1

Therefore T = [0 1 2 3 4 5 6 7 8 9 10]

2. **F= 20: 1: 100**

Therefore F = [20 21 22 23 24 …… 100]

3. **T= 0:1/pi: 1**

Therefore T= [0, 0.3183, 0.6366,

0.9549] 4. **zeros (1, 3)**

The above instruction creates a vector of one row and three columns whose values are zero
Output= [0 0 0]

5. **ones (5,2)**

The above instruction creates a vector of five rows and two columns Output=11

11

11

11

11

6. **zeros (2, 4)**

Output= 0 0 00

0 0 00

7. a = [1 2 3] , b = [4 5 6]

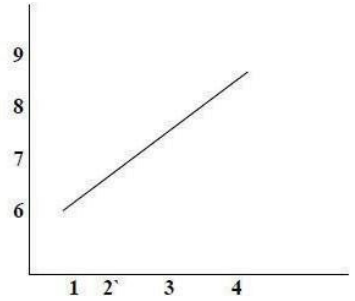
THEN **a.*b** = [4 10 18]

i.e. [4*1 5*2 6*3]

8. if C= [2 2 2] and b = [4 5 6]
   b.*C results in [8 10 12]

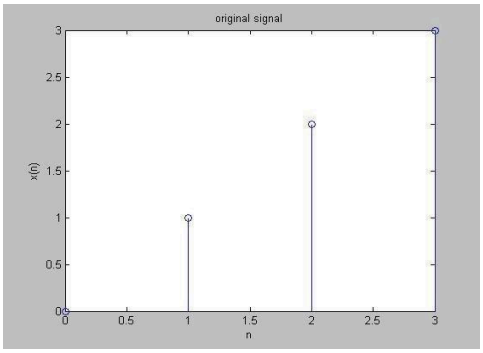
9. **plot (t,x)**

If x = [6 7 8 9] and t = [1 2 3 4]; This instruction will display a figure window which indicates the plot of x versus t

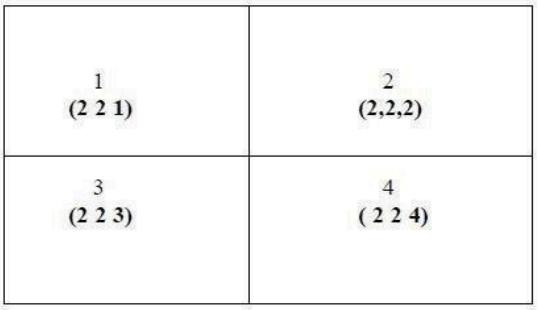


10. **stem(t,x)**

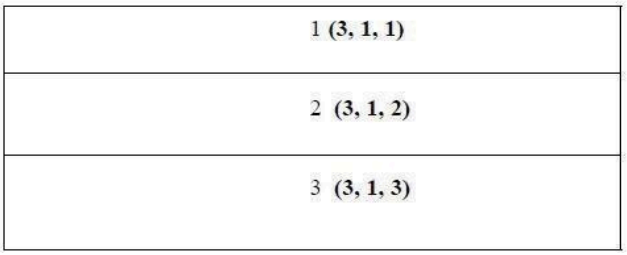
If x(n) = [0 1 2 3];This instruction will display a figure window as shown



11. **Subplot**: This function divides the figure window into rows and columns Subplot (2 2 1) divides the figure window into 2 rows and 2 columns 1 represent number of thefigure



Subplot(3, 1, 3)



12. Filter:**Syntax:** y =filter(b,a,X)

Description:y = filter(b,a,X) filters the data in vector X with the filter described by numerator coefficient vector b and denominator coefficient vector a. If a(1) is not equal to 1, filter normalizes the filter coefficients by a(1). If a(1) equals 0, filter returns an error.

13. Fliplr: **Syntax: B =**fliplr(A)

Description:B = fliplr(A) returns A with columns flipped in the left-right direction, that is, about a vertical axis.If A is a row vector, then fliplr(A) returns a vector of the same length with the order of its elements reversed. If A is a column vector, then fliplr(A) simply returns A.

14. Conv: **Syntax:** w =conv(u,v)

Description:w = conv(u,v) convolves vectors u and v. Algebraically, convolution is the same operation as multiplying the polynomials whose coefficients are the elements of u and v.

15. Impz:**Syntax:** [h,t] =impz(b,a,n)

Description:[h,t] = impz(b,a,n) computes the impulse response of the filter with numerator coefficients b and denominator coefficients a and computes n samples of

the impulse response when n is an integer (t = [0:n-1]'). If n is a vector of integers,impz computes the impulse response at those integer locations, starting the response computation from 0 (and t = n or t = [0 n]).If, instead of n, you include the empty vector for the second argument, the number of samples is computed automatically by default.

16. Disp:**Syntax:**disp(X)

    Description:disp(X) displays an array, without printing the array name. If X contains a

text string, the string is displayed.Another way to display an array on the screen is to type its name, but this prints a leading "X=," which is not always desirable.Note that disp does not display empty arrays.

17. xlabel:**Syntax:**xlabel('string')

    Description:xlabel('string') labels the x-axis of the current axes.

18. **ylabel:Syntax:**ylabel('string')

    Description:ylabel('string') labels the y-axis of the current axes.

19. **Title:Syntax:**title('string')

    Description:title('string') outputs the string at the top and in the center of the currentaxes.

20. **grid on: Syntax:**gridon

    Description:grid on adds major grid lines to the current axes.

In this laboratory, we will practice some fundamentals in MATLAB. The following MATLAB commands will be used. You can use "help topic" to get information for the usage of a specific command.

```
diag      - Generate diagonal matrix.
poly      - Polynomial.
polyval   - Evaluate polynomial.
roots     - Find polynomial roots.
exp       - Exponential.
det       - Determinant.
inv       - Matrix inverse.
rank      - Matrix rank.
eig       - Eigenvalues and eigenvectors.
expm      - Matrix exponential.
plot      - Linear plot.
title     - Graph title.
xlabel    - X-axis label.
ylabel    - Y-axis label.
print     - Print graph or save graph to file
```

You may also need to use the following operators and/or symbols:

Char	Name	HELP topic
+	Plus	arith
-	Minus	arith
*	Matrix multiplication	arith
.*	Array multiplication	arith
^	Matrix power	arith
.^	Array power	arith
\	Backslash or left division	slash
/	Slash or right division	slash
./	Array division	slash
:	Colon	colon
( )	Parentheses	paren
.	Decimal point	punct
;	Semicolon	punct
%	Comment	punct
'	Transpose and quote	punct
=	Assignment	punct

or you can try

```
help matlab/ops
```

to get more infomation. To exit from MATLAB, type **quit** or **exit** at the MATLAB prompt, followed by the ``return''key.

1.1 Fundamental Expressions

Working in the MATLAB environment is straightforward since most commands are entered as you would write them mathematically. For example, entering

```
a=4/3
```

yields the MATLAB response

```
a=

    1.3333
```

and assigns to variable **a** the value of 4 divided by **3**.

If you do not care to create a new variable but want to know the value of an expression, you can type the expression by itself, e.g.,

```
4/3
```

which yields

```
ans=

   1.3333
```

where **ans** is a dummy variable that stands for "answer."

If you prefer to create a new variable but do not wish to see the MATLAB response, type a semicolon at the end of the expression. For example,
b=4+7;
will create a new variable **b** whose value is 11, but MATLAB will not display the value of **b**. However, you can check the value of a variable at any time by entering the variable name at the prompt:
b
which yields
b=

11
Since **a** and **b** have been defined we can do the following:
c=a*(b-1)
which yields
c=

13.3333
If you are typing in an expression that does not fit on one line, use an ellipsis (three or more periods) at the end of the line and continue typing on the next line, e.g.,
p=1+2+...
3+4+6;
Arithmetic operators are the same as those commonly used except that * represents multiplication \ performs left devision, and ^ is the power operator. For the order in which MATLAB performs operations, the power operator ^ has precedence over multiplication * and division / and \, which have precedence over addition + and subtraction -. Precedence of like operators proceeds from left to right, but parentheses can be used affect the order of operation. Try
1+2^3/4*2

1+2^3/(4*2)

(1+2)^3/(4*2)
You should get the results of **5**, **2**, and **3.3750**.
MATLAB has several predefined variables such as
i and **j** stand for square root of **-1** which can be used to represent the complex numbers;

pi stands for pai;

Inf stands for infinity;

MaN stands for not a number (e.g. 0/0).

Try
c=4/0

d=Inf/Inf

y=sqrt(1+4*i)

z=exp(-1+3*j)

You should get the results of **Inf**, **NaN**, **1.6005+1.2496i**, and **-0.3642 + 0.0519i**.

You have used functions **sqrt** and **exp** which stand for "square root" and " exponential. There are other MATLAB functions that are very useful. You can use **help** followed by a command name to get the information about the usage of the command.

So far, you have learned how to define your variables. If you want to save them for your future use, you can use **save** command and latter on use **load** command to re-load them. Use **help** to get information about how to use these commands.

You can create a script file (an m-file) called "lab1.m" in your editor and include all the commands you want in that file. Later on you can run the m-file by typing the name of the m-file under the MATLAB prompt, e.g.,

>> lab1

1.2 Matrices, Vectors, and Polynomials

Vectors are entered into MATLAB by enclosing the vector elements within a pair of brackets. Vectors may either be _row_ or _column_ vectors. For example, a row vector V1 can be defined as,

```
>>V1 = [1 2 3]

V1=
   1  2  3
```

And a column vector V2 can be defined as,

```
>>V2 = [1; 2; 3]

V2=
   1
   2
   3
```

As you can see, row elements are separated by spaces (or commas), and column elements are separated by semicolons. A column vector may be transformed into a row vector, and visa-versa, through the transpose operation, defined in MATLAB by placing a single quote (') after the vector definition. For example, the transpose of our column vector V2 is,

```
>>V2'

ans=

   1  2  3
```

A **Matrix** is a series of vectors of like-dimension appended together into a two-dimensional array. Matrices are entered into MATLAB by listing the elements of the matrix and enclosing them within a pair of brackets. Elements of a single row are separated by commas or blanks, and rows are separated by semicolons or carriage returns. For example,

```
>> A=[1 2; 3 4]
```

yields the MATLAB response

A=

```
   1  2

   3  4
```
>>A=[1,2

3,4]
would produce the same result.

To find the dimensions of a matrix use the **size** command, e.g.,
>>size(A)

```
ans=

   2  2
```

SUBMATRICES

Individual matrix elements can be referenced using indices enclosed within parentheses. For example, to change the second element in the second row of matrix **A** to 5, type
>>A(2,2)=5

```
A =

   1  2

   3  5
```
If you add an element to a matrix beyond the existing size of the matrix then MATLAB automatically inserts zeros as needed to maintain a rectangular matrix:
>>A(3,3)=6

```
A =

   1  2  0

   3  5  0

   0  0  6
```
Since a vector is simply a 1 x n or an n x 1 matrix, where n is any positive integer, you can generate vectors in the same way as matrices:
>> v= [sin(pi/3) -7^3 a+1]

```
v=

   0.8660   -343.0000    2.3333
```
Alternatively, special vectors can be created using the **:** operator. The command **k=1:10** generates a row vector with elements from 1 to 10 with increment 1. Any other increment can be applied with a second **:** as follows:
>> knew = 1:0.25:2

knew =

1.0000 1.2500 1.5000 1.7500 2.0000

To add a row onto matrix **A** we type

```
>> A = [A;[7 8 9]]

A =

    1    2    0

    3    5    0

    0    0    6

    7    8    9
```

To extract the submatrix of **A** that consists of the first through third columns of the second through fouth rows use vectors as indices as follows:

```
>> B=A(2:4,1:3)

B =

    3    5    0

    0    0    6

    7    8    9
```

SPECIAL MATRICES

```
>> D=diag([1 2 3])

D =
    1    0    0
    0    2    0
    0    0    3
```

You can create a dummy matrix with all zero elements using the **zeros** command. This is very useful when you need to create a null matrix which you will fill in element-for-element later.

```
Z = zeros(2,3)

Z =
    0    0    0
    0    0    0
```

POLYNOMIALS AS MATRICES

Polynomials are described in MATLAB by row vectors with elements that are equal to the polynomial coefficients in order of decreasing powers. For example, to enter the polynomial p=s^2+5s +6 type

p=[1 5 6]

p =

    1   5   6

Zero coefficients must be included to avoid confusion, i.e., q=s^3+5s+6 is entered as
q=[1 0 5 6]

q =

    1   0   5   6

A polynomial can be evaluated using the **polyval** command. For example,
>> polyval(p,1)

ans =

   12

gives the value of the polynomial **p** at s=1. Using the **roots** command is a convenient way to find the roots of a polynomial, e.g.,
>> r= roots(p)

r =

   -3

   -2

1.3 Matrix Operations and Functions

MATLAB performs matrix arithmetic as easily as it does scalar arithmetic. To add two matrices simply type
>> B+D

ans =

    4   5   0

    0   2   6

    7   8   12

Similarly, to multiply two matrices do as you would for scalars,
>> B*D

ans =

    3   10   0
    0   0   18
    7   16   27

Dividing by matrices is also straightforward once you understand how MATLAB interprets the divide symbols (/ and \). Suppose you want to solve for x in the equation P x = Q. To express the solution x=P^{-1}Q in MATLAB use left division as x=P\Q. Now suppose you want to solve y P = Q for y. The solution to this problem is y=Q P^{-1} which you can write in MATLAB using right division as y=Q/P.
Be careful with the inner dimensions of the two matrices being multiplied or divided which should be the same. Otherwise, MATLAB will tell you there are some bugs.
The power operator ^ also operates on the matrix as whole as long as the matrix is square. Other functions that perform operations on matrices include **det(B), inv(B), rank(B), eig(B)**, and **expm(B)** which produce the determinant, inverse, rank, eigenvalues, and e^X respectively. Many of them require that the matrices be square.

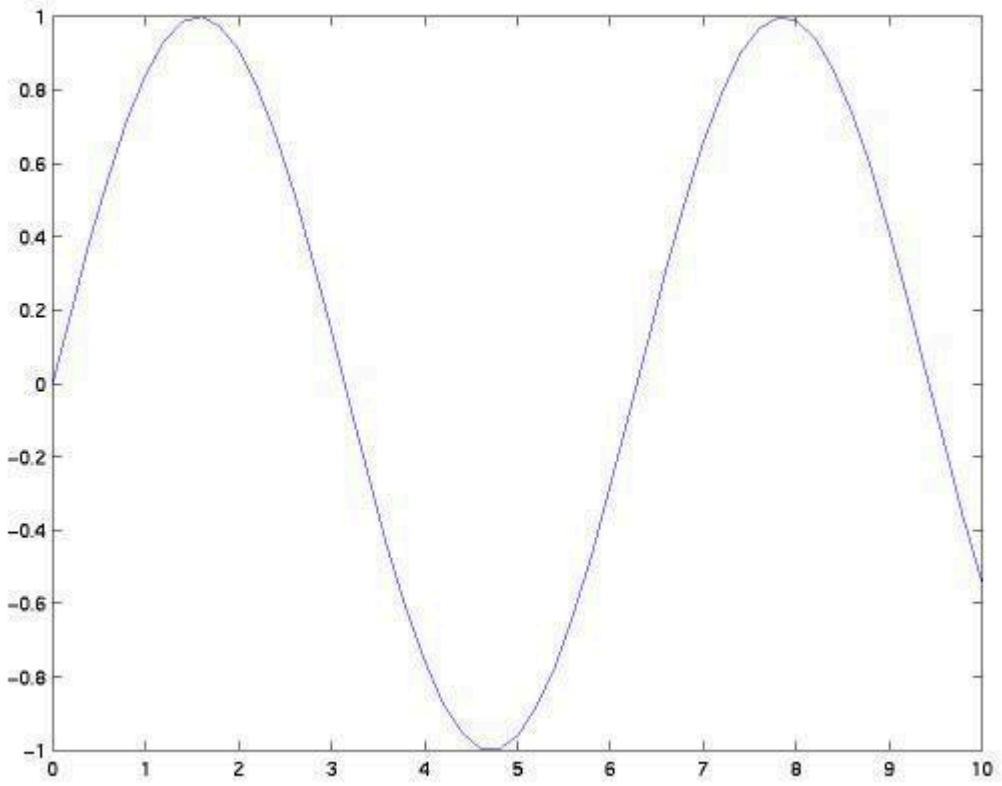
At times you may want to consider a matrix as simply an array of numbers and operate on the array element by element. Specifically, you will create arrays to represent tables of data that you want to manipulate. MATLAB provides different ways of using functions to operate on arrays instead of matrices. For example, suppose you have a table of data that you have entered as an array called **Data**. Now suppose you would like to perform a root-mean-square calculation and need to find the square of each element in **Data**. Using a **.** you can convert arithmetic matrix operation into element-by-element operations (addition and subtraction are the same in either case). Specifically, preceding the operator with the . indicates array operations. To square each element in the array type **Data.^2**. Similarly, to multiply two arrays **R** and **S** (of the same dimensions) element by element type **R.*S** as follows:

```
>> R=[4 5
    0 1];
>> S=[2 3
    4 6];
>> R.*S
ans =
   8   15
   0    6
```

1.4 Plotting

Let's see how to plot a vector using the "plot" command. First try

```
t=0:0.2:10;
y=sin(t);
plot(t,y)
```



This is a very basic plot; the first variable is on the horizontal axis and the second variable is on the vertical axis. You can change the style of the line by using a third option to "plot", try
plot(t,y,':')
Note that the colon is enclosed in ***single quotes*** and separated from the other arguments by a comma. You should get a dotted line. You can make it green (if you have a color monitor) by using
plot(t,y,'g:')
Other options that you might want to try are:
solid - red r
dashed -- green g
dotted : blue b
dashdot -. white w
or try
help plot
for more information.
Now let's put some useful labels on the plot so someone who looks at it knows what it is. Enter:
xlabel('Time in seconds');
ylabel('Sin(t)');
title('Sine Wave -- J. Doe, ENME362, LAB1');

Assignment

Create a MATLAB m-file which will solve each of the following problems. For each problem, you need to turn in the following:

(1) Printout of the m-file.
(2) Printout of the MATLAB results when the m-file is run.
Use the **diary** command to save the output to a file,
then print this file to turn in.
(3) Printouts of any plots requested.

1. Consider the following matrices and vectors:

$$A = \begin{bmatrix} -5 & 1 & 0 \\ 0 & -2 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad c = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}$$

(a) Suppose A x = b, find x.

(b) Suppose y A = c, find y.

(c) Let G(s) = c * inv(sI-A) * b, find G(0) and G(1).

(d) Define C_m = [b Ab A^2b], find the rank of C_m.

2. Consider the function

$$H(s) = \frac{n(s)}{d(s)} \quad \text{where} \quad n(s) = s^3 + 6.4\,s^2 + 11.29\,s + 6.76$$

$$d(s) = s^4 + 14\,s^3 + 46s^2 + 64s + 40$$

(a) Find n(-12), n(-10), d(-12), d(-10).

(b) Find H(-12), H(-10) .

(c) Find all the values of s for which H(s) =0.
(d) Plot H(s) for s=0 through s=20. Please label your axes appropriately.

EXPERIMENT NO. 1

Experiment 1

Aim: Determination of step & impulse response for a second‑order unity feedback system.

Apparatus: MATLAB software

Theory:
MATLAB tool is used for analyzing the time-domain response of systems, both linear and nonlinear.
Key Commands:

- tf() lsim()
  tfdata() step()
  roots() impulse()
  tf2zp() residue()
  zp2tf() poly()

A. Overview of the MatLab Control System Toolbox
MatLab offers a special toolbox for analyzing control systems called (surprise!) the Control System Toolbox. To see the various commands provided by this toolbox, type the command:

- help control

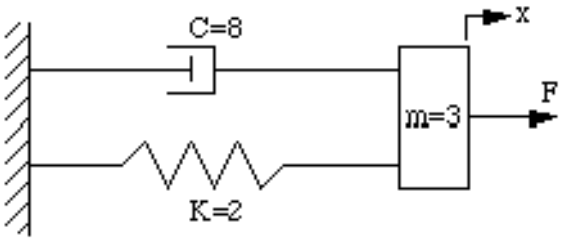
A simple demonstration of some of the features of this toolbox can be seen by running the command:

- ctrldemo

Run this command and you will get a number of plots.

B. Model Creation via tf()

Recall that transfer functions provide a convenient way of expressing the relationship between the input and output of a linear system when all initial conditions are zero. The roots of the numerator of a transfer function are called the zeros of the system, and the roots of the denominator are called the poles of the system.
Consider a simple mechanical spring-mass-damper system:



Assuming zero initial conditions, a free body diagram of this system results in the following differential equation of motion:

Taking the Laplace transform of this equation and solving for the transfer function X(s) / F(s) results in:

$$\frac{X(s)}{F(s)} = \frac{1}{3s^2+8s+2}$$

Let's define the numerator and denominator of the transfer function separately.
The numerator is simply a scalar. At the MatLab prompt, type:

- num = 1

The denominator is a little trickier, since it is a polynomial of the Laplace variable s. In MatLab, a polynomial is represented as a row vector containing the polynomial coefficients in descending order. For example, the polynomial 2s3 + s + 5 would be represented by the vector [2, 0, 1, 5]. For the current problem, the denominator polynomial is 3s2 + 8s + 2. To define the polynomial in MatLab, type:

- den = [3 8 2]

Now that we have defined the numerator and denominator of the transfer function, a SYS data object for the spring-damper system can be created, using the tf() function. To define the SYS data object for this system (let's call it ***sys***), type:

- sys = tf(num, den)

MatLab should respond by showing the transfer function of the system. We now have a variable ***sys*** which can be used to analyze the spring-mass-damper system. Unfortunately, not all MatLab functions are able to take a SYS object as an argument. For these functions, we will have to describe the system by its numerator and denominator polynomials instead.
Once you have a SYS object, the original numerator and denominator of the system can be recovered using the tfdata() function. Try typing:

- [num, den] = tfdata(sys,'v')

MatLab shouold respond by displaying the original numerator and denominator.

C. Poles and zeros

A transfer function can be described by its poles (roots of the denominator) and zeros (roots of the numerator). The roots() function can be used to find the roots of a polynomial. To find the zeros and poles of our example system, type:

- zeros = roots(num)
- poles = roots(den)

Alternately, the zeros and poles may be found simultaneously using the tf2zp() function, where ***tf2zp*** stands for "transfer function to zeros & poles". To do this, type:

- [zeros2, poles2] = tf2zp(num, den)

You should find that ***zeros*** and ***poles*** are the same as ***zeros2*** and ***poles2*** . In both cases, you should find there are no finite zeros, and the poles are located at s = -2.3874 and s = -0.2792.
Another useful MatLab function, called zp2tf() , can be used to define a transfer function from a set of poles and zeros. In this case, the arguments of the function are the poles and zeros placed in row vector form, and the result of the function is 2 vectors containing the numerator and denominator polynomials. For example, the poles of our example are located at s = -2.3874 and s = -0.2792, and there are no finite zeros. To find the transfer function given this information, type:

- [num2, den2] = zp2tf([], [-2.3874, -2.3874], 1/3)

Which should return the following:

- num2 = 0 0 0.3 333
- den2 = 1.0000 2.6 666 0.6666

The third argument of the zp2tf() function is the scalar gain of the transfer function. Why is this needed? Given a set of poles, there are an infinite number of polynomials which have those poles as their roots; if we find one such polynomial, then any scalar multiple of that polynomial will possess the same roots. When you give MatLab a set of zeros and poles, it chooses polynomials for the numerator and denominator which are scaled so that the highest order coefficient of each polynomial is equal to one. In order to produce the desired transfer function, you need to tell MatLab what the scalar gain should be based on this fact. For our example system, a scalar gain of 1/3 is used since, by observation of the original transfer

function, we would need to divide both the numerator and denominator through by 3 to produce numerator and denominator polynomials with the highest order coefficient of each polynomial equal to one.
A final way to go between a transfer function representation of a system and a pole/zero representation is to use the poly() function. This is the inverse of the roots() function, but now we provide the roots of a polynomial as arguments to the function (in vector form), and the result of the function is a polynomial which possesses those roots. Just as with zp2tf() , poly( ) returns a polynomial with a highest-order coefficient of one. Using our previously defined variables, type:

- num3 = poly(zeros)
- den3 = poly(poles)

These commands will produce the following output:

- num3 = 1
- den3 = 1.0000 2.6667 0.666

To produce the original transfer function, a scalar value of 1/3 must be multiplied on the numerator. This is analogous to including the scalar gain of 1/3 to the zp2tf() function.

D. Partial Fraction Expansion and Residues

MatLab can also be used to perform partial fraction expansion of transfer function expressions. This can be very handy for examining the response of individual components of a larger transfer function. First, verify that you still have the original numerator and denominator polynomials, ***num*** and ***den*** , properly defined. To perform a partial fraction expansion on this transfer function, type:

- [R, P, K] = residue(num, den)

R is a vector containing the residues, ***P*** is a vector containing the component poles, and ***K*** is sometimes called the ***direct term*** .
The residue() function breaks the transfer function defined by ***num*** and ***den*** into its ***n*** component residues, where ***n*** is the order of the denominator:

$$\frac{num(s)}{den(s)} = \frac{R(1)}{s - P(1)} + \frac{R(2)}{s - P(2)} + \ldots + \frac{R(n)}{s - P(n)} + K(s)$$

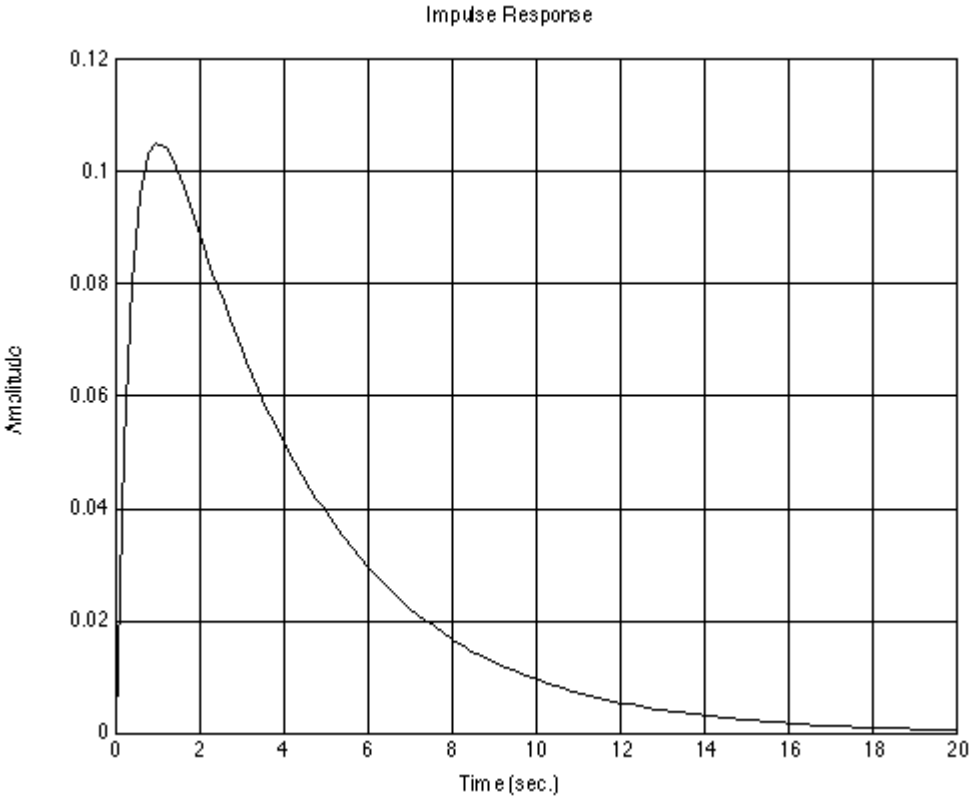
If the order of the denominator is larger than the order of the numerator, then K will be zero

E. Impulse response

Now that you know how to create SYS data objects and represent systems in terms of transfer functions and poles/zeros, we can look at the time response of linear systems. In this studio we'll consider the response of a linear system to two types of inputs: impulse and step functions.
The impulse response of an LTI system can be plotted using the impulse() function. To try this with our example system, type:

- impulse(sys)

The output should look like:



Note: you won't see the grid lines unless you execute the "grid" command after making the impulse plot

This plot shows the time response of the mass when a unit impulse force is applied to the mass at time t=0s. If you want to see the response for a longer period of time, the impulse() function will also take a time vector as an argument. Try this:

- impulse(sys, [0 : 0.1 : 40])

You should get the same basic plot as before, but with a 40s time axis.
Note that instead of using the SYS data object as input to the function, we could also have used the numerator and denominator polynomials, ***num*** and ***den*** , defined earlier. In general, it is better to stick with using SYS, since some MatLab functions will require this, and future versions of MatLab may require it for ***all*** control toolbox functions (get used to it now!).
In some cases, getting a pretty plot isn't enough. Instead, you may want the actual numbers for the mass position as well as the associated time vector. To get this information, type:

- [x, t] = impulse(sys);
- plot(t,x)

Make sure you type the semicolon at the end of the first line, or else the entire output and time vectors will be displayed on the screen. The values describing the position of the mass are now held in ***x*** , and the corresponding values of time are held in ***t*** .

F. Step response

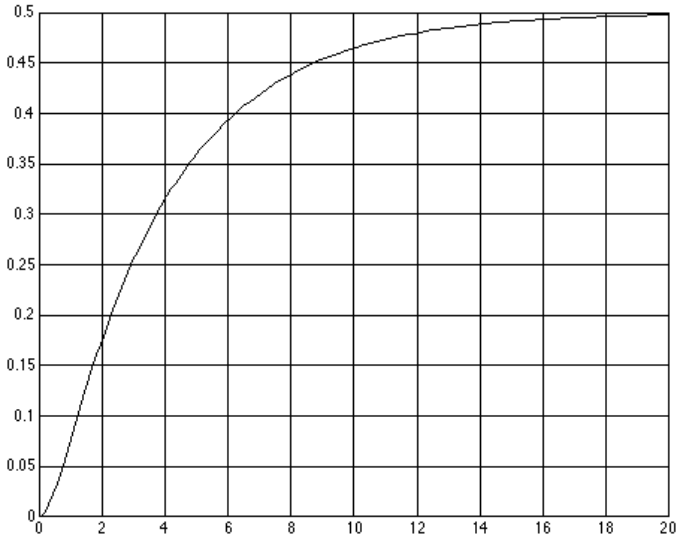
Step response can be found using MatLab's step() function, which operates the same way as impulse(). In this case, the forcing function is a unit step applied at time t=0s. To plot the step response of our system, type:

- step(sys)

or

- [x, t] = step(sys)
- plot(t,x)

After typing the plot command, you should see the following:



The x-axis corresponds to the time vector returned by [x, t] = step(sys), and the y-axis shows the response vector, x(t).
If a longer time period is desired in the plot, say t = 0 to 50 sec, you can create the required time vector as:

t = [0 : 0.2: 50];

and then pass this vector to the step() function:

x = step(sys, t)
or
x = step(num, den, t)

G. General linear simulation

A more general technique for simulating the response of a linear system to an arbitrary input signal is the lsim() function. To see how this function works, say we want to simulate the response of our system to a function F(t) = 1 for t<10s, F(t) = 0 for t>10s. First, generate a time vector containing the time points of interest, say for 0 < t < 20s in 0.1s increments:

- t = [0:.1:20];

Next, create an input vector F where the ith element of the vector contains the value of F(ti), and ti is the value contained in the ith element of the time vector:

- F = zeros(size(t));
- F(1:(length(t)+1)/2) = ones((size(t)+1)/2);

If you don't fully understand the syntax of setting up the forcing function, don't worry about it too much. With more experience using MatLab this sort of expression will become second nature to you. To plot the response of the system to this input, type:

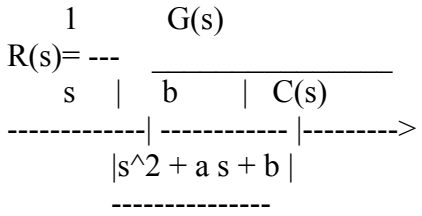
- x = lsim(sys, F, t);

As you can see, lsim() takes a ***SYS*** , input ***F*** , and time ***t*** as arguments, and returns the output ***x*** . This is a very powerful function which allows you to simulate any arbitrary input function you might want to investigate.

Time response: 2nd order systems

In this lab, we will study time responses of control systems. The time response of a control system is usually divided into two parts: the **transient response** and the **steady-state response**. We will study these responses for the second order systems. For simplicity, we will mostly use "step input."

Let us look at the following second order (open-loop) system whose transfer function is:

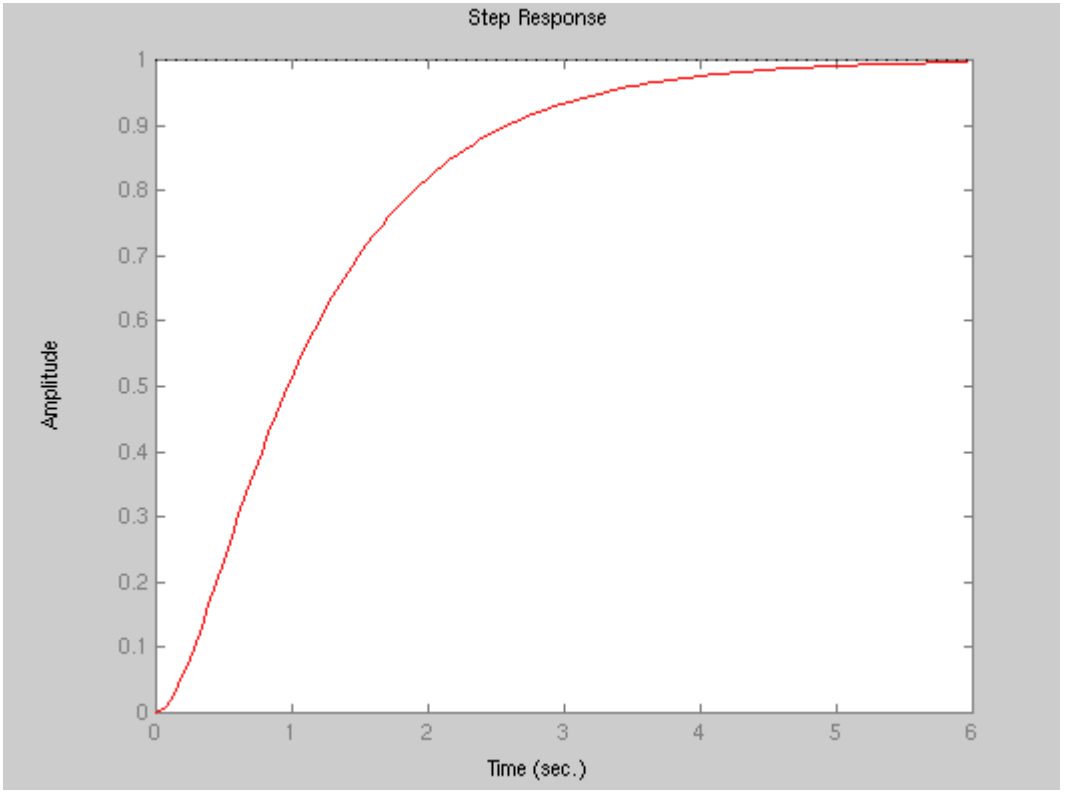


Four typical cases are as follows:

(1) Overdamped

$$G(s) = \frac{4}{s^2 + 5s + 4} \qquad (1)$$

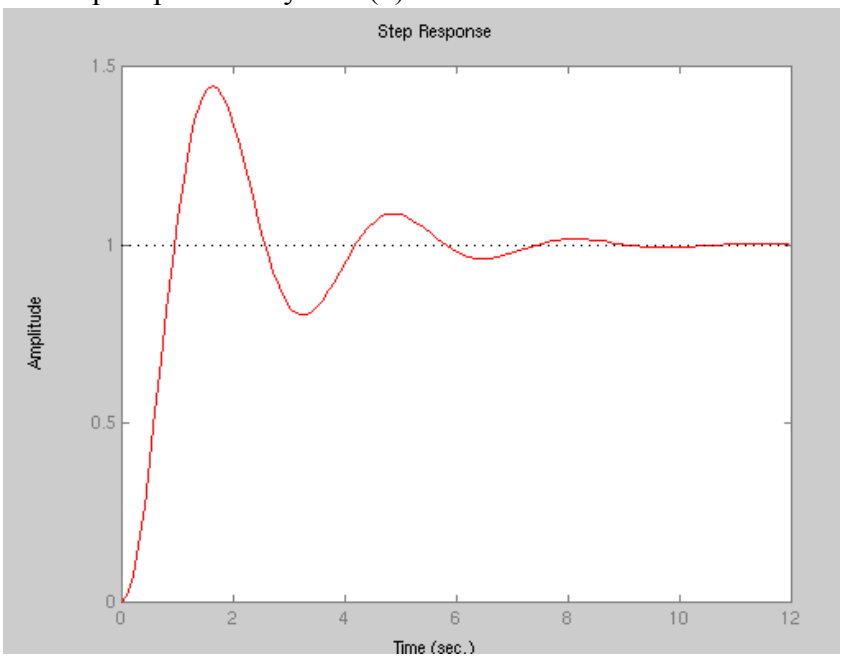
The step response of system (1) is



(2) Underdamped

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

The step response of system (2) is

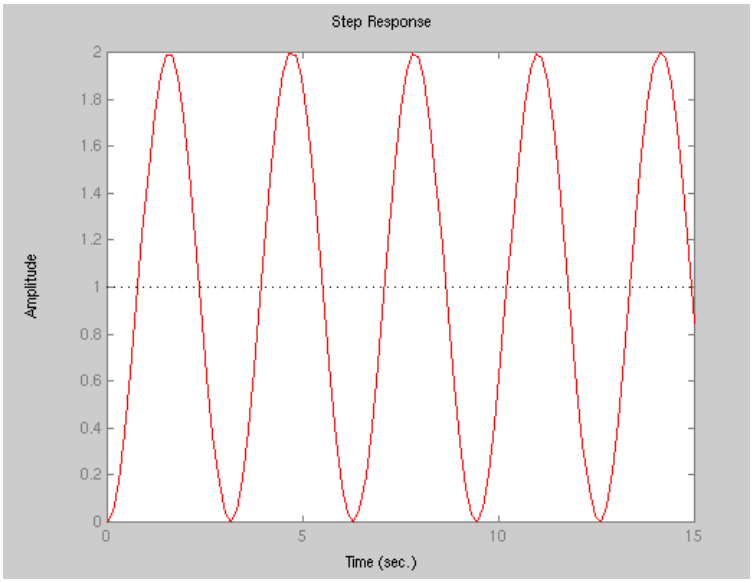


(3) Undamped

4

$$G(s) = \frac{}{s^2 + 4} \qquad (3)$$

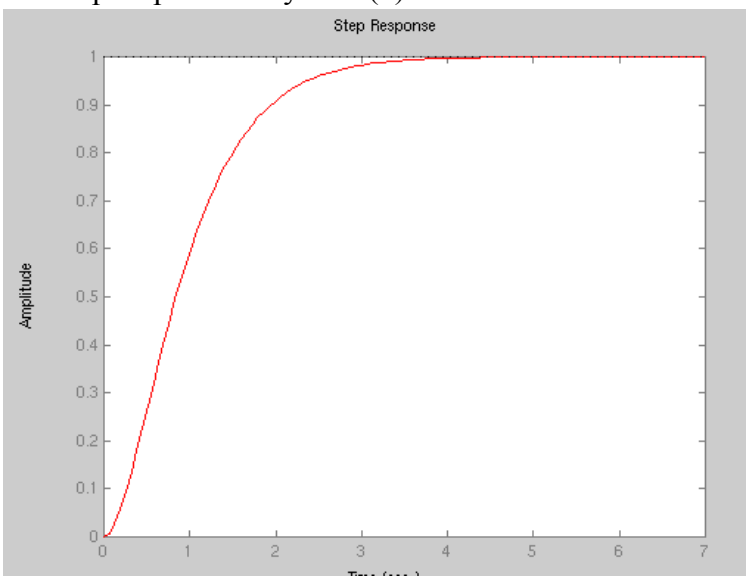
The step response of system (3) is



(4) Critically damped

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

The step response of system (4) is



EXPERIMENT NO. 2

Aim: Comparison of open loop & closed loop control in speed control of D.C. motor & to find the transfer function.

Apparatus: - D.C servomotor set up, multimeter, connecting wires.

Theory:

Controlled Variable and Manipulated Variable:

The controlled variable is the quantity or condition that is measured and controlled. The manipulated variable is the quantity or condition that is varied by the controller so as to affect the values of controlled variable. Normally the control variable is the output of the system. Control means measuring the values of controlled variable of the system and applying the manipulated variable to the system to correct or limit deviation of the measured value from a desired value.

In study control engineering, we need to define additional terms that are necessary to describe control systems. The Block diagram of open loop control system is shown in figure -1 for an open loop control system would not have feedback path i.e feedback path is open.

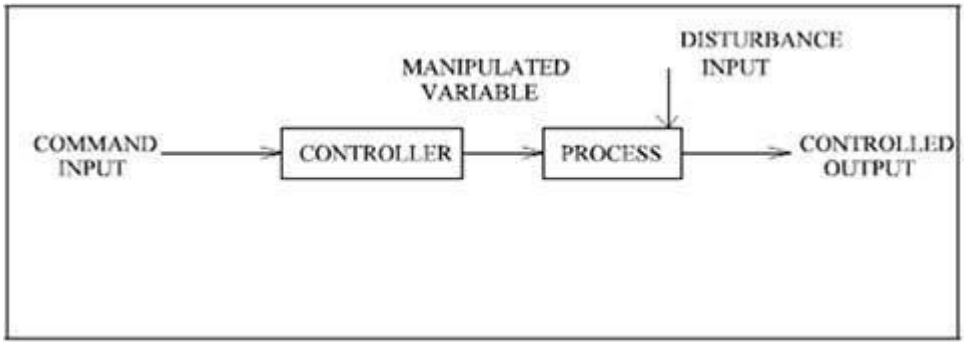


Figure-05 **open loop control system**

Plant:

A plan may be a piece of equipment, perhaps that a setup machine parts functioning together, the purpose of which is to perform a particular operation. In this manual we shall call any physical object to be controlled a plant.

Process:

The Merrian-Webster Dictionary defines a process to be a natural progressively continuing operation or development marked by a series of gradual changes that succeed one another in

a relatively fixed way a lead towards a particular results or end. Or an artificial or progressively continuing operation that consists of a series of controlled option or movements systematically directed toward a particular result or end. Examples are chemical, economic and biological processes.

Systems:

A system is a combination of components that act together and perform a certain objective. A system is not limited to physical ones. The concept of the system can be applied to abstract, dynamic phenomena such as those encountered in economics. The word system should interrupt to imply physical, biological, economic and the like systems.

Disturbance:

A Disturbance is a signal that tends to adversely affect the value of the output of a system, It is disturbance is generated within the system, it is called internal, while an external disturbance is generated outside the system and is an input.

Feedback Control:

Feedback control refers to an operation that, in the presence of disturbances, tends to reduce the difference between the output of a system and some reference input and that does .so on the basis of this difference. The block diagram of closed loop control system is shown in figure.

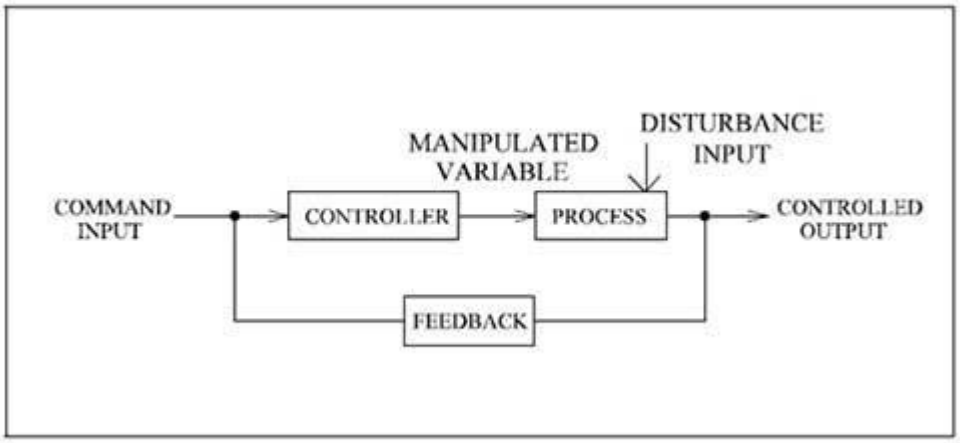


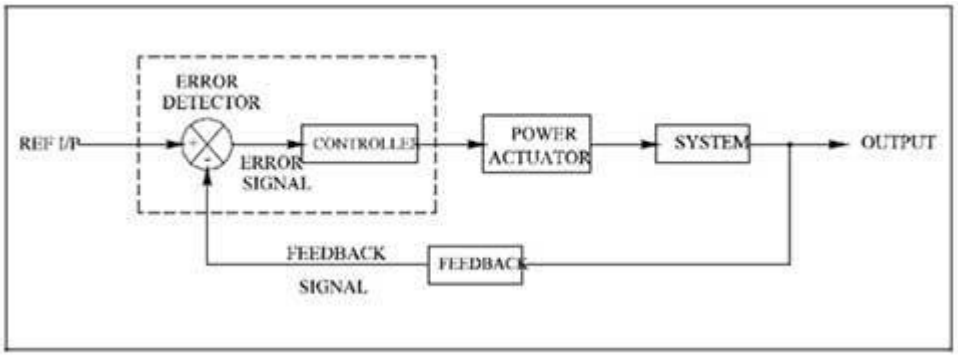
Figure-06 **closed loop control system**

Control System:

Control systems Control a certain physical quantity so that it changes in prescribed ways according to arbitrary input changes.

AUTOMATIC CONTROL SYSTEM:

The basic components of an automatic control system are error detector, amplifier and controller, power actuator, system and sensor (or) feedback system. The block diagram of an automatic control system is shown in figure.



ERROR AMPLIFIER:

Which compares the reference signal Vr with feed back signal? The output is a voltage proportional to the difference between the two signals.

CONTROLLER:

The controller process the error signal and gives an output voltage signal Vc known as the control voltage. This suggests the necessary corrective measures required in the actuating signal Va to be applied to the system.

POWER ACTUATOR:

Which takes the input as the control voltage Vc from the controller and produces the necessary actuating input signal to be applied to the system to achieve the desired output.

FEED BACK

The constitutes the output sensor and associated amplifier. The feedback signal Vf is the voltage proportional to the output variable of the system.

A D.C motor can be controlled by varying either the field current o the armature current. The types of D.C servomotors are series motors the shunt motors and the permanent magnet (PM) motor. These motors offer higher efficiency than an A.C motor of the same size, but radio frequency interference is a problem in some applications.

In its general form of construction the separately excited motor is similar to the series motor, but armature and field circuits are separated, control being applied either to armature or field. Field control may utilize connection of the amplifier single ended or more frequently pushpull, requiring the centre tapped arrangement. An advantage is gained over the series connection in avoiding the larger currents required to drive the armature. The field power is only a small fraction of the armature power, which implies a power gain in the system. The field control machine requires a constant current supply to its armature, often approximated by a constant voltage connected through a high external series resistance.

This device, which is relatively simple and inexpensive, leads to some waste of power in resistance drop and possibly, problems of heat dissipation, but in small machines this is usually unimportant. The closeness of the approximation depends on motor running at low enough speeds for the armature back emf to be small compared with the supply voltage. Armature control, though it requires higher current handling and current reversal, provides a nearly ideal linear performance. This linearity is achieved through excitation of the field under fixed condition, so that magnet type characteristics is not imposed on the machine. More over as the armature circuit posses a similar time constant than the field, a faster

response can be obtained. It is noteworthy that the constant field connection gives machine performance equivalent to the permanent magnet motor.

Block Diagram Of DC Servomotor

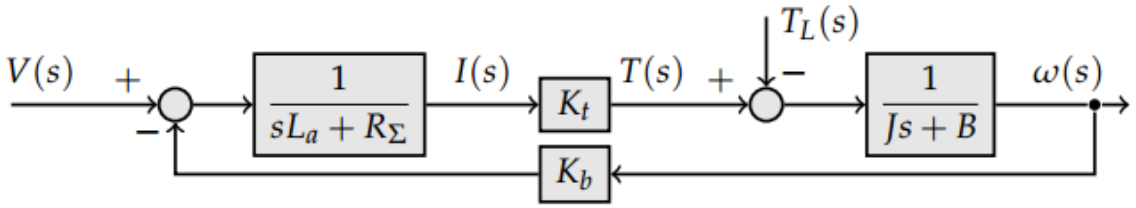


Figure 2.2: Block diagram of a PMDC motor. RΣ includes the armature resistance Ra

Procedure:-

Armature Control:

1. Adjust T1 at setting with the help of knob K.
2. Ensure the POT P1 (speed control) is in the maximum anti clock wise position. Switch on supply.
3. Connect a digital or analog Multimeter across the terminals marked armature to measure armature voltage in the range volts.
4. Adjust P1 and P2 so that Va= and Vf= volts.
5. Note down T1, T2 and speed and enter the results in the table 1.
6. Adjust T1 up to in the steps of to get a set of readings.
7. Now for Va= volts etc repeat step no 6.
8. From table no 1, plot the speed torque characteristics.

FIELD CONTROL METHOD

1. You may repeat above steps for various values of field voltages by controlling POT P2 and keeping Va at volts.

Table 1: For Armature Control Radius of pulley=R= 3.53cm Field voltage Vf =16v Armature voltage Va=13v

Table No. 1

S. NO.	VT	Vr	N Speed	Error ($V_r - V_T$)

Table or Field Control Field voltage Vf =16v Armature voltage Va=15v

Precautions:
1. The speed control knob should be always in the most anticlockwise position before switching on the equipment.
2. In order to increase the armature voltage, rotate the knob in the clockwise direction in a gentle fashion.
3. In order to increase the load on the servomotor adjust the knob.

Result: Torque-speed characteristics of D.C Servomotor at different Armature voltages and Field voltages are plotted.

Experiment 2(A)

Aim: To study the characteristics of positional error detector by angular displacement of two servo potentiometers excited with ac

Apparatus: A.C position control system unit.

Theory:
A pair of precision servo potentiometers is working as an error detector. The potentiometer marked as INPUT POTENTIOMETER translates information regarding the desired angular position into a proportional A.C voltage. The potentiometer marked OUTPUT POTENTIOMETER converts the information regarding the present position of mechanical load into a proportional A.C voltage. Note that this pair of input And output potentiometers is excited by a 6 volts 50 Hz supply. {V {t}}.
Any difference of potential between the wiper contacts of servo potentiometer is amplified by means of A.C power amplifier. The power amplifier output activates the control winding of the A.C servomotor {2 phase}.The other is activated {reference winding} by means of fixed

AC voltage. The load to be positioned is coupled to the output shaft of the output servomotor and gear train combination. The same shaft is also coupled to the output potentiometer.

This position control system works to make the output shaft position identical to the input shaft position. When the output shaft is being positioned, the mechanical load is also moved to a new desired position thus making error voltage always zero. The system works to make the error voltage zero after disturbance.

AC Servo motor position control

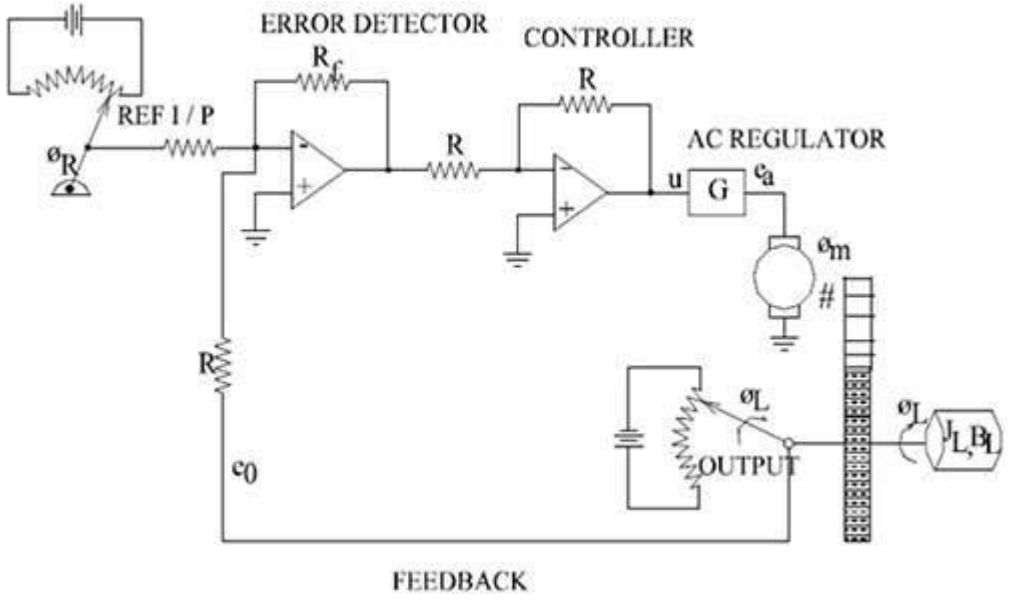
For the study of automatic control system, AC motor is used as a system to be controlled. Here the controlled variables are the position of the motor.

The main advantage of the AC servomotor used as system is

1. Control of AC servomotor is so much easier than induction motor, because of were control only control phase winding voltage like 12V or 24V not main winding voltage 230V AC

2. Direction of motor reversal is also obtained by interchanging the control phase winding voltage.

Block diagram of A.C position Control System:



ERROR DETECTORS:

In Position Control Systems the reference input will be an input signal proportional to desired output. The feedback signal is a signal proportional to current output of the system. The error detector compares the reference input and feedback signal and if there is a difference it produces an error signal. The error signal is used to correct the output if there is a deviation from the desired value.

CONTROLLERS:

A controller is a device introduced in the system to modify the error signal and to produce a control signal. The manner in which the controller produces the control signal is called control action. The controller modifies the transient response of the system. The controller

may be electrical, electronics, pneumatics and hydraulics depending upon the system. We are discussed only electronic controller.

Procedure:-

1. Switch ON the AC main supply, Switches SW1, SW2 and SW3 should be in the ON position.
2. Observe that the input and output potentiometers come in alignment.
3. Keep the gain pot in almost in maximum gain position { Almost fully clockwise position}
4. Take the input potentiometer to the starting position i.e. very near to the zero position. The output potentiometer will also follow the input potentiometer till the null indicator indicates null position.
5. Enter the observations in the tabular column given below. You may repeat the observations with lower amplifier gain, to observe that there is greater error, with higher gain {i.e. maximum gain} you may note that the output pot indicates sustained oscillations.
6. You may use the test points black, TP1, TP2, TP3 a.TP1- Black- Excitation voltage for potentiometer system.b.TP2-black –Variable point potential for I/P pot.c.TP3- Blackvariable point potential for O/P pot. Note that EXCITATION point {TP1-black} for O/P pot is floating with respect to the main ground.
7. Green terminal on the left hand side indicates main ground of the system. You may connect C.R.O across the TP4 and ground {green terminal}, TP5 and ground to observe the A.C preamplifier output and servo amplifier output respectively.
8. Please note that the switches SW1 SW2 and SW3 are in series with A.C preamplifier, servo amplifier output and at the input of the A.C servomotor reference and control winding

Table No. 2.1(A)
Observation:

Sl.No	I/P Angular position	O/P Angular position	Remarks

Precautions:

1. If the Red LED is not glowing, check for the front panel fuse {D.C fuse}.If it blows again; do not switch ON the unit.
2. The Switch SW3 is connected in series with A.C servomotor windings. The same may put in OFF condition when the unit is not being used.
3. Do not try to rotate the o/p by means of the knob by hand

Result- Ac position control is studied for various gains and it is observed that error is high for low values of gain and error is less fo high values of gain.

Discussion of Result: This position Control system works to make the output shaft position identical to the input shaft position. When the output shaft is being positioned ,the mechanical load is also moved to a new desired position thus making error voltage always zero.

Experiment 2(B)

Aim: To study the characteristics of positional error detector by angular displacement of two servo potentiometers excited with dc

Apparatus: D.C position control system unit.

Theory:

The D.C Position control system is so called because the D.C signals exist in the system. For example if the reference input and the controlled output are constant values a straight line can graphically represent the actuating signal. The signals in the other part of the system can be represented in the similar manner. For D.C voltage controlled system, the actuating signal e[t] is a D.C voltage. In the simplest form the output position and the reference position o1and 02 are measured and compared by a potentiometer pair whose output voltage is proportional to error in the angular position. The error voltage is amplified and applied to servomotor whose positions the load and the output potentiometer such that the error is reduced to zero.

NEED FOR STABILIZATION

With switch SW1in open position and the step change in the input shaft, the output shaft exhibits an oscillating behaviour. This happens because of the system elements, which are capable of storing energy i.e. capacitance, inductance inertia of moving components like rotor, load, gear train etc. Once the system is excited by change in the input signal, the various elements begin to store energy, even if the error voltage falls to zero. The stored energy causes the output shaft to move in the same direction. This creates an error of opposite polarity and the system is again instructed to work in the opposite direction. In this way, energy storing elements tend to produce overshoots and undershoot in the system. In the experimental setup, output Derivative Feedback is used for stabilization of the output. The Tachogenerator, which is coupled to the motor, generates an output voltage, which is proportional to the rate of change of displacement. This voltage is coupled to the input of error amplifier, either in the regenerative mode or degenerative mode. By the adjustment of

potentiometer P4ampunt of derivative feedback can be adjusted, while DPDT switch is meant for selection of mode of stabilizing feedback.

HOW DERIVATIVE FEEDBACK WORKS

When switch SW1 is closed [i.e. in TACHO IN position] and the degenerative feedbacks are suitably adjusted, we observe that the output shaft follows the input shaft in a smooth fashion without any unwanted oscillations.

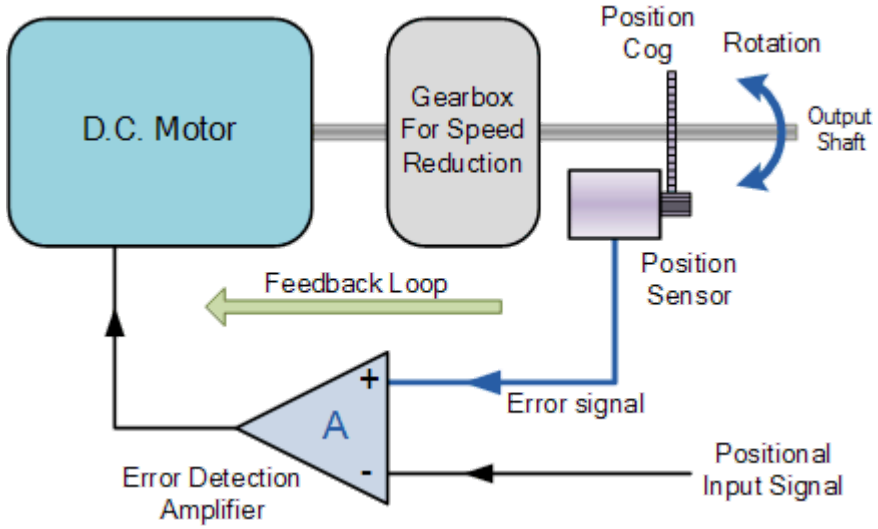
If the mode of feedback is regenerative, then the output never reaches the stable state. Instead it keeps on oscillating around the desired position. In short, for degenerative feedback, the damping factor of the system is decreased, thus resulting in breaking action on the moving components, prior to their final desired position. For a greater amount of feedback voltage, damping becomes excessive and the system exhibits a very sluggish response, settles to a final position. The tendency for oscillations is found to be dependent on the amplifier gain setting.

OPERATION WITH STABLIZING FEEDBACK

1. Now put the SW switch in lower position i.e. tacho in position, SW2 must be in downward position i,e degenerative mode. Keep P4 in fully anticlockwise direction.
2. The system can be tested for operation as follows
3. Now take the pot P1to 180 position& effect the step input change in one of the directions, output again indicates oscillations. Now advance the pot in P4 in clockwise direction and the output now is observed to follow the input in a smooth fashion without oscillation. If the P4 pot is too much advanced, the output now follow input in a sluggish fashion indicating over –damped system. Now take the pot P1 to 180 positions.
4. Now put the switch SW2 in upward position i.e. regenerative mode. Now if the pot P1 is disturbed the output pot P2 is found to oscillate continuously around the desired position. As the amount of feedback is adjusted the frequency and the amplitude of output is observed to vary.

DO NOT OPERATE THE D.C POSITION CONTROL IN THE REGENERATIVE MODE FOR A LONG TIME.

Block Diagram Of DC Position Control System:



Procedure: 1. Before switching on the main panel, see that the switches SW3, SW4 {on the LHS panel} are in the downward position i.e. ON position.
2. Ensure that SW1&SW2are in the off position i.e upward position.
3. Keep the input position P1 in 10 positions
4. Potentiometer p3 [amplifier gain adjustment] should be in mid position.
5. Now switch on the main unit LED 'Rand LED 'G" should glow. Operation without feedback [SW1in off position i.e. Tacho Out}
6. Now slowly advance the i/p potentiometer P1 in clockwise direction. The o/p potentiometer along with load will be seen to be following the change in the i/p potentiometer.
7. When the i/p is disturbed the null indicator will be showing some indication but when the o/p reaches desired positions, again the null indicator indicates almost zero. It may be noted that when i/p POT is moved in anti-clockwise direction, the o/p POT also moves in the reverse direction.

Step change in input

8. Now change the i/p POT in a step fashion (in fact approximating step input). The output will be observed to change in oscillatory mode.

Observations: Plot the output angle v/s input angle for both the system i.e. without and with stabilizing feedback.

Table No. 2.1(B)

Sl. No.	Input	Output	With stabilizing feedback Regenerative mode

Table No. 2.2(B)

Sl. No.	Input	Output	With stabilizing feedback Degenerative mode

Precautions:
1. Please do not cross zero degree position by moving POT P1 i.e. do not operate between 350 deg and zero deg.
2. Do not try to rotate output POT by hand .this may damage the potentiometer

Result: The output angle v/s input angle characteristics for both the system i.e. without and with stabilizing feedback are to be observed.

Discussion of the Result:
1. Student should be able to understand the effect of amplifier gain. Higher the gain, smaller is the error.
2. Student should be able to understand the function of output potentiometer.

EXPERIMENT NO. 5

EXPERIMENT NO. 5

Aim: Plot unit step response of given transfer function transfer function and finds delay time, rise time and peak overshoot.

`

Theory:

Code:

```
% Define transfer function
numerator = [1]; % Replace with your transfer function numerator
denominator = [1 3 2]; % Replace with your transfer function denominator

% Create transfer function
sys = tf(numerator, denominator);

% Plot unit step response
figure;
step(sys);
title('Unit Step Response');
grid on;

% Analyze step response
info = stepinfo(sys);

delay_time = info.DelayTime;
rise_time = info.RiseTime;
peak_overshoot = info.Overshoot;

% Display results
fprintf('Delay Time: %.4f seconds\n', delay_time);
fprintf('Rise Time: %.4f seconds\n', rise_time);
fprintf('Peak Overshoot: %.2f%%\n', peak_overshoot);
```

Experiment 6

Aim: Plot the plot-zero configuration in the s-plane for the given transfer function.

```
% Define transfer function
numerator = [1]; % Replace with your transfer function numerator
denominator = [1 3 2]; % Replace with your transfer function denominator

% Create transfer function
sys = tf(numerator, denominator);

% Plot pole-zero map
figure;
pzmap(sys);
title('Pole-Zero Configuration in the S-Plane');
grid on;
```

Experiment 8

Aim: To study the time response of simulated linear systems.

Apparatus: - Linear system simulator kit(SCL-103), Connecting wires, Digital storage oscilloscope(DSO) and Multimeter.

Theory:

This set up is designed to study the time response of simulated linear systems. The present setup has straight forward building blocks of simulated process and signal sources. A dynamic system can be configured connecting them in suitable manners. Specifications for a control system design often involve certain requirement associated with the time response analysis of the system. For time domain analysis of control systems, we need to subject the system to various test inputs. Test input signals are used for analysing how well a system responds to these known set of inputs.

The transient and steady state response of the system can be studied when the test inputs are applied to the system. Linear System Simulator provides such an experimental setup where time response of various configurations of linear system can be studied. To study the time response of a variety of simulated linear systems and to correlate the studies with theoretical results.

To determine the open loop transfer function of all the blocks viz. integrator, time constant, uncommitted amplifier and error detectors/adders experimentally. To determine the first order (type 0 & type1) open loop system response for various input signals like unit step, ramp,

square wave etc. To determine the closed loop response of first and second order systems. To study disturbance rejection of closed loop system.

Representation of controller monitoring the plant:

The transfer function manipulations give us a transfer function model M(s) between command input R(s) and output Y(s); model Mw(s) between disturbance input W(s) and output Y(s); a model between command input R(s) and control U(s), etc. It is now easy to use some of the control analysis commands available from the Control System Toolbox. impulse(M) and step(M) commands represent common control analysis operations that we meet in this book. Also frequently used in the book are frequency-response plots.

Consider the block diagram

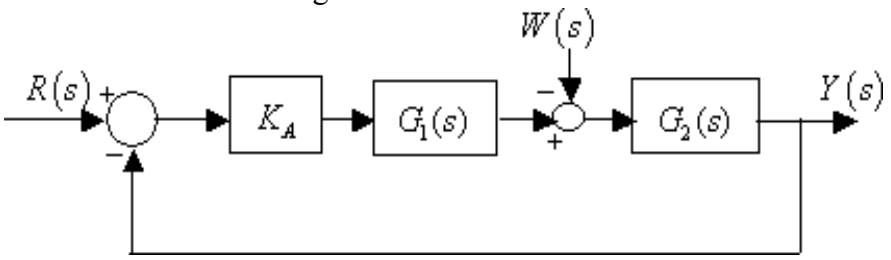


Fig. 5.1

Procedure:

For value of gain K_A = 80, the following two MATLAB sessions evaluate the step responses with respect to reference input $R(s)$ and disturbance signal $W(s)$ for

$$G_1(s) = \frac{5000}{(s+1000)}; G_2(s) = \frac{1}{s(s+20)}$$

```
>> %Step response with respect to R(s)
>> s = tf('s');
>> KA = 80;
>> G1 = 5000/(s+1000);
>> G2 = 1/(s*(s+20));
>> M = feedback(series(KA*G1,G2),1)
>> step(M)
```

The MATLAB responds with

Transfer function:

```
                400000
 ------------------------------------------
 s^3 + 1020 s^2 + 20000 s + 400000
```

and step response plot shown in Fig. M4.2. The grid has been introduced in the plot by right clicking on the plot and selecting **Grid** option.

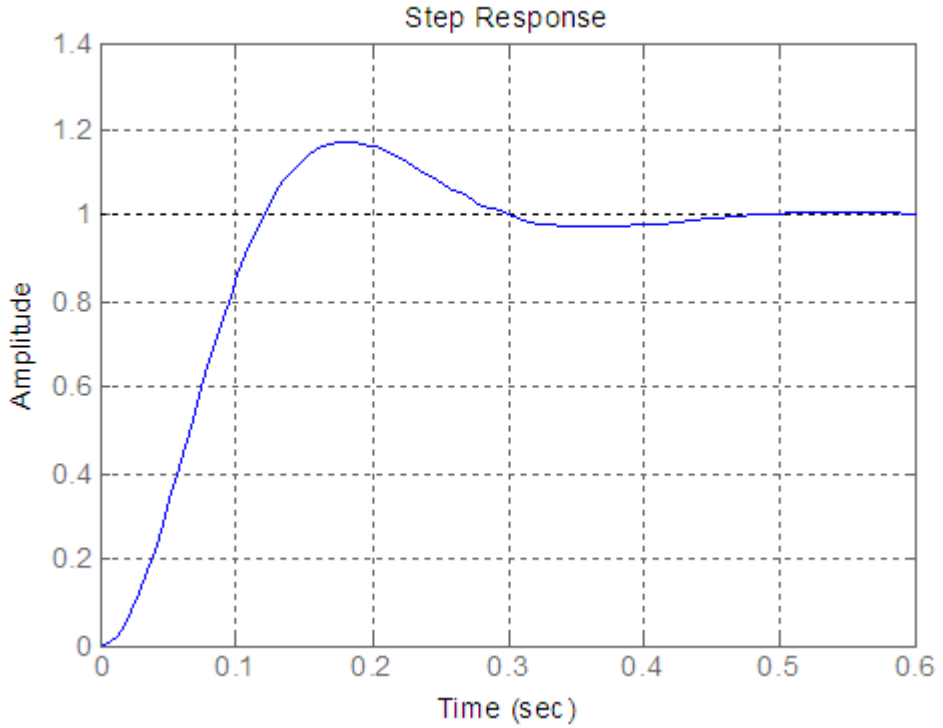


Fig. 5.2

```
>> %Step response with respect to W(s)
>> s = tf('s');
>> KA = 80;
>> G1 = 5000/(s+1000);
>> G2 = 1/(s*(s+20));
>> Mw = (-1) * feedback(G2, KA*G1)
>> step(Mw)
```

MATLAB responds with

Transfer function:

```
          -s - 1000
  ---------------------------------------------
  s^3 + 1020 s^2 + 20000 s + 400000
```

and step response plot shown in Fig. M4.3.

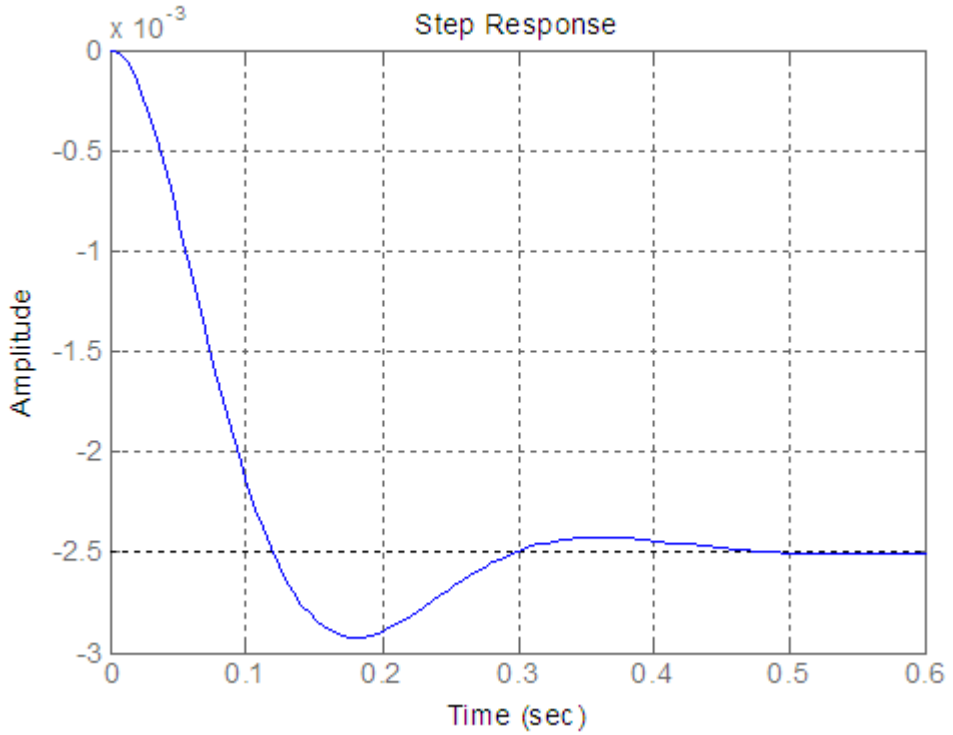


Fig. 5.3

Example M4.2

Let us examine the sensitivity of the feedback system represented by the transfer function

$$M(s) = \frac{K}{s^2 + s + K}$$

The system sensitivity to parameter K is

$$S_K^M = \frac{\Delta M / M}{\Delta K / K} = \frac{\partial M}{\partial K}\left(\frac{K}{M}\right) = \frac{s(s+1)}{s^2 + s + K}$$

Figure 5.4 shows the magnitudes of $\left|S_K^M(j\omega)\right|$ and $\left|M(j\omega)\right|$ versus frequency ω for K = 0.25; generated using the following MATLAB script. Text arrows have been introduced in the plot by following **Insert** from the main menu and selecting the option **Text Arrow**.

Note that the sensitivity is small for lower frequencies, while the transfer function primarily passes low frequencies.

```
w = 0.1:0.1:10;
M = abs(0.25./((j*w).^2+j*w+0.25));
SMK = abs((j*w .* (j*w + 1))./((j*w).^2 + j*w +0.25));
plot(w,M,'r',w,SMK,'b');
xlabel('Frequency (rad/sec)');
ylabel('Magnitude');
```

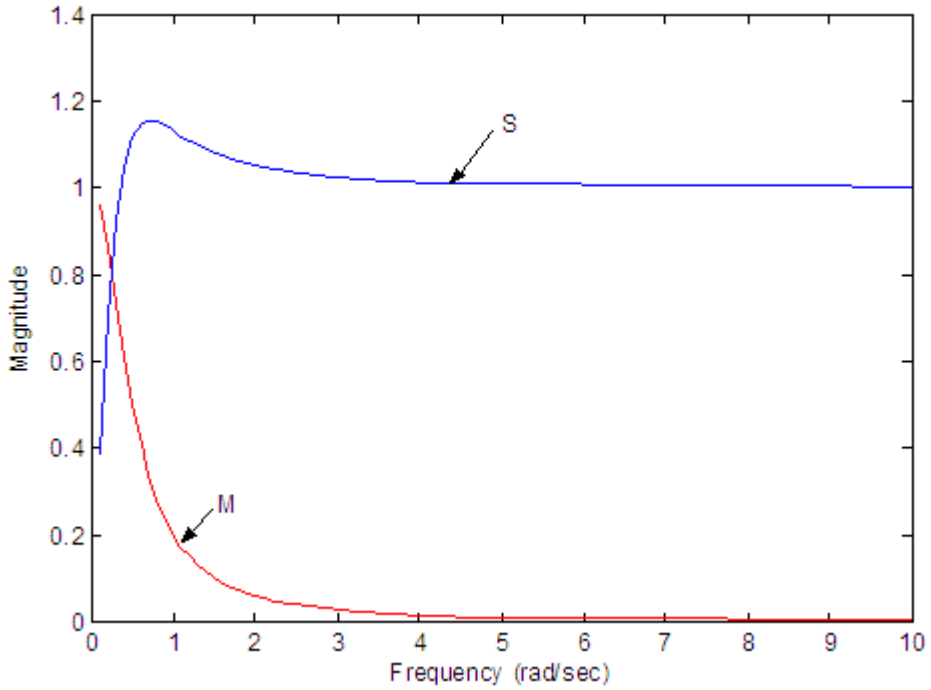


Fig. 5.4

Of course, the sensitivity S only represents robustness for small changes in gain K. If K changes from 1/4 within the range $K = 1/16$ to $K = 1$, the resulting range of step responses, generated by the following MATLAB script, is shown in Fig. M4.5. This system, with an expected wide range of K, may not be considered adequately robust. A robust system would be expected to yield essentially the same (within an agreed-upon variation) response to selected inputs.

```
s = tf('s');
S = (s*(s+1))/(s^2+s+0.25);
M1 = 0.0625/(s^2+s+0.0625);
M2 = 0.25/(s^2+s+0.25);
M3 = 1/(s^2+s+1);
step(M1);
hold on;
step(M2);
step(M3);
```

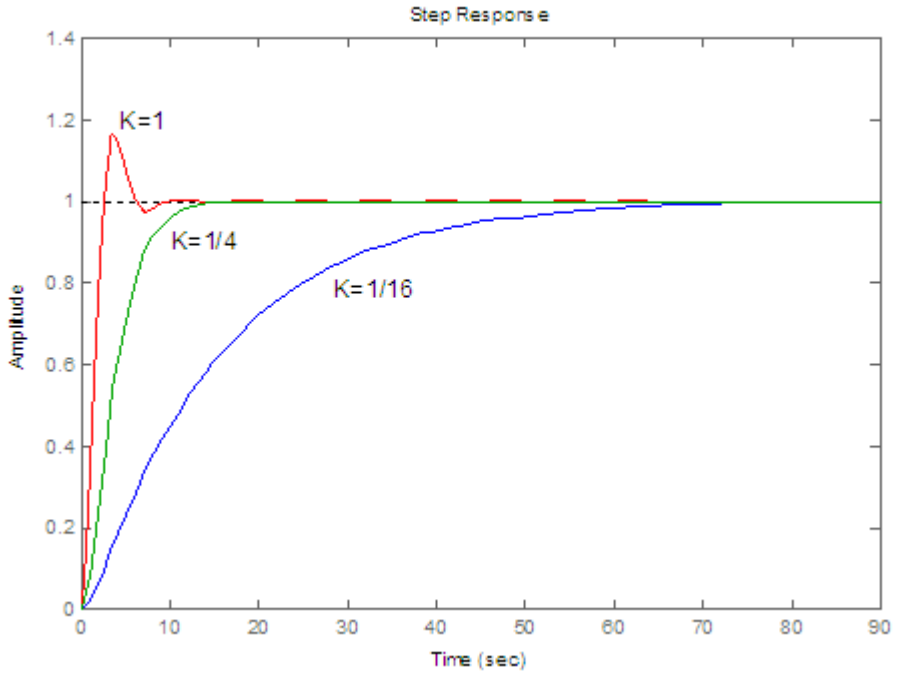


Fig. 5.5

Table No. 3.1

Observation:

Sl. No.	ACTUAL TEMPERATURE in Deg C	TIME in Seconds

FlowChart:

Characteristic: The Typical characteristic for second order system with step input may be as follows:

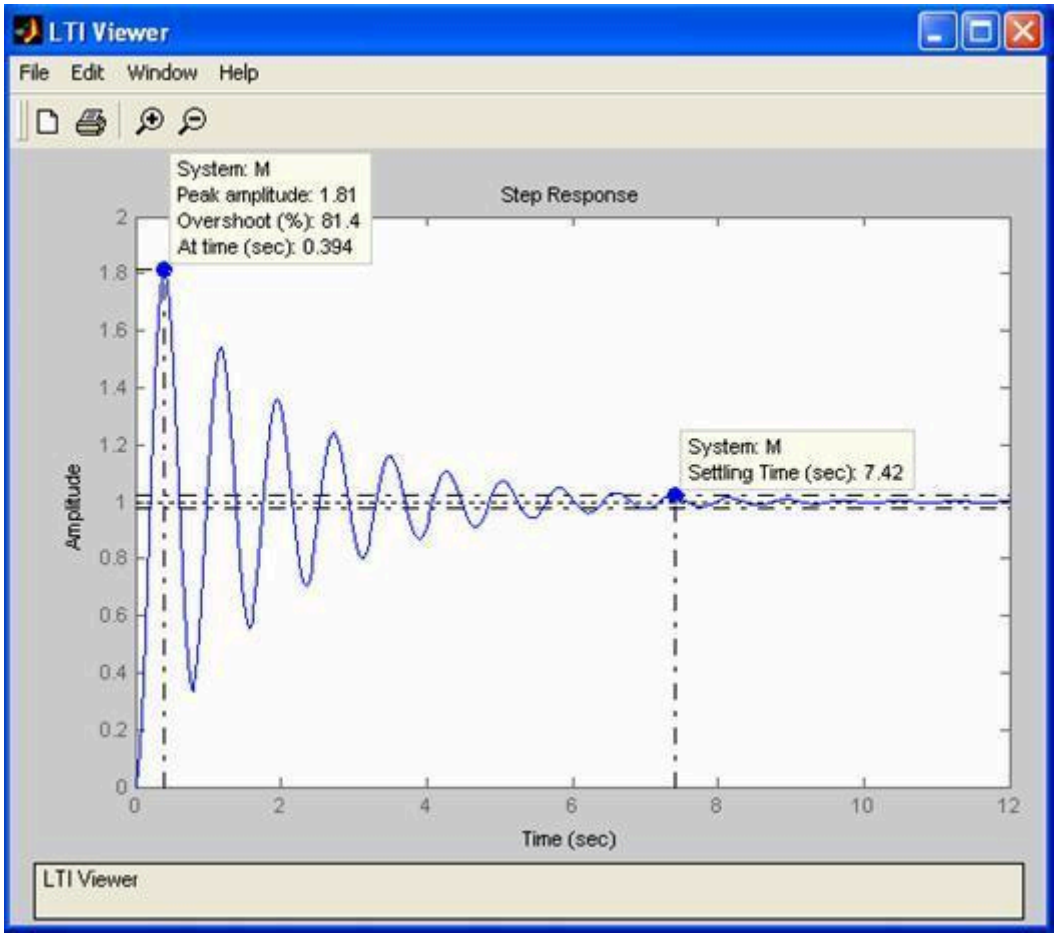


Fig. 5.6

Result: We have studied characteristics of 2nd order system in terms of damping ratio, time constant, rise time & settling time and verify with theoretical analysis.

Discussion of Result:

1. What is the difference between type and order of a system?
2. Why are we doing this experiment and what does the pattern tell?

3. Differentiate between linear and non-linear systems. Which blocks on the setup are non-linear?
4. Name the type of non-linearity. Why does the 3rd order system oscillate at very low value of K?

Experiment 10:

Aim: Plot root locus for any 2nd order system (with complex poles). For Mp=30%, find the value of K using MATLAB.

Root Locus

In this lab, we will learn the following new Matlab commands:

rlocus - calculates and plots the locus of the roots of:

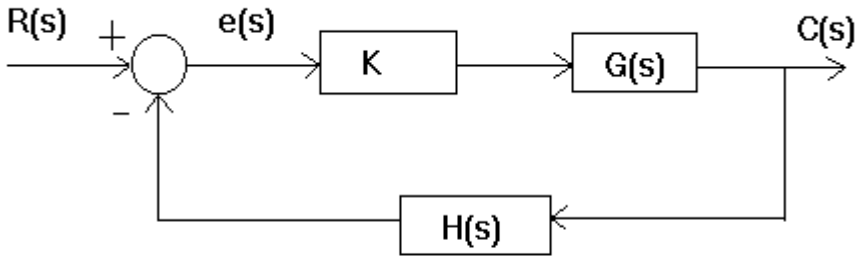
$$H(s) = 1 + k \frac{NUM(s)}{DEN(s)} = 0$$

rlocfind - Find the root locus gains for a given set of roots.

pzmap - Pole-Zero map of linear systems.
(for plotiing poles and zeros of a system)

sgrid or zgrid - Can be used to plot lines of constant damping ratio and natural frequency in the s-plane or z-plane.

Remember that the root locus of an (open-loop) transfer function H(s) is the locus of all possible locations of the closed loop poles with proportional gain k and unity feedback.



The closed-loop transfer function is

$$\frac{C(s)}{R(s)} = \frac{KG(s)}{1 + KG(s)H(s)}$$

and thus the poles of the closed loop system are values of s such that

1+ kG(s)H(s) = 0.
If we write G(s)H(s) = b(s)/a(s) then the denominator of the transfer function has the form:

a(s) + k b(s) = 0

a(s)
---- + b(s) = 0.
k

Let n = order of a(s) and m = order of b(s) [the order of a polynomial is the highest power of s that appears in it].
We will consider all positive values of k. In the limit as k -> 0, the poles of the closed-loop system are a(s) = 0 or the poles of G(s)H(s). In the limit as k -> infinity, the poles of the closed-loop system are b(s) = 0 or the zeros of G(s)H(s).
No matter what we pick k to be, **the closed-loop system must always have n poles**, where n is the number of poles of G(s)H(s). **The root locus must have n branches**, each branch starts at a pole of G(s)H(s) and goes to a zero of G(s)H(s). If G(s)H(s) has more poles than zeros (as is often the case), m < n and we say that G(s)H(s) has **zeros at infinity**. In this case, the limit of G(s)H(s) as s -> infinity is zero. The number of zeros at infinity is n-m, the number of poles minus the number of zeros, and is the number of branches of the root locus that go to infinity (asymptotes).
Consider a widget which has a loop transfer function of

$$G(s) = \frac{1}{s(s^2 + s + 4)}$$

Make a matlab file called rl.m. Enter the transfer function of the widget, and the command to plot the root locus:

```
numGH = 1;
denGH = [1 1 4 0];
rlocus(numGH,denGH)
```

This plot shows all possible closed-loop pole locations for a pure proportional controller. The plot, however, does not give directions of the loci. Can you tell the directions? And, can you draw the asymptotes?
Now try to find the value of k for which the locus crosses the jw-axis (the point at which the system becomes unstable). Type in the matlab window:
[k,poles] = rlocfind(numGH,denGH)
and then click on the crossing point in the graphics window. Matlab will return to you the locations of all three poles as well as the value of k. Try this for a few different points on the locus.
Let's say you decide to try a pure proportional controller, and pick the proportional gain to be 1/2 the gain at instability. Do an rlocfind again to save the value of k when the root locus crosses the imaginary axis (call it ku). In order to find the step response, you need to know the closed-loop transfer function. You could compute this using the rules of block diagrams, or let matlab do it for you:
[numCL, denCL] = cloop((ku/2)*numGH, denGH)

The two arguments to the function cloop are the numerator and denominator of the open-loop system. You need to include the proportional gain that you have chosen.

Unity feedback is assumed(i.e, H(s)=1).
If you have a non-unity feedback situation, look at the help file for the matlab function feedback, which can find the closed-loop transfer function with a gain in the feedback loop.
Check out the step response of your closed-loop system:
step(numCL,denCL)

For large K, the system is unstable. Also the rise time is not great.....
What if we reposition the closed loop poles for larger Wd (damping frequency), which should yield a smaller Tr ? You can try to improve the rise time yourself by choosing a range of K that gives a particular Wd which is large and then pick a value of K in that range to plot the step response.
Let us concentrate on stabilising the system.
Adding a zero to the system can pull the root locus farther to the left-half plane. Let's try a PD controller,
Note:you will learn about this in class lectures soon

K(s) = Kp + s Kd
Since we can only plot the root locus with respect to one parameter at a time, let's choose Kp = 2 and plot the locus with respect to Kd. Here we will need to do a little algebraic manipulation before we start. The characteristic polynomial of the closed-loop system is:

$$1 + K(s)G(s)H(s) = 0$$

$$1 + \frac{(Kp + s\,Kd)}{s\,(s^2 + s + 4)} = 0$$

$$s^3 + s^2 + 4s + Kp + Kd\,s = 0$$

$$1 + \frac{s\,Kd}{s^3 + s^2 + 4s + Kp} = 0$$

Note that a pure derivative term (if Kp were equal to zero) just adds a zero at the origin (a pure integral term adds a pole at the origin). The denominator now has an extra term due to Kp. To find the root locus of this system,

```
numD = [1 0];
denD = [1 1 4 2];
rlocus(numD,denD)
```

Now the system is stable for all values of Kd and Wd can be made very large for large K values. Pick a point on the root locus with Wd = 5 rad/sec using:
[kd,poles] = rlocfind(numD,denD)
and find the step response as mentioned before.

Note: You may need to show the root locus when Wd =5. To do this specify the range of K in the rlocus( ) command.

Let us see the step response at a point on the root locus with maximum damping coefficient (the open-loop transfer function is K(s) G(s)H(s)), where H(s)=1 i.e, unity gain :

```
numOL = [kd 2];
denOL = [1 1 4 0];
[numCL,denCL] = cloop(numOL,denOL);
step(numCL,denCL)
```

Notice that now there is no overshooting and see how the addition of the zero has increased the magnitude of the oscillations.
Now try a lead controller,

$$K(s) = K \frac{s+2}{s+10}$$

You could multiply out the numerator and denominator polynomials to find K(s) G(s)H(s), or you could let matlab do the work. Although matlab can't deal with symbolic expressions (like s), two polynomials are multiplied by taking the convolution of their coefficients (remember convolution?)

```
numK = [1 2];
denK = [1 10];
numL = conv(numK,numGH);
denL = conv(denK,denGH);
```

Now find the root locus with respect to the parameter k in your lead controller:
rlocus(numL,denL)
It looks like

If you always add a compensator which has the same number of zeros as poles (a good idea in practice), the closed-loop system will always have three zeros at infinity, or three asymptotes, which go off at angles of 120 degrees to each other. If two of them start from the poles near the jw-axis, they will become unstable quickly. One way to deal with those two poles is to add zeros near them in a configuration called a **notch filter**, as will be shown in the assignment.

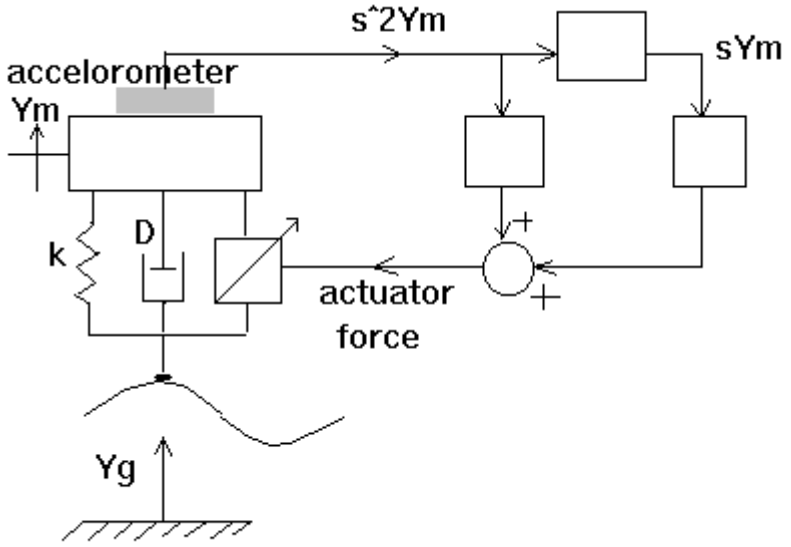
Assignments

1.Use the **notch filter**

$$K(s) = K\,\frac{s^2 + s + 3.5}{(s + 10)^2}$$

as the controller, plot the root locus of K(s)G(s)H(s) (H(s)=1 i.e, unity feedback)with respect to the gain k. Mark the direction of the loci and draw the asymptotes (with a little derivation based on your knowledge) on the plot. What is the difference from Figure 5? Choose the dominant poles to have a damping coefficient (zeta) of 0.707 (a 45 degree angle). Userlocfind to get the gain k at those poles, and plot the step response of the compensated system. Hand in a report bearing your plots and simple calculations as well as a little explanation.

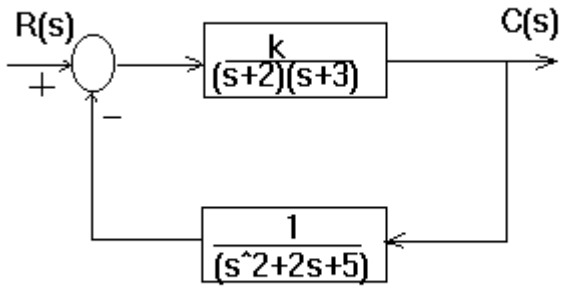
2.A possible active suspension system for AMTRACK trains has been proposed. The system uses a pneumatic actuator in parellel with the passive suspension system, as shown in the figure.



The force of the actuator subtracts from the force applied by the ground, as represented by dispacement, Yg(s). Aacceleration is sensed by an accelerometer, and signals proportional to acceleration and velocity are fed back to the force actuator. The transfer function relating acceleration to groound displacement is

$$\frac{Ym(s)}{Yg(s)} = \frac{s^2(Ds+K)}{(Ca+M)s^2 + (Cv+D)s + K}$$

Assume that M=1 and D=K=Cv=2.
Sketch a root locus for this system as Ca varies from zero to infinity.
3.For the system shown, do the following



a.Sketch the root locus.
b.Find the jw-axis crossing and the gain,K,at the crossing.
c.Find the gain **K**, for a damping ratio of 0.2.

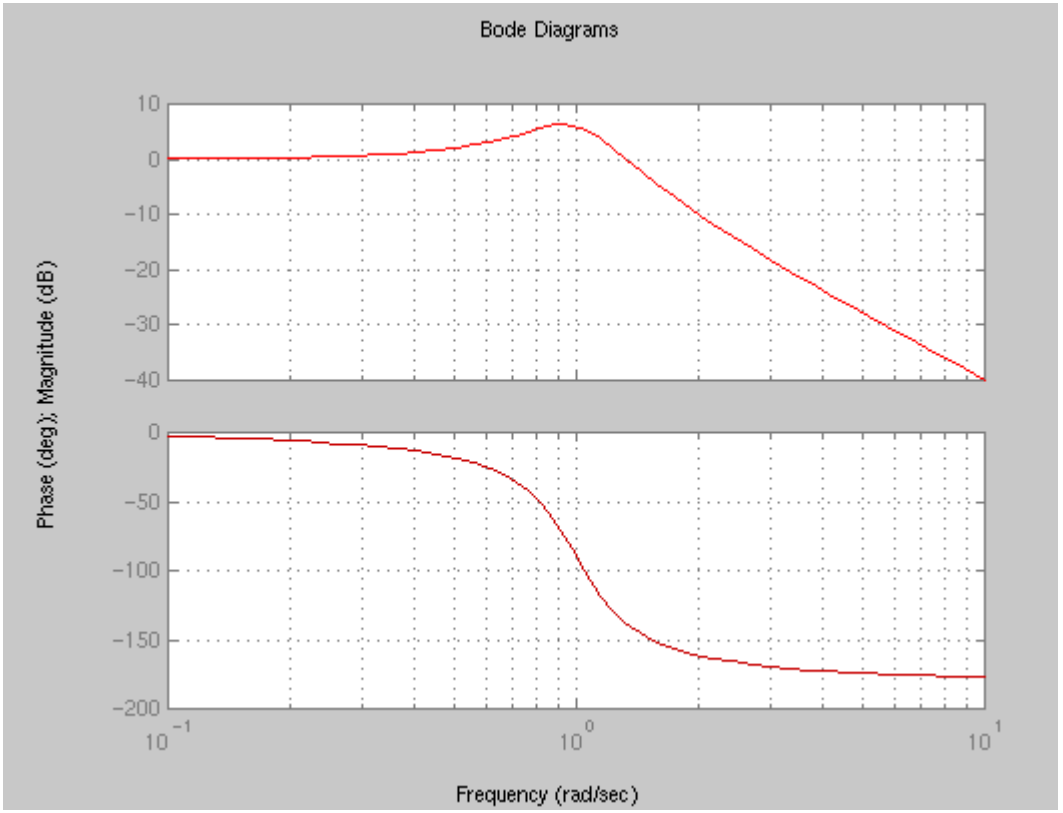
Aim: To design lead-lag compensator for the given process using Bode plots in MATLAB.
Frequency domain analysis
In this lab, we will learn the following new Matlab commands:
bode - Bode plot (frequency response).
loglog - Log-log scale plot.
hold - Hold current graph.
nyquist - Nyquist frequency response of LTI systems.

Frequency response methods provide a useful alternative to root locus methods for control system design. Frequency domain methods, however, can also be used in situations when the exact transfer function is not known - the frequency response of a system can be measured in the laboratory by choosing different sinusoidal inputs and measuring the magnitude and phase of the sinusoidal output, or by using a special type of instrument called a network analyzer.
In this lab, you will use MatLab to analyze the frequency response of some simple systems. Both Bode and Nyquist analysis will be performed.
Bode plots.
If a system has a transfer function G(s), its frequency response is G(jw) (it is only evaluated for values of s which are on the jw axis). Since G(jw) is a complex number, it is usually represented in two different plots, one for its magnitude and the other for its phase, both as functions of w. These two plots together are called the Bode plots of the system. By using a log scale for the frequency, a linear scale for the magnitude (in dB), and a linear scale for the phase, the composition of two systems (in series) is easily accomplished by adding together their Bode plots. For example, consider a standard second-order system (mass-spring-damper):

$$G1(s) = \frac{1}{s^2 + s/2 + 1}$$

and find its Bode plot using matlab:

```
numG1 = 1;
denG1 = [1 0.5 1];
bode(numG1,denG1);
```

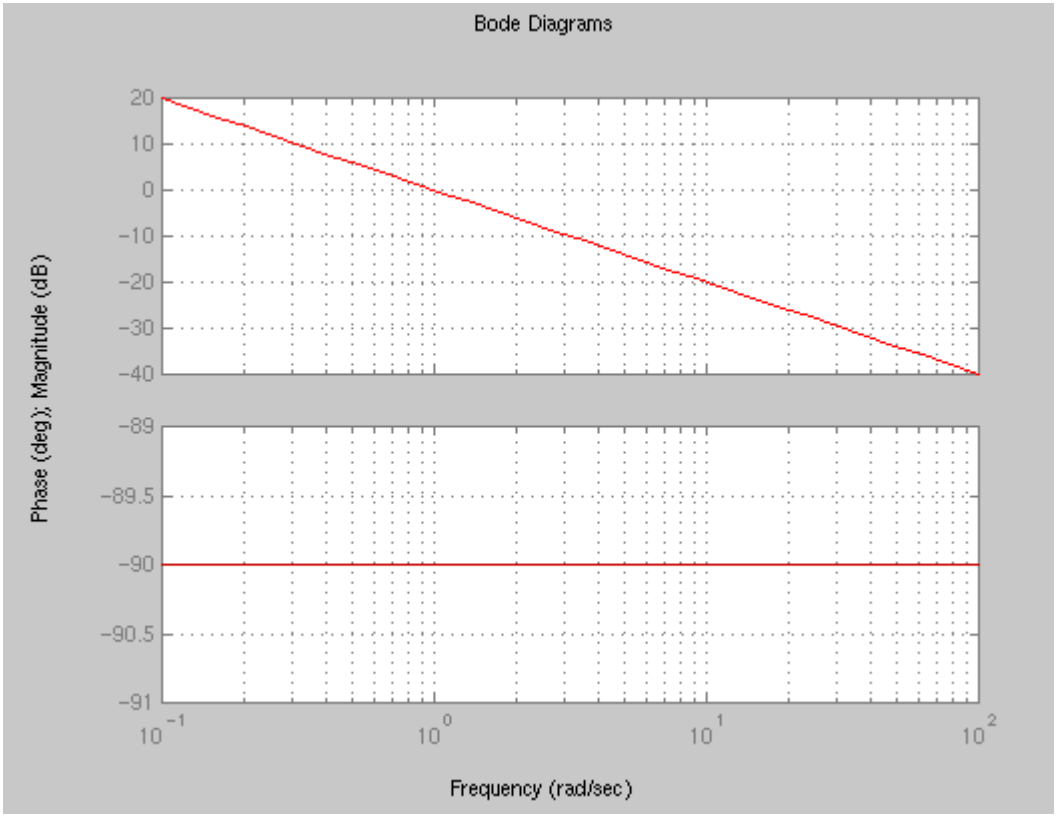


(note that this plot is the true frequency response, not the asymptotic Bode response you learned in lecture)

Now plot the Bode response of an integrator (a pole at the origin):

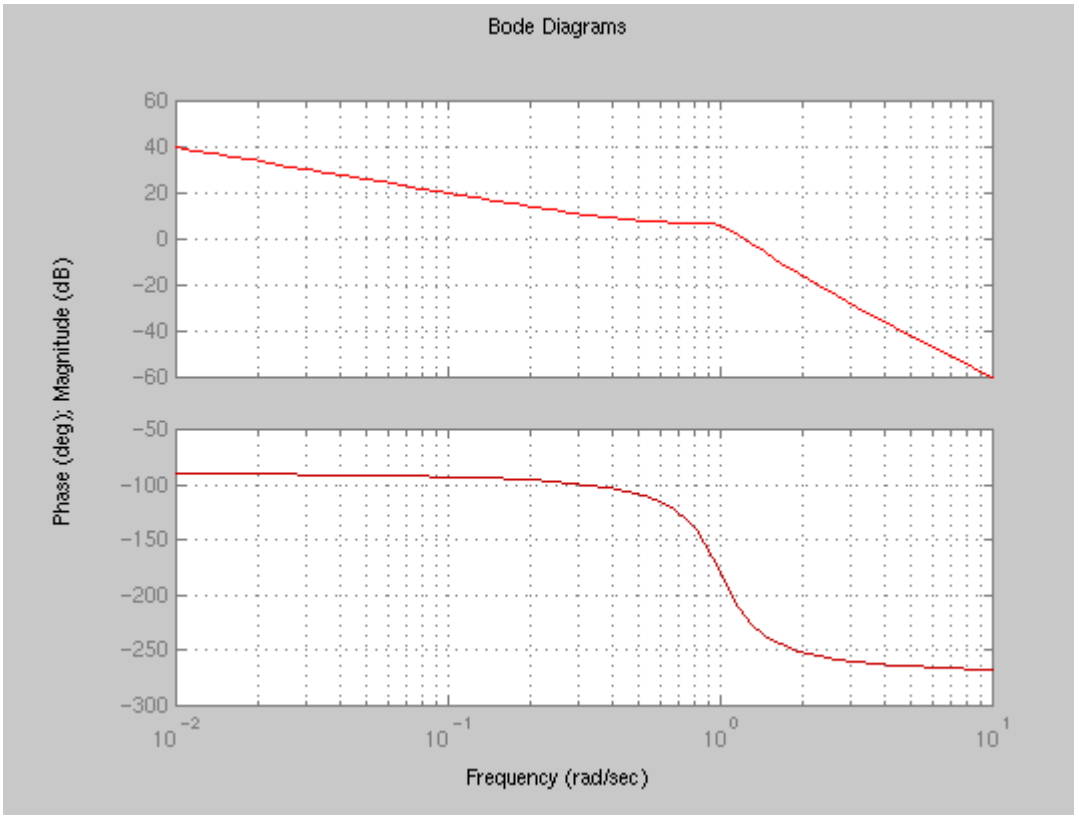
G2(s) = 1/s

```
numG2 = 1;
denG2 = [1 0];
bode(numG2,denG2);
```



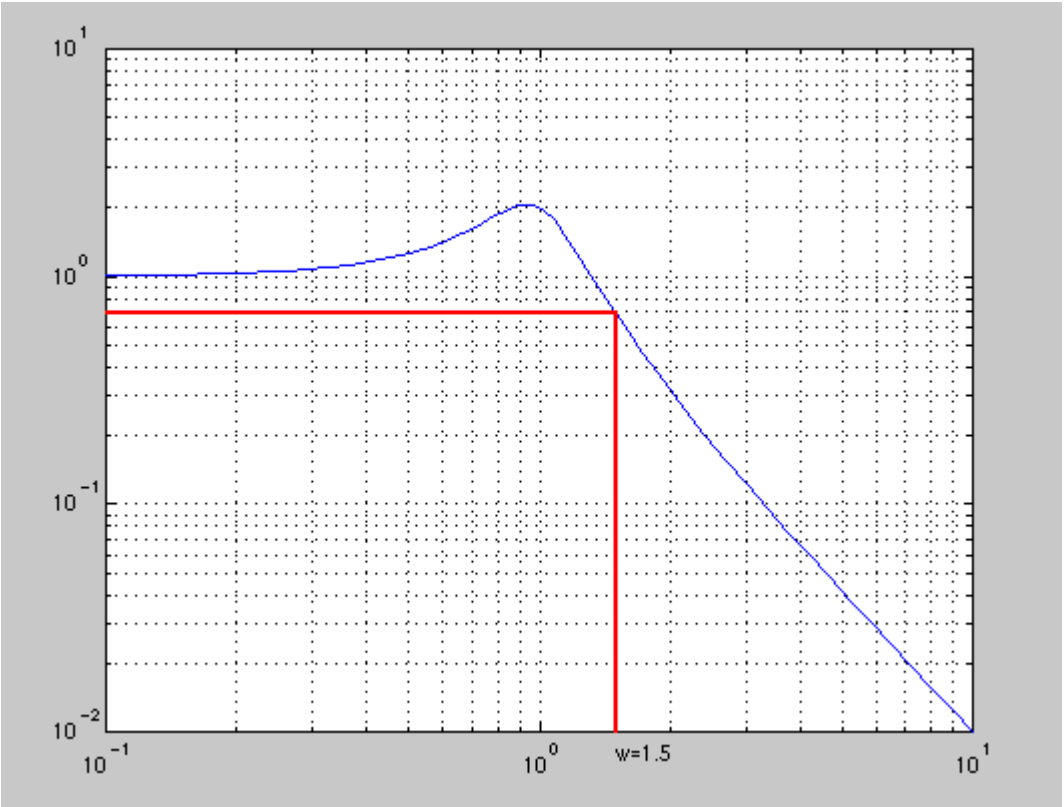
The combined system has a transfer function given by:
G(s) = G1(s) G2(s)

[numG,denG] = series(numG1,denG1,numG2,denG2);
bode(numG,denG);



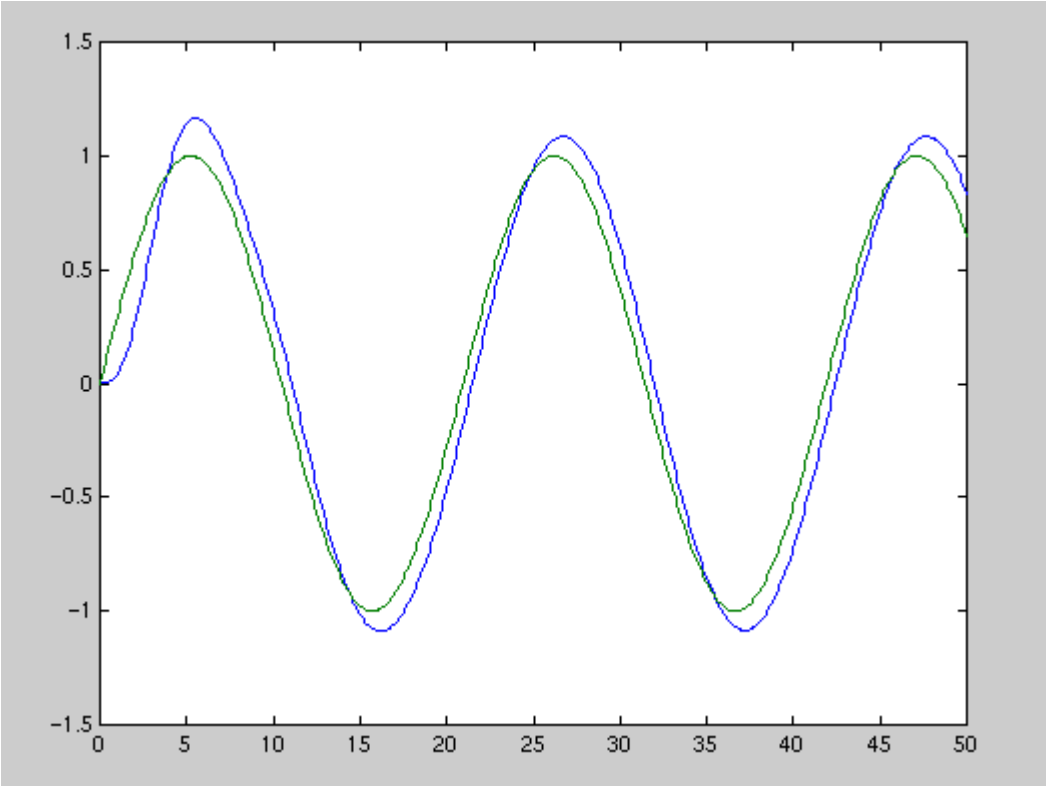
which is the sum of the two individual Bode plots. Note the subplot and axis commands which change the scale on the magnitude plot only.
There are several characteristics of a system that you can read directly from its Bode plot:
Bandwidth. Sinusoidal inputs with frequency less than w_BW are amplified by the system's DC gain, Mdc, while sinusoidal inputs with frequency greater than w_BW are attenuated. For a second-order system, the output is attenuated by a factor of 0.707*Mdc or greater. For your original 2nd order system (G1), plot the magnitude response on a log-log scale, i.e. do not express the magnitude in dB. Locate the point M = 0.7 on the magnitude axis (or Mdb = -3dB), and draw a line across until it hits the plot of |G(s)|. Read down to find w_BW, as shown in the figure:

```
[m,p,w]=bode(numG1,denG1);
loglog(w,m);
grid;
hold on;
plot([0.1 1.5], [0.7 0.7],'r-');
plot([0.1 1.5], [0.69 0.69],'r-');
plot([1.5 1.5], [0.01 0.7],'r-');
plot([1.49 1.49], [0.01 0.7],'r-');
text(1.5,0.008,'w=1.5');
```

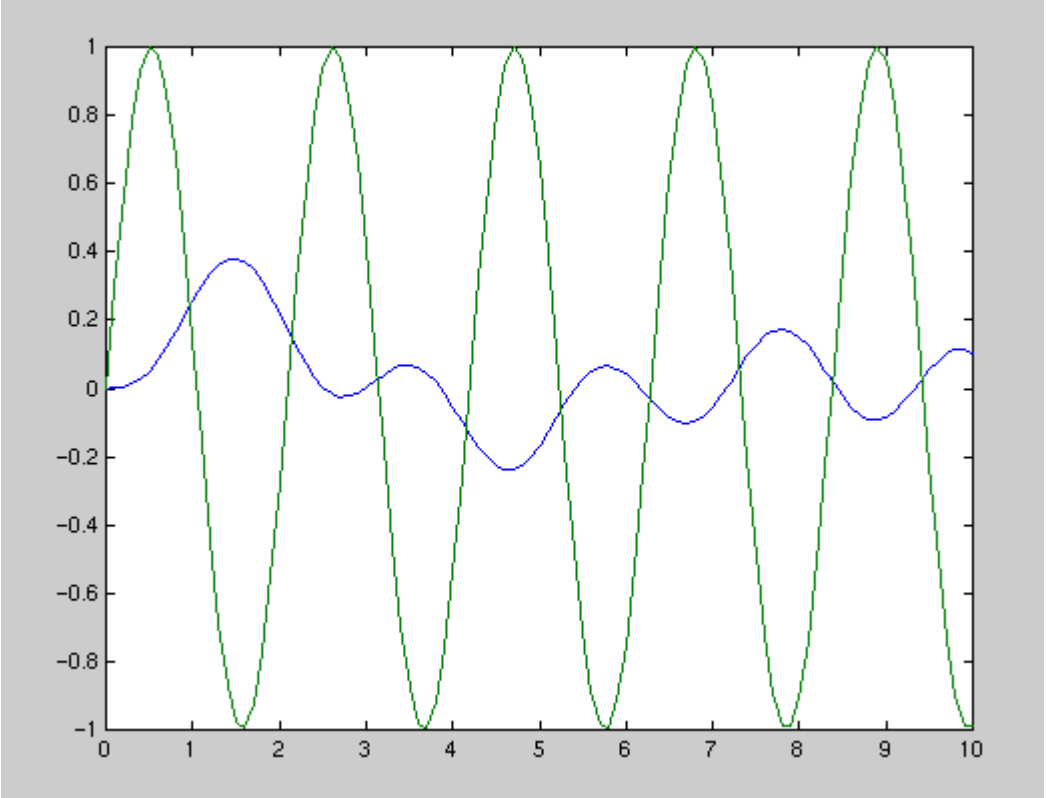


You can find the time-domain output of the system to a sinusoidal input by using the lsim command in matlab. Try applying two sinusoids with different frequencies to the system, and observe the response:

```
t=0:0.1:50;
u = sin(0.3*t);
[y,x] = lsim(numG1,denG1,u,t);
plot(t,y,t,u)
```




```
t=0:0.1:10;
u = sin(3*t);
[y,x] = lsim(numG1,denG1,u,t);
plot(t,y,t,u)
```



Note the phase lag as well as the magnitude attenuation in the second plot.

Questions:

1. Why are the last two plots so different in their output magnitude?

2. What is the bandwith of system G2?

*3. What is the bandwith of system G1*G2?*

System Type. The system type is the number of (open-loop) poles at s=0, and is the (negative of the) slope of the low-frequency portion of the magnitude Bode plot. By Bode's gain-phase relationship:

<G(jw) = n x 90 (n is the slope of |G(jw)|)

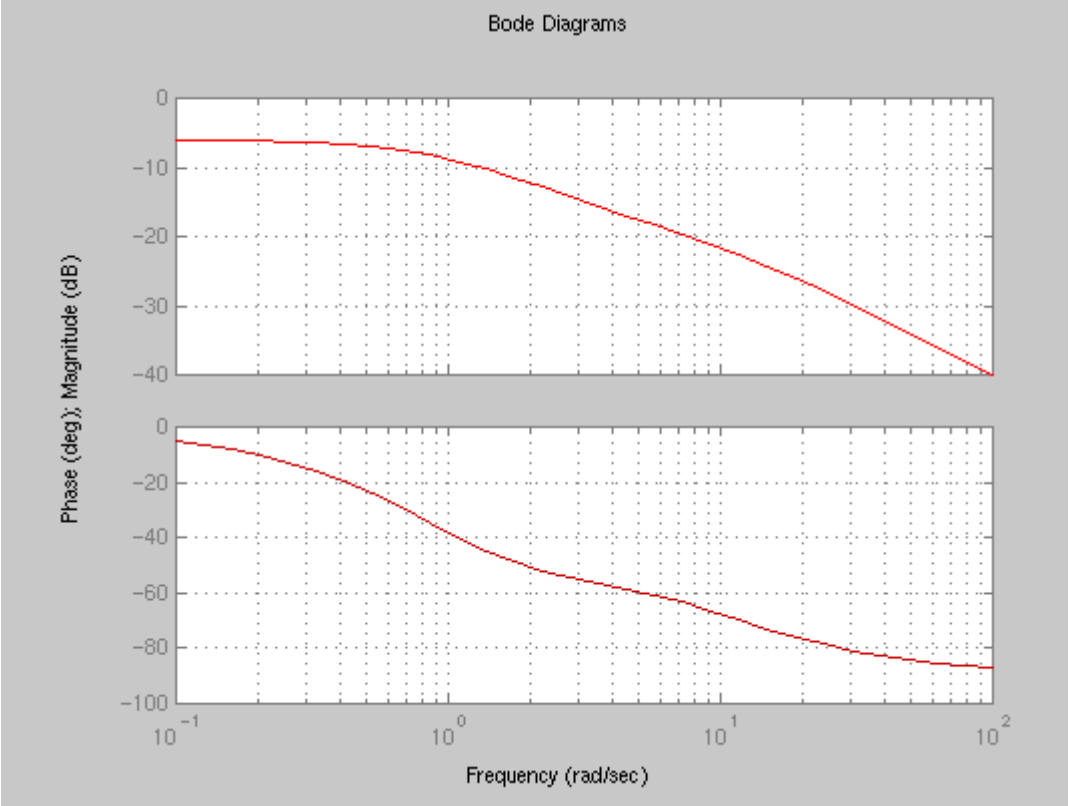
Thus you should be able to determine the type of a system by observing the low-frequency phase of G(jw) and dividing by 90 degrees.

With unity-gain feedback, a Type 1 system has zero steady-state error to a step input and a Type 2 system has zero steady-state error to a ramp input.

Question:

4. Plot the Bode plots of systems G1, G2, and G1*G2. From the plots, determine the types each system. Explain your answer graphically.

Determining the transfer function from a Bode plot. The bode plots can be used to estimate the transfer function of the system. For example, if we have measured the following bode plots of the system:



from which we observe that there are three "break points" at w=1, w=4, and w=8, and that the gain of the system is such that 20 log K = -6 so that K=10^{-6/20}=0.5. Therefore the system's transfer function is:

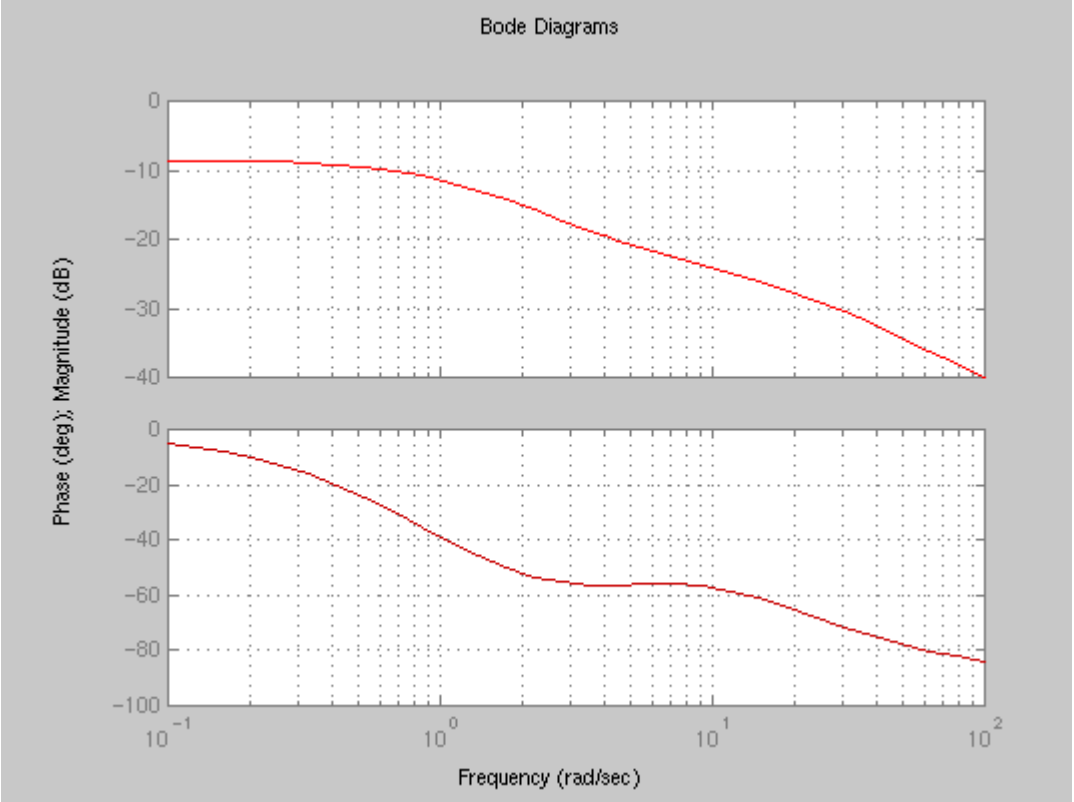
$$T(s) = \frac{K\,(s/4 + 1)}{(s/1 + 1)\,(s/8 + 1)} = \frac{(s + 4)}{(s + 1)\,(s + 8)}$$

The frequency response (bode plots) of a system can be measured in the laboratory by choosing different sinusoidal inputs and measuring the magnitude and phase of the sinusoidal

output, and the transfer function of the system can be determined from this data as we just showed.

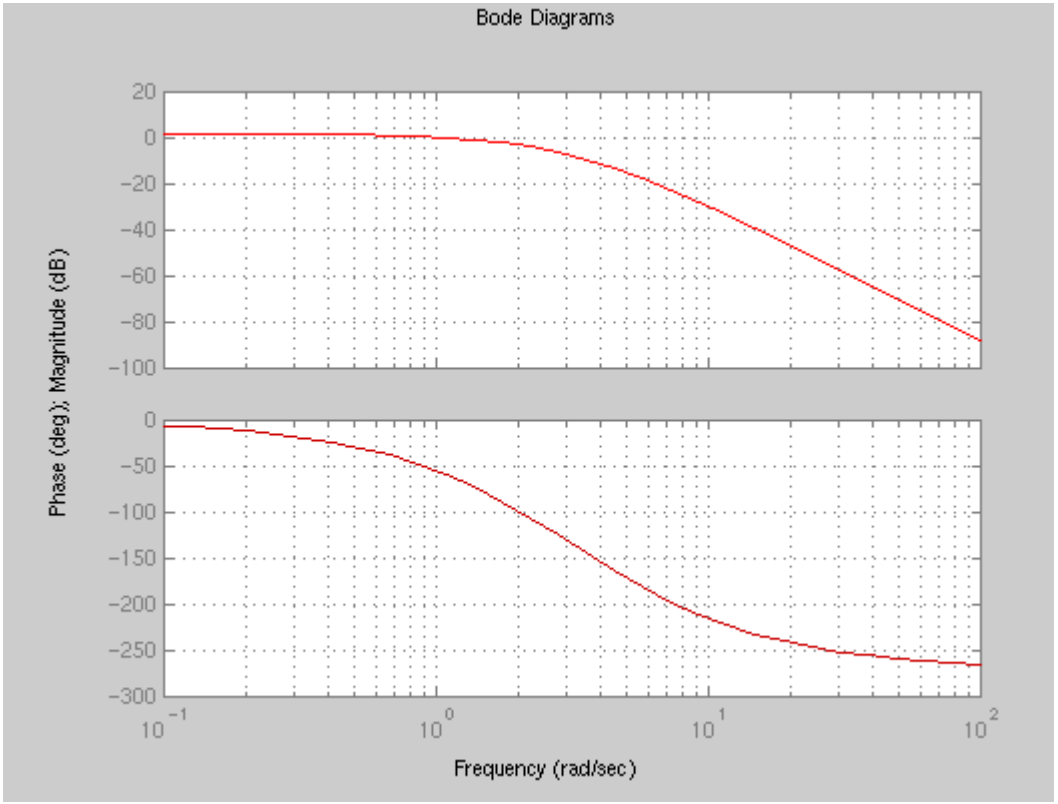
Question:

5. What is the transfer function corresponding to the following bode plot?



The stability can also be judged using Bode plots:

```
figure(4)
bode(numG3,denG3);
```



Questions: Consider the unit feedback system with the open-loop transfer function

$$G4(s) = \frac{K(s+3)(s+5)}{(s-2)(s-4)}$$

Is the closed-loop system stable for K=3? K=0.7? Why? Can you draw a general conclusion for K so that the closed-loop system is always stable?

Stability Margins

Gain margin. The closed-loop system is marginally stable when the root-locus crosses the jw axis, or 1 + KG(jw) = 0. In terms of magnitude and phase, |KG(jw)| = 1 and <KG(jw) = -180. Since most systems become unstable as K increases, the gain margin is defined as how much the proportional gain K can increase (in a unity feedback situation) before instability results. This can be read directly from the Bode plot by finding the point when the phase crosses 180 degrees, and finding the magnitude at that frequency:

If the gain is greater than 1 (0dB) then the system is unstable at K=1, and the gain margin is negative (dB) or < 1 (magnitude). If the phase never crosses -180, then the system is stable for all gains and the gain margin is infinite.

Phase margin. The analog of the gain margin is the phase margin. This is found from the Bode plot as the difference from the phase from -180 when the magnitude is equal to 1 (or 0dB).

If the phase is less than -180 when the magnitude is equal to one, then the closed-loop system is unstable for K=1. The phase margin for different proportional gains K can also be found from the Bode plot. When |KG(jw)| crosses magnitude 1, then |G(jw)| crosses magnitude 1/K. Shown above is the calculation for K = 1/3 (20 log (1/K) = 9.5 dB).

Question:

8. Find the phase margin and gain margin of G1(jw).
*9. Find the phase margin and gain margin of G1*G2(jw).*
10. Use bode, nyquist, AND root locus arguements to prove or disprove the following statement: "Any second-order minimum phase system with at least one finite zero can be stabilized by using a high-gain unit feedback".

Aim: To study the stability of control system using Nyquist plot
Nyquist Plot:
A Nyquist plot is a parametric plot of a frequency response used in automatic control and signal processing. The most common use of Nyquist plots is for assessing the stability of a system with feedback.
Nyquist Stability Criterion:

A feedback system or closed loop system is stable if the contour 'T' of the open loop transfer function G(s)H(s) corresponding to the Nyquist Contour in the s-place encircles the point(-1+j0) in counterclockwise direction and the number of counterclockwise encirclements about the (-1+j0) equals the number of poles of G(s)H(s) in the right half of s-place i.e., with positive real parts.
Or Closed Loop System is stable if the contour 'T' G(s)H(s) does not pass
through or does not encircle (-1+j0) point.
Procedure to plot

Step 1 – Check for the poles of G(s) H(s) of jω axis including that at origin
Step 2 – Select the proper Nyquist contour – a) Include the entire right half of s-plane by drawing a semicircle of radius R with R tends to infinity.
Step 3 – Identify the various segments on the contour with reference to Nyquist path
Step 4 – Perform the mapping segment by segment substituting the equation for respective segment in the mapping function. Basically we have to sketch the polar plots of the respective segment.
Step 5 – Mapping of the segments are usually mirror images of mapping of respective path of +ve imaginary axis.
Step 6 – The semicircular path which covers the right half of s plane generally maps into a point in G(s) H(s) plane.
Step 7- Interconnect all the mapping of different segments to yield the required Nyquist diagram.
Step 8 – Note the number of clockwise encirclement about (-1, 0) and decide stability by N = Z – P

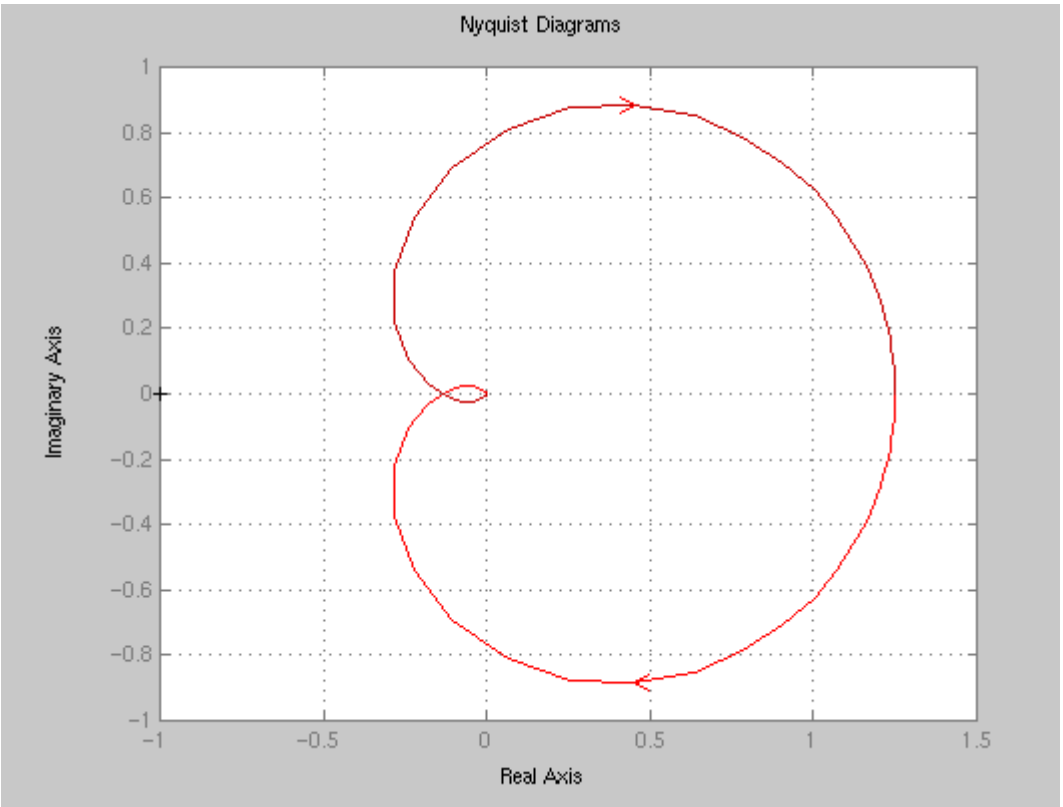
Nyquist diagram and stability.

Consider the unit feedback system with the open-loop transfer function

$$G3(s) = \frac{40\,K}{(s+2)(s+4)(s+5)}$$

For K=1, find its Nyquist diagram using matlab:

```
numG3 = 40;
denG3 = conv([1 2],[1 4]);
denG3 = conv(denG3,[1 4]);
nyquist(numG3,denG3);
grid;
```



Based on Nyquist criterion, it is seen that P=0, N=0, Z=P-N=0. Therefore, the closed-loop system is stable. From this diagram, it can be seen that, approximately, for $0 < K < 10$, the system will remain to be stable.

Experiment 10

Aim: To find the transfer function of control system using block diagrams and Feedback

MatLab's Control Toolbox provides a number of very useful tools for manipulating block diagrams of linear systems. There are three basic configurations that you will run into in typical block diagrams. These are the parallel, series, and feedback configurations. MatLab is a very useful tool for removing some of the drudgery from this task. In this practical, we will talk about MatLab's functions for automated block diagram manipulation, and also look at how matlab can be used to manually manipulate block diagrams. Using these tools, we will investigate some important properties of feedback systems such as tracking and steady-state error.

Key Commands:

- parallel
- series
- feedback
- cloop
- conv
- deconv

To begin this exercise, create two linear systems by typing:

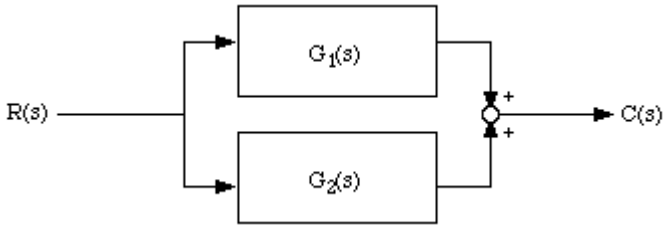
```
num1 = [1 0]
den1 = [1 0 2]
##############
num2 = 1
den2 = [3 1]
```

$$G_1(s) = \frac{s}{s^2 + 2}$$

$$G_2(s) = \frac{1}{3s + 1}$$

Parallel blocks

Consider a diagram with two blocks in parallel, as shown here:



The overall transfer function, C(s)/R(s), is given by T(s) = G1(s) + G2(s). The MatLab function parallel() can be used to determine the overall transfer function of this system. To see how this works, type:

[num, den] = parallel(num1, den1, num2, den2)

The result should be two polynomials which we have placed into the variables 'num' and 'den' : These polynomials describe the overall transfer function defined by the parallel() operation, i.e. a third-order system:

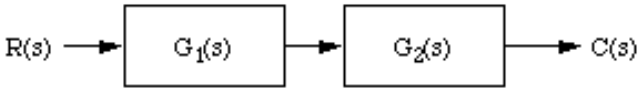
$$T(s) = \frac{4s^2 + s + 2}{3s^3 + s^2 + 6s + 2}$$

Series Blocks

A series connection of transfer functions yields an overall transfer function of

T(s) = G1(s) G2(s).

The matlab function series() can be used to determine this transfer function. Using the example systems, find the series connection by typing:



[num, den] = series(num1, den1, num2, den2)

As in the parallel connection, the result should be a third-order system:

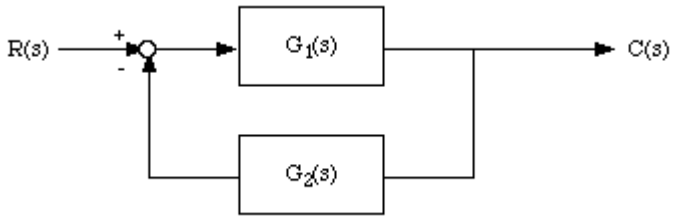
$$T(s) = \frac{s}{3s^3 + s^2 + 6s + 2}$$

Feedback Blocks

Feedback connections are what the topic of control systems is all about. You should already know that for the following negative feedback connection, the resulting transfer function is given by

$$T(s) = \frac{G(s)}{1 + G(s)H(s)}$$

Rather than simplifying this by hand, T(s) can be found using MatLab's feedback() function.



To try this out, type:
[num, den] = feedback(num1, den1, num2, den2, -1)
The result should be:

$$T(s) = \frac{3s^2 + s}{3s^3 + s^2 + 7s + 2}$$

Note that although MatLab's help file on this function says that positive feedback is assumed, this is not always the case - negative feedback is the default in some versions of MatLab. To be sure you are applying the proper feedback, a fifth arguement (SIGN) must be added to the function as shown above. If SIGN = 1, positive feedback is used. If SIGN = -1, negative feedback is used.
Another way to produce a feedback system is to use the cloop() function. This function produces the transfer function of a unity-gain feedback system, i.e. the case when G2(s) = 1. If G2(s) = 1, then the feedback connection can be determined as follows:
[num, den] = cloop(num1, den1, -1)
Note that just as with feedback(), cloop() takes a SIGN arguement to specify negative or positive feedback. Unity-gain feedback is very common in control systems, so cloop() is a very useful function to have at your command.

Manual Block Manipulation
The previous block manipulations could have been done "by hand", instead of using the automated functions, by employing the conv() and deconv() functions. These functions perform matrix convolution and deconvolution, which is effectively the same thing as polynomial multiplication and division.
For example, to multiply S^2+1 and S^2+2S+3
answer = conv([1 0 1], [1 2 3])
which will result in ***answer = [1 2 4 2 3]*** , or equivalently

$$s^4 + 2s^3 + 4s^2 + 2s + 3$$

Similarly, to divide ***answer*** by

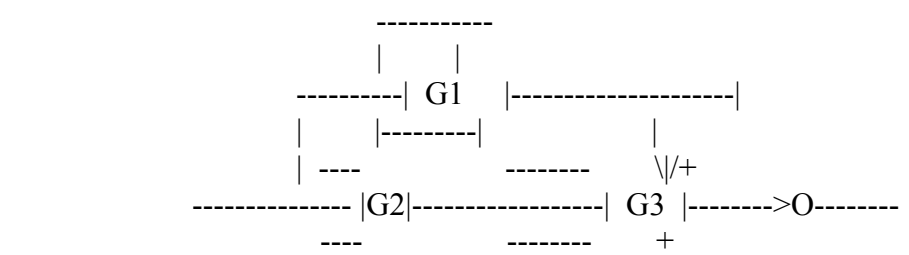
$$s^2 + 1$$

answer2 = deconv([1 2 4 2 3], [1 0 1])
which results in ***answer2 = [1 2 3]*** , or equivalently

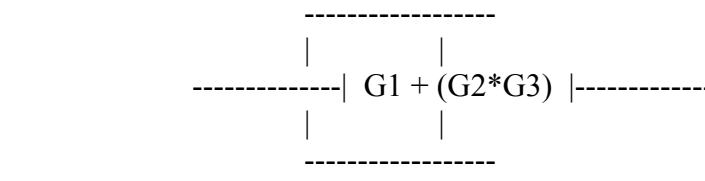
$$s^2 + 2s + 3$$

conv() and deconv() are very useful tools for many control system analysis and design applications, and you should keep them in mind for when you need to multiply or divide polynomials. They are mentioned here since block reduction is effectively a process of polynomial multiplication, even though the automated functions parallel() , series() , and feedback() are usually more convenient.

Example problem:
Let
G1(s)= 1, G2(s)=-1,
G3(s)= s/(s+1)
Consider the following block diagram.



Sol: This can be reduced to

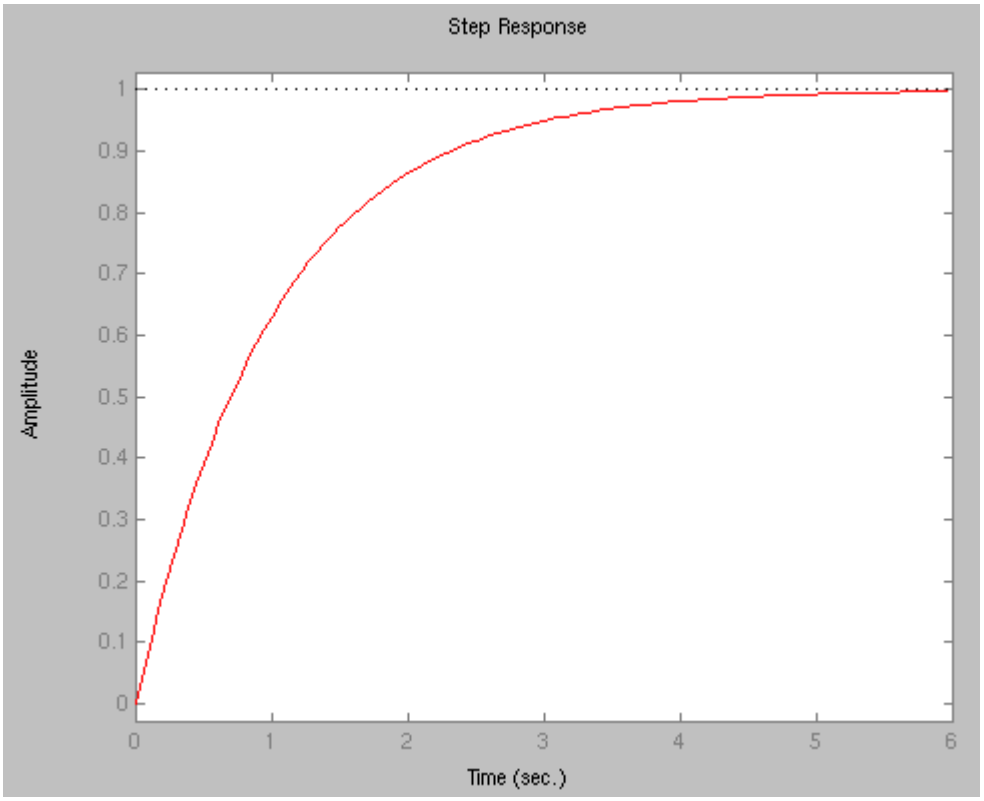


Matlab can be used to give the same result and step response can be plotted as shown below:
An m-file can be written like this:

```
num1=1;
den1=1;
num2=-1;
den2=1;
num3=[1 0];
den3=[1 1];
sys1=tf(num1,den1);
sys2=tf(num2,den2);
sys3=tf(num3,den3);
sys4=series(sys2,sys3);
sys=parallel(sys1,sys4)
step(sys)
```

Matlab will respond

Transfer function:
1/(s+1)
and opens a window showing the following plot:

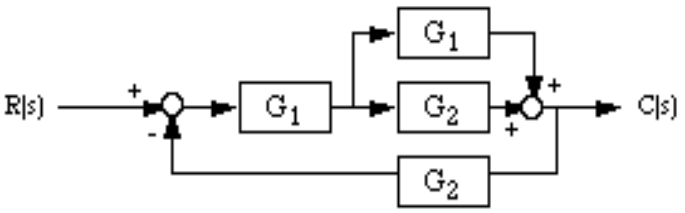


Assignments

1. Block diagram reduction - write an m-file to find the overall transfer function of the following system, where

$G_1(s) = \frac{s}{s+2}$, and

$G_2(s) = \frac{1}{s}$:



Note that if the version of MatLab you are using does not support the parallel() function, you will need to manually calculate the parallel connection of G1 and G2 in the above diagram. Make sure you simplify the resulting transfer function as much as possible, and print out your results. Also plot the step response for the transfer function.

Experiment 12

Aim: To study the performance of PID Controller

Apparatus: - Temperature measurement system Solid-state relay for driving heater bulbs Micro controller based control unit LCD display

Theory:

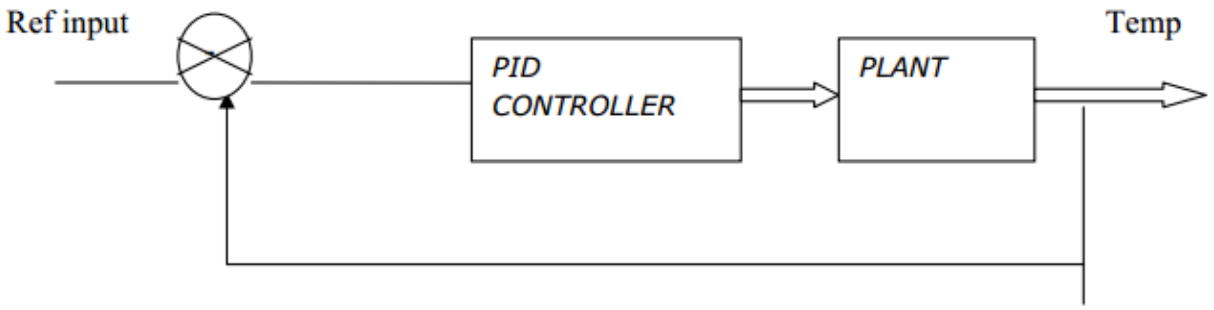
This set up is designed to demonstrate the working of a typical temperature controller using PID mode of operation. Proportional controller is a mode of control action in which there is a

continuous linear relationship between values of deviation and manipulated variable. In order to remove the offset associated with proportional action , combination of P+I is widely used, As a result of integral action, the offset error is almost reduced to zero but the transient response is adversely affected. A derivative control action may be added to proportional control to form P+D action. Derivative control action may be defined as control action in which the magnitude of the manipulated variable is proportional to the rate of change of error. P+I+D action produces smallest maximum deviation and offset is eliminated because of integral action. The derivative action provides improved transient response against load variations.

In short PID approach to control problem can be summarized in terms of the mathematical equations governing the operation

A simple analysis would show that the derivative block essentially increase the damping ratio of the system and therefore improves dynamic performance by reducing overshoot, the integral action eliminates the steady state error.

Representation of controller monitoring the plant:



Procedure:

For Proportional Control (P)

1. Keep SW3 in Test mode.
2. Keep SW2 in Mode Check
3. Then Keep KI=0 &KD=0, now system will be configured for proportional mode
4. Make proper connection for heater cable, RTD cable &fan cable.
5. Now select system as SW3 in TEST SW2 in Normal SW1 in PID side.
6. If selection is kept as above you will be able to set temperature with the help of P1. You will see display as ST= AT=
7. Now select system as SW3 in START SW2 in NORMAL SW1 in PID 8. Now system will start in proportional mode

For P+I mode:

All procedure is same as described for proportional mode Only at pt(3) keep KP>0, KI>0 &KD=0 then system will be configured for P+I mode.

For P+I+D mode:

All procedure is same as described for proportional mode. Only at pt(3) keep KP>0, KI>0 &KD>0 then system will be configured for P+I+D mode.

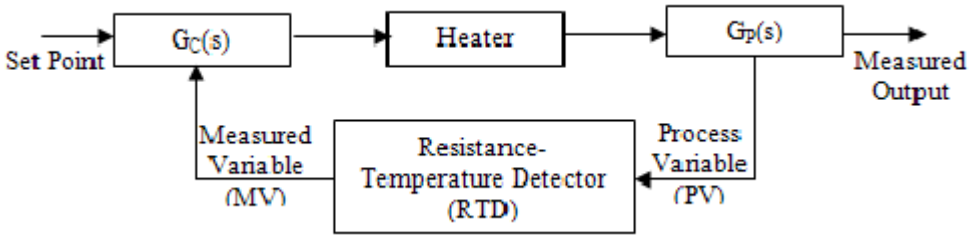
NOTE: For Modes: You can take observation with fan load on/off or by varying fan speed for creating disturbances on the system. Fan provided acts as load on the system by taking more/less away from the modes

Table No. 3.1

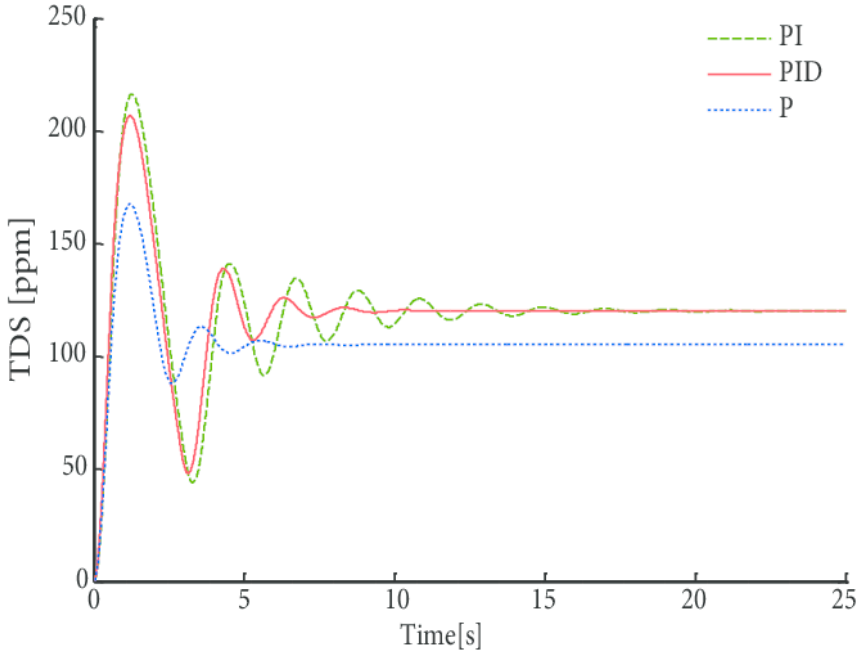
Observation:

Sl. No.	ACTUAL TEMPERATURE in Deg C	TIME in Seconds

FlowChart:



Characteristic: The Typical PID characteristic for different settings may be as follows Temp set pt Time



Result: Temperature of oven is controlled by using PID controller. By varying P,I,D values two graphs of temperature vs time is plotted.

Discussion of Result:

1. The steady state error may be reduced by proper setting of Integral controller setting.
2. The transient state is improved by adjustment of Kp & Td.
3. For different settings of P I D different response Curves may be obtained.
4. The response Characteristics of P+I+D controller exhibit

COURSE EXIT SURVEY

BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING
(Approved by AICTE, New Delhi & Affiliated to Guru Gobind Singh Indraprastha University, Delhi)
(An ISO 9001:2008 Certified Institution)
A-4, Paschim Vihar, Main Rohtak Road, New Delhi – 110063

Department of Electronics and communication
Course Exit Survey
2024-25

Subject Name: Control Systems Lab
Subject Code: EEC-355
Semester: 5th

Please rate how well you understood the course (Tick the most appropriate option)
(1 – Poor, 2- Good, 3- Excellent)

EEC-355.1 Do you understand the concept of transfer function for given control system problems and can you plot them on MATLAB

1. ☐ **2.** ☐ **3.** ☐

EEC-355.2 Are you able to apply time domain analysis to analyze the transient response of the systems.

1. ☐ **2.** ☐ **3.** ☐

EEC-355.3 Are you able to analyze and evaluate the stability of control systems with different methods.

1. ☐ **2.** ☐ **3.** ☐

ETEL-355.4 Are you able to design Lead, Lag, and Lead-Lag systems in control systems

1. ☐ **2.** ☐ **3.** ☐

Suggestions to improve the teaching methodology:

Overall how do you rate your understanding of the subject (tick whichever applicable)

1. Below 50%. 2. 50%-70%. 3.70%-90% 4. Above 90%

Name of student

Enrollment number **Signature**