MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE

(UGC AUTONOMOUS)

ANGALLU, MADANAPALLE – 517 325

SIMULATION AND CONTROL
LABORATORY MANUAL
2016-17

STUDENT COPY

DEPARTMENT

OF

ELECTRONICS & COMMUNICATION ENGINEERING

MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE

(UGC AUTONOMUS)

MADANAPALLE – 517 325

SIMULATION AND CONTROL
LABORATORY MANUAL
2016-17

DEPARTMENT

OF

ELECTRONICS & COMMUNICATION ENGINEERING

Lab In-charge HOD

MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE

(UGC AUTONOMOUS)

ELECTRONICS AND COMMUNICATION ENGINEERING

B. Tech. II Year II Semester

14ECE204 SIMULATION AND CONTROL PRACTICALS

Course Prerequisite: 14ECE102 & 108

LIST OF EXPERIMENTS:

SIGNAL EXPERIMENTS:

1. Basic Operations on Matrices
2. Generation of Various signals and Sequences (Periodic and Aperiodic), Such as Unit Impulse, Unit Step, Square, Saw Tooth, Triangular, Sinusoidal, Ramp, sinc function.
3. Operations on Signals and Sequences such as Addition, Multiplication, Scaling, Shifting, Folding, Computation of Energy and Average Power.
4. Finding the Even and Odd Parts of Signal or Sequence and Real and Imaginary Parts of Signal.
5. Convolution between Signals and Sequences.
6. Autocorrelation and Cross correlation between Signals and Sequences.
7. Verification of Linearity and Time Invariance Properties of a Given Continuous / Discrete System.
8. Computation of Unit Sample, Unit Step and Sinusoidal Responses of the given LTI System and verifying its Physical Realizability and Stability Properties.
9. Gibbs Phenomenon.
10. Finding the Fourier Transform of a given Signal and plotting its Magnitude and Phase Spectrum.
11. Waveform Synthesis using Laplace Transform.
12. Locating Zeros and Poles, and plotting the Pole-Zero maps in S-Plane and Z-Plane for the given Transfer Functions.
13. 13. Sampling Theorem Verification.

CONTROL SYSTEM EXPERIMENTS:

1. Transfer Function of DC Machine.
2. Effect of Feedback on DC Servo Motor.
3. Characteristics of AC Servo Motor.
4. Effect of P, PD, PI, PID Controller on a Second Order Systems.
5. Lag and Lead Compensation – Magnitude and Phase Plot.
6. Temperature Controller Using PID.

7. Stability Analysis (Bode, Root Locus, Nyquist) of Linear Time Invariant System Using MATLAB.

Course Outcomes:

Upon successful completion of the course, students will be able to

1. Differentiate between continuous time and discrete time signals.
2. Estimate Time domain response of a system using convolution.
3. Use PID Controller in feedback Systems.
4. Analyze stability of a given Linear Time Invariant System.

TABLE OF CONTENTS

PART A

Signal Experiments

INTRODUCTION

What Is MATLAB?

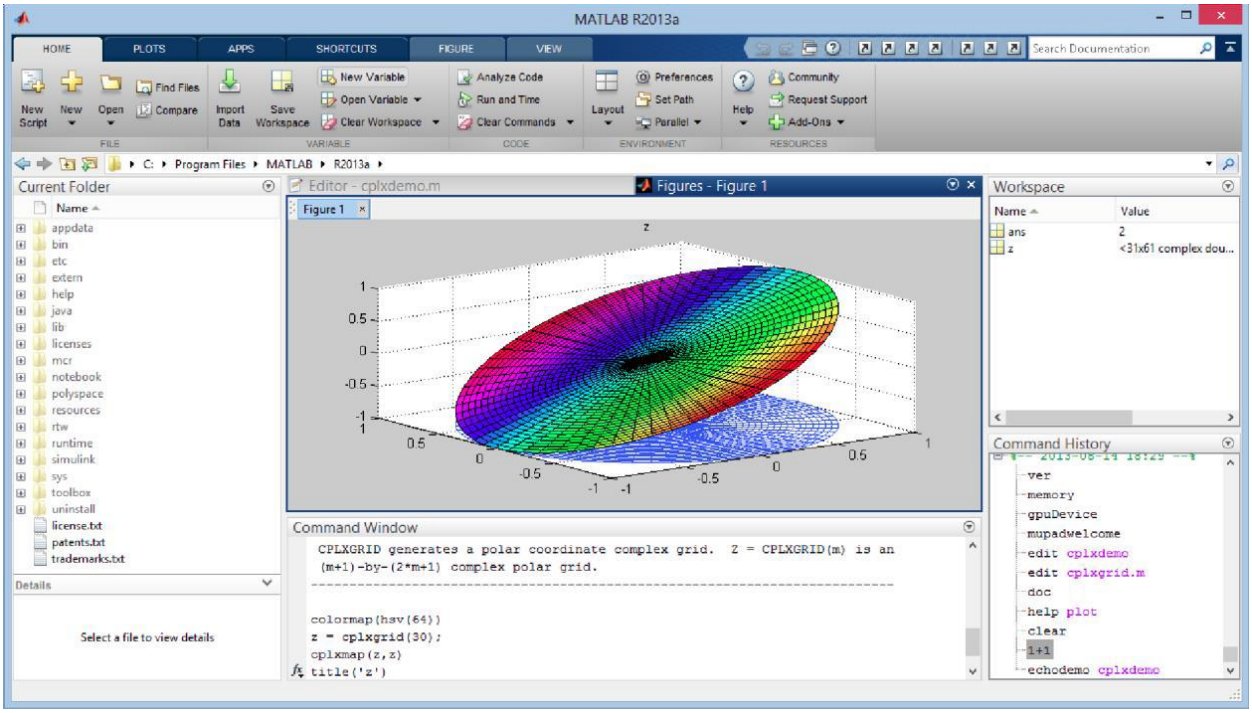
MATLAB is a software package for high performance numerical computation and visualization. The name stands for MATrixLABoratory. A proprietary programming language developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python. The fundamental data-type is the array and the basic building block is the matrix.

In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

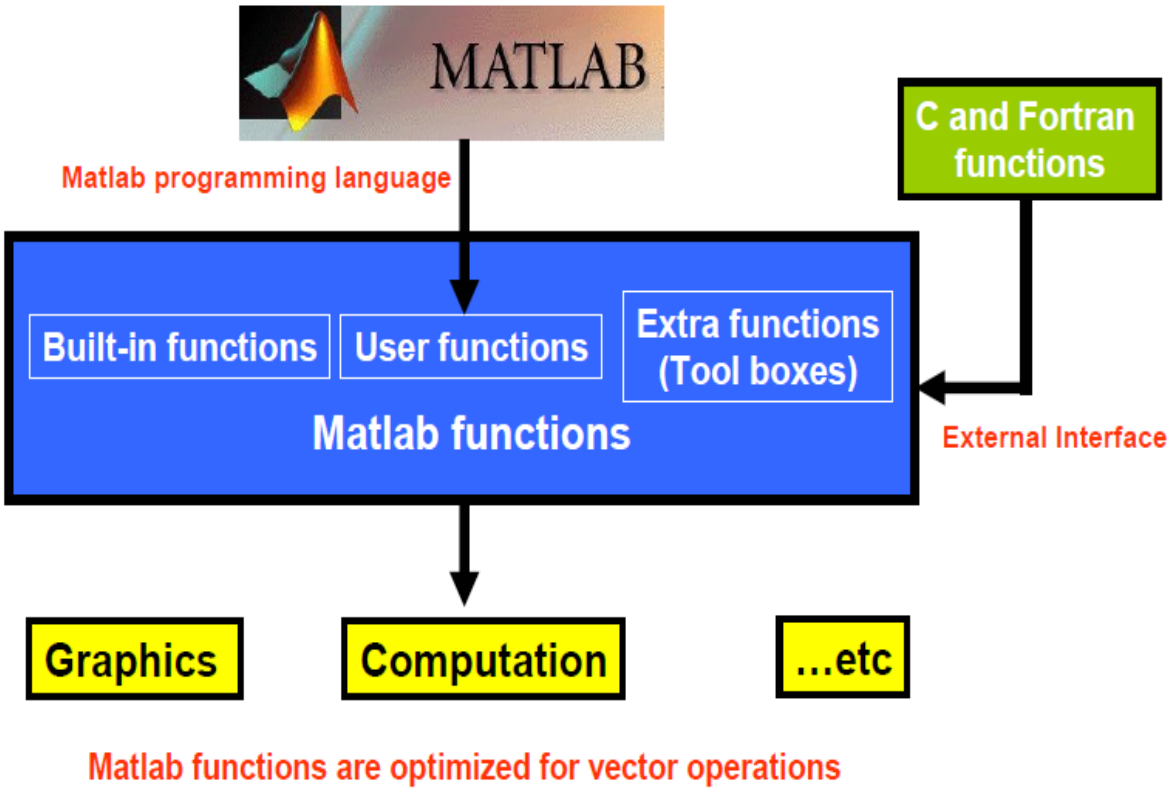
Why MATLAB?

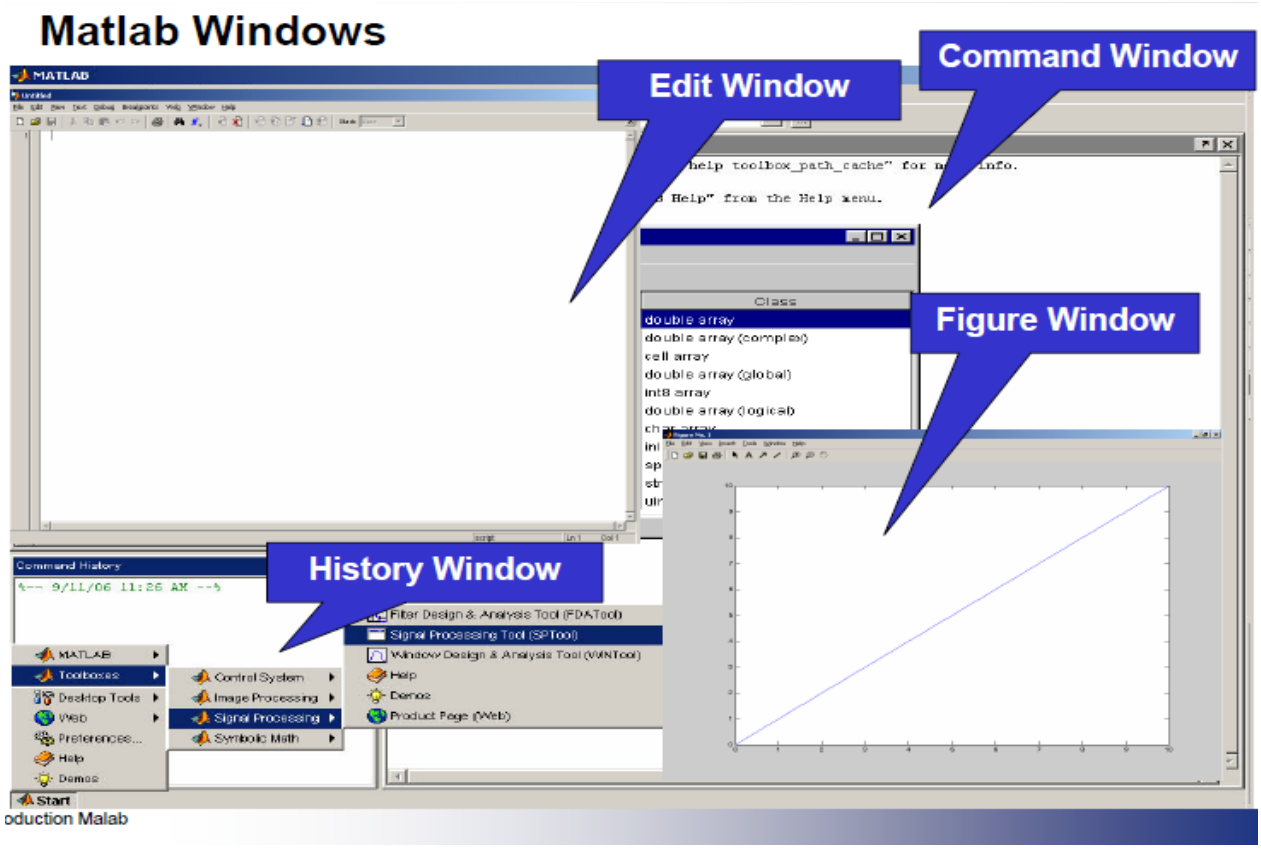
MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide. It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as *toolbox*. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.



MATLAB Main Features





Getting help in MATLAB

You can launch MATLAB help be selecting **Help> MATLAB Help**. From there you can easily go to the MATLAB function reference and the alphabetical list of the available functions. The easiest and fastest way to get help in MATLAB is by using,

The command **help** and The keyword search **look for**

Type help

Brings out a list of categories in which help is organized.

Type help category

e.g. help elfun gives a list of elementary math functions with the full name of each function.

Type help function name

e.g. help sin gives a brief description of the sinusoidal function.

Creating a directory and saving files

There is a default folder called **"work"** where MATLAB saves the files if no other location was specified.

If you need to store the files somewhere else, you have to specify the path to the files or change the working directory of MATLAB to the desired directory using,

The command **path** The command **cd**

Type path	Shows the MATLAB search path
Type cd	Shows the current directory
Type addpath C:\mywork	Adds the directory mywork to the existing path
Type rmpath C:\mywork	removes the directory mywork from the
Type cd C:\mywork	Sets the current directory to C:\mywork

Scripts Files

1. Script files are useful when you have to repeat a set of commands several times.
2. Script files are sequences of any number of commands, including commands calling built-in functions or user functions.
3. Script files can be created using an editor or word processing application, e.g. notepad (Windows) or the M-file editor (part of the standard MATLAB installation under Windows) and saved as M-files, i.e. using the extension .m.
4. To execute a script file, you just have to type the name of the file on the command line without the extension .m.
5. MATLAB executes the commands one by one as if you have typed all the commands stored in the file one by one at the command line.
6. Scripts can operate on existing data in the workspace, i.e. on global variables, and any variables that they create remain in the workspace

Operators and Special Characters

+	Plus; addition operator.
-	Minus; subtraction operator
*	Scalar and matrix multiplication operator.
.*	Array multiplication operator.

^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator
\	Left-division operator.
/	Right-division operator.
.\	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
( )	Parentheses; encloses function arguments and array indices; overrides precedence.
[ ]	Brackets; enclosures array elements.
.	Decimal point.
…	Ellipsis; line-continuation operator.
,	Comma; separates statements and elements in a row.
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.
_	Quote sign and transpose operator.
._	Nonconjugated transpose operator.
=	Assignment (replacement) operator.

Commands for Managing a Session

Clc	Clears Command window.
Clear	Removes variables from memory.
Exist	Checks for existence of file or variable.
Global	Declares variables to be global.
Help	Searches for a help topic.
Lookfor	Searches help entries for a keyword.

Quit	Stops MATLAB.
Who	Lists current variables.
Whos	Lists current variables (long display).

Special Variables and Constants

Ans	Most recent answer.
Eps	Accuracy of floating-point precision.
i,j	The imaginary unit -1.
Inf	Infinity.
NaN	Undefined numerical result (not a number)
Pi	The number p .

System and File Commands

Cd	Changes current directory.
Date	Displays current date.
Delete	Deletes a file.
Diary	Switches on/off diary file recording.
Dir	Lists all files in current directory.
Load	Loads workspace variables from a file.
Path	Displays search path.
Pwd	Displays current directory.
Save	Saves workspace variables in a file.
Type	Displays contents of a file.
What	Lists all MATLAB files in the current directory.
Wkl	read Reads .wk1 spreadsheet file.

Input/Output Commands

Disp	Displays contents of an array or string.
Fscanf	Read formatted data from a file.

Format	Controls screen-display format.
Fprintf	Performs formatted writes to screen or file.
Input	Displays prompts and waits for input.
;	Suppresses screen printing

Array Commands

Cat	Concatenates arrays.
Find	Finds indices of nonzero elements.
Length	Computers number of elements.
Linspace	Creates regularly spaced vector.
Logspace	Creates logarithmically spaced vector.
Max	Returns largest element.
Min	Returns smallest element.
Prod	Product of each column.
Reshape	Change size
Size	Computes array size.
Sort	Sorts each column.
Sum	Sums each column

Special Matrices

Eye	Creates an identity matrix.
Ones	Creates an array of ones.
Zeros	Creates an array of zeros.

Matrix Arithmetic

Cross	Computes cross products.
Dot	Computes dot products.

Matrix Commands for Solving Linear Equations

Det	Computes determinant of an array.

Inv	Computes inverse of a matrix.
Pinv	Computes pseudoinverse of a matrix.
Rank	Computes rank of a matrix.
Rref	Computes reduced row echelon form.

Plotting Commands

Basic xy Plotting Commands

Axis	Sets axis limits.
Fplot	Intelligent plotting of functions.
Grid	Displays gridlines.
Plot	Generates xy plot.
Print	Prints plot or saves plot to a file
Title	Puts text at top of plot.
Xlabel	Adds text label to x-axis.
Ylabel	Adds text label to y-axis.

Plot Enhancement Commands

Axes	Creates axes objects.
Close	Closes the current plot.
close all	Closes all plots.
Figure	Opens a new figure window.
Gtext	Enables label placement by mouse.
Hold	Freezes current plot.
Legend	Legend placement by mouse.
Refresh	Redraws current figure window.
Set	Specifies properties of objects such as axes.
Subplot	Creates plots in subwindows.
Text	Places string in figure.

Specialized Plot Commands

Bar	Creates bar chart.
Loglog	Creates log-log plot.
Polar	Creates polar plot.
Semilogx	Creates semilog plot (logarithmic abscissa).
Semilogy	Creates semilog plot (logarithmic ordinate).
Stairs	Creates stairs pot.
Stem	Creates stem plot

Colors, Symbols and Line Types

Color Symbol Line

y	Yellow		**r**	Red
.	Point		**g**	Green
-	Solid		**b**	Blue
m	Magenta		**c**	Cyan
o	Circle		**k**	Black
:	Dotted		<	triangle (left)
h	Hexagram		**w**	White
x	x-mark		>	triangle (right)
-.	dash dotted		**p**	Pentagram
h	Hexagram		^	triangle (up)
--	Dashed		**v**	triangle (down)
d	Diamond		*****	Star
+	Plus			

Logical and Relational Operators

==	Relational operator: equal to.
~=	Relational operator: not equal to.

<	Relational operator: less than.
<=	Relational operator: less than or equal to.
>	Relational operator: greater than.
>=	Relational operator: greater than or equal to.
&	Logical operator: AND.
|	Logical operator: OR.
~	Logical operator: NOT.
xor	Logical operator: EXCLUSIVE OR.

Exponential and Logarithmic Functions

exp(x)	Exponential; ex.
log(x)	Natural logarithm; ln(x).
log10(x)	Common (base 10) logarithm; log(x)= log10(x).
sqrt(x)	Square root; *x*.

Complex Functions

abs(x)	Absolute value; |x|.
angle(x)	Angle of a complex number x
imag(x)	Imaginary part of a complex number x.
real(x)	Real part of a complex number x.
conj(x)	Complex conjugate of x.

Procedure to Execute programs in MATLAB

1. Start MATLAB R2013a - either from shortcut present on the Desktop or Click START and search for MATLAB.
2. Click on “New Script”.
3. Enter the program.

4. Save the program using a ".m" extension. Note: The file names should not start with a number or any other special character. Pre-existing keywords cannot be used as file names (eg: "sin.m", is not valid).
5. Press "F5" to run the program.
6. Note down the output shown in the Command Window.

Experiment 1:

Basic Operations on Matrices

Aim: Write a MATLAB program to perform some basic operation on matrices such as addition, subtraction, multiplication.

Software Required: MATLAB

Theory:

As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

MATLAB is case-sensitive, which means that ***a + B*** is not the same as ***a + b***. The MATLAB ***prompt (»)*** in command window is where the commands are entered.

Matrices :

1**. Row matrix:** Elements in a row are separated either by using white spaces or commas

$$\boldsymbol{a} = [\mathbf{1} \quad \mathbf{2} \quad \mathbf{3}]$$

2. **Column matrix:** Elements which differ by a column are separated by enter or semicolon.

$$\boldsymbol{b} = [\mathbf{1}; \quad \mathbf{2}; \quad \mathbf{3};]$$

Looking into matrix: a(row, column) allows to look the particular element in the matrix "$\boldsymbol{a}$"

Vectors:

$$\boldsymbol{d} = [\mathbf{0}: \mathbf{7}]$$

"$\boldsymbol{d}$" is a vector or row matrix with first element as ***0*** and last element as ***7*** and increment is by default ***1***. The default increment can be changed (to ***0.1***) by using increment field in between as

$$\boldsymbol{e} = [\mathbf{0}: \mathbf{0}.\mathbf{1}: \mathbf{7}]$$

$\boldsymbol{d}[\mathbf{1}: \mathbf{2}]$allows to look into vector with increment $\boldsymbol{1e}[\mathbf{1}: \mathbf{2}: \mathbf{4}]$look with increment***2***

The Identity Matrix

In this section we will restrict our attention to ***square matrices***; i.e., matrices of dimension ***n*** × ***n***, i.e., matrices having an equal number of rows and columns. Use MATLAB's ***eye*** command to create an identity matrix.

>>**I=eye(3)**

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Indexing

Indexing matrices in MATLAB is similar to the indexing we saw with vectors. The difference is that there is another dimension to access the element in row 2 column 3 of matrix A, enter this command.

>> A=[1 2 3 0;5 -1 0 0;3 -2 5 0]

A =

1 2 3 0

5 -1 0 0

3 -2 5 0

>>A(2,3)

ans =

0

This is indeed the element in row 2, column 3 of matrix A. You can access an entire row with Matlab's colon operator. The command A(2,:) essentially means "row 2 every column" of matrix A.

A(2,:)

ans =

5 -1 0 0

Note that this is the second row of matrix A.Similarly, you can access any column of matrix A. The notation **A(:,2)** is pronounced "every row column 2" of matrix A.

>>A(:,2)

ans =

2

-1

-2

The Transpose of a Matrix

You can take the transpose of a matrix in exactly the same way that you took the transpose of a row or column vector. For example, form a "magic" matrix with the following command.

>> A=magic(4)

A =

16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

You can compute $\mathbf{A}^{\mathbf{T}}$ with the following command.

>> A**.'**

ans =

16 5 9 4

2 11 7 14

3 10 6 15

13 8 12 1

Building Matrices

MATLAB has some powerful capabilities for building new matrices out of one or more matrices and/or vectors. For example, start by building a 2 ×3 matrix of ones.

>> A=ones(2,3)
A =
1 1 1
1 1 1

>> D=zeros(2,3)
D =
0 0 0
0 0 0

Program

Result:

Conclusion:

Experiment 2:

Generation of various signals and sequences (Periodic and Aperiodic).

Such as Unit impulse, Unit step, square, saw tooth, triangular, sinusoidal, ramp, sinc function.

Aim: To generate various periodic and aperiodic signals and sequences such as Unit Impulse, Unit step, Square, Saw Tooth, Triangular, Sinusoidal, Ramp, Sinc functions using MATLAB.

Software required: MATLAB

Theory:

1. **Common Periodic Waveforms:**

The toolbox provides functions for generating widely used periodic waveforms: sawtooth generates a sawtooth wave with peaks at ±1 and a period of 2π. An optional width parameter specifies a fractional multiple of 2π at which the signal's maximum occurs. square generates a square wave with a period of 2π. An optional parameter specifies duty cycle, the percent of the period for which the signal is positive.

2. **Common Aperiodic Waveforms:**

The toolbox also provides functions for generating several widely used aperiodic waveforms: gauspuls generates a Gaussian-modulated sinusoidal pulse with a specified time, center frequency, and fractional bandwidth. Optional parameters return in-phase and quadrature pulses, the RF signal envelope, and the cutoff time for the trailing pulse envelope. chirp generates a linear, log, or quadratic swept-frequency cosine signal. An optional parameter specifies alternative sweep methods. An optional parameter phi allows initial phase to be specified in degrees.

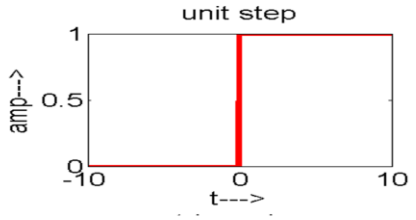
3. **The pulse train Function:**

The pulse train function generates pulse trains from either continuous or sampled prototype pulses. The following example generates a pulse train consisting of the sum of multiple delayed interpolations of a Gaussian pulse.

4. **Unit Step Function**

The unit step function and the impulse function are considered to be fundamental functions in engineering, and it is strongly recommended that the reader becomes very familiar with both of these functions.

The unit step function, also known as the Heaviside function, is defined as such:



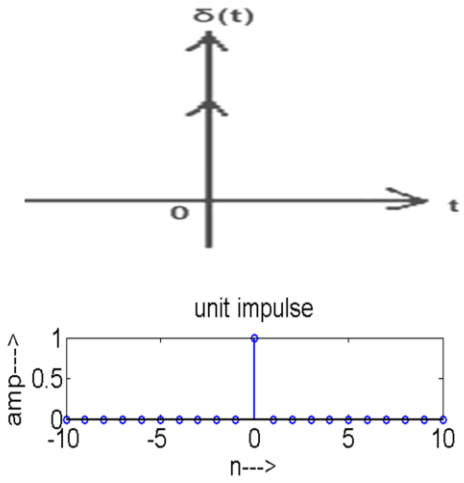
$$u(t) = \begin{cases} 0 & if\ t < 0 \\ 1 & if\ t \geq 0 \end{cases}$$

$$u[n] = \begin{cases} 0 & if\ n < 0 \\ 1 & if\ n \geq 0 \end{cases}$$

- It is used as best test signal.
- Area under unit step function is unity.

5. Unit Impulse Function

Impulse function is denoted by δ.

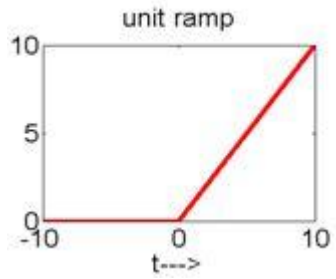


$$\delta(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases}$$

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

6. Ramp Signal

Ramp signal is denoted by r(t),



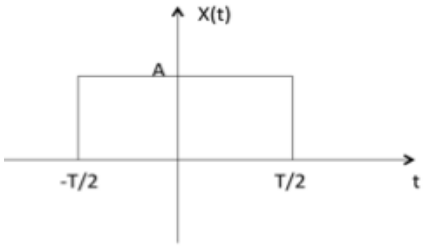
$$r(t) = \begin{cases} t\ ,\ t \geq 0 \\ 0\ ,\ t < 0 \end{cases}$$

$$r(n) = \begin{cases} n\ ,\ n \geq 0 \\ 0\ ,\ n < 0 \end{cases}$$

Area under unit ramp is unity.

7. Rectangular Signal

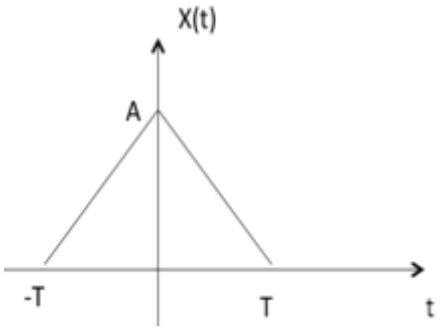
Let it be denoted as x(t)



$$\boldsymbol{x(t) = \begin{cases} A\ rect\left[\frac{r}{T}\right] & |t| \leq \frac{T}{2} \\ 0 & otherwise \end{cases}}$$

8. Triangular Signal

Let it be denoted as x(t)

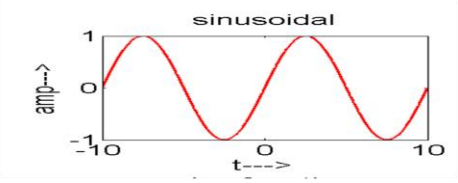


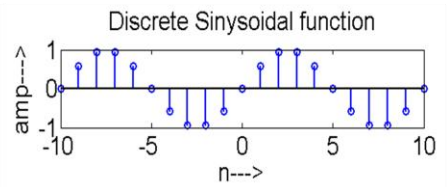
$$\boldsymbol{x(t) = \begin{cases} A\left[1 - \frac{|t|}{T}\right] & |t| \leq \frac{T}{2} \\ 0 & otherwise \end{cases}}$$

9. Sinusoidal Signal

Sinusoidal signal is in the form of

$$x(t) = A\ cos(w_o \pm \emptyset)\ or\ A\ sin(w_o \pm \emptyset)$$



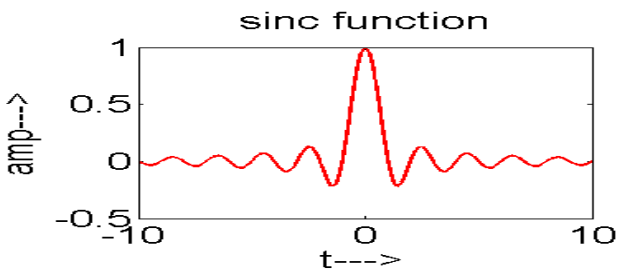


$$Where\ T_O = \frac{2\pi}{w_o}$$

10. Sinc Function

It is denoted as sinc(t) and it is defined as sinc

$$sinc(t) = \frac{\sin \pi t}{\pi t}$$



Program:

Result:-

Continuous:

Discrete:

Conclusion:

Experiment 3:

Operations on signals and sequences such as addition, multiplication, scaling, shifting, folding computation of energy and average power

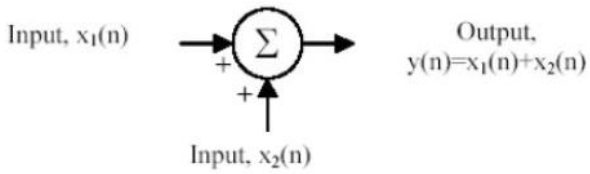
Aim: To perform operations on signals and sequences such as Addition, Multiplication, Scaling, Shifting, Folding, Computation of Energy and Average Power using MATLAB.

Software Required: MATLAB

Theory:

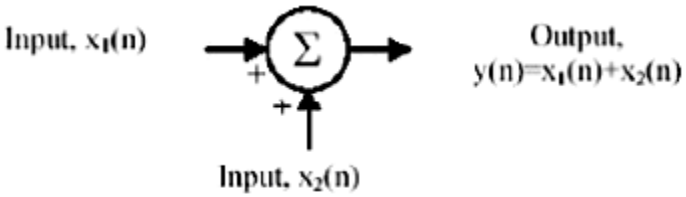
Time shifting: y(t)=x(t-T)The effect that a time shift has on the appearance of a signal. If T is a positive number, the time shifted signal, x (t -T) gets shifted to the right, otherwise it gets shifted left.

Signal Shifting and Delay:

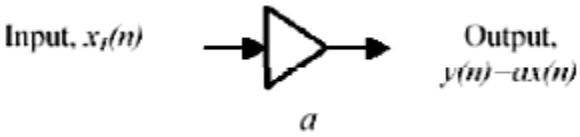


Time reversal/folding: Y(t)=y(-t)

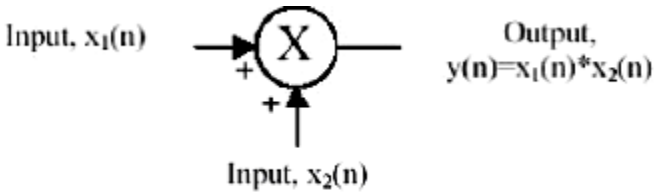
Addition: any two signals can be added to form a third signal, z (t) = x (t) + y (t)



Signal Amplification/Attenuation: For input x(n), output is y(n)=a*x(n)



Signal Multiplication/Division: For inputs, x(n) and y(n), outputs are z(n)=x(n)*y(n) and x(n)/y(n)



Signal Folding: y(n)={x(-n)} ; y=fliplr(x); n=-fliplr(n);

Signal energy and power: The energy of a signal is the area under the squared signal.

The Energy of continuous signal ***x(t)*** can be calculated using equation 1

The power of continuous signal ***x(t)*** can be calculated using equation 2

The Energy of discrete signal ***x[n]*** can be calculated using equation 3

The power of discrete signal ***x[n]*** can be calculated using equation 4

Continuous	**Discrete**
$\boldsymbol{E = \int_{-\infty}^{\infty} \lvert x(t)^2 \rvert dt}$ **(1)**	$\boldsymbol{E = \sum_{n=-\infty}^{\infty} \lvert x(n) \rvert^2}$ **(3)**
$\boldsymbol{P = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \lvert x(t) \rvert^2 \, dt}$ **(2)**	$\boldsymbol{P = \lim_{N\to\infty} \left(\frac{1}{2N+1} \sum_{n=-N}^{N} \lvert x(n) \rvert^2 \right)}$ **(4)**

Power is a time average of energy (energy per unit time). This is useful when the energy of the signal goes to infinity.

Note:

1. "Energy signals" have finite energy and zero power.
2. "Power signals" have finite or non-zero power and infinite energy.

Program:

Result:

Conclusion

Experiment 4:

Finding the even and odd parts of signals or sequences and real and imaginary parts of signals.

Aim: To Find the Even and Odd Parts of Signal or Sequence and Real and Imaginary Parts of signals and sequences using MATLAB.

Software required: MATLAB

Theory:

Even and Odd Signal

One of characteristics of signal is symmetry that may be useful for signal analysis. Even signals are symmetric around vertical axis, and Odd signals are symmetric about origin.

Even Signal: A signal is referred to as an even if it is identical to its time-reversed counterparts; x(t) = x(-t).

Odd Signal: A signal is odd if x(t) = -x(-t).

An odd signal must be 0 at t=0, in other words, odd signal passes the origin. Using the definition of even and odd signal, any signal may be decomposed into a sum of its even part, $\boldsymbol{x_e(t)}$ and its odd part, $\boldsymbol{x_o(t)}$ as follows:

Continuous	**Discrete**
$\boldsymbol{x(t) = x_e(t) + x_o(t)}$	$\boldsymbol{x[n] = x_e[n] + x_o[n]}$
$\boldsymbol{x_e(t) = \frac{1}{2}\{x(t) + x(-t)\}}$	$\boldsymbol{x_e[n] = \frac{1}{2}\{x[n] + x[-n]\}}$
$\boldsymbol{x_o(t) = \frac{1}{2}\{x(t) - x(-t)\}}$	$\boldsymbol{x_o[n] = \frac{1}{2}\{x[n] - x[-n]\}}$

It is an important fact because it is relative concept of Fourier series. In Fourier series, a periodic signal can be broken into a sum of sine and cosine signals. Notice that sine function is odd signal and cosine function is even signal.

Program:

Result:

Even and odd part of sequence and signal

Real and imaginary sequence/ signal

% SOLVE

$$x(t) = \begin{Bmatrix} t & 0 < t < 1 \\ 1 & 1: t < 2 \\ 0 & else\ where \end{Bmatrix}$$

Conclusion

Experiment 5:

Convolution between signals and sequences

Aim: To perform convolution of two signals and sequences using MATLAB.

Software Required: MATLAB

Theory:

Convolution is an integral concatenation of two signals. It is used for the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system. Note that convolving two signals is equivalent to multiplying the Fourier transform of the two signals. Linear Convolution involves the following operations.

1. Folding
2. Multiplication
3. Addition
4. Shifting

Discrete	Continuous
$\boldsymbol{y[n] = \sum_{k=-\infty} x[k]h[n-k]}$	$\boldsymbol{y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau = x(t) * h(t)}$

x[n]=Input signal samples

h[n-k]=impulse response co-efficient

y[n]=convolution output

n=Number of input samples

h=Number of impulse response coefficient

If a continuous-time system is both linear and time-invariant, then the output y(t) is related to the input x(t) by a convolution integral.

Program:

Result:

Conclusion:

Experiment 6:
Autocorrelation and Cross correlation between signals and sequences

Aim: To perform autocorrelation and cross correlation of signals and sequences using MATLAB.

Software Required: MATLAB

Theory:

In Signal processing, when the autocorrelation function is normalized by mean and variance, it is sometimes referred to as the autocorrelation coefficient. Given a signal f(t), the continuous autocorrelation Rff(T) is most often defined as the continuous cross-correlation integral of f(t)with itself, at lag T

$$R_{ff}(\tau) = (f(t) * \overline{f}(-t))(\tau) = \int_{-\infty}^{\infty} f(t+\tau)\overline{f}(t)\,\mathrm{d}t = \int_{-\infty}^{\infty} f(t)\overline{f}(t-\tau)\,\mathrm{d}t$$

The discrete autocorrelation Rat lag j for a discrete signal x(n) is

$$R_{xx}(j) = \sum_{n} x_n\,\overline{x}_{n-j}.$$

In signal processing, cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. This is also known as a sliding dot product or sliding inner-product. It is commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, single particle analysis, electron tomography averaging, cryptanalysis, and neurophysiology.

For continuous functions f and g, the cross-correlation is defined as:

$$(f \star g)(\tau) \ \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f^*(t)\ g(t+\tau)\,dt,$$

where f* denotes the complex conjugate of f and t is the time lag. Similarly, for discrete functions, the cross-correlation is defined as**:**

$$(f \star g)[n] \ \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f^*[m]\ g[m+n].$$

Program:

Result:

Continuous signal

Discrete sequence

Conclusion:

Experiment 7:

Verification of Linearity and Time Invariance Properties of a Given Continuous / Discrete System

Aim: To compute linearity and time invariance properties of a given continuous/discrete system using MATLAB.

Software required: MATLAB

Theory:

Linear system:

A system is said to be linear if it satisfies the principle of superposition. Let us consider a system with two inputs $X_1(t)$ and $X_2(t)$ then the superposition will be defined as follows

$$f(aX_1(t)+bX_2(t)) = af(X_1(t))+bf(X_2(t))$$

where a and b are the weighs added to the inputs and $f(X(t)) = y(t)$ is the response of the continuous time system to the input $X(t)$. The same is also true for discrete time system and is given as

$$f(aX_1(n)+bX_2(n)) = af(X_1(n))+bf(X_2(n))$$

Linear Time Invariant system(LTI system):

A linear System in which an input-output pair is invariant to a shift, t_0 in time is called a Linear Time Invariant System. i.e. in continuous time if $y(t)$ is the output to a corresponding input $x(t)$ to a system then for a time invariant system $y(t-t_0)$ will be the output to a input $x(t-t_0)$. Similarly in discrete domain if $y[n]$ is the output of a system to an input $x[n]$ then for a time invariant system $y[n-n_0]$ will be the output to an input of $x[n-n_0]$. Output of an LTI system is nothing but the convolution sum of system and its applied input.

Program to check linearity of a system

Program for LTI system:

Results:

For linearity:

discrete:

Continuous:

For time invariance:

Discrete:

Conclusion:

Experiment No.:8
Computation of Unit Sample, Unit Step and Sinusoidal Responses of the given LTI System and verifying its Physical Realizability and Stability Properties

Aim: To compute the Unit Sample, Unit Step and Sinusoidal Responses of the given LTI System and verifying its Physical Realizability and Stability Properties using MATLAB.

Software Required: MATLAB

Theory:

A discrete time system performs an operation on an input signal based on predefined criteria to produce a modified output signal. The input signal x(n) is the system excitation, and y(n) is the system response. If the input to the system is unit impulse i.e. x(n) = δ(n) then the output of the system is known as impulse response denoted by h(n) where, h(n) = T[δ(n)]

we know that any arbitrary sequence x(n) can be represented as a weighted sum of discrete impulses. Now the system response is given by

$$y[n] = T[x[n]] = T[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)]$$

For a linear system this reduces to

$$y[n] = \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)]$$

In MATLAB, the responses of Unit Sample, Unit Step and Sinusoidal can be computed using the command, *filter*. and the syntax can be given as

y = filter(b,a,x)

where y filters the input data, x, using a rational transfer function defined by the numerator and denominator coefficients b and a, respectively.

The input-output description of the filter operation on a vector in the Z-transform domain is a rational transfer function. A rational transfer function is of the form,

$$H(Z)= \frac{\sum_{k=0}^{M} b_k X(n-k)}{\sum_{k=1}^{N} a_k X(n-k)}$$

$$H(z)= \frac{b_0+b_1Z^{-1}+b_2Z^{-2}+\ldots\ldots\ldots\ldots b_{N-1}Z^{(N-1)}+b_NZ^{-N}}{1+ a_1Z^{-1}+a_2Z^{-2}+\ldots\ldots\ldots\ldots a_{N-1}Z^{(N-1)}+a_NZ^{-N}}$$

In MATLAB, *zplane*, checks for the stability of the given system. The syntax of which can be written as zplane(b,a) where b and a are row vectors, first uses roots to find the zeros and poles of the transfer function represented by numerator coefficients b and denominator coefficients a.

Program:

Result:

Conclusion:

Experiment No.:9

Gibbs phenomenon

Aim: To verify Gibbs Phenomenon using MATLAB.

Software Required: MATLAB

Theory:

Fourier series can be used to represent extremely large class of periodic signals. A periodic function f(t) can be expressed in the form of a trigonometric series as

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos n\omega_0 t + \sum_{n=1}^{\infty} b_n \sin n\omega_0 t$$

where,

$$a_0 = \frac{1}{T}\int_0^T f(t)$$

$$a_n = \frac{2}{T}\int_0^T f(t)\cos n\omega_0 t\, dt$$

$$b_n = \frac{2}{T}\int_0^T f(t)\sin n\omega_0 t\, dt$$

Similarly a periodic signal can be represented in exponential flourier series as

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t}$$

where,

$$c_n = \frac{1}{T}\int_0^T f(t) e^{-jn\omega_0 t}\, dt$$

Gibb's Phenomenon:

The peculiar manner in which the Fourier series of a piecewise continuously differentiable periodic function behaves at a jump discontinuity: the nth partial sum of the Fourier series has large oscillations near the jump, which might increase the maximum of the partial sum above that of the function itself. The overshoot does not die out as the frequency increases, but approaches a finite limit.

The Gibbs phenomenon involves both the factthat Fourier sums overshoot at a jump discontinuity, and that this overshoot does not die out as the frequency increases. The best known version of the Gibbs phenomenon is the overshoot that arises when a discontinuous function is represented by a truncated set of Fourier expansion terms. The situation is similar

if the truncated Fourier expansion is instead obtained by means of interpolation on an equispaced grid.

Program:

Result:

Conclusion:

Experiment 10:

Finding the FT of a signal and plotting its Magnitude and Phase spectrum

Aim: To obtain Fourier Transform and Inverse Fourier Transform of a given signal and to plot its Magnitude and Phase Spectra.

Software Required: MATLAB

Theory:-

Continuous Fourier transform:

Let a signal $f(t)$ have a period of T and if $T \to \infty$ then the repetition period becomes infinity and hence the wave $f(t)$ becomes non-periodic. We know that a periodic signal $f(t)$ can be represented using Fourier series as follows

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t}$$

where,

$$c_n = \frac{1}{T}\int_0^T f(t)\, e^{-jn\omega_0 t}\, dt$$

if $T \to \infty$ then

$$\omega_0 = \frac{2\Pi}{T} \to d\omega$$

$$or\, 1/T = \omega_0 / 2\Pi \to d\omega / 2\Pi$$

The nth harmonic in Fourier series $n\omega_0 \to nd\omega$ and it can be written as $n\omega_0 \to \omega$. In the limit $\sum$ leads to an integral so the signal/ wave $f(t)$ can be represented as

$$f(t) = \int_{-\infty}^{\infty} \frac{d\omega}{2\Pi}\left[\int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt\right] e^{j\omega t}$$

In the above equation the quantity in the square bracket is a function of frequency and is denoted as $F(j\omega)$ and is the Fourier transform of the signal $f(t)$. So the Fourier transform of a continuous signal $f(t)$ is given as

$$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

And its inverse Fourier transform is given as

$$f(t) = \frac{1}{2\Pi}\int_{-\infty}^{\infty} F(j\omega) e^{j\omega t} d\omega$$

Discrete time Fourier transform:

Similarly a we can find the Fourier transform of a discrete time signal x[n] as follows

$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$ and inverse of it is given as

$$x[n] = \frac{1}{2\Pi}\int_{2\Pi} X(e^{j\omega})e^{j\omega n}d\omega$$

Program:

Result:

Conclusion:

Experiment 11:

Waveform Synthesis using Laplace Transform

Aim: To perform waveform synthesis using Laplace Transform of a given signal

Software required: MATLAB

Theory:-

Bilateral Laplace transform:

When one says "the Laplace transform" without qualification, the unilateral or one-sided transform is normally intended. The Laplace transform can be alternatively defined as the bilateral Laplace transform or two-sided Laplace transform by extending the limits of integration to be the entire real axis. If that is done the common unilateral transform simply becomes a special case of the bilateral transform where the definition of the function being transformed is multiplied by the Heaviside step function.

The bilateral Laplace transform is defined as follows:

$$\boldsymbol{F(s) = L\{f(t)\} = \int_{-\infty}^{\infty} e^{-st} f(t) dt}$$

Inverse Laplace transform

The inverse Laplace transform is given by the following complex integral, which is known by various names (the Bromwich integral, the Fourier-Mellin integral, and Mellin's inverse formula):

$$\boldsymbol{f(t) = L^{-1}\{F(s)\} = \frac{1}{2\pi i} \lim_{T\to\infty} \int_{\gamma-iT}^{\gamma+iT} e^{st} F(s) ds}$$

Program:-

Result:-

Conclusion:-

Experiment 12:

Locating Zeros and Poles and plotting the Pole-Zero maps in S-plane and Z-plane for the given transfer function

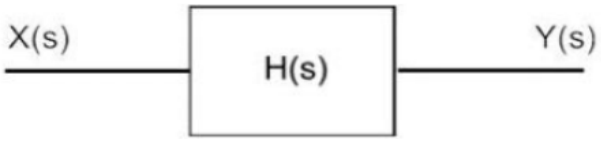
Aim: -To locating the zeros and poles and plotting the pole zero maps in s-plane and z-plane for the given transfer function.

Software Required: MATLAB

Theory:- A Transfer Function is the ratio of the output of a system to the input of a system, in the Laplace domain considering its initial conditions to be zero. If we have an input function of X(s), and an output function Y(s), we define the transfer function H(s) to be:

$$\boldsymbol{H(s) = \frac{Y(s)}{X(s)}}$$

transfer function is the Laplace transform of a system‟ s impulse response.



$$\boldsymbol{F(s) = L\{f(t)\} = \int_{-\infty}^{\infty} e^{-st} f(t) dt}$$

Given a continuous-time transfer function in the Laplace domain, *H*(*s*) or a discrete-time one in the Z-domain, *H*(*z*), a zero is any value of *s* or *z* such that the transfer function is zero, and a pole is any value of *s* or *z* such that the transfer function is infinite.

***Zeros*:**1. The value(s) for *z* where the *numerator* of the transfer function equals zero2. The complex frequencies that make the overall gain of the filter transfer function zero.

Poles: 1. The value(s) for *z* where the *denominator* of the transfer function equals zero2. The complex frequencies that make the overall gain of the filter transfer function infinite.

Z-transforms

the Z-transform converts a discrete time-domain signal, which is a sequence of real or complex numbers, into a complex frequency-domain representation. The Z-transform, like many other integral transforms, can be defined as either a one-sided or two-sided transform.

Bilateral Z-transform

The bilateral or two-sided Z-transform of a discrete-time signal x[n] is the function X(z) defined as

$$X(z) = Z\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

Unilateral Z-transform

Alternatively, in cases where x[n] is defined only for n ≥ 0, the single-sided or unilateral Z-transform is defined as

$$X(z) = Z\{x[n]\} = \sum_{n=0}^{\infty} x[n]z^{-n}$$

In signal processing, this definition is used when the signal is causal

$$Wherez = r.e^{jw}$$

$$X(z) = \frac{P(z)}{Q(z)}$$

The roots of the equation P(z) = 0 correspond to the 'zeros' of X(z)

The roots of the equation Q(z) = 0 correspond to the 'poles' of X(z)

Program:

Result:

S-Plane

Z-plane

Conclusion

Experiment 13:

Sampling Theorem Verification

Aim: -To Demonstrate Sampling Theorem and aliasing effect using MATLAB.

Software Required: MATLAB

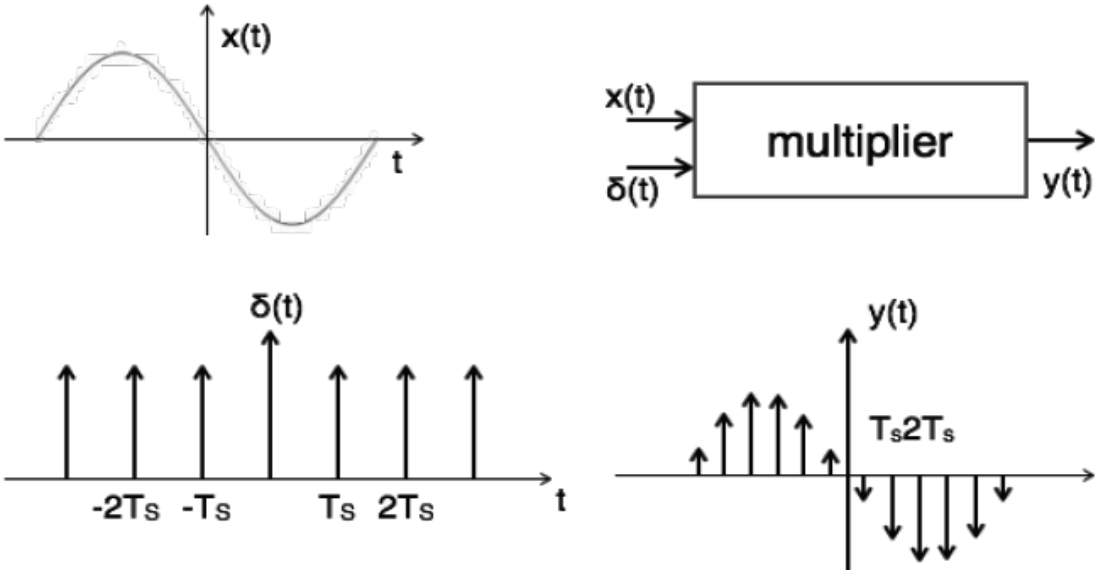
Theory:-

Sampling Theorem: Statement: A continuous time signal can be represented in its samples and can be recovered back when sampling frequency f_s is greater than or equal to the twice the highest frequency component of message signal. i.e.

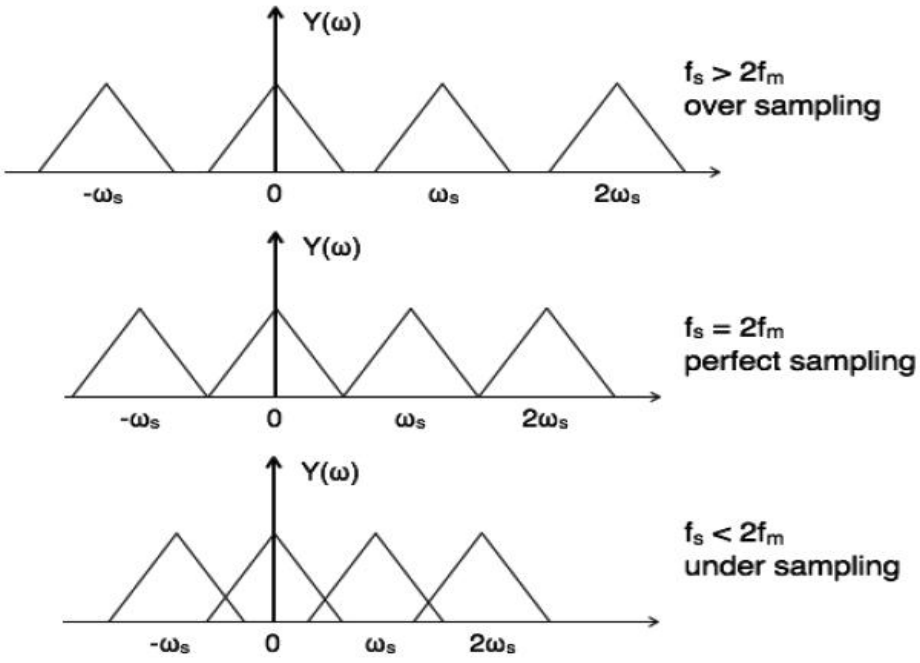
$$f_s \leq 2f_m$$

Proof: Consider a continuous time signal x(t). The spectrum of x(t) is a band limited to f_m Hz i.e. the spectrum of x(t) is zero for $|\omega|>\omega_m$.

Sampling of input signal x(t) can be obtained by multiplying x(t) with an impulse train δ(t) of period T_s. The output of multiplier is a discrete signal called sampled signal which is represented withy(t) in the following diagrams



Possibility of sampled frequency spectrum with different conditions is given by the following diagrams:



Aliasing Effect

The overlapped region in case of under sampling represents aliasing effect, which can be removed by

- considering $f_s > 2f_m$
- By using anti aliasing filters

Program:

Result:

Conclusion:

Exercise Problems:-

1. a. Perform multiplication between the matrices manually and verify the same using by writing script. A= $\begin{bmatrix} 4 & 6 \\ 1 & 9 \end{bmatrix}$ B=$\begin{bmatrix} 2 & -3 & 1 \\ -2 & 0 & 5 \end{bmatrix}$

   b. Explain test signals and generate them by writing MATLAB script.
2. Generate saw tooth, triangular wave with period of 2 by writing MATLAB script and draw manually.
3. Perform addition, subtraction, multiplication, time scaling, and magnitude scaling, for the following sequences theoretically and verify the result using MATLAB.

   X={1,2,3 ,4,5} y={3, 4, 5, 6, 7}
4. a. Calculate energy of the signal and verify by writing MATLAB script.

   X={3 4 5 6 7 8 9}

   b. find the even and odd parts of the signal x=[1,1,1,1,-1,1-1,1] theoretically and verify using MATLAB script.
5. Perform graphical convolution between the sequences x=[1,2,3] and h=[1 1 1] and verify using MATLAB.
6. a. Perform correlation between the sequences x=[1,2,3] and h=[1 1 1] and verify using MATLAB.

   b. Find the real and imaginary parts of the signal x=[1+j3 ,2+j4,3-j5,4+j6 ,-5-j7,6+j8, -7+j9] theoretically and verify using MATLAB script.
7. Verify linearity and time invariance of the following system y(n)= n*x(n) theoretically and verify the result using MATLAB.
8. Verify linearity and time invariance of the following system y(n)= cos(x(n)) theoretically and verify the result using MATLAB.
9. Find unit step and impulse response of the following system $H(z) = \frac{1}{1-\frac{1}{4}z^{-1}}$ and verify the result using MATLAB.
10. Calculate Fourier sires coefficients of the square wave with period 3 and observe Gibbs phenomenon by considering 10^{th},15^{th} harmonics.
11. Find Fourier transform of the signal x(t)=exp(-3*t)u(t) and its spectra manually verify the results using MATLAB

12. Write MATLAB script to find the inverse Laplace transform of and $H(s)=\frac{1}{(s+1)(s+2)}$ plot the poles and zeros and comment on stability.
13. Write MATLAB script to find the inverse Laplace transform of and $H(s)=\frac{s}{(s^2+s+1)}$ plot the poles and zeros and comment on stability.
14. a. Calculate poles and zeros of the transfer function $H(s)=\frac{s+1}{(s^2+2s+2)}$ and plot them using MATLAB and comment on stability.

    b. Calculate poles and zeros of the transfer function $H(z)=\left(\frac{1+\frac{1}{4}z^{-1}}{1+\frac{1}{2}z^{-1}}\right)\left(\frac{1-2z^{-1}}{1-\frac{1}{4}z^{-1}}\right)$ and plot them using MATLAB and comment on stability.
15. Write Nyquist theorem and verify sampling theorem using MATLAB.
16. Find transfer function of an armature controlled DC motor using following specifications. J=3.2284E-6, B=3.577E-6, K=0.0274, L=275E-6, R=4
17. Find step response of control system with transfer function $H(s)=\frac{100}{s^2+12s+100}$ and note the time domain specification, and note the effects in time domain specifications when used P, PI and PID controllers with K_p=5, Ti=2, Td=3.
18. Draw the bode plot manually for open loop transfer function $G(s)=\frac{20}{s(1+3s)(1+4s)}$. Find gain margin phase margin, phase crossover frequency, gain margin, gain crossover frequency and comment on stability. Verify all by writing MATLAB script.
19. Draw the bode plot manually for open loop transfer function $G(s)=\frac{5(1+2s)}{(1+4s)(1+0.25s)}$. Find gain margin phase margin, phase crossover frequency, gain margin, gain crossover frequency and comment on stability. Verify all by writing MATLAB script.
20. Draw the Nyquist plot for the system with open loop transfer function is $G(s)H(s)=\frac{40}{s(s+2)(s+10)}$ manually and write MATLAB script to plot Nyquist plot, and comment on stability.

21. Sketch the root locus of the system with open loop transfer function is $G(s) = \frac{K}{s(s+2)(s+4)}$. Find the range of K so that system is stable manually and verify the same using MATLAB.
22. The open loop transfer function of certain unity feedback control system is given by $G(s) = \frac{K}{s(s+4)(s+80)}$. It is desired to have the phase margin to be at least 33^0 and velocity error constant K_v=30 sec^{-1}. Design a lag series compensator manually and verify the result using MATLAB.
23. Design a phase lead compensator for a unity feedback control system is given by $G(s) = \frac{K}{s(s+1)}$ to satisfy the following specifications i) the phase margin of the system ≥ 45^0ii) steady state error for unit ramp input ≤ 1/15.
24. Consider a control system represented by $G(s) = \frac{K}{s(s+2)(s+30)}$ Design a phase-lag compensator such that the following specification are met: Steady state error for ramp input ≤ 0.05, Phase Margin ≥ 45^0

PART- B

Control System Experiments

Experiment no:1

Transfer Function Of A Dc Machine

Aim: To conduct a test on dc motor and to determine the transfer function of a armature controlled DC motor by conduction,

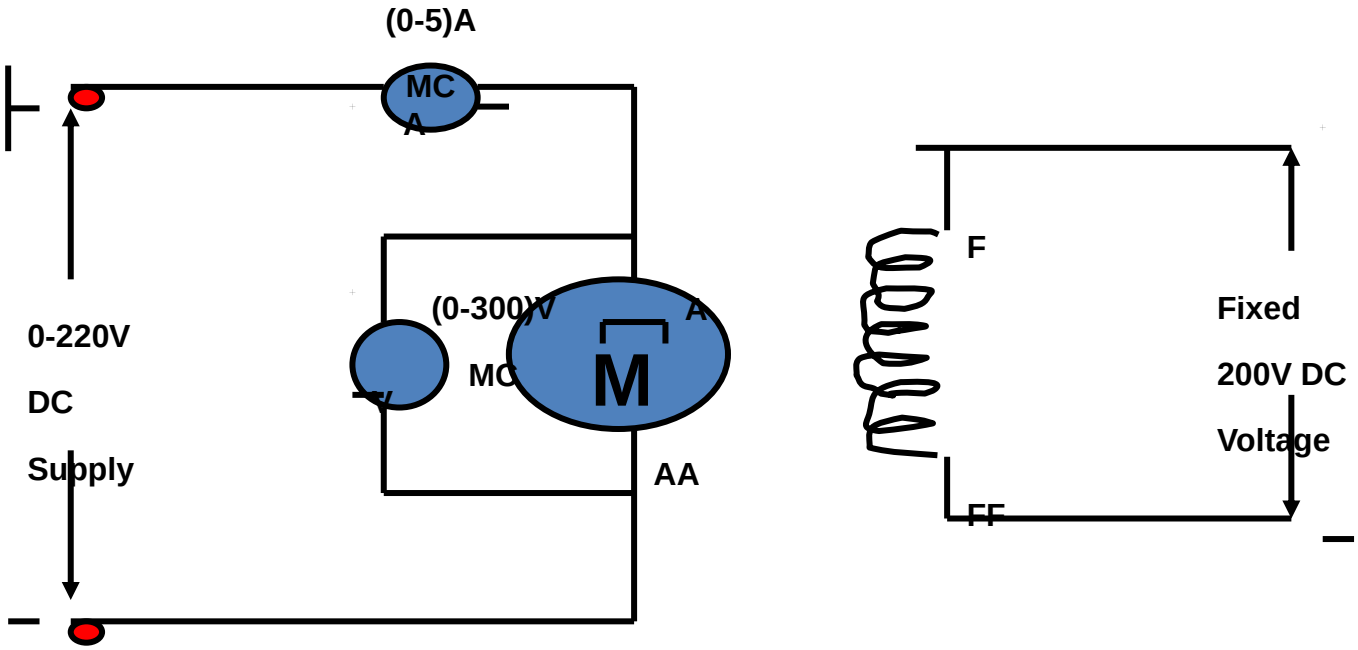
(i) load test on dc motor.
(ii) Speed control by armature voltage control**.**

Apparatus required:

1. DC Servo motor control system kit.
2. DC Voltmeter (0-15) V MC.
3. DC Ammeter (0-2) A MC.
4. Multimeter.
5. Patch chords.

Circuit Diagrams:

Load Test:

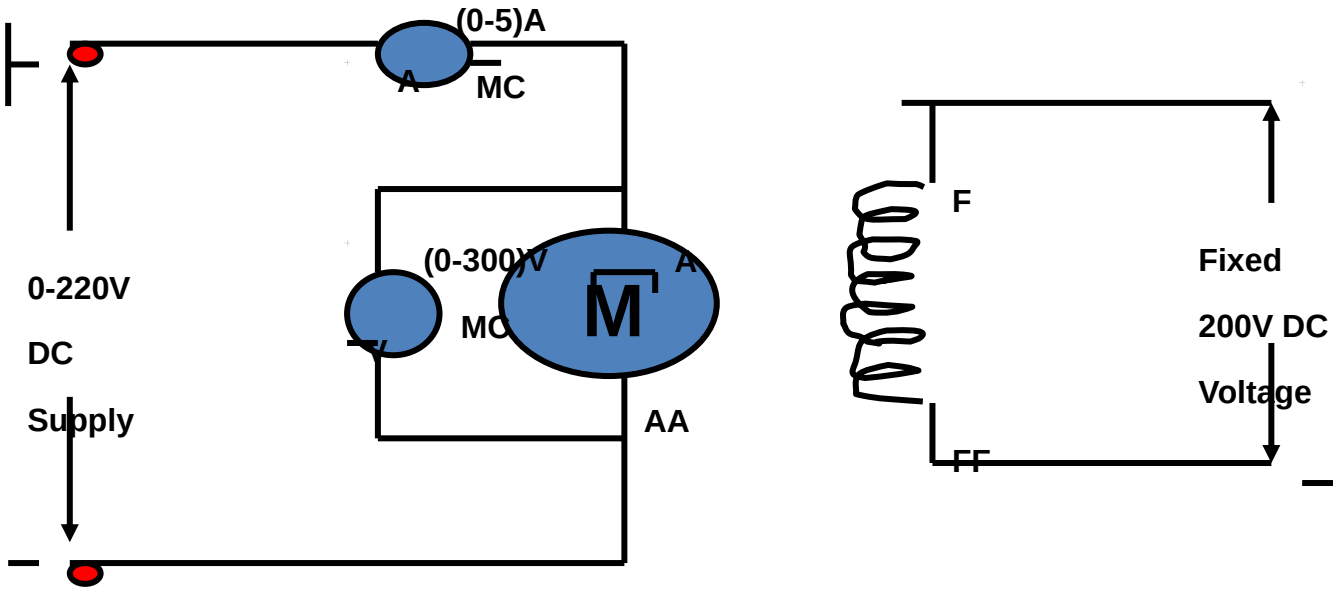


Procedure:

To Find toque constant K_T .

1. Connect the circuit as per circuit diagram.
2. Switch on the field with socket provided on the back panel.
3. Then Switch on the MCB after connecting AC 3-Phase core main cable to phase and neutral of 1-phase supply.
4. Keep the armature voltage pot in minimum position and switch on the shaft switch.
5. Vary the pot ,so that armature voltage as gradually increased which can be mentioned on the volt meter and bring the motor to its rated speed.
6. Apply load in steps and for each load note down the armature current ,spring balance readings and speed.
7. Tabulate the readings for each step.
8. Draw the graph b/w torque and armature current and get the torque constant k_t from the slope obtained from the graph.

Armature Voltage Control:

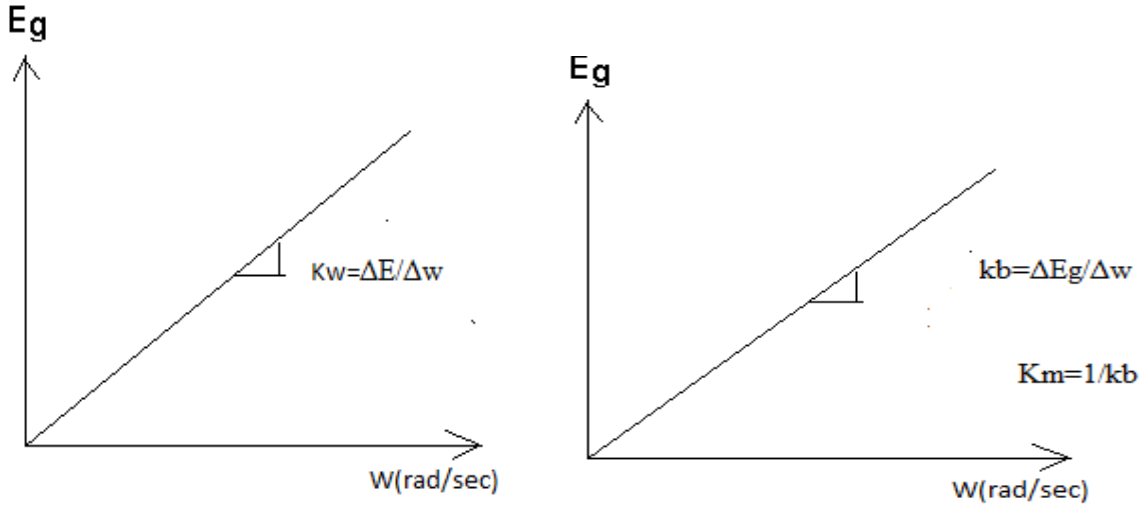


Armature Voltage Control Method.

To find back EMF constant k_b.

1. Make the connection as per the circuit diagram.
2. Keep the field on by plugging 3 core power and cord to socket provided on the back panel of the control unit.
3. Note down the corresponding armature current ,speed for various armature voltage.
4. Tabulate the reading as per tabular column.
5. Plot the graph between Back EMF and Speed and deduce the back EMF constant k_b.

Expected Graphs:



Calculations:

Motor back emf $E_b = V - I_aR_a$

W=2ΠN/60 rad/sec

From the graph E_bVs W

$K_b=\Delta E_b / \Delta w$

$K_m=1/K_b$

K_m= motor gain constant

G= Rated output voltage of the chopper / control voltage level (or) rated out put voltage

$G= V_{dc} / V_c$

In this condition G=12.4/3.3

Ks= Reference voltage for rated speed / Rated speed in rad/sec

= Vr/W

The closed loop transfer function is given by

$T.F=K_A.K_M / S.T+1+K_MK_W.K_A$

$K_A = G.K_p$

K_P = position error constant

$K_p = V_c / V_{error}$

$V_{error} = V_{ref} - V_f$

Motor time constant $T = J R_a / K_b.Kt$

Moment of inertia $J = 0.03$ N / m^2

K_t = Motor torque constant ≈ 1.3

= K_b when speed is expressed in radians / sec (w.unit)

Tabular column:

(i) Load Test:

S. No	IL(A)	F1(kg)	F2(kg)	N(rpm)	T=F1-F2x9.8xr (N-M)	V(v)

(ii) *Armature Voltage Control*:

Armature current Ia(A)	V(v)	N(rpm)	Eb=V-IaRa(v)	W=2xΠxN/60 (rad/sec)

Sample calculations:

To find transfer function:-

Viscous friction co-efficient

Transfer function = (G/(1+GH))= (Km/(sBm(1+Tms)Ra(1+sTa)+KmKbs)

Experiment No.:2

Effect Of Feedback On Dc Servomotor

Part – A:

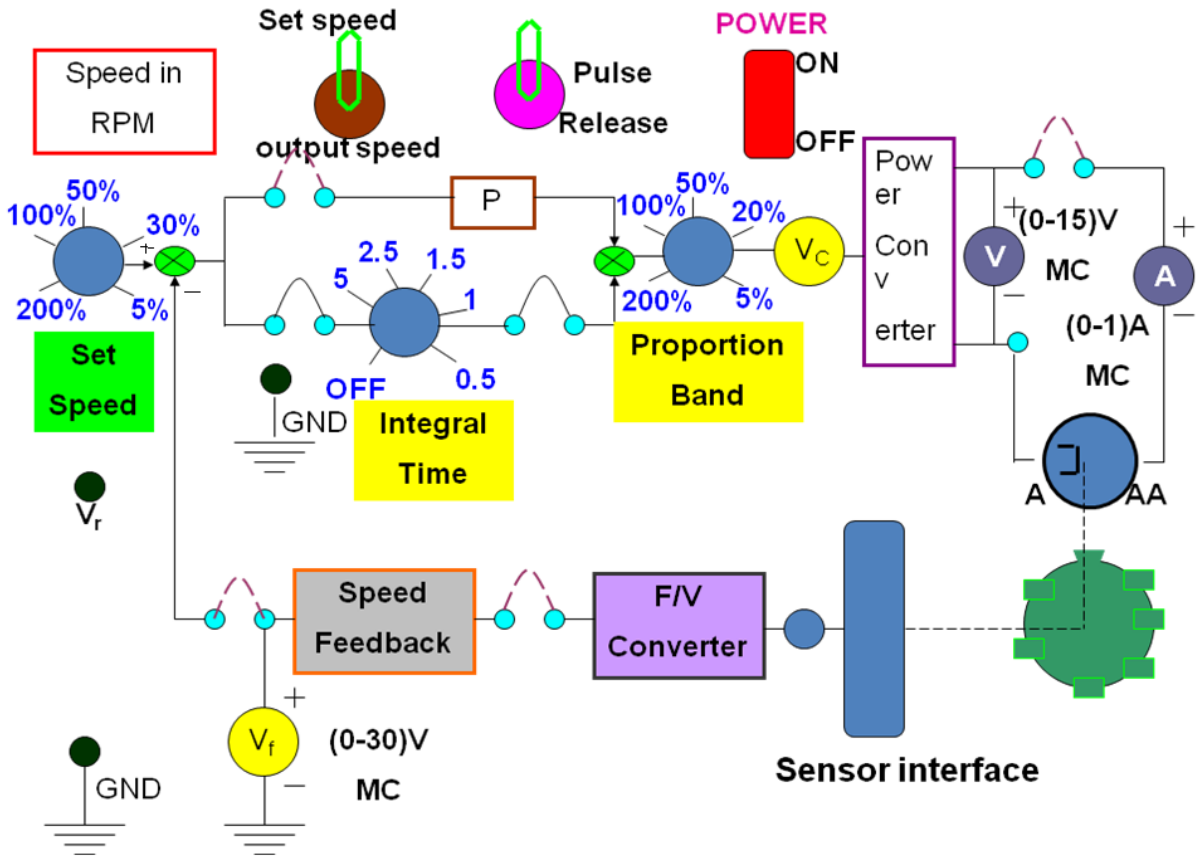
Closed loop performance in terms of steady state error

Aim: To evaluate the closed loop performance in terms of steady error

Apparatus Required:

1. DC servomotor control system kit.
2. Patch chords
3. Voltmeter (0-30) V MC – 3
4. Ammeter (0-1) A MC - 1

Circuit Diagram:



Procedure:

1. Ensure power is OFF to the motor control unit and pulse ON/OFF switch is in OFF position.
2. Connect the circuit as per circuit diagram
3. Switch ON the power to the motor controller.

4. By keeping toggle switch in SP (set point) position, vary the set speed knob and Set V_r = 5 V (= ω_s corresponding to 1500rpm).
5. Run the motor by switching the pulse ON/OFF switch to ON position.
6. Note down the speed, speed feedback voltage V_f, armature voltage V_a, control voltage V_c and ammeter reading.
7. By varying proportional band to different values, repeat step 6
8. Reduce the speed by decreasing the proportional gain and switch OFF power to the motor controller.

Tabular Column:

Using P controller.

S. No	V_f (V)	V_c (V)	V_a (V)	I_a (A)	N (rpm)	$E_b = V_a - I_a.R_a$(V)	$\omega = 2\Pi N/60$ (rad/sec)	$V_{error} = V_r - V_f$	$K_p = V_c/V_{error}$	$K_a = G.K_p$	e_{ss} (cal)	$e_{ss} = \omega_s - \omega$ (exp)

Part – B:

Study of the speed control system with PI controller.

Aim: To study of the speed control system with PI controller.

Apparatus Required:

1. DC servomotor control system kit.
2. Patch chords.
3. Voltmeter (0-30) V MC – 1
4. Ammeter (0-1) A MC - 1

Procedure:

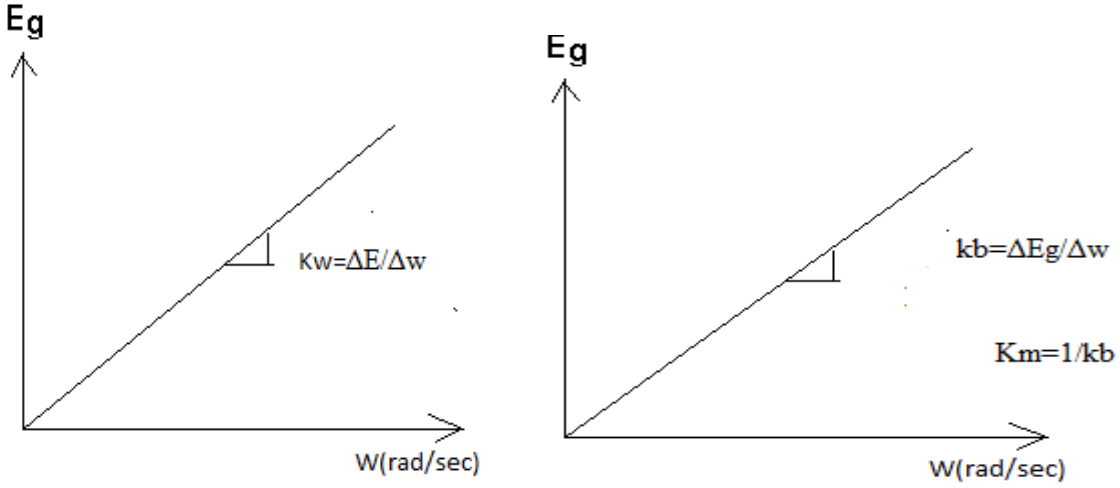
1. Set the controller to be PI controller.
2. Ensure the speed feedback loop is closed and pulse ON/OFF switch is in OFF position.
3. Set the speed controller at the low value.
4. Make the connections as shown in the figure.
5. Switch ON the power to the motor.
6. Set $V_{ref} = W_{ref} = 1$ V, switch ON pulse ON/OFF switch, run the motor at varying speeds by varying the proportional gain.
7. For each speed note down the speed feedback voltage V_f, V_a, V_c and ammeter reading.
8. Reduce the speed by decreasing proportional gain.
9. Switch OFF the power to the motor controller.
10. Measure the armature resistance of the motor using a multimeter.
11. Calculate the K_w and K_b by plotting a graph between the feedback voltage and speed and E_b and speed respectively.

Tabular Column:

Using PI controller.

S. No	V_f (V)	V_c (V)	V_a (V)	I_a (A)	$E_b = V_a - I_a.R_a$(V)	N (rpm)	W = 2 Π N/60 (rad/sec)	$V_{error} = V_r - V_f$	$K_p = V_c/V_{error}$	$K_a = G.K_p$	E_{ss} (cal)	$E_{ss} = W_s$ (exp)

Expected Graph:



Calculations:

K_a = total gain of the amplifier.

= K_p.g

G = gain of the amplifier.

= Rated motor voltage/V_r(max)

Steady state error = e_{ss}= $K_m.V_r/(1+K_a.K_m.K_w)$ $K_p = V_c/V_{error}$.

$W_s = 2\Pi N_s/60$.

K_w from graph of EbvsVf

Km = 1/Kb, kb is obtained from graph Ebvs ω

Sample calculations:-

For P controller:-

From graph:-

From graph:-

Experiment No.:3

Characteristics Of Ac Servo Motor

Aim: To obtain the torque – speed Characteristics of AC servo motor

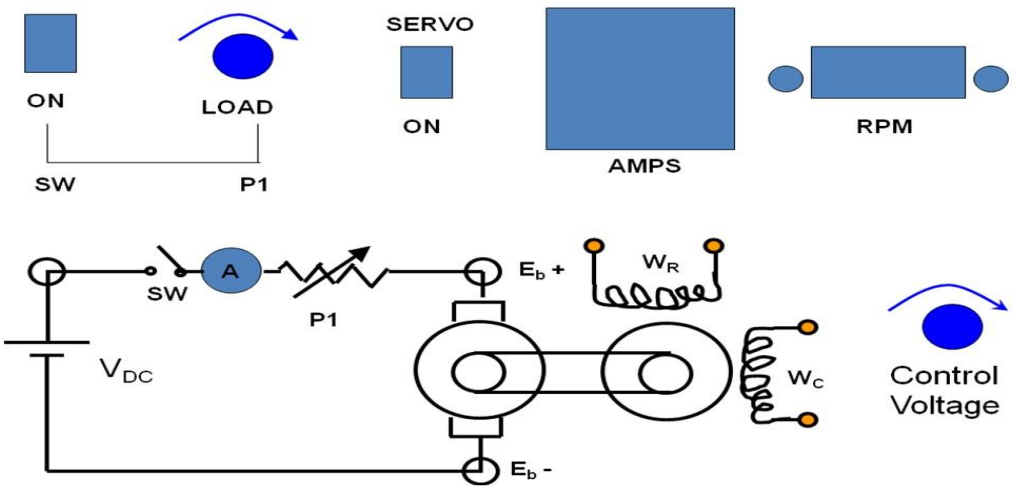
Apparatus Required:

1. AC servo motor kit,
2. Patch cords,
3. multimeter

Procedure:

1. Keep P1 in the minimum position & control knob at maximum position, Switch on the main supply power switch
2. Measure the control winding voltage (AC) & reference winding voltage (AC) by multimeter. Adjust control winding voltage $W_c$by control voltage knob say 230 or 220V
3. Now slowly load the motor by switching ON SW & by varying P1 in steps of I_a . Note down back EMF E_b and speed.
4. Vary P_1upto 0.5A by seeing ammeter and tabulate the readings.
5. Potentiometer P_1 is brought back to minimum position & switch OFF SW switch.
6. Set the control winding voltage P_2 to a new value say 210V or 200V and repeat steps 3 & 4.
7. Plot the graphs of speed vs torque for two control winding voltages

   ***Note**: If the control winding voltage is reduced less than 190V, the motor may not rotate due to insufficient voltage*



Tabular Column:

Reference winding voltage V_r = 206 V

Control winding voltage V_c = 218 V

S.No	Ia(A)	N(rpm)	Eb=V-IaRa	P (w)	T(N-M)

Formulae:

$P = E_b .I_a$

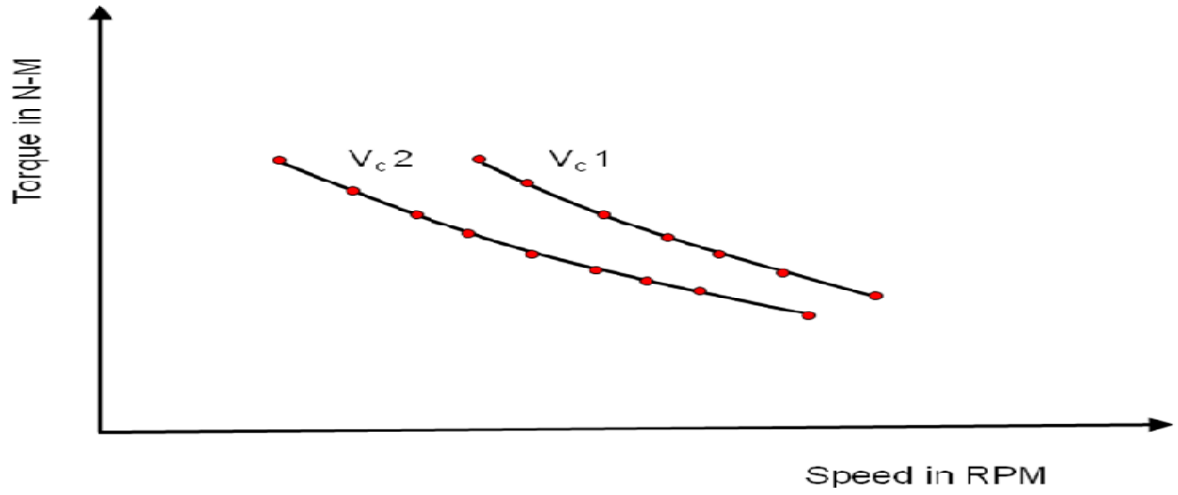
$T = 60 P / 2 \Pi N$

WhereEb is the back emf in V

I_a is the armature current in A

N is the speed in rpm

MODEL GRAPH:



Sample calculations:-

Experiment No.:4

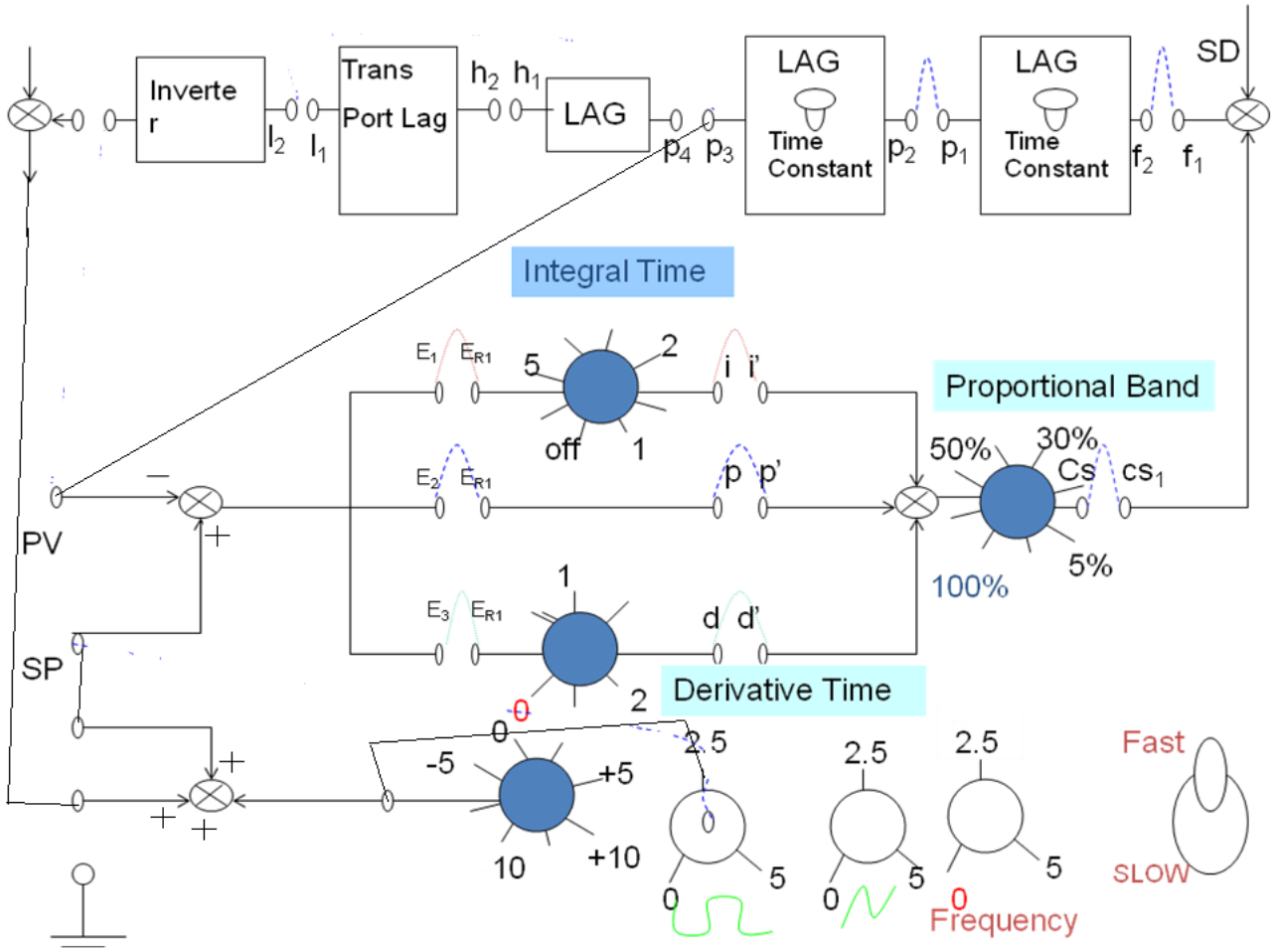
Effects Of P, PI, PD, PID Controller On Second Order System

Aim: To study the effects of P, PI, PID controllers on a second order system.

Apparatus Required:

1. Model PCS – 01
2. Patch chords
3. CRO

Circuit Diagram:



P-Controller:

Aim: To study the closed loop response of P controller on a second order system.

Procedure:

1. Make the connections as per the circuit diagram.
2. Switch on the power supply to the unit.
3. Square wave of amplitude 2V (p-p) is given as the input with optimum frequency settings so as to view the response well on as CRO.
4. Adjust the value of the proportional band knob and measure the peak over shoot from the response.
5. Note down the raise time, peak time, settling time for a given step input and tabulate the readings.
6. Repeat the same procedure for various band values.

Tabular Column:

Sl.No.	Proportional Band (P_b) %	Peak Over Shoot (M_p) volts	Raise Time T_r(msec).	Peak Time Tp (msec)	Settling time T_s(msec)

PI Controller:

Aim: To study the closed loop response of a PI controller on a second order system.

Procedure:

1. Make the connections as per the circuit diagram.
2. Switch ON the supply to the unit.
3. Square wave of amplitude 2V (p-p) is given as input with optimum frequency setting.
4. Adjust the value of proportional band and integral time knobs and view the response on CRO.
5. Note down peak over shoot; raise time, peak time and settling time for the response.
6. Repeat the above procedure for various proportional and integral time values.

Tabular Column:

S. No	Proportional Band (P_b) %	Integral Time (Ti) sec	Peak Over Shoot (V) μ_p	Raise Time T_r (msec)	Peak Time T_p(msec)	Settling Time T_s (msec)

PD Controller:

Aim: To study the closed loop response of a PD controller on a second order system.

Procedure:

1. Make the connections as per the circuit diagram.
2. Switch ON the supply to the unit.
3. Square wave of amplitude 2V (p-p) is given as input with optimum frequency setting.
4. Adjust the value of proportional band and derivative time knobs and view the response.
5. Note down peak over shoot, raise time, peak time and settling time for the response.
6. Repeat the above procedure for various derivative time values.

Tabular Column:

S. No	Proportional Band (P_b) %	Derivative Time (T_d) sec	Peak Over Shoot (V) μ_p	Raise Time T_r (msec)	Peak Time T_p(msec)	Settling Time T_s (msec)

PID Controller:

Aim: To study the closed loop response of PID controller on a second order system.

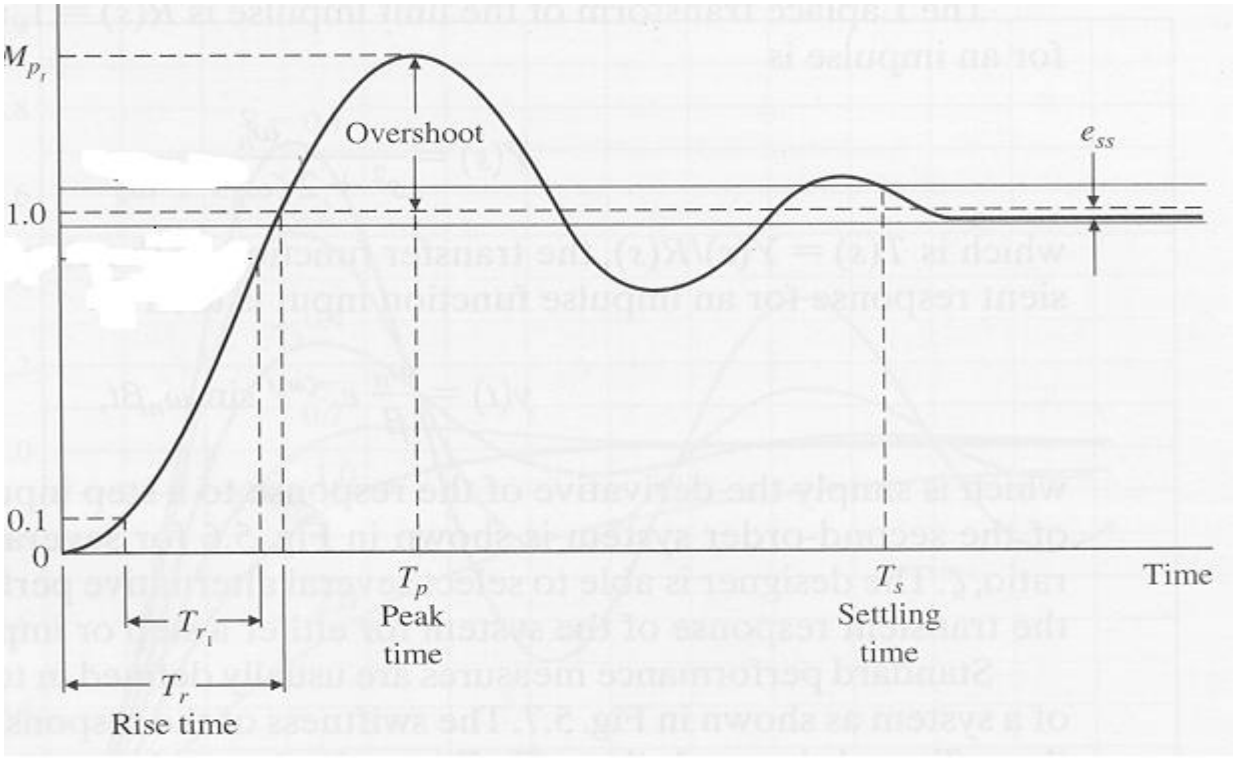
Procedure:

1. Make the connections as per the circuit diagram.
2. Switch ON the supply to the unit.
3. Square wave of amplitude 2V (p-p) is given as input with optimum frequency setting.
4. Adjust the value of proportional band integral time and derivative time knobs and view the response on CRO screen.
5. Note down peak over shoot; raise time, peak time and settling time for the response.
6. Repeat the above procedure for various proportional and derivative time values and tabulate the readings.

Tabular Column:

S. No	Proportional Band (P_b) %	Integral Time (Ti) sec	Derivative Time (T_d) sec	Peak Over Shoot (V) μ_p	Raise Time T_r (msec)	Peak Time T_p(msec)	Settling Time T_s (msec)

Expected Graph:



Sample calculations:

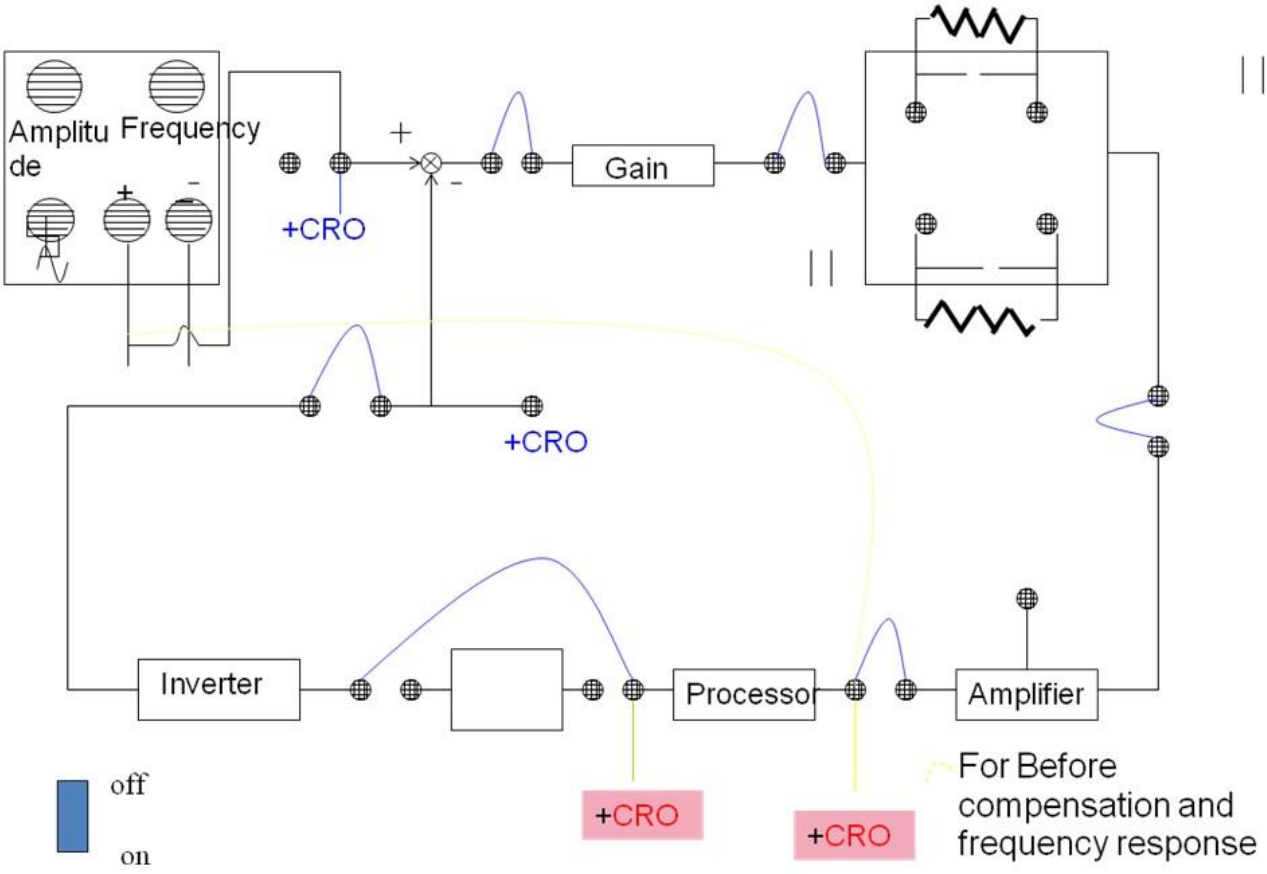
Experiment No.5
Compensation Design Using Lead – Lag Network

Aim: To compensate the phase lag or lead in the system by designing compensation circuit to meet the specifications.

Apparatus Required:

1. Lead – lag Network kit
2. Patch chords
3. Resistors of required rating corresponding to design CRO and probes

Circuit Diagram:



Procedure:

part – a:

1. Switch ON the power to the instrument
2. Connect the individual blocks using path cards by passing the compensation network and gain.
3. Give a sinusoidal input as the set value to the error detector
4. Measure the amplitude and frequency of the input single.

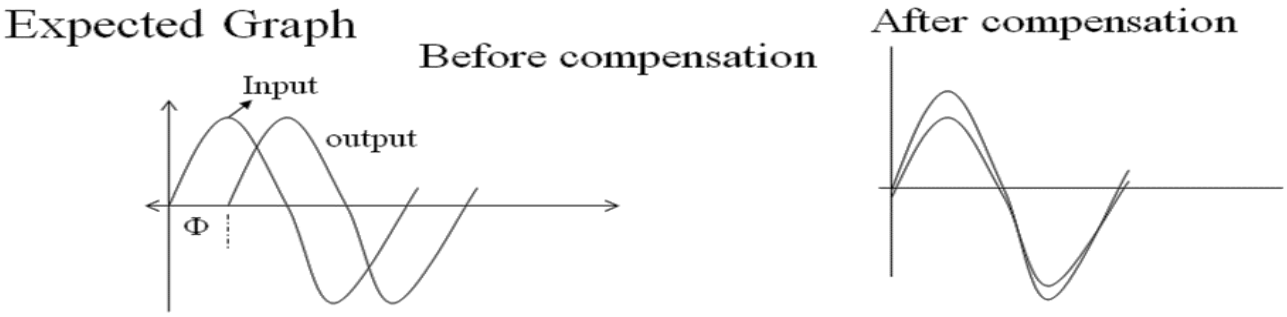
5. Measure the amplitude and phase shift of the output single with respect to the input sine wave
6. Using the technique explained previously calculate the values of R1, c1 r2 and C2 to compensate for the phase shift of the output single.

Tabular column:

Part A:

S. No	Amplitude		Frequency F(Hz)	Phase Ue Ø (degree)	R_1 Ω	C_1 µfd	R_2 Ω	C_2 µfd
	i/p	o/p						

Expected Graph



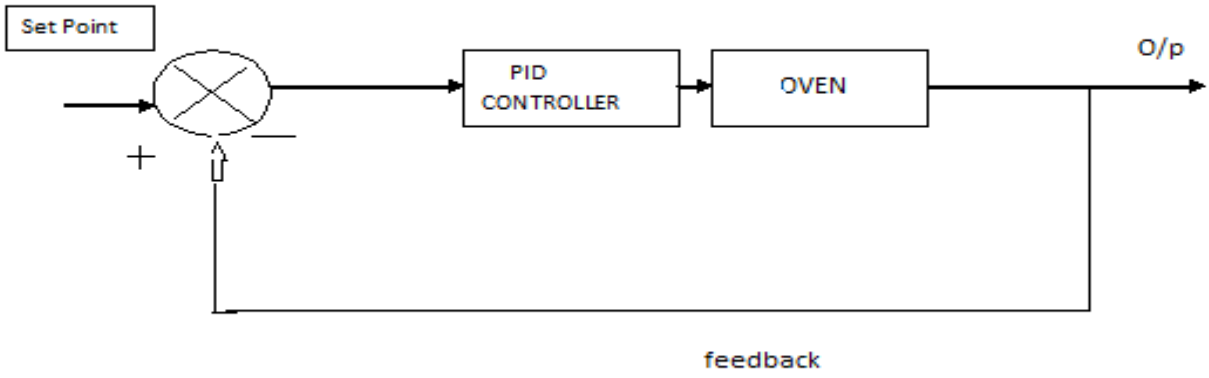
Experiment No.: 6
Temperature Control Using PID Controller

Aim: To study the steady state characteristics of temperature by using pid controller.

Apparatus Required:

1. Temperature control kit.
2. Patch chords.
3. Process Heater
4. Temperature sensor.

Circuit Diagram:



Procedure:

PI Controller:

1. Connect the power chord to main supply of 230V, 50Hz.
2. Put the power switch of the kit ON position.
3. Set the proportional gain K_p and Integral gain K_i at particular value.
4. Note down the time taken for every 5 degree raise in temperature.
5. Draw the graph between temperature and time

PID Controller:

1. Connect the power chord to main supply of 230V, 50Hz.
2. Put the power switch of the kit ON position.
3. Set the proportional gain K_p, Integral gain $K_i$and derivative gain K_d at particular value.
4. Note down the time taken for every 5 degree raise in temperature .
5. Draw the graph between temperature and time

Tabular Columns:

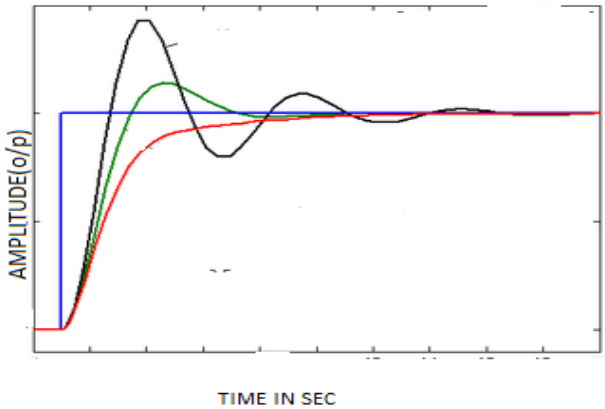
Using Pi Controller: K_P= 50 K_I=45

S.No	Temperature	Time (sec)

Using PID Controller: K_P= 50 K_I=45 K_D=56

S.No	Temperature	Time (sec)

Expected Graph:



Experiment No. 7:
Stability Analysis

(i) ROOT LOCUS

Aim: To plot the Root locus for the given transfer function and to verify it using MATLAB.

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

Apparatus Required: PC with MATLAB software.

Root Locus Plot Using MATLAB:

The characteristic equation can be written as $1+k\frac{num}{den}=0$.

The command rlocus (num, den) gives the root locus plot.

If the system is defined in state space, root locus is obtained by the command rlocus (A, B, C, D).

Theoretical Calculations:

Root locus: Given transfer function H(s)= (k(s+10)/(s^2+6s+8))

Let K=1

Open loop poles = S1,S2=-2,-4

No of branches of root locus = no of poles=n=2

No of open loop zeros =m=1

S=-10

No of branches terminating at ∞ =>n-m=2-1=1

Angle of asymptotes = 180(2q+1)/(n-m) q=0,1,2 ……

n-m-1=0

Ø a=180

Centroid = (-2-4)-(-10)/(2-1)=4

Break away point=dk/ds = 0

Characteristic equation= 1+G(s)H(s)=0

(1+k(S+10))/(S^2+6s+8)=0

K=-(S^2+6s+8)/(S+10)

dk/ds=0=> -((S+10)(2S+6)-(S^2+6S+8))/((S+10)^2) = 0

S^2+20S+5=0

S=-3.07 , -16.92

Program:

OUTPUT:

(ii) Bode Plot

Aim: To obtain the Bode Plot for the given transfer function and to verify it using MATLAB.

Apparatus: PC with MATLAB software.

Theoretical Calculations:

Bode plot:

$H(s)= (S+10)/(S^2+6S+8) = (1.25(1+0.1j\omega)/((1+0.5j\omega)(1+0.25j\omega))$

Corner frequencies:

$0.1\omega=1$ $0.5\omega=1$ $0.25\omega=1$

$\omega_{c3}=10$ $\omega_{c2}=4$ $\omega_{c1}=2$

Magnitude of G(jw) = | G(jw)| =(1.25(1+(0.1ω)^2)^(1/2))/((1+0.5ω^2)(1+0.25ω^2))^(1/2))

Angle G(jω) = 1/(tan(0.1ω)-1/(tan(0.5ω)-1/(tan(0.25ω)

S.NO	term	Cornerfreq (rad/sec)	Slope db/dc	Change in slope (db/dec)

ωl=0.1 rad/sec ωh=30 rad/sec

At

$\omega=\omega_c=0.1$ A=20log(1.25/0.1)=21.93 db

$\omega=\omega_{c1}$ A=20log(1.25/2)= - 4.082 db

$\omega=\omega_{c2}$ A= - 20log(4/2)-4.08=-10.102 db

$\omega=\omega_{c3}$ A= - 40log(10/4)-10.102= - 26.019 db

$\omega=\omega_h$ A= -40log(30/10)-26.019= - 45.105 db

ω	0.1	0.5	2	4	10	20	30
∟G(jω)	-3.72	-18.29	-60.25	-86.63	-101.8	-99.54	-97.02

Program:

Output: