

**DEPARTMENT OF ELECTRONICS & COMMUNICATION  
ENGINEERING**

**COLLEGE OF ENGINEERING, THIRUVANANTHAPURAM**

**AE431: CONTROL SYSTEM AND SIGNAL PROCESSING LAB**

**LAB MANUAL**



**2021**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION  
ENGINEERING**

**COLLEGE OF ENGINEERING, THIRUVANANTHAPURAM**



**CERTIFICATE**

This is a controlled document of **Department of Electronics and Communication** of **College of Engineering, Trivandrum**. No part of this can be reproduced in any form by any means without the prior written permission of the Head of the Department, Electronics & Communication, College of Engineering, Trivandrum. This is prepared as per **2015 B.Tech** Electronics and Communication scheme.

<p><b>Prepared by:</b></p> <p><b>Ms. Jeeshma Mary Paul,</b>  <b>Ms. Kavya Manohar</b>          Research Scholar          Department of Electronics and          Communication,          College of Engineering, Trivandrum</p>	<p><b>Compiled by:</b></p> <p><b>Dr. Anitha Edison,</b>  <b>Ms. Narendramudra N G</b>          Assistant Professor          Department of Electronics and          Communication,          College of Engineering, Trivandrum</p>
<p><b>Reviewed by:</b></p>	<p><b>Approved by:</b></p>

Course Code	Course name	L-T-P Credits	Year of Introduction
AE431	CONTROL SYSTEM AND SIGNAL PROCESSING LAB	0-0-3-1	2016
<p><b>SIGNAL PROCESSING LAB</b></p> <ol style="list-style-type: none"> <li>1. Familiarization of signal processing commands used in MATLAB Software.</li> <li>2. Developing elementary signal function modules (m-files) for unit impulse, step, exponent and ramp sequence.</li> <li>3. Generating continuous and discrete time sequences.</li> <li>4. Carrying out mathematical operations on signals.</li> <li>5. Response of LTI system described by difference and differential equation.</li> <li>6. Developing a program for computing inverse Z-Transform.</li> <li>7. Developing program for finding magnitude &amp; phase response of LTI System 8. Developing program for computing DFT &amp; IDFT.</li> <li>8. Developing a program for computing circular convolution.</li> <li>9. Design of filter: FIR, IIR, ECG Signal filter (can be done as 3 separate experiments).</li> </ol> <p><b>CONTROL SYSTEM LAB using MATLAB</b></p> <ol style="list-style-type: none"> <li>1. Familiarization of MATLAB commands used in control system design</li> <li>2. Representation of system in MATLAB: state space representation &amp; transfer function representation</li> <li>3. Stability analysis using Bode plot, root locus &amp; their pole-zero-gain representation.</li> <li>4. Implementation of PID control using both m-file and Simulink.</li> <li>5. Pole placement technique applied to stabilize a system.</li> <li>6. Realization of a compensator design.</li> <li>7. Modelling and analysis of a first order system.</li> <li>8. Modelling of an unstable system (inverted pendulum, ball &amp; plate system etc.)</li> </ol> <p><b>PC Based Control</b></p> <ol style="list-style-type: none"> <li>1. PLC programming: familiarization of instruction set.</li> <li>2. PLC programming: simulation of process control.</li> </ol> <p><b>LabVIEW based Virtual Instrumentation</b></p> <ol style="list-style-type: none"> <li>1. Getting started with LabVIEW: Basic operations, controls, indicators, and simple Programming structures.</li> <li>2. Debugging a VI and sub-VI.</li> <li>3. Familiarization of DAQ card.</li> </ol>			

(i) Course Outcomes (COs)

Course Outcome	Course Outcome (CO) Description	Cognitive level	Assessment tool
	At the end of the course, the student should be able to:		
AE431/CO 1	Build signal processing systems using MATLAB	K4	Continuous Evaluation/ Practical Exam
AE431/CO 2	Model control systems using MATLAB	K4	Continuous Evaluation/ Practical Exam
AE431/CO 3	Make use of PLC, LabVIEW for modelling basic control systems	K3	Continuous Evaluation/ Practical Exam
AE431/CO 4	Develop a custom filter design toolbox using MATLAB	K6	Course Project
AE431/CO 5	Prepare record of lab experiments	K3	Evaluation of Record
AE431/CO6	Present the course project	K3	Demo and presentation of course project

(ii) CO-PO Mapping

Course Outcome	CO-PO/PSO matrix showing level of correlation (1-Low, 2-Medium, and 3-High)														
	PO-1	PO-2	PO-3	PO-4	PO-5	PO-6	PO-7	PO-8	PO-9	PO-10	PO-11	PO-12	PSO-1	PSO-2	PSO-3
AE431/CO 1	3				3								3		
AE431/CO 2	3				3								3		
AE431/CO 3					3								3		
AE431/CO 4		3	3	3	3			3	3				3	3	3
AE431/CO 5										3					
AE431/CO 6										3					

# **PART A**

## **Signal Processing Lab**

# **EXPERIMENT 1: FAMILIARIZATION OF SIGNAL PROCESSING COMMANDS USED IN MATLAB SOFTWARE**

## **Aim**

To familiarize basic MATLAB functions and signal processing toolbox.

## **MATLAB**

MATLAB is a software package for high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. The name MATLAB stands for matrix laboratory. MATLAB features a family of add-on application-specific solutions called toolboxes. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include Image processing, signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

### **(a) DEFINITION OF VARIABLES**

Variables are assigned numerical values by typing the expression directly Eg:  $a=1+2$  yields  $a=3$ . MATLAB utilizes the following arithmetic operations: + addition, - subtraction, \* multiplication, / division, ^ power operation, ' transpose

There are certain predefined variables which can be used in the same manner as user defined variables:  $i=\sqrt{-1}$ ,  $j=\sqrt{-1}$ ,  $\pi = 3.1416$

There are also a number of predefined functions that can be used when defining a variable. Some common functions that are used are Abs- magnitude of a number, Angle- angle of a complex number, Cos- cosine function, assume arguments in radian. Exp-exponential functions.

### **(b) DEFINITION OF MATRICES**

MATLAB is based on matrix and vector algebra. Even scalars are treated 1x1 matrix. Therefore, vector and matrix operation are simple as common calculator operations. Vectors can be defined in two ways. The first method is used for arbitrary elements,  $v=[1\ 3\ 5\ 7]$  creates 1x4 vector elements with elements 1 3 5 & 7. Note that commas would have been used in the place of spaces to separate the elements. Additional elements can be added to the vector  $v(5)=8$  yields the vector  $v=[1\ 3\ 5\ 7\ 8]$ . Previously defined vectors can be used to define a new vector. For example, with we defined above  $a=[9\ 10]$ ;  $b=[v\ a]$ ; creates the vector  $b=[1\ 3\ 5\ 7\ 9\ 10]$ . The second method is used for creating vector with equally spaced elements  $t=0:0.1:10$ ; creates 1x101 vector with elements 0,0.1,0.2. . . . 10. Note that the middle number defines the increments is set to a default of 1  $k=0:10$  creates 1x11 vector with the elements 0,1,2. . . . 10.

Matrices are defined by entering the element row by row.  $M = [124; 368]$  creates the matrix  $M = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 6 & 8 \end{bmatrix}$ . There are number of special matrices that can be defined Null matrix:  $[ ]$ ;  $N \times m$  matrixes of zero:  $M = \text{zeros}(m,m)$ ;  $N \times m$  matrix of ones:  $M = \text{ones}(n,m)$ ;  $N \times n$  matrix of identity:  $M = \text{eye}(n)$ .

A particular elements of matrix can be assigned  $M(1,2)=5$  place the number 5 in the first row, 2nd column. Operations and functions that were defined for scalars in the previous section can be used on vectors and matrices. For example  $a=[1 \ 2 \ 3]$ ;  $b=[4 \ 5 \ 6]$ ;  $c=a+b$  yield  $c=[5 \ 7 \ 9]$ . Functions are applied element by element. For example  $t=0:0.1:10$ ;  $x=\cos(2*t)$  creates a vector 'x' with elements equal to  $\cos(2t)$  for  $t=0,1,2, \dots, 10$ .

### (c) GENERAL INFORMATION

MATLAB is case sensitive. So 'a' and 'A' are two different names. Comment statements are preceded by '%'.

1) M-files M-files are macros of MATLAB commands that are stored as ordinary text file with the extension 'm' that is 'filename.m'. An m-file can be either a function with input and output variables or a set of commands. MATLAB requires that the m-file must be stored either in the working directory or in a directory that is specified in the MATLAB path list. The following commands typed from within MATLAB demonstrate how this m-file is used.  $X=2, y=3, z=y$  plus  $x(y,x)$  MATLAB m-files are most efficient when written in a way that utilizes matrix or vector operations, loops and if statements are available, but should be used sparingly since they are computationally inefficient. An example is for  $k=1:10$   $x(k)=\cos(k)$  end; This creates a  $1 \times 10$  vector 'x' containing the cosine of the positive integers from 1 to 10. This operation is performed more efficiently with the commands  $k=1:10$   $x=\cos(k)$  which utilizes a function of a vector instead of a for loop. An if statement can be used to define combinational statement.

## **Expected Output**

Familiarize basic MATLAB functions and signal processing toolbox

## EXPERIMENT 2: GENERATION OF ELEMENTARY SIGNALS

### Aim

Plot elementary signals

- a. Impulse b. Step c. Ramp d. Exponential

### Steps

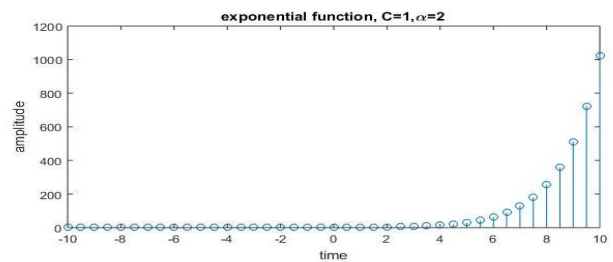
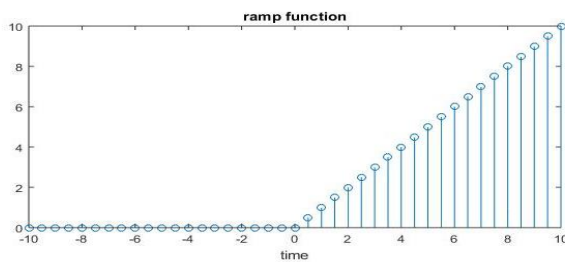
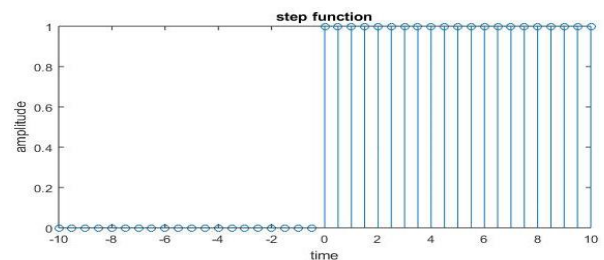
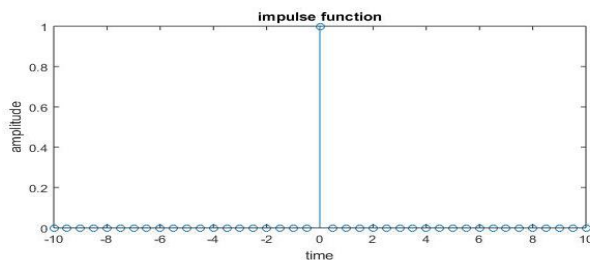
1. Define a variable for time  $n$
2. Define unit impulse function  $\delta[n] = \begin{cases} 1 & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases}$
3. Define unit step function  $u[n] = \begin{cases} 1 & \text{for } n \geq 0 \\ 0 & \text{for } n < 0 \end{cases}$
4. Define unit ramp function  $r[n] = \begin{cases} n & \text{for } n \geq 0 \\ 0 & \text{for } n < 0 \end{cases}$
5. Define exponential function  $x[n] = C\alpha^n$ . Get  $C$  and  $\alpha$  as user inputs. Ask each student to select a different  $C$  and  $\alpha$
6. Plot the functions

### MATLAB functions and methods to be familiarised

1. Use MATLAB Help to familiarise basic functions: plot, stem, subplot, xlabel, ylabel, title, axis, ones, zeros, clc, clear all, close all, input
2. Familiarise the usage of conditional statements

### Expected Output

Typical waveform for time varying from -10 to 10 and parameters for exponential function  $C = 1$  and  $\alpha = 2$





## EXPERIMENT 3: GENERATION OF PERIODIC SIGNALS

### Aim

Plot discrete time periodic signals of given amplitude and frequency

a. Sine wave b. Square wave c. Triangular wave

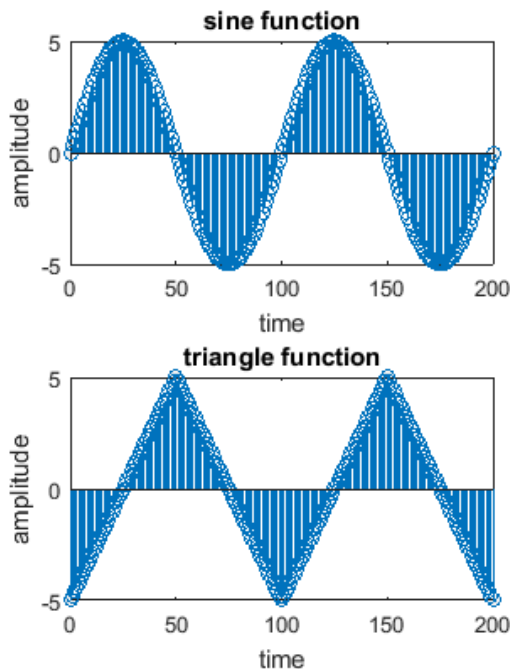
### Steps

1. Define a variable for amplitude  $amp$ . Get  $amp$  as user input. Ask each student to select a different  $amp$
2. Define a variable for frequency  $f$ . Get  $f$  as user input. Ask each student to select a different  $f$
3. Define frequency dependant time base  $tp$  and  $n$
4. Define periodic sine wave  $y_{sin} = amp * \sin(2 * \pi * ft * p)$
5. Define periodic square wave  $y_{square} = amp * \text{square}(2 * \pi * ft * p)$
6. Define periodic triangular wave  $y_{triangle} = amp * \text{sawtooth}(2 * \pi * ft * p, 0.5)$
7. Plot the functions

### MATLAB functions and methods to be familiarised

1. Use MATLAB Help to familiarise basic functions: plot, stem, subplot, xlabel, ylabel, title, axis, clc, clear all, close all, input
2. Familiarise the usage of MATLAB inbuilt functions sin, square and sawtooth

### Expected Output



#### Inputs

$amp=5$

$f=50$

Frequency dependant time base are defined as

$tp=-10: 0.01/f :10$

$n=0:(\text{length}(tp)-1)$

## EXPERIMENT 4: SIGNAL OPERATIONS

### Aim

Perform following signal operations on discrete time signals

- a. Amplitude Scaling   b. Addition of two signals   c. Time Scaling   d. Time Shifting

### Steps

1. Define first sequence  $x1[n]$ . Get  $x1[n]$  as user input. Ask each student to select a different  $x1[n]$
2. Define second sequence  $x2[n]$ . Get  $x2[n]$  as user input. Ask each student to select a different  $x2[n]$
3. Perform amplitude scaling  $y\_amplitude\_scaling = aa.* x1[n]$ . Get  $aa$  as user input. Ask each student to select a different  $aa$
4. Perform addition  $y\_addition = x1[n] + x2[n]$
5. Perform time scaling  $y\_time\_scaling = x1[at.* n]$ . Get  $at$  as user input. Ask each student to select a different  $at$
6. Perform time shifting  $y\_time\_shift1 = x1[n - ts]$  and  $y\_time\_shift2 = x1[n + ts]$ . Get  $ts$  as user input. Ask each student to select a different  $ts$
7. Plot the functions

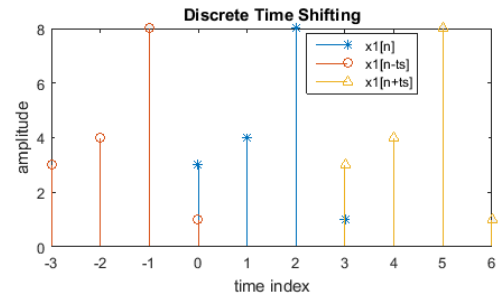
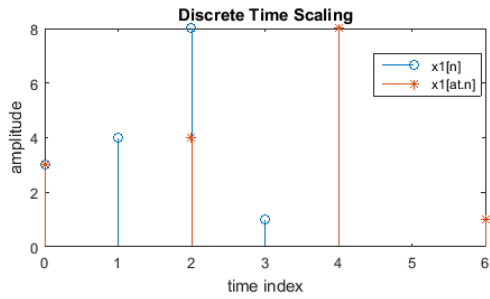
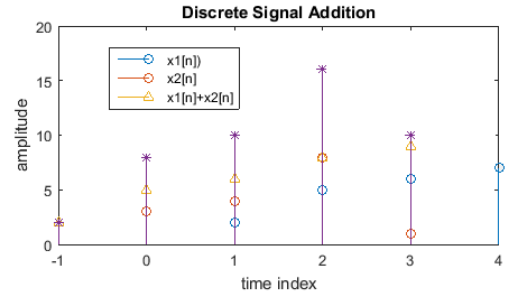
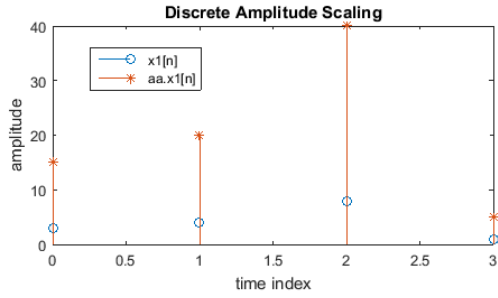
### MATLAB functions and methods to be familiarised

1. Use MATLAB Help to familiarise basic functions: plot, stem, subplot, xlabel, ylabel, title, axis, clc, clear all, close all, input
2. Familiarise the usage of basic signal operations time shift, time scale, amplitude scale and addition

### Expected Output

Input:

$aa = 5$ ,  $at = 2$ ,  $ts = 3$ ,  $x1[n] = [3 \ 4 \ 8 \ 1]$ , starting point of  $x1[n] = 0$ ,  $x2[n] = [2 \ 5 \ 6 \ 8 \ 9]$ , starting point of  $x2[n] = -1$



## EXPERIMENT 5: Z TRANSFORM AND INVERSE Z TRANSFORM

### Aim

Obtain z-transform and inverse z-transform of following functions.

a.  $x[n] = a^n u[n]$    b.  $x[n] = \sin(\omega n) u[n]$

### Theory

The Z-transform converts a discrete-time signal, which is a sequence of real or complex numbers, into a complex frequency-domain representation. It can be considered as a discrete-time equivalent of the Laplace transform. The z-transform is useful for the manipulation of discrete data sequences and has acquired a new significance in the formulation and analysis of discrete-time systems. It is used extensively today in the areas of applied mathematics, digital signal processing, control theory, population science, and economics. These discrete models are solved with difference equations in a manner that is analogous to solving continuous models with differential equations. The role played by the z transform in the solution of difference equations corresponds to that played by the Laplace transforms in the solution of differential equations.

### Steps

1. Define input  $x1[n] = a^n u[n]$ . Get  $a$  as user input. Ask each student to select a different  $a$
2. Define z transforms of  $x1[n]$ ,  $y1 = \text{ztrans}(x1[n])$
3. Define input  $x2[n] = \sin(\omega n) u[n]$ . Get  $w$  as user input. Ask each student to select a different  $w$
4. Define z transforms of  $x2[n]$ ,  $y2 = \text{ztrans}(x2[n])$
5. Define inverse z transform of  $y1$ ,  $z1[n] = \text{iztrans}(y1)$
6. Define inverse z transform of  $y2$ ,  $z2[n] = \text{iztrans}(y2)$
7. Display the values

### MATLAB functions and methods to be familiarised

1. Use MATLAB Help to familiarise basic functions: display, clc, clear all, close all, input
2. Familiarise the usage of `ztrans()` and `iztrans()` functions

### Expected Output

For input:  $a = 5$ ,  $w = 20$

z-transform of  $5^n \cdot u[n]$ :

$$z/(z - 5)$$

z-transform of  $\sin(20n) \cdot u[n]$ :

$$(z \sin(20)) / (z^2 - 2 \cos(20) z + 1)$$

Inverse z-transform of  $-z/(5 - z)$ :

$$5^n$$

Inverse z-transform of  $(z \sin(20)) / (z^2 - 2 \cos(20) z + 1)$ :

$$\sin(20 \cdot n)$$

## EXPERIMENT 6: IMPULSE RESPONSE OF LTI SYSTEM

### Aim

Obtain impulse response of first order and second order LTI system

### Steps

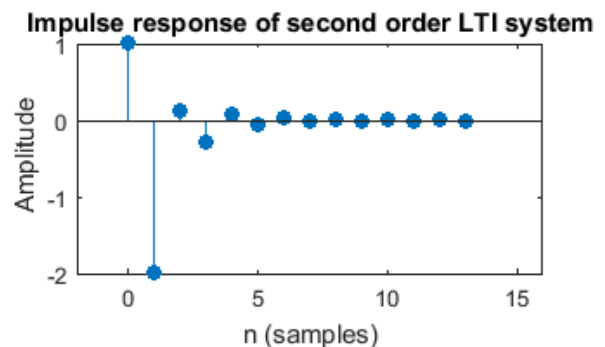
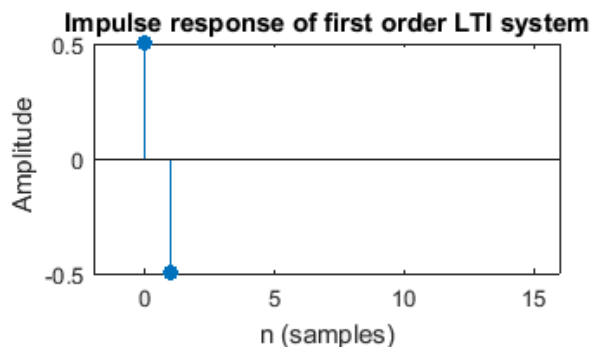
1. Give numerator [b1] and denominator [a1] coefficients of first order LTI system. Get numerator and denominator as input. Ask each student to select a different [b1] and [a1]
2. Find impulse response of first order system using Matlab function `impz([b1],[a1])`
3. Give numerator [b2] and denominator [a2] coefficients of second order LTI system. Get numerator and denominator as input. Ask each student to select a different [b2] and [a2]
4. Find impulse response of second order system using Matlab function `impz([b2],[a2])`
5. Plot the functions

### MATLAB functions and methods to be familiarised

1. Use MATLAB Help to familiarise basic functions: `stem`, `subplot`, `xlabel`, `ylabel`, `title`, `axis`, `clc`, `clear all`, `close all`, `input`
2. Familiarise the usage of `impz()` function

### Expected Output

Input:  $b1 = [1 \ -1]$ ,  $a1 = 2$ ,  $b2 = [1 \ (-7/4) \ (-0.5)]$ ,  $a2 = [1 \ (1/4) \ (-1/8)]$



## EXPERIMENT NO: 7

### LTI SYSTEM RESPONSE

#### **AIM:**

Write a MATLAB program to plot the magnitude and phase response of an LTI system

#### **THEORY:**

Linear Time Systems (LTI Systems) are a class of systems used in DSP that are both linear and time invariant. Linear systems are systems whose output for a linear combination of inputs are the same as a linear combination of individual responses to those inputs. The time invariant systems are systems where the output does not depend on when an input was applied. These properties make LTI systems easy to represent and understand graphically.

LTI systems can be described in terms of impulse response (eg:  $h[n] = [1 \ 1]$ ) or in terms of

transfer functions. (eg:  $H(z) = \frac{1 - \frac{7}{4}z^{-1} - \frac{1}{2}z^{-2}}{1 + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}}$ ,  $H(z) = \frac{1}{2} - \frac{1}{2}z^{-1}$ )

Frequency response of an LTI system is the DFT of its impulse response.

#### **STEPS**

1. Define the impulse response of LTI system as a discrete sequence
2. Use inbuilt function to compute DFT of the sequence
3. Plot the magnitude and phase response in terms of frequency
4. Define the coefficients of numerator and denominator of transfer function as discrete sequences.
5. Use inbuilt function to find the corresponding frequency response
6. Plot its magnitude and phase response in dB and degrees respectively

#### **MATLAB FUNCTIONS USED:**

- `fft()` : Computes DFT using FFT algorithm
- `freqz()` : Computes the complex frequency response from the Transfer function coefficients
- `plot()`: •`ABS( )` : `abs(X)` returns an array Y such that each element of Y is the absolute value of the corresponding element of X.
- `abs()`: To compute absolute value of magnitude of a complex function

## **SAMPLES OF EXPECTED OUTPUT**

Need to simulate a paste output

## **RESULT**

The MATLAB program to obtain the time and frequency response of an LTI system is executed and output is obtained.



## EXPERIMENT NO: 8

### DFT AND IDFT

#### AIM

Write a MATLAB program to find DFT and IDFT of input sequence.

#### THEORY

Discrete Fourier Transform is the transformation used to represent the finite duration frequencies. DFT of a discrete sequence  $x(n)$  is obtained by performing sampling operations in both time domain and frequency domain. It is the frequency domain representation of a discrete digital signal.

The DFT of a sequence  $x(n)$  of length  $N$  is given by the following equation,

$$X(k) = \left\{ \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} ; 0 \leq k \leq N-1 \right\}$$

IDFT performs the reverse operation of DFT, to obtain the time domain sequence  $x(n)$  from frequency domain sequence  $X(k)$ . IDFT of the sequence is given as,

$$x(n) = \frac{1}{N}$$

#### STEPS

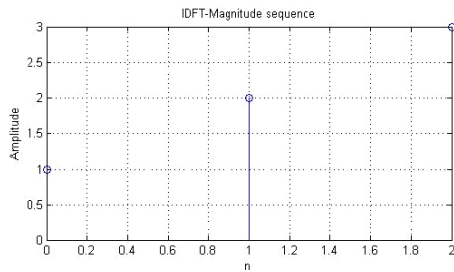
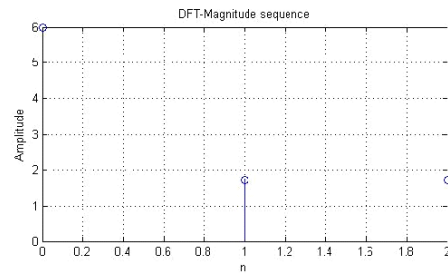
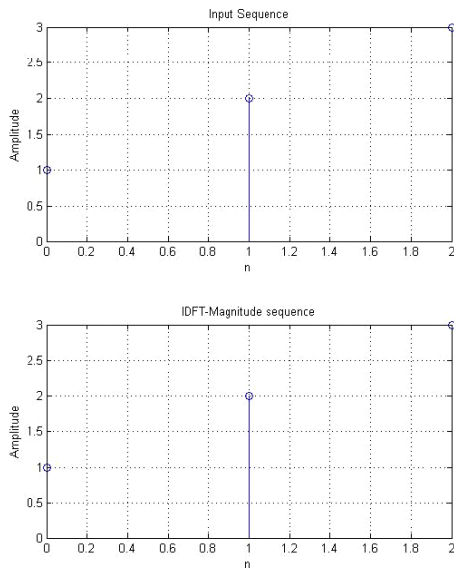
1. Accept an input sequence from the user, interactively
2. Without using inbuilt functions, compute the DFT of the sequence. Plot the magnitude response and phase response along with the input sequence
3. Using the sequence obtained as DFT in the previous step as input, compute the IDFT without using inbuilt functions.
4. Plot the IDFT and verify it is same as the original input.

#### MATLAB FUNCTIONS USED:

- sqrt(): Square root function
- zeros(): Defines sequence of zeros
- exp(): Exponential function
- subplot(): For multiple plots in a single figure
- xlabel(); ylabel(): For labeling the axes
- title(): For providing title to Figure
- disp(): For displaying a sequence
- fft(): DFT function using FFT Algorithms
- ifft(): IDFT function using Inverse FFT algorithm

## SAMPLES OF EXPECTED OUTPUT

Input Sequence=[1 2 3]



## RESULT

The MATLAB program to find DFT and IDFT of input sequence is executed and output is obtained.

## EXPERIMENT NO: 9

### LINEAR CONVOLUTION

#### AIM

Write a MATLAB program to perform linear convolution of two input sequences without using inbuilt function and verify the result using MATLAB function.

#### THEORY

Convolution is used to find the output response of a digital system. The linear convolution of two continuous time signals  $x[n]$  and  $h[n]$  is defined by  $y[n] = x[n] * h[n]$ . The length of the output sequence  $y[n] = \text{length of } x[n] + \text{length of } h[n] - 1$ .

#### STEPS

1. Interactively accept two input sequences from the user
2. Make the lengths of the sequences equal by padding zeros
3. Perform linear convolution by user defined functions
4. Repeat the same using builtin functions
5. Plot both signals and verify the result

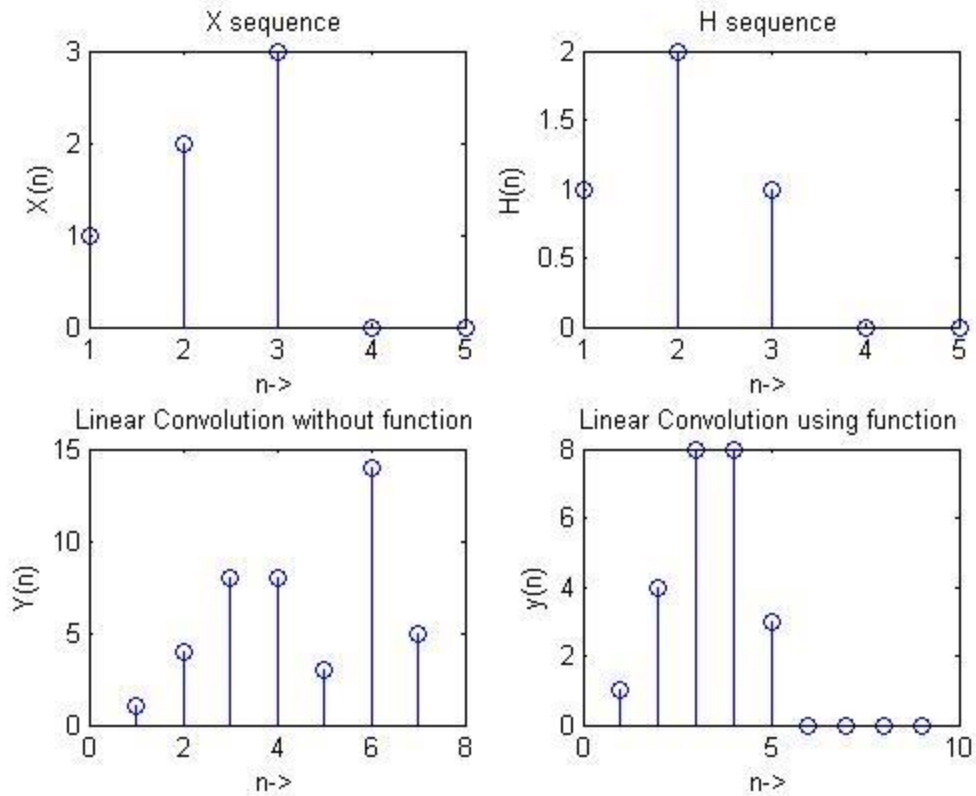
#### **MATLAB FUNCTIONS USED:**

- `length()`: Finding the length of a sequence
- `zeros()`: Defines a sequence of zeros
- `for ()`: Looping function
- `conv()`: Linear Convolution function

#### **SAMPLES OF EXPECTED OUTPUT**

The first sequence, X: [1 2 3]

The second sequence, H: [1 2 1]



## RESULT

The MATLAB program to find the linear convolution of two input sequences is executed and output is verified using inbuilt function.

## **EXPERIMENT NO:10**

### **CIRCULAR CONVOLUTION**

#### **AIM**

Write a MATLAB program to obtain circular convolution of two sequences.

#### **THEORY**

Circular Convolution of two sequences  $x[n]$  and  $y[n]$ , each of length  $N$  is given by ,

$$p[n] = x[n] \otimes y[n].$$

If the length of sequence is not equal, zero padding is done to get the maximum length among the two. The resulting convolved signal would be zero outside the range  $n = 0, 1, \dots, N-1$ .

#### **STEPS**

1. Interactively accept two input sequences from the user
2. Make the lengths of the sequences equal by padding zeros
3. Perform circular convolution by using matrix method
4. Repeat the same using builtin functions
5. Plot both signals and verify the result

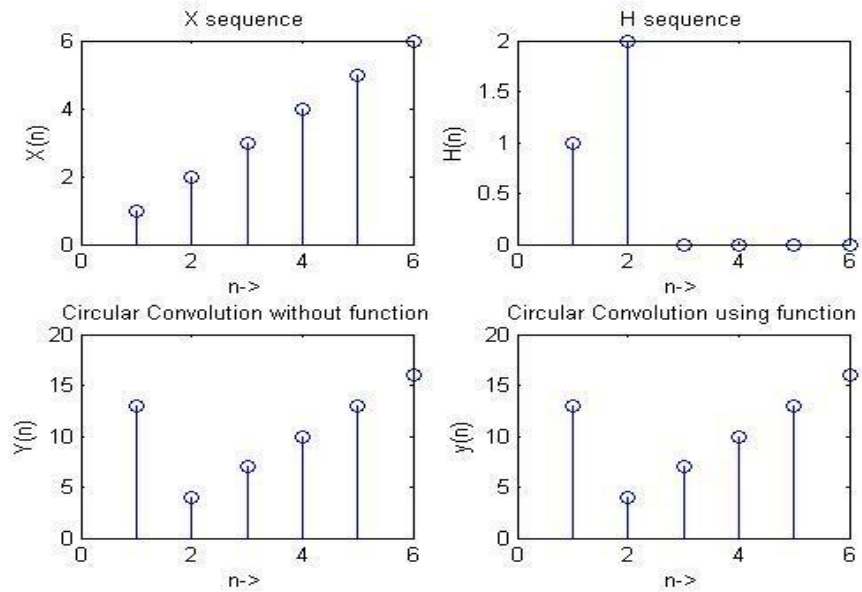
#### **MATLAB FUNCTIONS USED:**

- `length()`: Finding the length of a sequence
- `fliplr ()`: Flip matrix left to right. `fliplr(A)` returns  $A$  with columns flipped in the left-right direction, that is, about a vertical axis.
- `zeros()`: Defines a sequence of zeros
- `cconv()`: Circular convolution function

#### **SAMPLES OF EXPECTED OUTPUT**

The first sequence,  $X = [1 \ 2 \ 3 \ 4 \ 5 \ 6]$

The second sequence,  $H = [1 \ 2]$



## RESULT

The MATLAB program to find the circular convolution of two input sequences is executed and output is verified using inbuilt function.

## EXPERIMENT NO: 11

### IIR FILTER DESIGN

#### AIM

Write a MATLAB program to design an IIR Filter (Butterworth and Chebyshev).

#### THEORY

The Butterworth filter is a type of signal processing filter designed to have as flat a frequency response as possible in the pass band so that it is also termed a maximally flat magnitude filter. The frequency response of the Butterworth filter is maximally flat (has no ripples) in the passband and rolls off towards zero in the stopband. When viewed on a logarithmic Bode plot the response slopes off linearly towards negative infinity. A first-order filter's response rolls off at  $-6$  dB per octave ( $-20$  dB per decade) (all first-order lowpass filters have the same normalized frequency response). A second-order filter decreases at  $-12$  dB per octave, a third-order at  $-18$  dB and so on. Butterworth filters have a monotonically changing magnitude function with  $\omega$ , unlike other filter types that have non-monotonic ripple in the passband and/or the stopband.

Chebyshev filters are analog or digital filters having a steeper roll-off and more pass band ripple (type I) or stop band ripple (type II) than Butterworth filters. Chebyshev filters have the property that they minimize the error between the idealized and the actual filter characteristic over the range of the filter, but with ripples in the passband. Because of the passband ripple inherent in Chebyshev filters, filters which have a smoother response in the passband but a more irregular response in the stopband are preferred for some applications.

These are the most common Chebyshev filters. The gain (or amplitude) response as a function of angular frequency  $\omega$  of the  $n$ th order low pass filter is

$$G_n(\omega) = |H_n(j\omega)| = \frac{1}{\sqrt{(1 + \varepsilon^2 T_n^2)\left(\frac{\omega}{\omega_0}\right)}}$$

Where  $\varepsilon$  is the ripple factor,  $\omega_0$  is the cutoff frequency and  $T_n()$  is a Chebyshev polynomial of the  $n^{\text{th}}$  order.

#### INSTRUCTIONS

1. Accept the butterworth filter design parameters for the user (Pass band and stop band cut-off frequencies, and attenuations)
2. For each type of lowpass, bandpass, high pass filter, find the filter order and implement the filter design using inbuilt functions
3. Repeat the same for Chebyshev filter design
4. Plot filter magnitude and phase response

**MATLAB FUNCTIONS USED**

- `abs()`: `abs(X)` returns an array Y such that each element of Y is the absolute value of the corresponding element of X.
- `gridon()` : Grid lines for 2-D and 3-D Plots
- `input()` : Request user input
- `buttord()`: Butterworth filter order selection.[
  - `[N, Wn] = buttord(Wp, Ws, Rp, Rs)` returns the order N of the lowest order digital Butterworth filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to  $\pi$  radians/sample)
- `butter()`: Butterworth digital and analog filter design.
  - `[B,A] = butter(N,Wn)` designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be  $0.0 < Wn < 1.0$ , with 1.0 corresponding to half the sample rate.
- `cheby1()`: Chebyshev Type I digital and analog filter design.
  - `[B,A] = cheby1(N,R,Wp)` designs an Nth order lowpass digital Chebyshev filter with R decibels of peak-to-peak ripple in the passband. CHEBY1 returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The passband-edge frequency Wp must be  $0.0 < Wp < 1.0$  with 1.0 corresponding to half the sample rate.
- `cheblord()` Chebyshev Type I filter order selection.
  - `[N, Wp] = cheblord(Wp, Ws, Rp, Rs)` returns the order N of the lowest order digital Chebyshev Type I filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to  $\pi$  radians/sample).
- `freqz()` : Digital filter frequency response.
  - `[H,W] = freqz(B,A,N)` returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter

**SAMPLES OF EXPECTED OUTPUT:****BUTTERWORTH****a) Low pass**

The pass band frequency  $w_p = 1500$

The stop band frequency  $w_s = 3000$



The pass band attenuation  $r_p = .15$

The stop band attenuation  $r_s = 60$

The sampling frequency  $f_s = 70000$

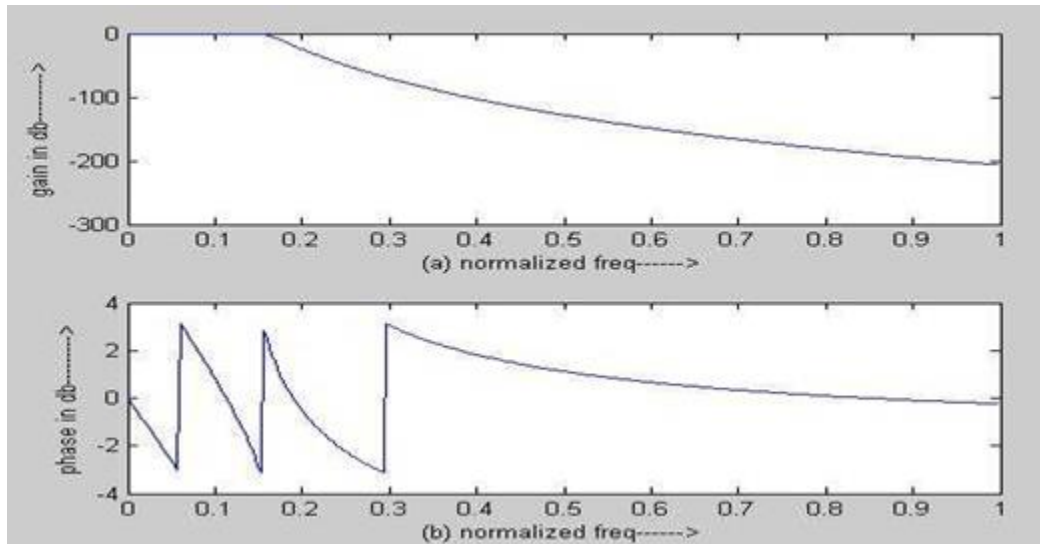
$n = 8$

$wn = 0.1151$

$n2 = 5$

$wn2 = 0.1000$

$N = 512$



### b) High pass

The pass band frequency  $w_p = 2000$

The stop band frequency  $w_s = 3500$

The pass band attenuation  $r_p = 0.2$

The stop band attenuation  $r_s = 40$

The sampling frequency  $f_s = 80000$

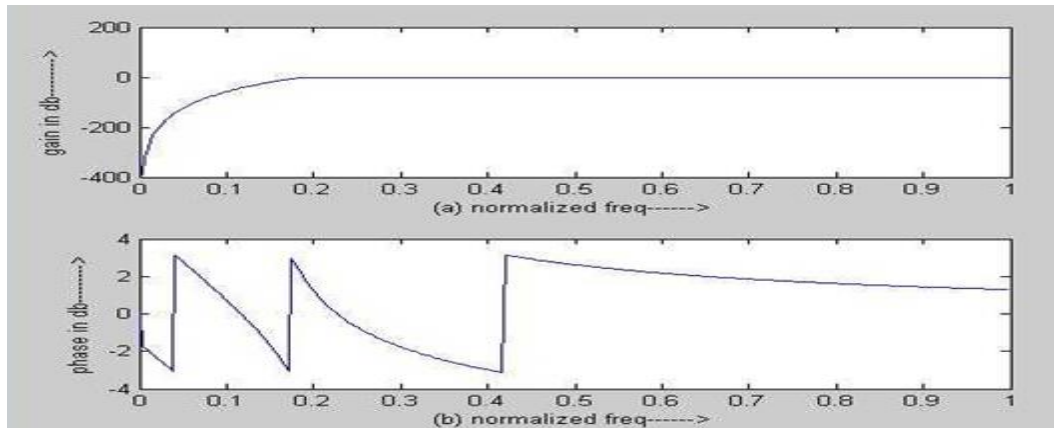
$n = 8$

$wn = 0.1151$

$n2 = 5$

$wn2 = 0.1000$

$N = 512$



### c) Bandpass

The pass band frequency  $\omega_p = 1500$

The stop band frequency  $\omega_s = 2000$

The pass band attenuation  $r_p = 0.3$

The stop band attenuation  $r_s = 40$

The sampling frequency  $f_s = 90000$

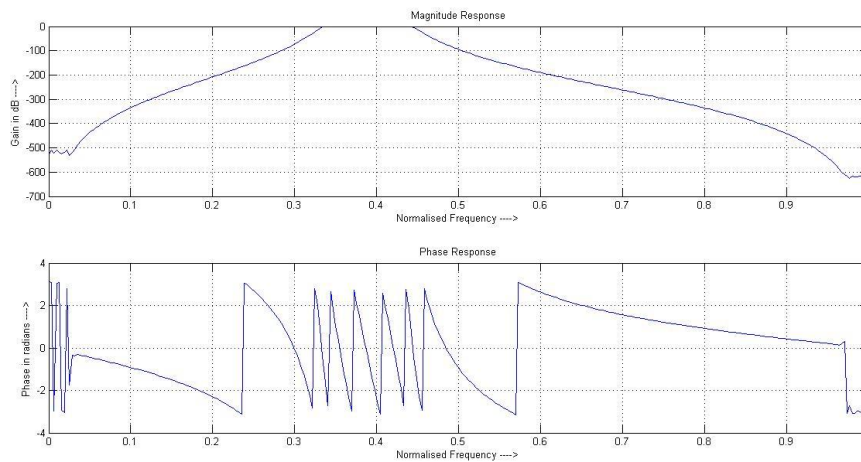
$n = 8$

$\omega_n = 0.1151$

$n_2 = 5$

$\omega_{n2} = 0.1000$

$N = 512$



### CHEBYSHEV

#### a) Lowpass

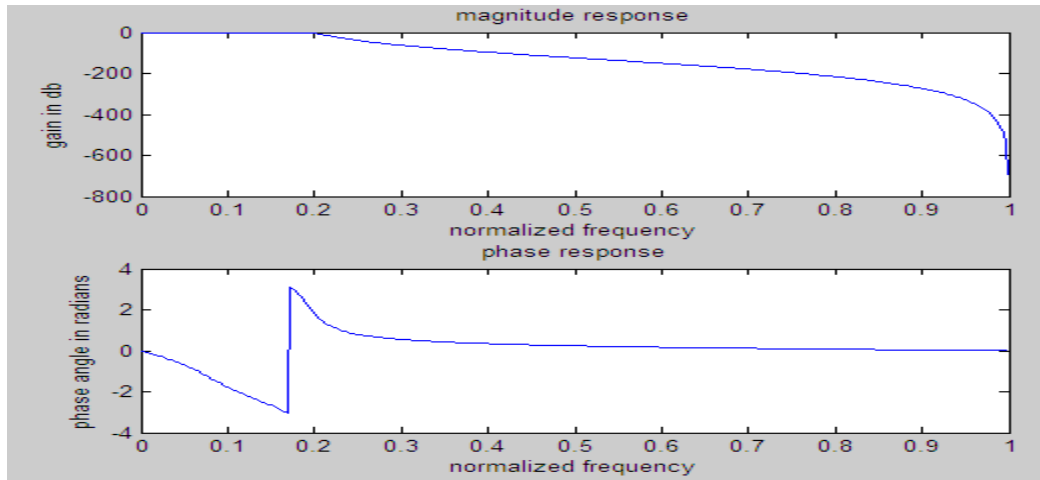
The passband attenuation=2

The stopband attenuation=20

The stopband edge frequency=200

The passband edge frequency=300

The sampling frequency=2000



## b)Highpass

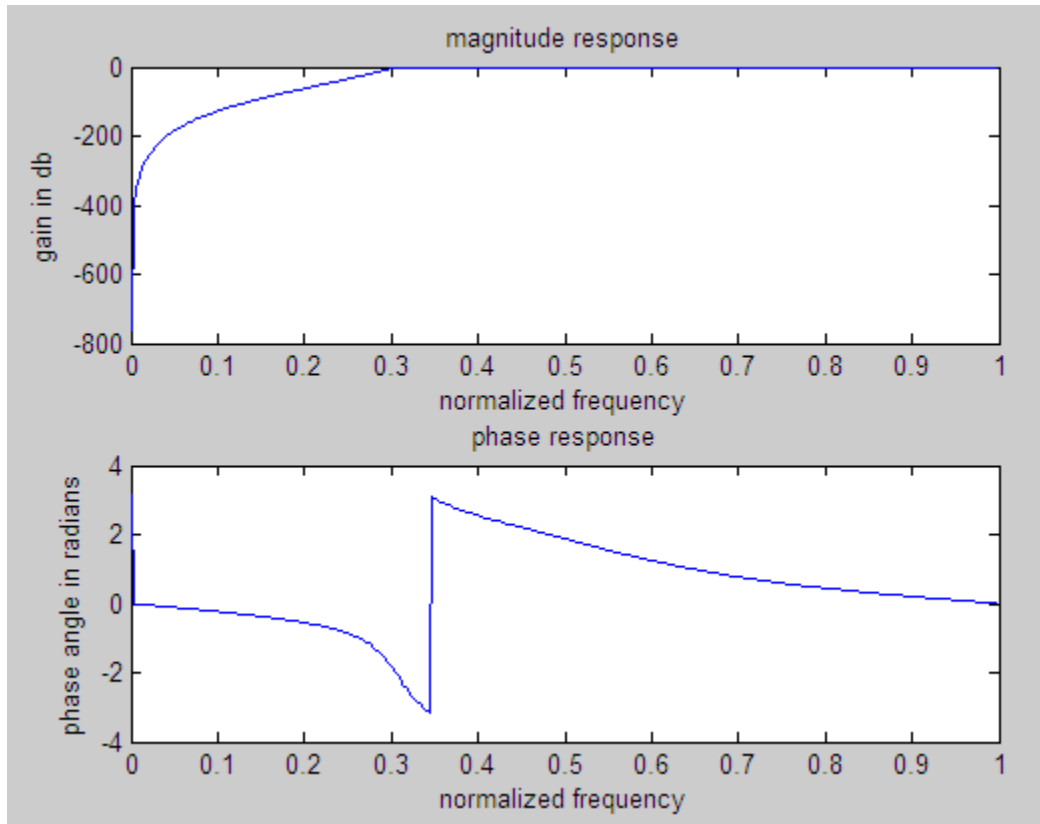
The passband attenuation=2

The stopband attenuation=20

The stopband edge frequency=200

The passband edge frequency=300

The sampling frequency=2000



### c)Bandpass

The passband attenuation=2

The stopband attenuation=20

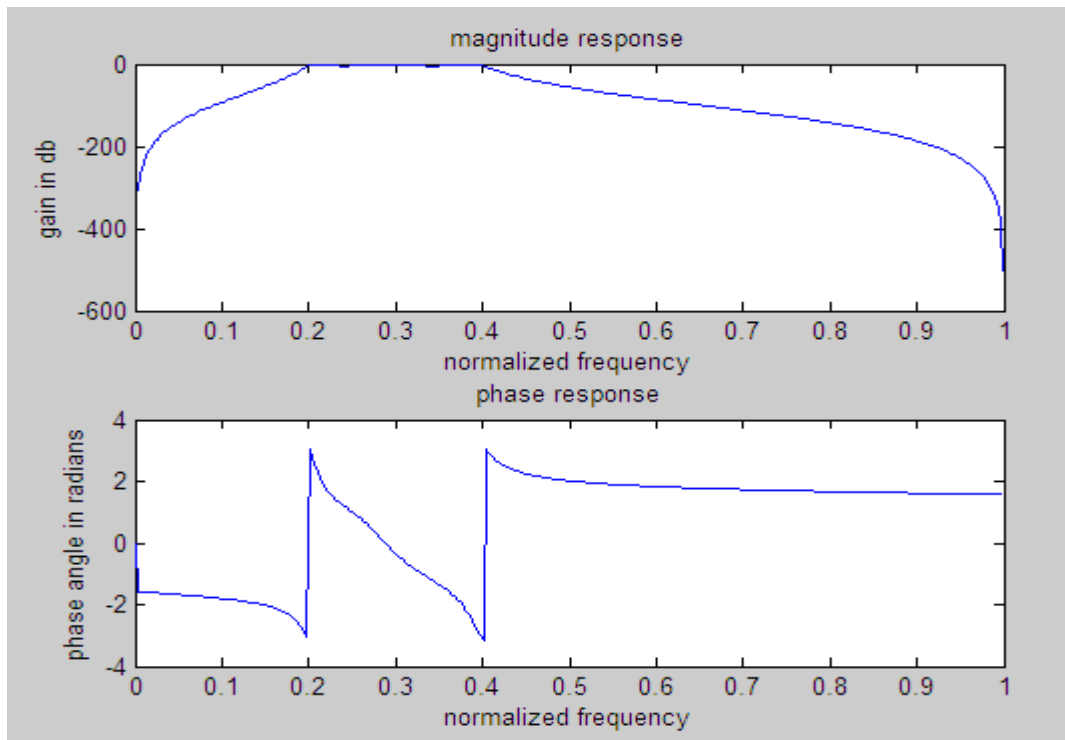
The upper stopband edge frequency=500

The lower stopband edge frequency=100

The upper passband edge frequency=400

The lower passband edge frequency=200

The sampling frequency=2000

**RESULT:**

Designed Butterworth and Chebyshev Filters.

## EXPERIMENT NO: 12

### FIR FILTER DESIGN

#### AIM:

Write a MATLAB program to design FIR Filter using Window method.

#### THEORY

The rectangular window sequence is given by

$$\omega_R(n) = 1 \text{ for } -\frac{N-1}{2} \leq n \leq \frac{N-1}{2} \text{ 0 otherwise}$$

In the design of FIR filters using any window technique, the order can be calculated using the formula given by

$$N = \frac{-20 \log(\sqrt{\delta_p \delta_s})}{14.6 \frac{(f_s - f_p)}{F_s}} - 13$$

Where  $\delta_p$  is the pass band ripple,  $\delta_s$  is the stop band ripple,  $f_p$  is the pass band frequency,  $f_s$  is the stop band frequency and  $F_s$  is the sampling frequency.

The equation for hamming window is given by

$$\omega_H(n) = 0.54 + \frac{0.46 \cos 2\pi n}{N-1} \text{ for } -\frac{N-1}{2} \leq n \leq \frac{N-1}{2} \text{ 0, otherwise}$$

The hanning window sequence can be obtained by substituting  $\alpha=0.5$  in the raised cosine window function.

#### STEPS

1. Generate the ideal impulse response (Sinc function).
2. Call window functions for - Rectangular, Hamming, Hanning and Kaiser windows
3. Multiply them with ideal impulse response.
4. Plot the frequency responses of these windowed sequences to obtain the FIR filter responses of the respective windows.

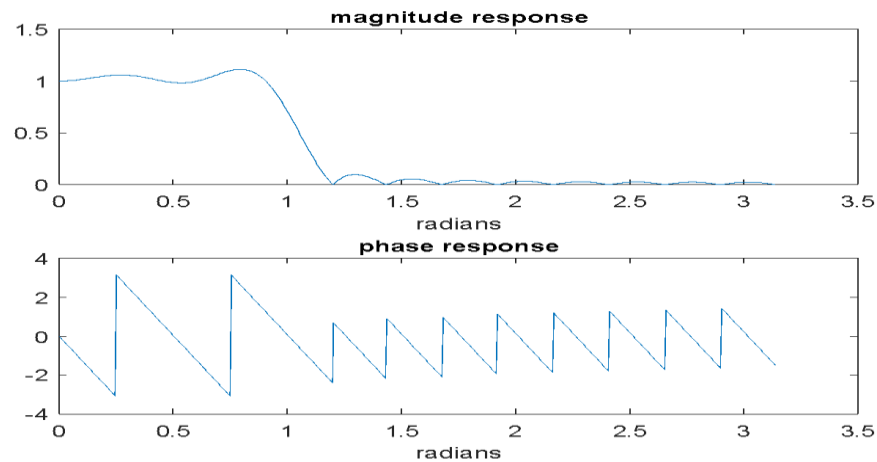
## MATLAB Functions To be familiarised:

length (), fft(X), fft(X,N), abs(X), abs(X), hann(N), hamming(N): Hamming window, rectwin(N)

## SAMPLES OF EXPECTED OUTPUT

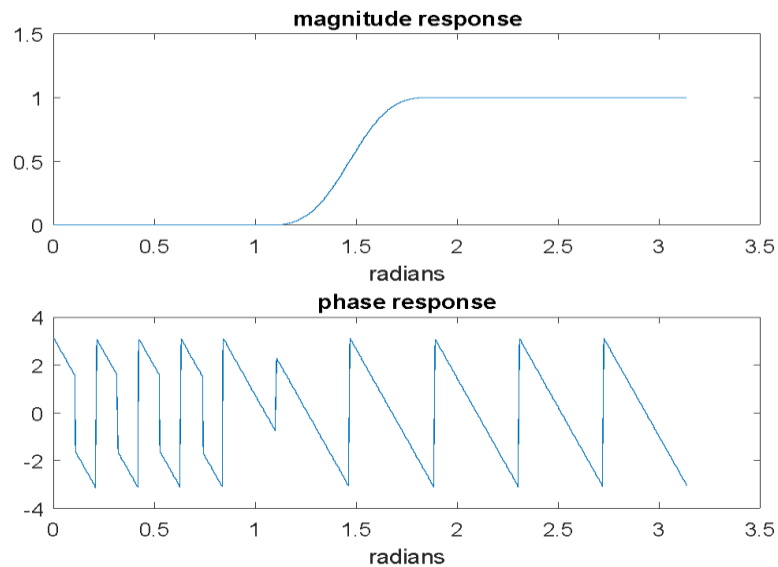
### 1. Low pass Filter using Rectangular window

Order of the filter : 25, Cutoff frequency : 500, Sampling frequency : 3000



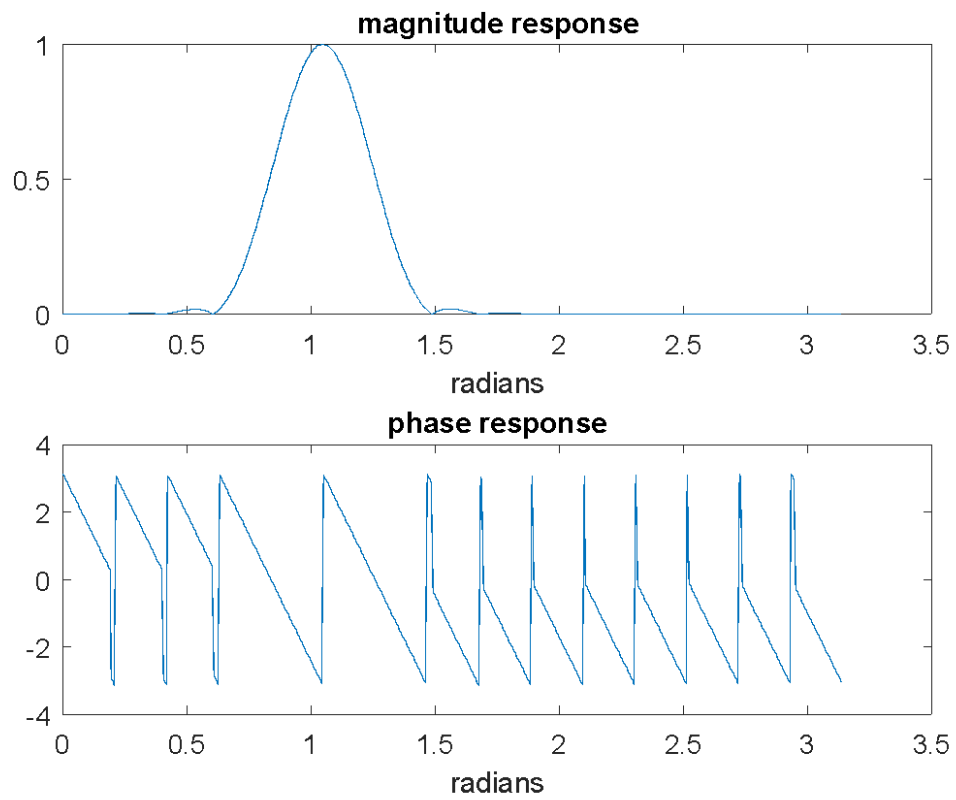
### 2. High pass Filter using Hamming window

order of the filter : 30, cutoff frequency : 700, Sampling frequency : 3000



### 3. Band pass Filter using Hanning window

Order of the filter: 30, Cut-off frequency1 : 460, Cut-off frequency2 :540, Sampling frequency: 3000





# **PART B**

## **Control System Using MATLAB**

## **EXPERIMENT 1: Familiarisation Of MATLAB commands using Control System Design**

### **AIM**

To familiarize with MATLAB commands used in control system design.

### **ABOUT COMMANDS**

The first step in the control design process is to develop appropriate mathematical models of the system to be controlled. These models may be derived either from physical laws or experimental data. Visually, a dynamic is represented by state- space and transfer function. Here we get familiarized with various control system design commands such as tf, tf2zp, zp2tf, tf2ss, ss2tf, pzmap, feedback, bode, rlocus.

### **RESULT:-**

Various commands of MATLAB used in control system design were familiarized using help command.

## **EXPERIMENT 2: Representation Of System in MATLAB: State Space Representation & Transfer Function Representation**

**AIM:-**To represent a system in MATLAB in both state space and transfer function form.

### **THEORY:-**

Transfer function models describe the relationship between the inputs and outputs of a system using a ratio of polynomials. The model order is equal to the order of the denominator polynomial. The roots of the denominator polynomial are referred to as the model poles. The roots of the numerator polynomial are referred to as the model zeros. The parameters of a transfer function model are its poles, zeros and transport delays. A state-space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations or difference equations. State variables are variables whose values evolve through time in a way that depends on the values they have at any given time and also depends on the externally imposed values of input variables. Output variables' values depend on the values of the state variables. The "state space" is the Euclidean space in which the variables on the axes are the state variables. The state of the system can be represented as a vector within that space.

### **Steps:-**

1. Define transfer function.
2. Define state space form
3. To Transform a given Transfer Function to State Space Model using MATLAB.
4. Display the functions.

### **MATLAB functions and methods to be familiarised**

Use MATLAB Help to familiarise basic functions: tf(num,den), ss(A,B,C,D,Ts), tf2ss(b,a), clc, clear all, close all, input

### **EXPECTED RESULT:-**

System has been represented using MATLAB using both state space and transfer function form.

Transfer Function of given system is

Transfer Function:

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

---


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

Corresponding State Space Model A, B, C, D are:

$$A = \begin{bmatrix} -0.5000 & -0.5000 & -1.0000 & 0 \\ 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1.0000 & 1.5000 & 1.000 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

## **EXPERIMENT 3: Stability Analysis Using Bode Plot, Root Locus & Their Pole-Zero-Gain Representation.**

**AIM:-** To obtain stability analysis of a system using bode plot, root locus, and pole-zero gain representation.

### **THEORY:-**

Root locus analysis is a graphical method for examining how the roots of a system change with variation of a certain system parameter, commonly a gain within a feedback system. This is a technique used as a stability criterion in the field of classical control theory developed by Walter R. Evans which can determine stability of the system. The root locus plots the poles of the closed loop transfer function in the complex  $s$ -plane as a function of a gain parameter. The root locus of a feedback system is the graphical representation in the complex  $s$ -plane of the possible locations of its closed-loop poles for varying values of a certain system parameter. The points that are part of the root locus satisfy the angle condition. The value of the parameter for a certain point of the root locus can be obtained using the magnitude condition.

Bode plot is the graphical tool for drawing the frequency response of a system. It is represented by two separate plots, one is the magnitude vs frequency and the other one is phase vs frequency. The magnitude is expressed in dB and the frequency is generally plotted in log scale. One of the advantages of the Bode plot in  $s$ -domain is that the magnitude curve can be approximated by straight lines which allow the sketching of the magnitude plot without exact computation.

A pole-zero plot is a graphical representation of a rational transfer function in the complex plane which helps to convey certain properties of the system such as: Stability, Causal system / anticausal system, Region of convergence (ROC), Minimum phase / non minimum phase. A pole zero plot shows the location in the complex plane of the poles and zeros of the transfer function of a dynamic system, such as a controller, compensator, sensor, equalizer, filter, or communications channel. By convention, the poles of the system are indicated in the plot by an X while the zeros are indicated by a circle or O. A pole-zero plot can represent either a continuous-time (CT) or a discrete-time (DT) system. For a CT system, the plane in which the poles and zeros appear is the  $s$  plane of the Laplace transform. In this context, the parameter  $s$  represents the complex angular frequency, which is the domain of the CT transfer function. For a DT system, the plane is the  $z$  plane, where  $z$  represents the domain of the Z-transform.

### **Steps:-**

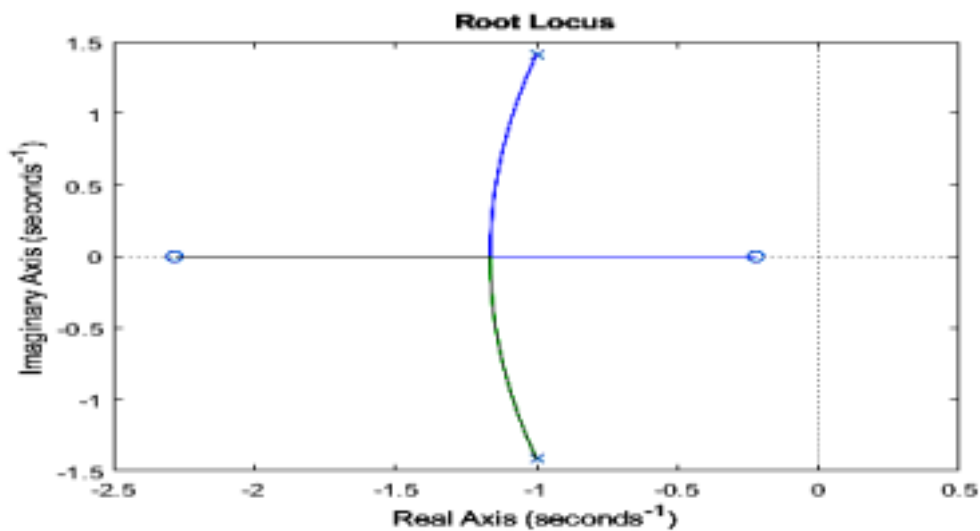
1. Plot the root locus of the transfer function
2. Draw the Bode Plot for the given transfer function
3. Find Gain Margin, Phase Margin, Gain Cross over Frequency, Phase Cross over Frequency, Resonant Peak, Resonant Frequency, Bandwidth
4. Draw the pole-zero gain plot

### **MATLAB functions and methods to be familiarised**

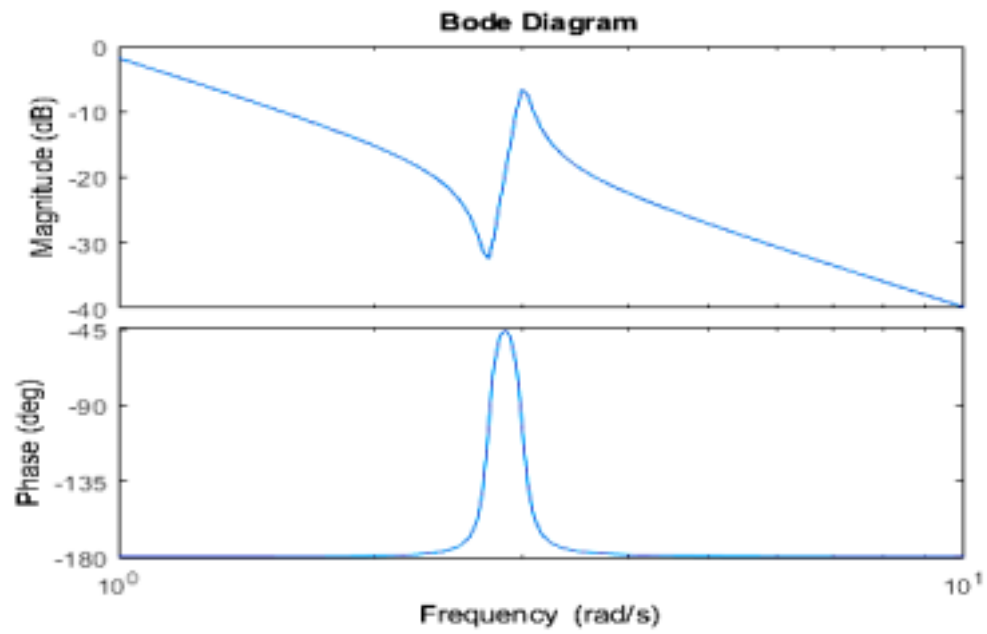
Use MATLAB Help to familiarise basic functions: clc, clear all, close all, input, figure, rlocus, title, grid, logspace, title, grid, bode, margin, tf(num,den), pzmap(g).

### **EXPECTED RESULTS:-**

ROOT LOCUS: - Typical plot for Root locus of the transfer function  $G(s)=1/(S^3+8S^2+17S)$



BODE PLOT:- Typical plot forfor the given transfer function  $G(S)=1/S(S^2+2S+3)$



## **EXPERIMENT 4: Modelling and Analysis Of First Order System**

**AIM:-** To obtain closed loop step and impulse response of a first order unity feedback system.

### **THEORY:-**

In closed loop control, the control action from the controller is dependent on feedback from the process in the form of the value of the process variable (PV). A closed loop controller, therefore, has a feedback loop which ensures the controller exerts a control action to manipulate the process variable to be the same as the "Reference input" or "set point". For this reason, closed loop controllers are also called feedback controllers.

Impulse calculates the unit impulse response of a dynamic system model. For continuous-time dynamic systems, the impulse response is the response to a Dirac input  $\delta(t)$ . For discrete-time systems, the impulse response is the response to a unit area pulse of length  $T_s$  and height  $1/T_s$ , where  $T_s$  is the sample time of the system. (This pulse approaches  $\delta(t)$  as  $T_s$  approaches zero.) For state-space models, impulse assumes initial state values are zero. Impulse (sys) plots the impulse response of the dynamic system model sys. This model can be continuous or discrete, and SISO or MIMO. The impulse response of multi-input systems is the collection of impulse responses for each input channel. The duration of simulation is determined automatically to display the transient behaviour of the response.

### **Steps:-**

1. Plot impulse response.
2. Plot step response.

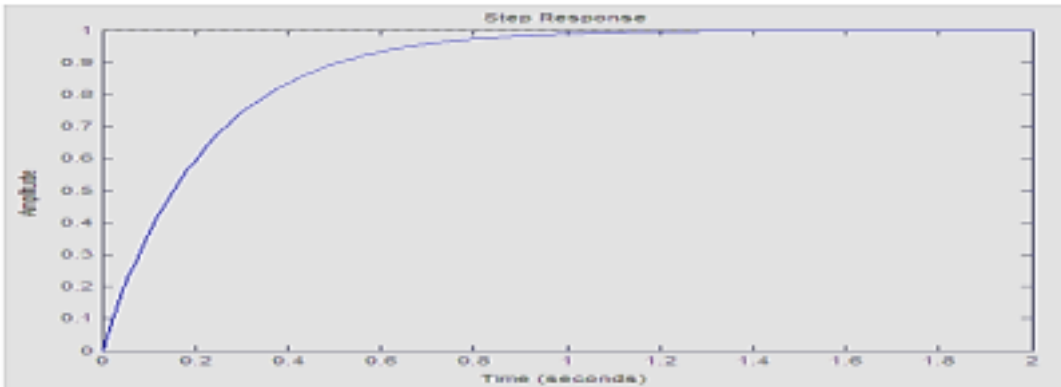
### **MATLAB functions and methods to be familiarised :-**

Use MATLAB Help to familiarise basic functions: clc, clear all, close all, input, tf(a,b); feedback(g,1), Subplot, step(sys), impulse(sys).

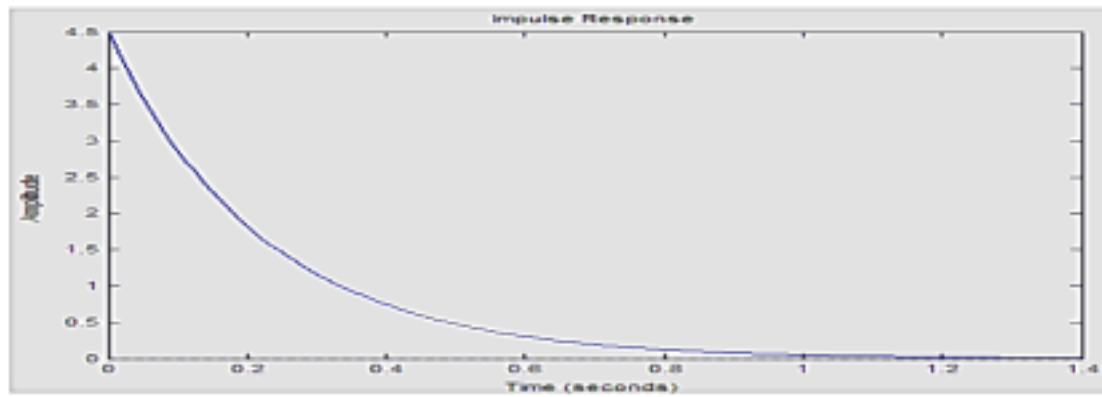
### **EXPECTED RESULTS:-**

#### **Step response:-**





### Impulse Response:-



## **EXPERIMENT 5: Realization of a compensator design**

**AIM:-** To design phase lead, phase lag compensators with the given parameters

### **THEORY:-**

There are three types of compensators — lag, lead and lag-lead compensators. A compensating network is one which makes some adjustments in order to make up for deficiencies in the system. Compensating devices may be in the form of electrical, mechanical, hydraulic etc. Most electrical compensators are RC filter. The simplest network used for compensator is known as lead, lag network.

A system which has one pole and one dominating zero (the zero which is closer to the origin than all other zeros is known as dominating zero.) is known as lead network. If we want to add a dominating zero for compensation in control system then we have to select lead compensation network.

A system which has one zero and one dominating pole (the pole which is closer to origin than all other poles is known as dominating pole) is known as lag network. If we want to add a dominating pole for compensation in control system then, we have to select a lag compensation network.

### **Steps:-**

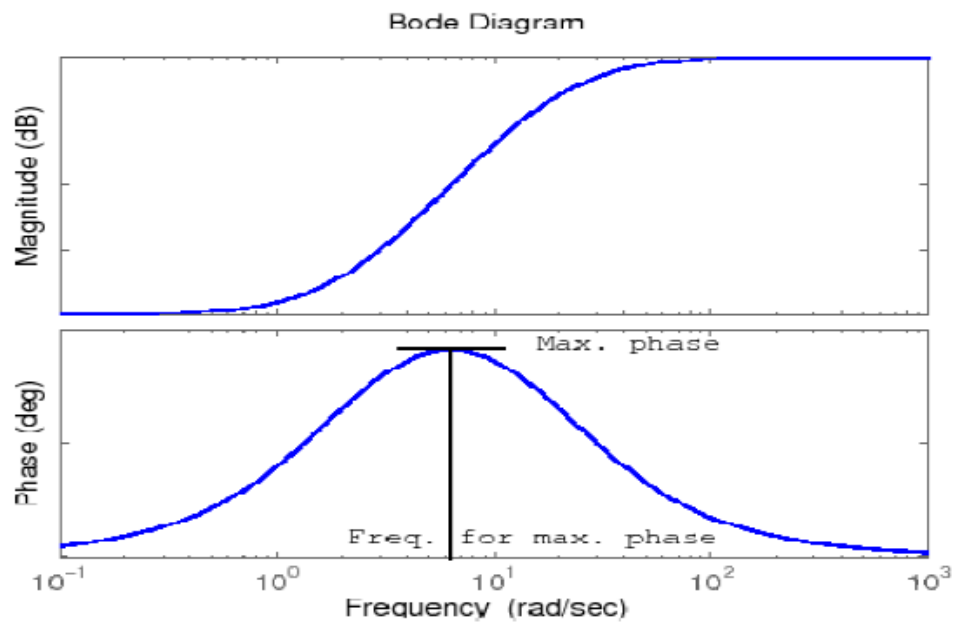
1. Design a phase lead compensator
2. Draw the bode plot
3. Find  $\alpha$ ,  $\phi$  degree
4. Draw bode plot of compensated system
5. Find compensated gain margin.
6. Design a lag compensator
7. Draw the bode plot of uncompensated system and compensated system
8. Display the compensated gain margin.

### **MATLAB functions and methods to be familiarised :-**

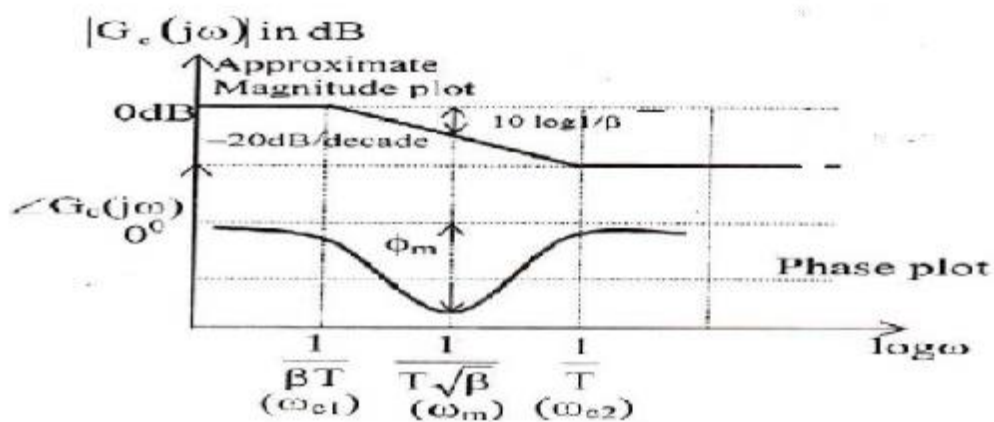
Use MATLAB Help to familiarise basic functions: `clc`, `clear all`, `close all`, `input`, `tf(a,b)`, `Subplot`, `bode`, `title`, `margin`, `display`, `conv`

## EXPECTED RESULTS:-

Lead Compensator



Lag compensator



# **PART C**

## **PC Based Control**

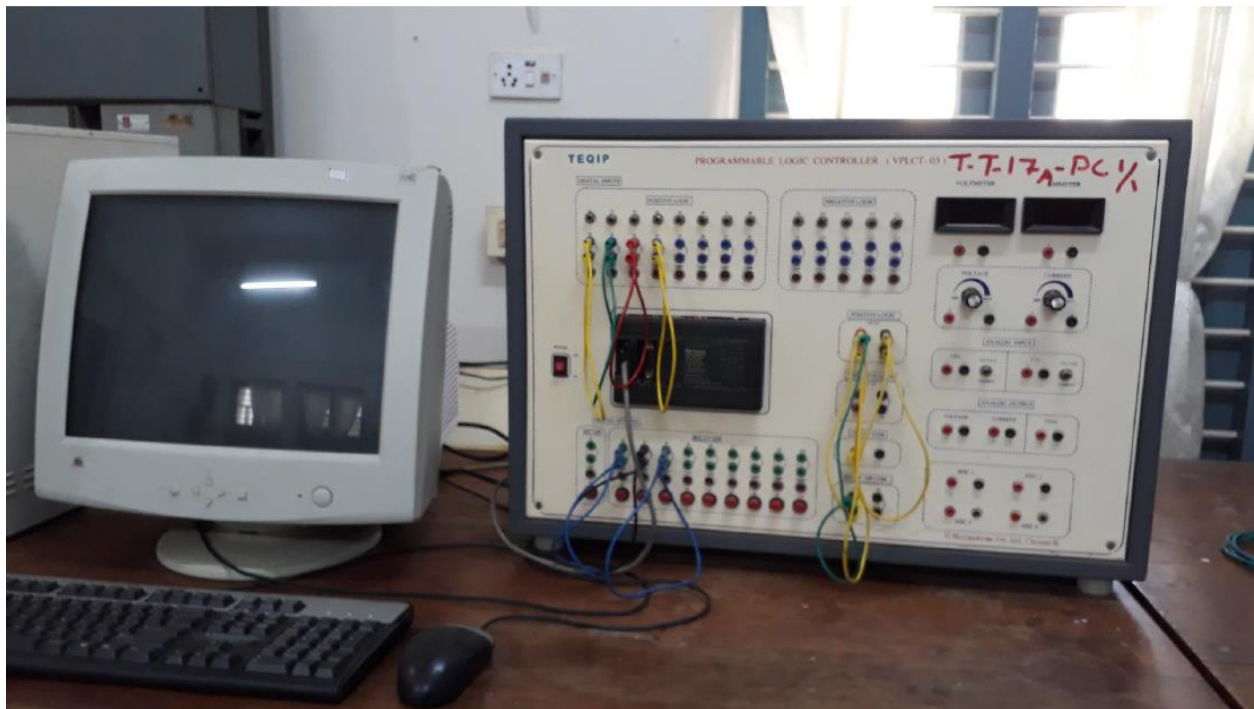
## EXPERIMENT 1: FAMILIARIZATION OF PLC INSTRUCTION SET

### Aim

Familiarise basic instruction set in ladder programming

### Programmable Logic Controller

Programmable logic controller (PLC) is a specialized control computer for industrial and commercial applications. It can be programmed to do a variety of logical functions. It replaces electromechanical relays logic elements with a solid-state digital computer with stored program, which can emulate the interconnection of many relays to perform certain logical tasks. A PLC has many input terminals, through which it interprets 'high' and 'low' logical states from sensors and switches. It also has many output terminals, through which it outputs 'high' and 'low' signals to power lights, solenoids, contactors, small motors and other devices lending themselves to on/off control




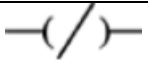


VPLCT-03 from Vi Microsystems

## **Ladder Logic Programming Basics**

Ladder logic (also known as ladder diagram or LD) is a programming language used to program a PLC. It is a graphical programming. A ladder logic is drawn between the two lines (power line/Signal line) Each line of code is called a rung, The PLC takes one rung and executes that and then goes to the next line. Each PLC has its own specific IDE to create ladder programme. PLC in the lab is programmed using a Versa Pro.

Basic ladder logic symbols

Symbol	Logic
	Normally open contact
	Normally closed contact
	Normally open output
	Normally closed output

## **To do**

Create ladder programs for basic logic gates like AND, OR, XOR etc and verify output

# **PART D**

## **LabVIEW based Virtual Instrumentation**

## **EXPERIMENT 1: INTRODUCTION TO LABVIEW**

### **Aim**

Familiarise:

- a. Block diagram and front panel windows in LabVIEW
- b. Controls and indicators in LabVIEW
- c. Generate and display sinewave in LabVIEW

### **LabVIEW**

LabVIEW is a graphical programming tool developed by National Instruments. In LabVIEW, we programme by connecting different functional blocks.

LabVIEW programs are called Virtual Instruments (VI). A VI has two parts: a front panel and a block diagram. Front panel is to build user interface. Front panel has options to place controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates. After you build the user interface, you add graphical code using VIs and structures to control the front panel objects. The block diagram contains this graphical source code.

### **Steps**

Familiarize controls and indicators

1. Open LabVIEW and create a blank VI. There will be two windows: front panel and block diagram
2. Place some indicators and controls in front panel window
3. Connect the controls and indicators using connecting wires in block diagram window
4. Observe the change in indicators as controls are changed

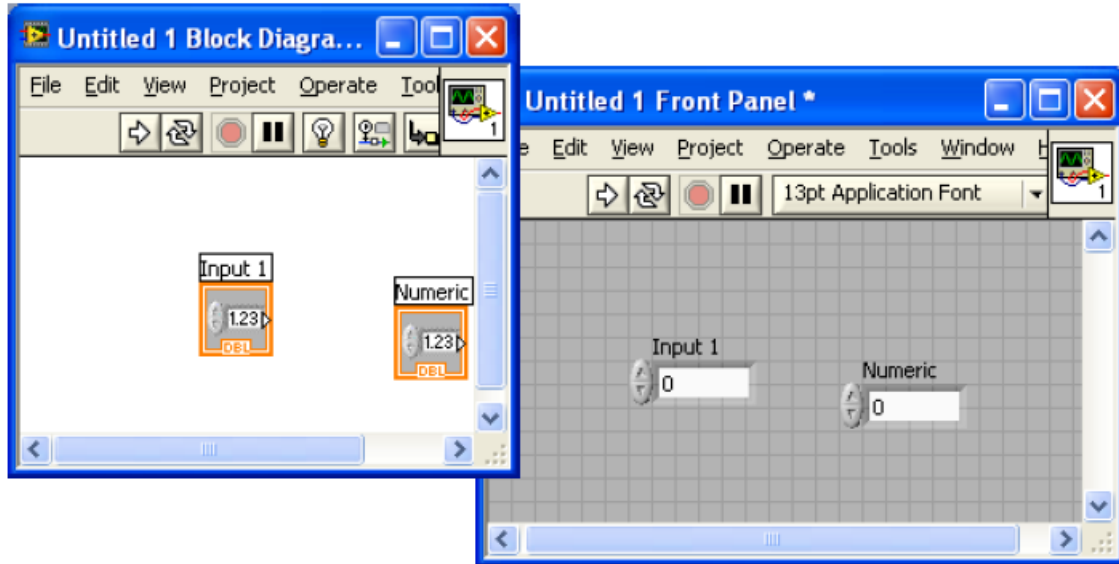
Generate a sinewave and display the output waveform

1. In front panel, place waveform graph indicator to display the output
2. Switch to block diagram panel and create waveform generation function. Waveform generation functions are available in signal processing toolbox.
3. Place controls to change the amplitude and frequency of waveform

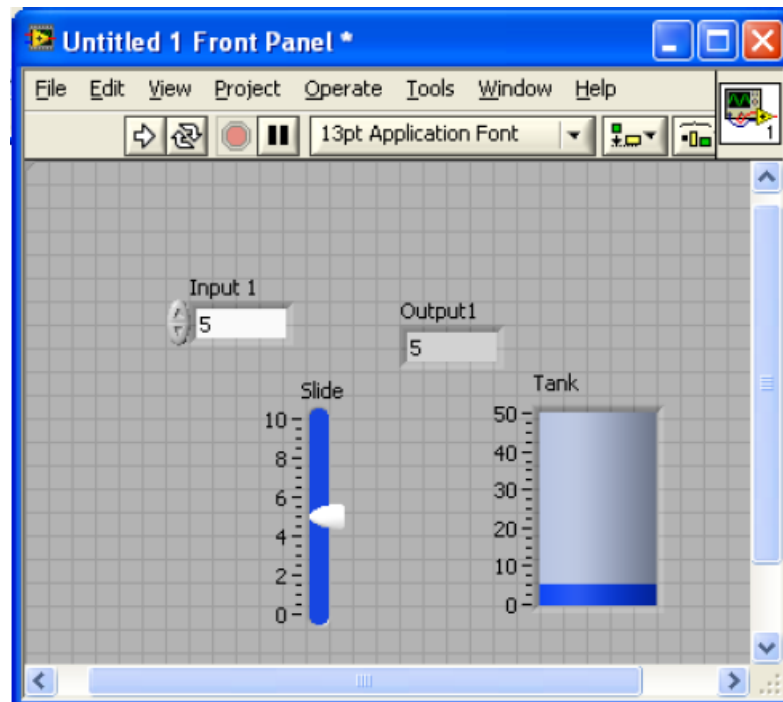


## Expected Output

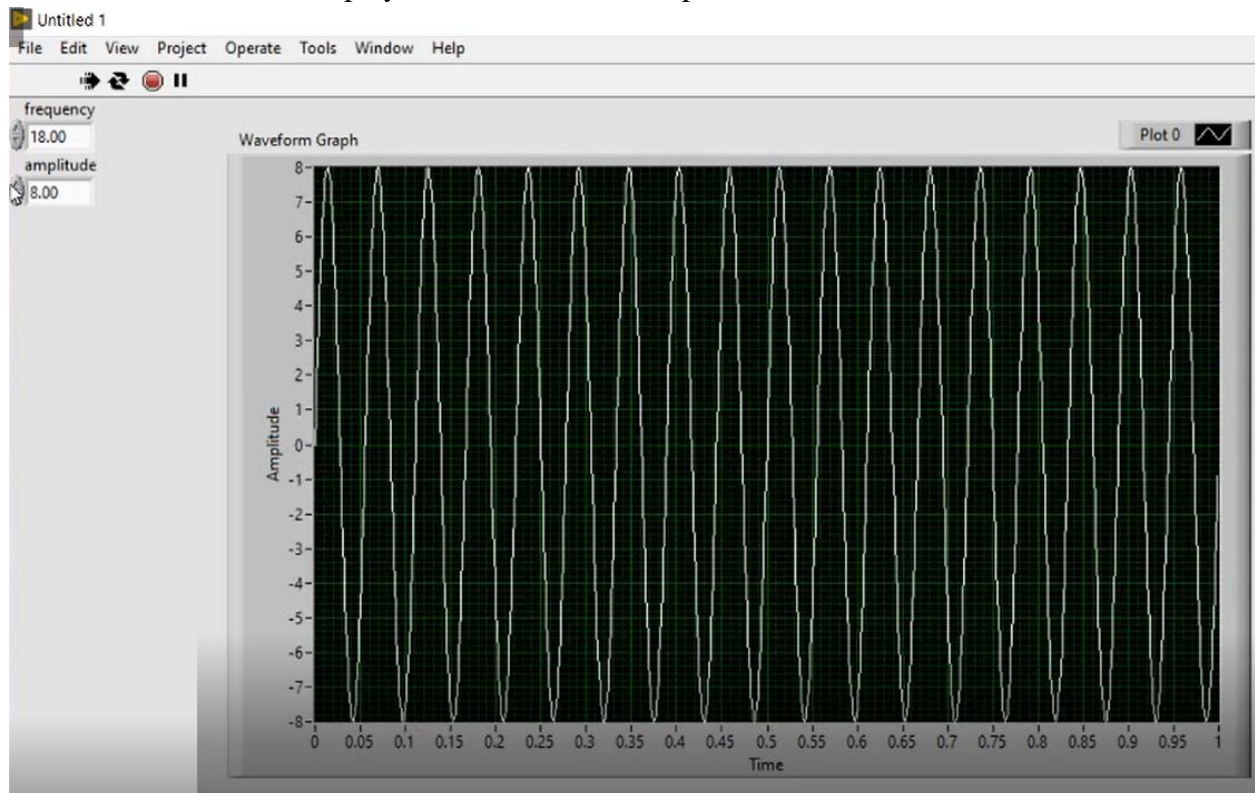
- Familiarisation of front panel and block diagram panel with numeric indicators and controls



- Front panel with slider control and tank indicator



c. Sine waveform displayed in LabVIEW front panel



## EXPERIMENT 2: RESPONSE OF A SECOND ORDER SYSTEM USING LABVIEW

### Aim

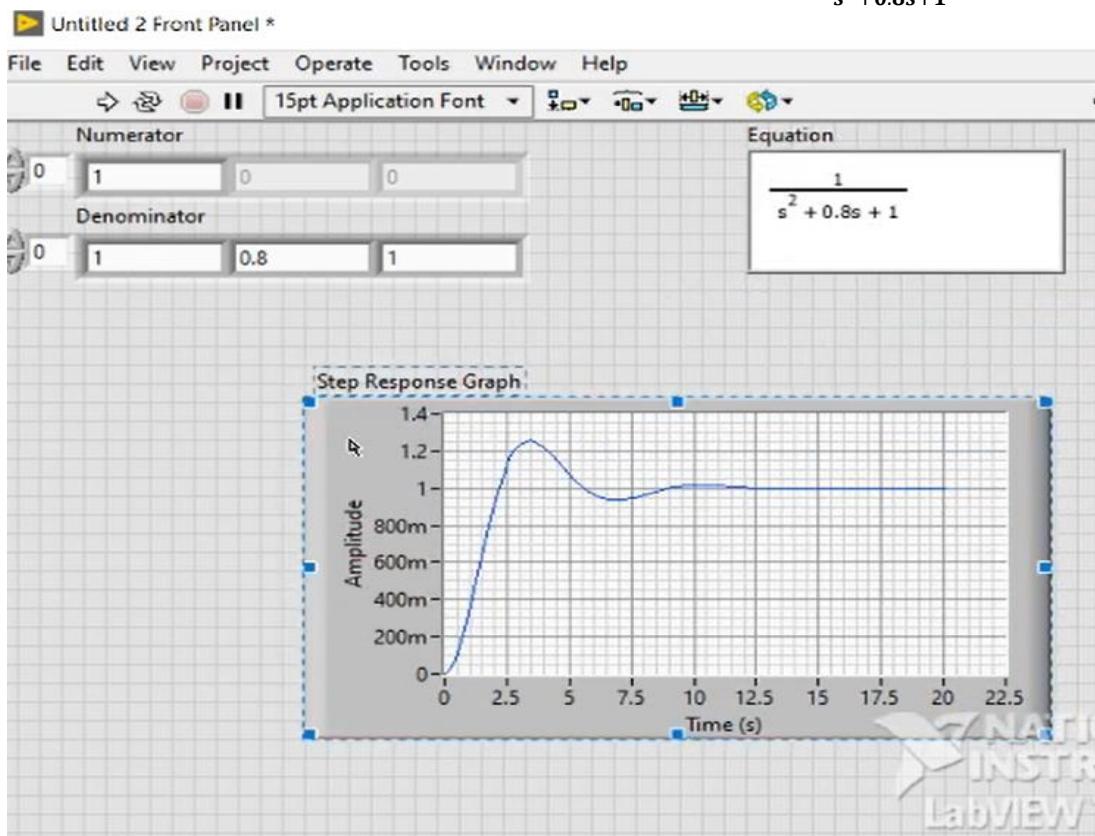
Obtain the step response of a second order system specified by the transfer function  $H(s)$

### Steps

1. In block diagram panel, create block to generate transfer function (Already available as subVI in control and simulation>>control design>>model construction>>CD construct transfer function model.vi)
2. Create controls to change numerator and denominator of transfer function
3. Use draw transfer function block to display the transfer function (subVI in control and simulation>>control design>>model construction>>CD draw transfer function equation.vi) and create an indicator for it
4. To obtain the step response, use the corresponding VI (subVI in control and simulation>>control design>>Time Response>>CD step response.vi)
5. Create a graph indicator to view step response

### Expected Output

step response of the system specified by the transfer function  $H(S) = \frac{1}{s^2 + 0.8s + 1}$



## **EXPERIMENT 3: FAMILIARIZATION OF DAQ CARD**

### **Aim**

To set up hardware and to develop a LabVIEW VI for monitoring the speed of a dc motor.

### **Steps**

1. A disc with perforations is attached to the shaft of the motor
2. An IR LED is kept ON on one side of disc and a phototransistor at the other.
3. As the motor rotates a train of pulses will be produced at the collector of the photo transistor. It can be connected to an analog input channel of the DAQ card.
4. Develop a VI in LabVIEW for data acquisition. There is an express VI called DAQ assistant for data acquisition. Set the channel number, type of the data etc. and data can be acquired.
5. For speed monitoring, train of pulses from the collector of the phototransistor is acquired and processed to the data into rpm.
6. The rpm data can be displayed numerically in a text indicator or in a meter indicator display.

### **Expected Output**

A GUI displaying the speed of motor in rpm