S J P N Trust's

Hirasugar Institute of Technology, Nidasoshi

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Digital Signal Processing

Laboratory Manual

18EEL67

Faculty Incharge

Prof. A. U. Neshti & Prof. M. P. Yenagimath

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

VISION

"To be the centre of excellence in teaching and learning to produce the competent & socially responsible professionals in the domain of Electrical & Electronics Engineering."

MISSION

I. To educate students with core knowledge of Electrical and Electronics Engineering to excel in their professional career.

II. To develop problem solving skills, professional skills and ethical values among the students for the betterment of mankind.

III. To prepare technically competent and socially responsible Electrical Engineer to serve the future needs of the society.

Program Educational Objectives (PEOs):

Engineering Graduates will be able to:

PEO1: Achieve successful professional career in Electrical Engineering and allied disciplines.

PEO2: Pursue higher studies and continuously engage in upgrading the professional skills.

PEO3: Demonstrate professional & ethical values, effective communication skills and teamwork to solve issues related to profession, society and environment.

Program Specific Outcomes (PSOs):

Engineering Graduates will be able to:

PSO1: Apply knowledge & competencies to analyze & design Electrical & Electronics circuits, control and power systems, machines & industrial drives.

PSO2: Use software/hardware tools for the design, simulation and analysis of Electrical and Electronics Systems.

Overview

Year / Semester	3rd Year /6th Semester	**Academic Year**	2021 - 2022
Laboratory Title	Digital Signal Processing Laboratory	**Laboratory Code**	18EEL67
Total Contact Hours	42	**Exam Hours**	3 Hours
CIE Marks	40	**SEE Marks**	60

Objectives

- To explain the use of MATLAB software in evaluating the DFT and IDFT of given sequence
- To verify the convolution property of the DFT
- To design and implementation of IIR and FIR filters for given frequency specifications.
- To realize IIR and FIR filters.
- To help the students in developing software skills

Course Outcomes

The student, after successful completion of the course, will be able to

1. Show the physical interpretation of sampling theorem in time and frequency domains.
2. Evaluate the impulse response of a system.
3. Perform convolution of given sequences to evaluate the response of a system
4. Compute DFT and IDFT of a given sequence using the basic definition and/or fast methods.
5. Provide a solution for a given difference equation.
6. Design and implement IIR and FIR filters.

Prerequisites

- MATLAB programming language.
- Basic operation such as creating file, delete, copy, rename etc should be known.
- DSP algorithm operation should be understood.

Base Course

- Signals and Systems
- Digital Signal Processing

Resource Required

- MATLAB software.

Introduction

MATLAB, which stands for Matrix Laboratory, is a state-of-the-art mathematical software package for high performance numerical computation and visualization provides an interactive environment with hundreds of built in functions for technical computation, graphics and animation and is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. At its core ,MATLAB is essentially a set (a "toolbox") of routines (called "m files" or "mex files") that sit on your computer and a window that allows you to create new variables with names (e.g. voltage and time) and process those variables with any of those routines (e.g. plot voltage against time, find the largest voltage, etc). It also allows you to put a list of your processing requests together in a file and save that combined list with a name so that you can run all of those commands in the same order at some later time. Furthermore, it allows you to run such lists of commands such that you pass in data.

MATLAB Windows:

MATLAB works with through these basic windows

Command Window

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

The Current Directory Window

The Current Directory window displays a current directory with a listing of its contents. There is navigation capability for resetting the current directory to any directory among those set in the path. This window is useful for finding the location of particular files and scripts so that they can be edited, moved, renamed, deleted, etc. The default current directory is the Work subdirectory of the original MATLAB installation directory

The Command History Window

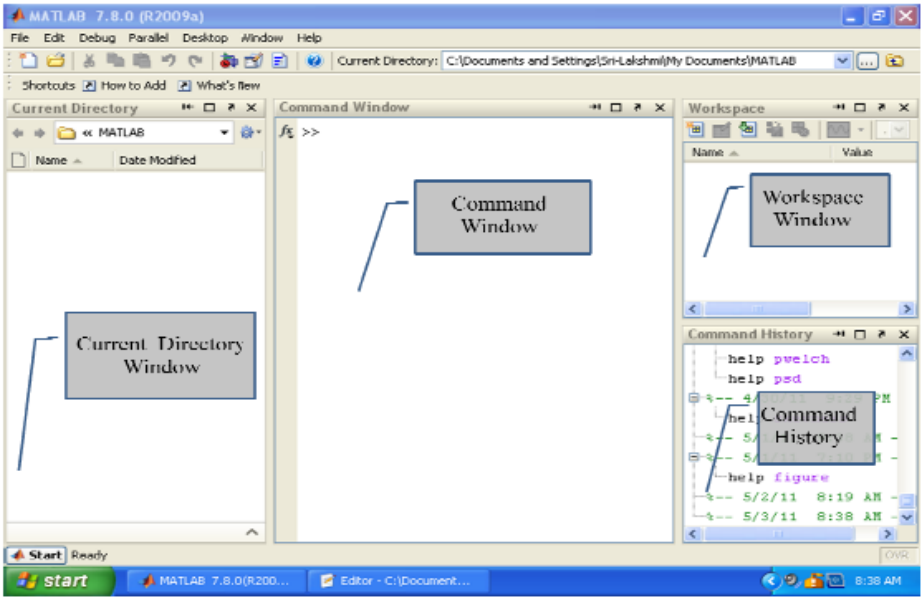
The Command History window, at the lower left in the default desktop, contains a log of commands that have been executed within the Command window. This is a convenient feature for tracking when developing or debugging programs or to confirm that commands were executed in a particular sequence during a multistep calculation from the command line.

Graphics Window

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

Edit Window

This is where you write edit, create and save your own programs in files called M files.



Input-output

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

Data Type

The fundamental data distinct data objects- integers, real numbers, matrices, character strings, structures and cells. There is no need to declare variables as real or complex, MATLAB automatically sets the variable to be real.

Dimensioning

Dimensioning is automatic in MAT required for vectors or arrays .we can find the dimensions of an existing matrix or a vector with the size and length commands.

Where to work in MATLAB?

All programs and commands can be entered either in the

a) Command window

b) As an M file using MATLAB editor

Note: Save all M files in the folder 'work' in the current directory. Otherwise you have to locate the file during compiling. Typing quit in the command prompt>> quit, will close MATLAB Development Environment. For any clarification regarding plot etc, which are built in functions type help topic i.e. help plot.

Operations on vector and matrices in MATLAB

MATLAB utilizes the following arithmetic operators;

+ Addition

- Subtraction

* Multiplication

/ Division

^ Power Operator

‘transpose

Relational operators in MATLAB

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater
>=	Greater or equal to
==	Equal to
~=	Not equal to

Basic Functions in MATLAB

1) **Plot** Syntax: plot (x, y)

Plots vector y versus vector x. If x or y is a matrix, then the vector is plotted versus the rows or columns of the matrix.

2) Stem Syntax: stem(Y)

Discrete sequence or "stem" plot.

Stem (Y) plots the data sequence Y as stems from the x axis terminated with circles for the data value. If Y is a matrix then each column is plotted as a separate series.

3) Subplot Syntax: Subplot (2 2 1)

This function divides the figure window into rows and columns.

Subplot (2 2 1) divides the figure window into 2 rows and 2 columns 1represent number of the figure.

1 (2 2 1)	**2 (2 2 2)**
3 (2 2 3)	**4 (2 2 4)**

Subplot (3 1 2) divides the figure window into 3 rows and 1 column 2 represent number of the figure

1 (3 1 1)
2 (3 1 2)
3 (3 1 3)

4) **Disp** Syntax: disp(X)

Description: disp(X) displays an array, without printing the array name. If X contains a text string, the string is displayed.Another way to display an array on the screen is to type its name, but this prints a leading "X=," which is not always desirable.Note that disp does not display empty arrays.

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

6) **ylabel** Syntax : ylabel('string')

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

7) **Title** Syntax : title('string')

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

8) **grid on** Syntax : grid on

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

General Instructions

- Go through the Instruction manual before start of the experiment.
- Strictly follow the instructions given by the Teacher/ Lab Instructor.
- It is mandatory to come to lab in a formal dress (Shirts, Trousers, ID card, and Shoes for boys). Strictly no Jeans for both Girls and Boys.
- It is mandatory to come with work book and lab record in which previous experiment should be written.
- Lab record of the present lab experiment should be corrected on the same day.
- Mobile Phones should be Switched OFF in the lab session.
- Students have to come to lab in-time. Late comers are not allowed to enter the lab.
- Prepare for the viva questions. At the end of the experiment, the lab faculty will ask the viva questions and marks are allotted accordingly.

Contents

Expt. No.	Experiment
1	Verification of Sampling Theorem both in time and frequency domains.
2	Evaluation of impulse response of a system.
3	To perform linear convolution of given sequences.
4	To perform circular convolution of given sequences using (a) the convolution summation formula (b) the matrix method and (c) Linear convolution from circular convolution with zero padding.
5	Computation of N – point DFT and to plot the magnitude and phase spectrum.
6	Linear and circular convolution by DFT and IDFT method.
7	Solution of a given difference equation.
8	Calculation of DFT and IDFT by FFT
9	Design and implementation of IIR filters to meet given specification (Low pass, high pass, band pass and band reject Filters)
10	Design and implementation of FIR filters to meet given specification (Low pass, high pass, band pass and band reject filters) using different window functions
11	Design and implementation of FIR filters to meet given specification (Low pass, high pass, band pass and band reject filters) using frequency sampling technique.
12	Realization of IIR and FIR filters.

Reference

- Proakis & Monalakis, Digital Signal processing Principles Algorithms & Applications, Pearson Education, Edition, New Delhi, 2007.
- Oppenheim & Schaffer, Discrete Time signal Processing, **PHI.** 2003.
- S. K. Mitra, Digital Signal processing, **Tata Mc-Graw Hill, 2nd** Edition, 2004.
- Lee Tan: Digital signal processing, **Elsevier publications**, 2007.
- Practical C++ programming by Oreilly.
- Introduction to programming with MATLAB for scientists and Engineers, 2e Broenkow,ML 2007.
- MATLAB and introduction with applications 2nd edition- Amos Gillat.

Evaluation Scheme

CIE Marks: 40
Internal Assessment: 16 Marks
Write-up: 03 Marks
Conduction & result: 10 Marks
Viva-voce: 03 Marks
Continues Assessment: 24
Scheme of External Examination
External Exam will be conducted for 100 Marks and obtained marks will be scaled down for 60 Marks by university

Experiment 1

1.0 Verification of Sampling Theorem both in time and frequency domains.

1.1 Aim

To verify Sampling theorem for a signal of given frequency.

1.2 Theory

Sampling is a process of converting a continuous time signal (analog signal) x (t) into a discrete time signal x[n], which is represented as a sequence of numbers. (A/D converter). Converting back x[n] into analog (resulting in $\hat{x}(t)$) is the process of reconstruction. (D/A converter). For $\hat{x}(t)$ to be exactly the same as x (t), sampling theorem in the generation of x(n) from x(t) is used. The sampling frequency fs determine the spacing between samples. Aliasing-A high frequency signal is converted to a lower frequency, results due to under sampling. Though it is undesirable in ADCs, it finds practical applications in stroboscope and sampling oscilloscopes.

Sampling theorem: Sampling theorem includes two definitions.

i. A band limit of the signal which is having finite energy, whose maximum frequency component is W Hz can be completely representing into its samples at the rate of 2W samples/sec.

ii. A band limit of the signal which is having finite energy, whose maximum frequency component is W Hz, can be completely recovered from its samples at the rate of 2W samples/sec.

Nyquist Rate Sampling: The Nyquist rate is the minimum sampling rate required to avoid aliasing, equal to the highest modulating frequency contained within the signal. In other words, Nyquist rate is equal to two sided bandwidth of the signal (Upper and lower sidebands) i.e. fs = 2W.To avoid aliasing, the sampling rate must exceed the Nyquist rate. i. e. $fs>f_{N,}$ where f_N =2W.

1.2 Algorithm

1. Select the frequency of analog signal f Hz.
2. To generate sine wave of f Hz defines a closely spaced time vector.
3. Generate the sinusoid and plot the signal.
4. Select the sampling frequency. Generate a suitable time scale for this sampling signal.
5. Sample the analog signal at the instant specified by n.
6. Modify the time vector n used for discrete simulation.
7. Reconstruct the analog signal from its discrete samples.
8. Compare the analog and reconstructed signal.
9. Repeat the values experiment for different values of f and verify reconstructed and analog signal.

1.4 MATLAB Implementation

Step 1: MATLAB can generate only discrete time signals. For an approximate analog signal xt, choose the spacing between the samples to be very small (≈0), say 50μs = 0.00005. Next choose the time duration, say xt exists for 0.05seconds. (tfinal in program) (For low frequency say <1000 Hz choose 0.05 secs, for higher choose 0.01 secs or lesser as appropriate). Now begin with the vector that represents the time base-

t = 0:0.00005:0.05;

The colon (:) operator in MATLAB creates a vector, in the above case a time vector running from 0 to 0.05 in steps of 0.00005. The semicolon (;) tells MATLAB not to display the result. Given t, the analog signal xt of frequency f is generated as (cos (ωt) =cos (2πft)):-

xt=cos(2*pi*f*t);

Where pi is recognized as 3.14 by MATLAB.

Step 2: To illustrate oversampling condition, choose sampling frequency fs_0=2.2*f. For this sampling rate T_0=1/fs_0, generate the time vector as **n1 = 0:T_0:0.05; &** over sampled discrete time signal **x1=cos (2*pi*f*n1);**

Step 3: Repeat step 2 for different sampling frequencies, i.e., fs=1.3*f & fs=2*f for under sampling and Nyquist sampling conditions.

1.5 MATLAB Program: Sampling Theorem in Time Domain

```
clc;
clear all;
close all;
tfinal=0.05;
t=0:0.00005:tfinal;
f=input('enter the analogy frequency f=');
xt=cos(2*pi*f*t);

%under sampling plot
fs1=1.3*f;
n1=0:1/fs1:tfinal;
xn=cos(2*pi*f*n1);
subplot(3,1,1);
plot(t,xt,'b',n1,xn,'r*-');
title('under sampling');
xlabel('time');
ylabel('amplitude');
```

```
%nyquist plot
fs2=2*f;
n2=0:1/fs2:tfinal;
xn=cos(2*pi*f*n2);
subplot(3,1,2);
plot(t,xt,'b',n2,xn,'r*-');
title('nyquist sampling');
xlabel('time');
ylabel('amplitude');

%over sampling plot
fs3=5*f;
n3=0:1/fs3:tfinal;
xn=cos(2*pi*f*n3);
subplot(3,1,3);
plot(t,xt,'b',n3,xn,'r*-');
title('over sampling');
xlabel('time');
ylabel('amplitude');
```

1.6 MATLAB Program: Sampling Theorem in Frequency Domain

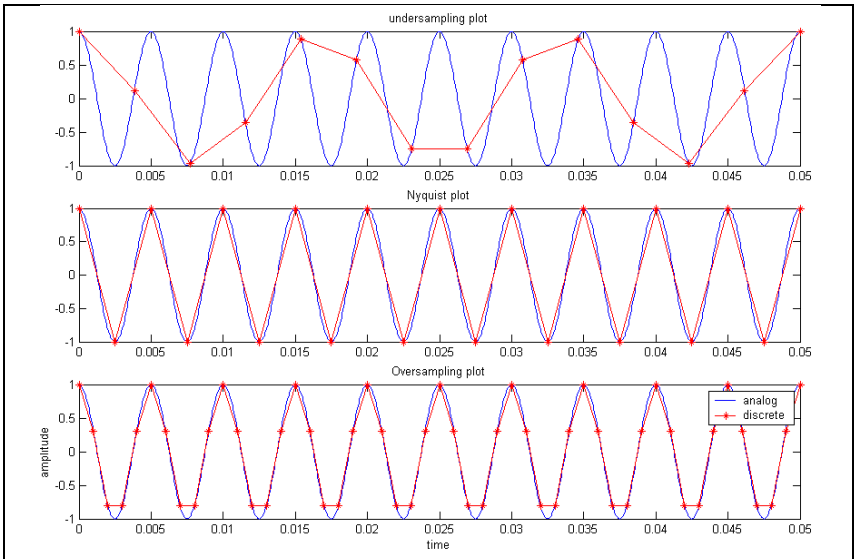
```
clc;
close all;
clear all;
f1 = input('Enter the first sine wave frequency = ');
f2= input('Enter the second sine wave frequency = ');
fn = 2*max(f1,f2);
fs = fn/2;
t = [0:1/fs:0.1];
x = cos(2*pi*f1*t)+cos(2*pi*f2*t);
xk = fft(x);
f = [0:length(xk)-1]*fs/length(xk);
figure(1);
plot(f,abs(xk));
xlabel('frequency');
ylabel('amplitude');
title('Under Sampling');
grid;
fs = fn;
t = [0:1/fs:0.1];
x = cos(2*pi*f1*t)+cos(2*pi*f2*t);
xk = fft(x);
f = [0:length(xk)-1]*fs/length(xk);
figure(2);
plot(f,abs(xk));
xlabel('frequency');
ylabel('amplitude');
title('Nyquist Rate Sampling');
grid;
fs = 2*fn;
t = [0:1/fs:0.1];
x = cos(2*pi*f1*t)+cos(2*pi*f2*t);
xk = fft(x);
f = [0:length(xk)-1]*fs/length(xk);

figure(3);
plot(f,abs(xk));
xlabel('freq');
ylabel('amplitude');
title('Over Sampling');
grid;
```

1.6 Inference

1. From the under sampling plot observe the aliasing effect. The analog signal is of 200Hz (T=0.005s). The reconstructed (from under sampled plot) is of a lower frequency. The alias frequency is computed as fd-fs1 = 200-1.3*200 = 200-260= -60Hz. This is verified from the plot. The minus sign results in an 180˚ phase shift.
2. Sampling at the Nyquist rate results in samples sin(πn) which are identically zero, i.e., we are sampling at the zero crossing points and hence the signal component is completely missed. This can be avoided by adding a small phase shift to the sinusoid. The above problem is not seen in cosine waveforms (except cos (90n)). A simple remedy is to sample the analog signal at a rate higher than the Nyquist rate. The **Fig1.2** shows the result due to a cosine signal (**x1=cos(2*pi*fd*n1);**
3. The over sampled plot shows a reconstructed signal almost similar to that of the analog signal. Using low pass filtering the wave form can be further smoothened.

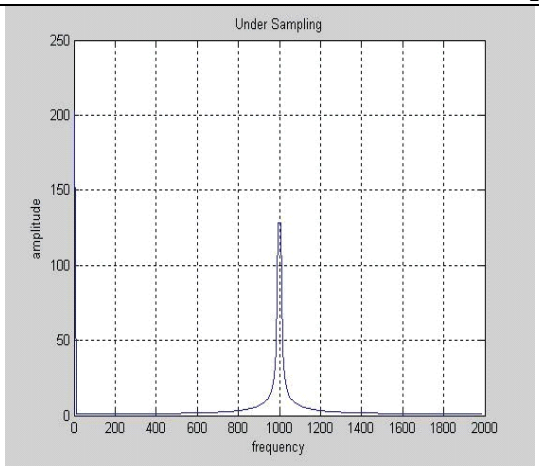
1.7 Output Waveforms

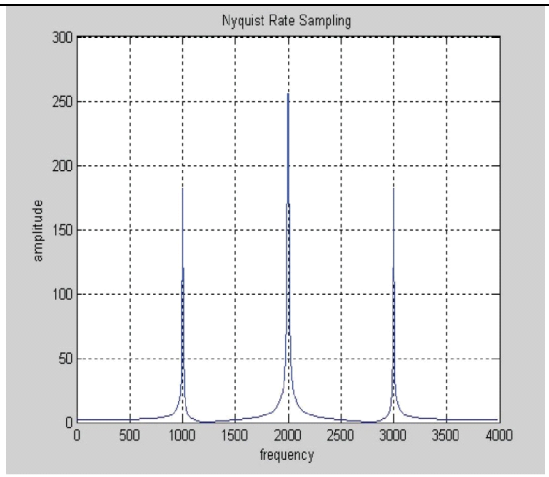
Time Domain

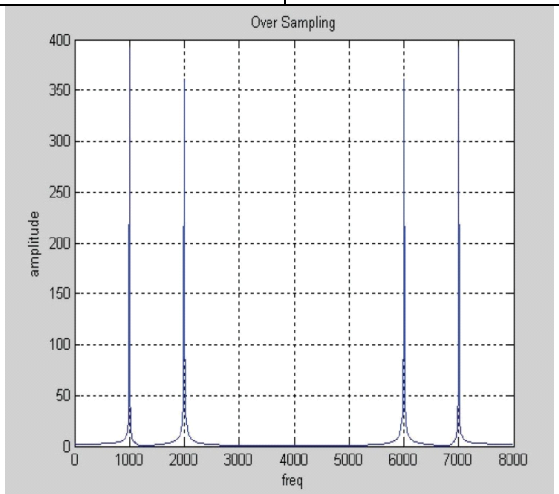


Plots of a sampled cosine wave of 200Hz

Frequency Domain







1.8 Results

1.9 Conclusion

Experiment 2

2.0 Evaluation of impulse response of a system

2.1 Aim

To write a MATLAB program to evaluate the impulse response of the system.

2.2 Theory

LTI Discrete time system is completely specified by its impulse response i.e. knowing the impulse response we can compute the output of the system to any arbitrary input. Let h[n] denotes the impulse response of the LTI discrete time systems. Since discrete time system is time invariant, its response to [n-1] will be h[n-1] .Likewise the response to [n+2] , [n-4] and [n-6] will be h[n+2], h[n-4] and h[n-6] .

From the above result arbitrary input sequence x[n] can be expressed as a weighted linear combination of delayed and advanced unit sample in the form k=+ and k=-

x[n] = x[k] h[n-k]

where weight x[k] on the right hand side denotes specifically the k th sample value of the sequence. The response of the LTI discrete time system to the sequence

x[k] [n-k] will be x[k] h [n-k].

As a result, the response y[n] of the discrete time system to x[n] will be given by

y[n] = x[k] h [n-k] ……..(1)

This can be alternately written as

y[n] = x[n-k] h [k]…….(2)

The above equation (1) and (2) is called the convolution sum of the sequences x[n] and h[n] and represented compactly as y[n] =x[n] * h[n] Where the notation * denotes the convolution sum.

Structure for Realization of Linear Time Invariant systems: Let us consider the first order system Y(n)=-a 1y(n-1)+b0 x(n) +b1 x(n-1).This realization uses separate delays(memory) for both the input and output samples and it is called as Direct form one structure. A close approximation reveals that the two delay elements contain the same input w (n) and hence the same output w(n-1).consequently these two elements can be merged into one delay. In contrast to the direct form I structure, this new realization requires only one delay for auxiliary quantity w (n), and it is more efficient in terms of memory requirements. It is called the direct form II structure and it is used extensively.

2.3 Algorithm

1. Create symbolic variables n and x, assume that they are integers.
2. Compute inverse z-transform of y
3. Get the impulse response of the system using command "impz(b,a,N)" where **'b**' is numerator Coefficients, **'a**' is denominator coefficients and number of samples is **N.**
4. Plot discrete sequence data.

2.4 MATLAB Implementation

MATLAB has an inbuilt function '***impz***' to solve difference equations numerically, given the input and difference equation coefficients (b, a).

y=impz(b,a,N)

Where x is the input sequence, y is the output sequence which is of same length as x.

2.5 Calculation

Let the Difference equation is given as $y(n) = x(n)+0.5x(n-1)+0.85x(n-2)+y(n-1)+y(n-2)$

$y(n) = x(n)+0.5x(n-1)+0.85x(n-2)+y(n-1)+y(n-2)$

$y(n) - y(n-1) - y(n-2) = x(n) + 0.5x(n-1) + 0.85x(n-2)$ Taking Z transform on both sides,

$y(z) - z^{-1} y(z)- z^{-2} y(z) = x(z) + 0.5 z^{-1} x(z) + 0.85 z^{-2} x(z)$

$y(z)[1 - z^{-1} - z^{-2}] = x(z)[1 + 0.5 z^{-1} + 0.85 z^{-2}]$

But, $h(z) = y(z)/x(z)$

$= [1 + 0.5 z^{-1} + 0.85 z^{-2}]/ [1 - z^{-1} - z^{-2}]$ By dividing we get

$H(z) = 1 + 1.5 z^{-1} + 3.35 z^{-2} + 4.85 z^{-3}$

h(n) = [1 1.5 3.35 4.85]

2.6 MATLAB Program:

```
clc;
close all;
clear all;
%difference equation of second order system
%y(n)=x(n)+.5x(n-1)+.85x(n-2)+y(n-1)+y(n-2)
b=input('enter the coefficient of x(n),x(n-1)..= ');
a=input('enter the coefficient of y(n),y(n-1)..= ');
N=input('enter the no of samples of imp response=');
[h,t]=impz(b,a,N);
subplot(2,1,1);
%figure(1)
plot(t,h);
title('plot of impulse response');
ylabel('amplitude');
xlabel('time index---->N');
subplot(2,1,2);
%figure(2)
stem(t,h);
title('plot of impulse response');
ylabel('amplitude');
xlabel('time index---->N');
disp(h);
grid on;
```

2.7 Output Waveforms

Input:

enter the coefficient of x(n),x(n-1)..= [1 0.5 0.85]

enter the coefficient of y(n),y(n-1)..= [1 -1 -1]

enter the no of samples of imp response=4

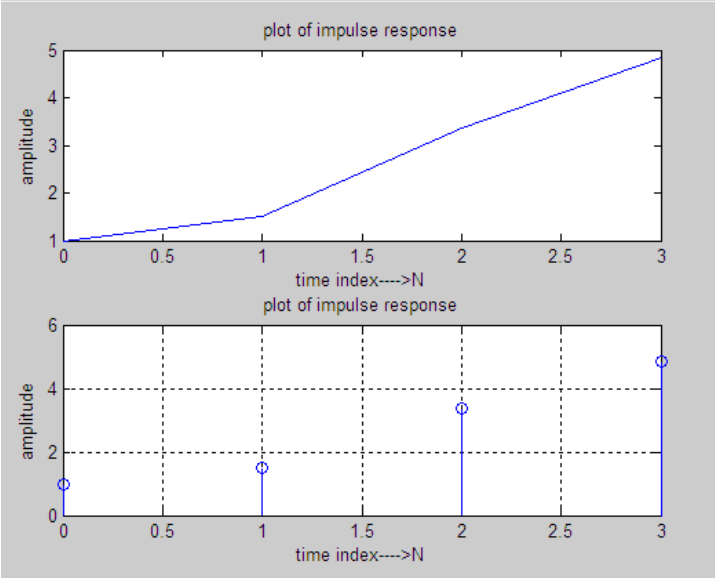
Output

1.0000

1.5000

3.3500

4.8500



2.8 Results

2.9 Conclusion

Experiment 3

3.0 Linear Convolution of two given sequences.

3.1 Aim

To obtain Linear convolution of two finite duration sequences.

3.2 Theory

Convolution is an integral concatenation of two signals. It has many applications in numerous areas of signal processing. The most popular application is 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. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal. In linear convolution length of output sequence is, length(y (n)) = length(x (n)) + length (h (n)) – 1.

Mathematical Formula:

The linear convolution of two continuous time signals x(t) and h(t) is defined by

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

For discrete time signals x(n) and h(n), is defined by

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Where x (n) is the input signal and h (n) is the impulse response of the system.

3.3 Algorithm

1. Read the input sequence, x[n] and plot.
2. Read the impulse response of the system, h[n] and plot
3. Convolve the two sequences using conv command and plot the results.

3.4 MATLAB Implementation

The timing information for a sequence is provided by another vector, say **n=-3:5;** creates a vector with values from -3 to 5 with an increment of 1. During plotting, the time vector size and the sequence size should be the same, i.e., the number of elements in the sequence x_1 and in its time vector n_1 should be the same. Similarly for x_2 and y.

3.5 Calculation

Example for finding Linear Convolution of Right sided Sequences.

$x_1 = [1,5,10,12]$ $x_2 = [5,10]$

$$x_1 = \delta(n) + 5\delta(n-1) + 10\delta(n-2) + 20\delta(n-3)$$

$$x_2 = 5\delta n + 10\delta(n-1)$$

$$z = x_1 * x_2$$

$$z = [\delta(n) + 5\delta(n-1) + 10\delta(n-2) + 20\delta(n-3)] * [5\delta(n) + 10\delta(n-1)]$$

$$z = \delta(n) * 5\delta(n) + \delta(n) * 10\delta(n-1) + 5\delta(n-1) * 5\delta(n) + 5\delta(n-1) * 10\delta(n-1)] + 10\delta(n-2) * 5\delta(n) + 10\delta(n-2) * 10\delta(n-1) + 20\delta(n-3) * 5\delta(n) + 20\delta(n-3) * 10\delta(n-1)$$

On simplification we get,

$$z = 5\delta(n) + 35\delta(n-1) + 100\delta(n-2) + 200\delta(n-3) + 200\delta(n-4)$$

$$z = \{5,35,100,200,200\}$$

Example for finding Linear Convolution of both sided sequences.

X1 = [1, 2, 3, 2, 1, 3, 4]

X2 = [2, -3, 4, -1, 0, 1]

$$X1 = \delta(n+3) + 2\delta(n+2) + 3\delta(n+1) + 2\delta(n) + 1\delta(n-1) + 3\delta(n-2) + 4\delta(n-3)$$

$$X2 = 2\delta(n+1) - 3\delta(n) + 4\delta(n-1) - 1\delta(n-2) + 0\delta(n-3) + 1\delta(n-4)$$

Z= X1 * X2

On Simplification, we get

$$Z = 2\delta(n+4) + 1\delta(n+3) + 4\delta(n+2) + 2\delta(n+1) + 9\delta(n-1) + 6\delta(n) + 3\delta(n-2) + 2\delta(n-3) + 15\delta(n-4) - 3\delta(n-5) + 3\delta(n-6) + 4\delta(n-7)$$

Z= {2, 1, 4, 2, 6, 9, 3, 2, 15, -3, 3, 4}

3.6 MATLAB Program: Linear Convolution of Right Sided Sequences

```
clc;
clear all;
close all;
x1=input('enter the first sequence x1(n)=');
x2=input('enter the second sequence x2(n)=');
y=conv(x1,x2);
disp('linear convolution of x1&x2 is=');
disp(y);
%graphical display
subplot(2,2,1);
stem(x1);
xlabel('n');
ylabel('x1(n)');
title('plot of x1(n)');
subplot(2,2,2);
stem(x2);
xlabel('n');
ylabel('x2(n)');
title('plot of x2(n)');
subplot(2,1,2);
stem(y);
xlabel('n');
ylabel('y(n)');
title('convolution output');
```

3.7 MATLAB Program: Linear Convolution of Both Sided Sequences

```
clc;
close all;
clear all;
x1=input('enter the first sequence x1(n)=');
x2=input('enter the second sequence x2(n)=');
n1=-3:3;
n2=-1:4;
ybegin=n1(1)+n2(1);
yend=n1(length(x1))+n2(length(x2));
ny=[ybegin:yend];
y=conv(x1,x2);
disp('linear convolution of x1&x2 is =');
disp(y);
subplot(2,2,1);
stem(n1,x1);
xlabel('n');
ylabel('x1(n)');
title('plot of x1(n)');
subplot(2,2,2);
stem(n2,x2);
xlabel('n');
ylabel('x2(n)');
title('plot of x2(n)');
subplot(2,1,2);
stem(ny,y);
xlabel('n');
ylabel('y(n)');
title('convolution output');
```

3.8 Output Waveforms

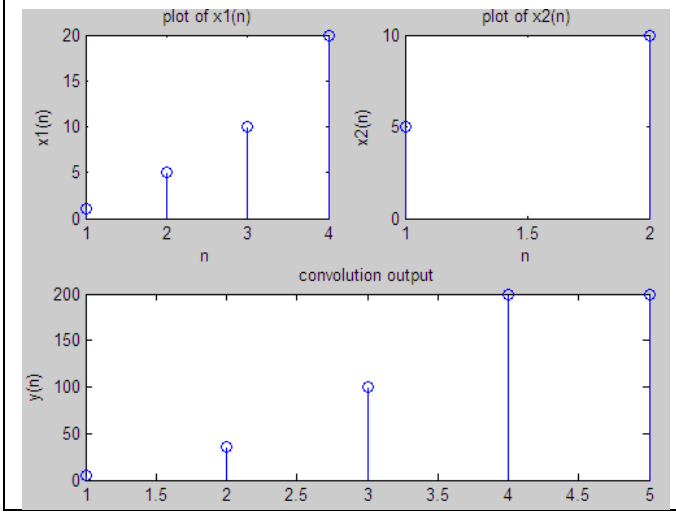
1. Linear Convolution of Right Sided Sequences

Input:

enter the first sequence x1(n)=[1 5 10 20]
enter the second sequence x2(n)=[5 10]

Output:

linear convolution of x1&x2 is=
5 35 100 200 200



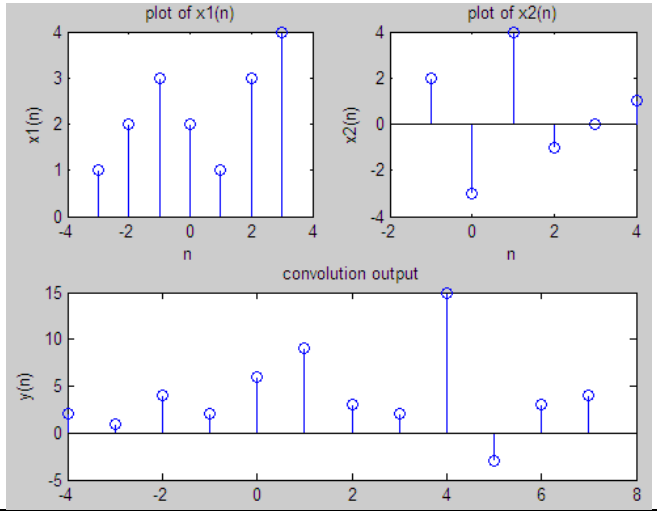
2. Linear convolution of both sided sequence

Input:

enter the first sequence x1(n)=[1 2 3 2 1 3 4]
enter the second sequence x2(n)=[2 -3 4 -1 0 1]

Output:

linear convolution of x1&x2 is =
2 1 4 2 6 9 3 2 15 -3 3 4



3.9 Results

3.10 Conclusion

Experiment 4

4.0 (a) Circular Convolution of two given sequences using summation formula.

(b) Circular Convolution of two given sequences using matrix method.

(c) Linear convolution from circular convolution with zero padding.

4.1 (a) Aim

To obtain circular convolution of two finite duration sequences.

(b) Aim

To write MATLAB program to find circular convolution by matrix method.

(c) Aim

To find linear convolution from circular convolution using MATLAB program with zero padding.

4.2 Theory

a) Circular Convolution of two given sequences using summation formula.

- As seen in the last experiment, the output y[n] of a LTI (linear time invariant) system can be obtained by convolving the input x[n] with the system's impulse response h[n]. The above linear convolution is generally applied to aperiodic sequences. Whereas the Circular Convolution is used to study the interaction of two signals that are periodic.
- The linear convolution sum is $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k]$. To compute this equation, the linear convolution involves the following operations:

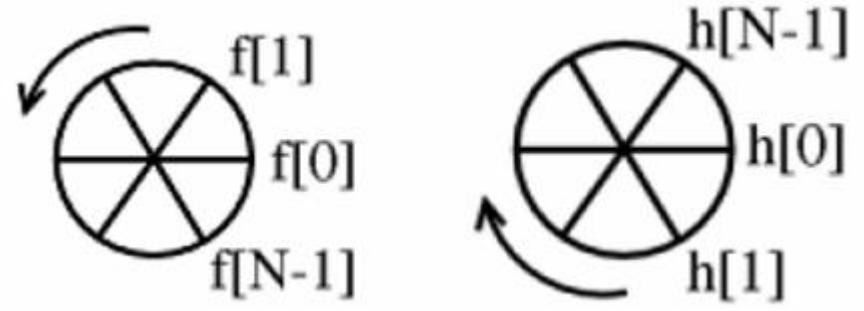
- Folding- fold h[k] to get h[-k] (for y[0])
- Multiplication – $v_k[n]$ = x[k] × h[n-k] (both sequences are multiplied sample by sample)
- Addition- Sum all the samples of the sequence $v_k[n]$ to obtain y[n]
- Shifting – the sequence h[-k] to get h[n-k] for the next n.

- The circular convolution sum is $y[n] = x[n](N)h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[\langle n-k \rangle_N]$ where the index $\langle n-k \rangle_N$ implies circular shifting operation and $\langle -k \rangle_N$ implies folding the sequence circularly.
- Steps for circular convolution are the same as the usual convolution, except all index calculations are done "mod N" = "on the wheel".

- Plot f [m] and h [−m] as shown in Fig. 4.1. (use f(m) instead of x(k))
- Multiply the two sequences
- Add to get y[m]
- "Spin" h [−m] n times Anti Clock Wise (counter-clockwise) to get h[n-m].

- Where x[n] and h[n] can be both finite and infinite duration sequences. If infinite sequences, they should be periodic, and the N is chosen to be at least equal to the period. If they are finite

sequences N is chosen as >= to max (xlength, hlength). Whereas in linear convolution N>= xlength+hlength-1.



Plotting of f(m) and h(-m) for circular convolution

Let $x_1(n)$ and $x_2(n)$ are finite duration sequences both of length N with DFT's $X_1(k)$ and $X_2(k)$. Convolution of two given sequences $x_1(n)$ and $x_2(n)$ is given by the equation,

$$x_3(n) = \text{IDFT}[X_3(k)] \text{where } X_3(k) = X_1(k)\, X_2(k)$$

b) Circular Convolution of two given sequences using matrix method.

Matrix Method for Convolution: Due to the importance of Discrete Fourier Transform (DFT) in signal processing application, it is critical to have an efficient method to compute this algorithm. DFT operates on a N -point sequence of numbers, referred to as x(n) . The value x(n) is presented in time domain data and usually can be taught as a uniformly sampled version of a finite period of a continuous function f (x) . The DFT of x(n) sequence is transformed to X(k) in frequency domain representation employing by using Discrete Fourier Transform. The functions x(n) and X(k) is generally represented in complex signal form, given by

$$\sum_{n=0}^{N-1} x(n)\, e - \frac{j2\pi kn}{N} = X(K)$$

Where x(n) is the input time domain representation and N is the number of input to the DFT. The value n represents the discrete time-domain index and k is the normalized frequency domain index. The description of efficient computation is discussed on DFT methods since the IDFT and DFT consumes the same type of computational algorithm. From the computation of each value of k , it is observed that direct computation of X(k) involves N complex multiplications (4N real multiplications) and N −1 complex additions (4N − 2 real additions). Eventually, to compute all N values of the DFT requires N2 complex multiplications and N2 − N complex additions. The multiplication of two discrete time signals in discrete Fourier transform is equivalent to the circular convolution of their sequences in time domain. For x(n) and h(n) signal convolution is express as:

$$\sum_{n=0}^{N-1} x(n)h((m-n))N = y(m)$$

Here the term h(m-n)N indicates the circular convolution. The convolution in time domain of two signal x and h is perform by multiplying its discrete fourier transform and the converting it in time domain by inverse discrete fourier transform. The equation of DFT is the summation of discrete signal multiplied by

twiddle factor given as:

$$X(K)= \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}$$

Where, $e^{-j2\pi kn/N}$ is called as twiddle factor.

For long convolution the FFT is faster method as compare to DFT. Fig. 1.

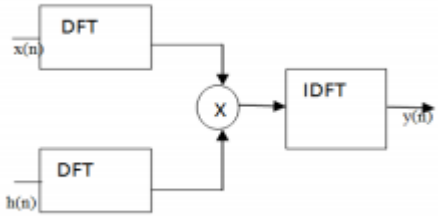


Fig. 1. Convolution by DFT-IDFT Method.

The same convolution process is done by matrix method as:

$$y(m) = \begin{bmatrix} 1 & 1 & 0 & 2 \\ 2 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 \\ 1 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 6 \\ 5 \end{bmatrix}$$

The NxM matrix multiplication equation is written as:

y(0) = h(0)x(0) + h(N-1)x(1) + h(N-2)x(2) + ---------- h(2)x(N-2) + h(1)x(N-1)

y(1) = h(1)x(0) + h(0)x(1) + h(N-1)x(2) + ---------- h(2)x(N-2) + h(1)x(N-1)

y(2) = h(2)x(0) + h(1)x(1) + h(0)x(2) + ----------h(4)x(N2) + h(3)x(N-1)

y(N-2) = h(N-2)x(0) + h(N-3)x(1) + h(N-4)x(2) + ---------- --h(0)x(N-2) + h(N-1)x(N-1)

y(N-1) = h(N-1)x(0) + h(N-2)x(1) + h(N-3)x(2) + ---------- --h(1)x(N-2) + h(0)x(N-1)

Thus circular convolution is obtaining quickly using matrix multiplication approach.

c) Linear convolution from circular convolution with zero padding.

We know that linear convolution in time domain results in multiplication in frequency domain and vice versa. Let X(k) and H(k) denote the DFTs of x(n), h(n) respectively. IDFT of Y(k), where Y(k) = (X(k)H(k)), is circular convolution of x(n) and h(n) not their linear convolution. So to avoid aliasing x(n), h(n) are zero padded to the length L+M-1 (L, M are lengths of x(n) and h(n), respectively) before finding their DFTs.

4.3 Algorithm

a) Circular Convolution of two given sequences using summation formula.

1. Read the first input sequence, x[n] and plot.
2. Read the second input sequence, h[n] and plot
3. Find the length of x[n] and y[n] , l1 and l2 respectively
4. Check if l1=l2. Proceed only if equal.
5. If l1 not equal to l2, zero padding is done to make l1=l2.

6. Initialize a loop variable for the number of output points.
7. For each output sample access the samples of y[n] in cyclic order.
8. Find the sum of products of x[n] and cyclically folded and shifted h[n] to get circular convoluted output.
9. Display and plot the output.

b) Circular Convolution of two given sequences using matrix method.

1. Read the first input sequence, x[n] and plot.
2. Read the second input sequence, h[n] and plot
3. Find the length of x[n] and y[n] , l1 and l2 respectively
4. Take transpose of the given sequences.
5. Use convmtx/circshift MATLAB command.
6. Display and plot the output

c) Linear convolution from circular convolution with zero padding.

1. Read the first input sequence, x[n] and plot.
2. Read the second input sequence, h[n] and plot
3. Find the length of x[n] and y[n] , l1 and l2 respectively
4. Check if l1=l2. Proceed only if equal.
5. If l1 not equal to l2, zero padding is done to make l1=l2.
6. Initialize a loop variable for the number of output points.
7. For each output sample access the samples of y[n] in cyclic order.
8. Find the sum of products of x[n] and cyclically folded and shifted h[n] to get circular convoluted output.
9. Display and plot the output.

4.4 MATLAB Implementation

a) Circular Convolution of two given sequences using summation formula.

MATLAB recognizes index 1 to be positive maximum. Index 0 is not recognized. Hence in the below program wherever y, x and h sequences are accessed, the index is added with +1. the modulo index calculation for circular convolution is carried out using the function - MOD Modulus (signed remainder after division). MOD(x, y) is x - y.*floor(x./y) if y ~= 0. By convention, MOD(x, 0) is x. The input x and y must be real arrays of the same size, or real scalars. MOD(x, y) has the same sign as

y while REM(x, y) has the same sign as x. MOD(x, y) and REM(x, y) are equal if x and y have the same sign, but differ by y if x and y have different signs.

b) Circular Convolution of two given sequences using matrix method.

MATLAB program to find Circular Convolution by matrix multiplication using circshift command or we can use convmtx command.

A = convmtx (h,n) returns the convolution matrix, A, such that the product of A and a vector, x, is the convolution of h and x. If h is a column vector of length m, A is (m+n-1)-by-n and the product of A and a column vector, x, of length n is the convolution of h and x. If h is a row vector of length m, A is n-by-(m+n-1) and the product of a row vector, x, of length n with A is the convolution of h and x. Convmtx handles edge conditions by zero padding.

Y = circshift(A,K) circularly shifts the elements in array A by K positions. If K is an integer, then circshift shifts along the first dimension of A whose size does not equal 1. If K is a vector of integers, then each element of K indicates the shift amount in the corresponding dimension of A.

Y = circshift(A,K,dim) circularly shifts the values in array A by K positions along dimension dim. Inputs K and dim must be scalars.

4.5 Calculation

a) Circular Convolution of two given sequences using summation formula.

Example for finding Circular Convolution of two Sequences.

Let's take x (n) = {1, 1, 2, 1} and h(n) = {1, 2, 3, 4}

y(0) = x(k) x_2(-k)

= x(0) h(0) + x(1) h(3) + x(2)h(2) + x(3) h(1)

= 1 + 4 + 6 +2 = 13

y(1) = x(k) h(1-k)

= x(0) h(1) + x(1) h(0) + x(2) h(3) + x(3) h(2)

= 2 + 1 + 8 + 3= 14

y(2) = x(k) h(2-k)

= x(0) h(2) + x(1) h(1) + x(2) h(0) + x(3) h(3)

= 3 + 2 + 2+ 4= 11

y(3) = x(k) h(3-k)

= x(0) h(3) + x(1) h(2) + x(2) h(1) + x(3) h(0)

= 4 + 3 + 4 + 1= 12

The circular convoluted signal is,

y(n) = {13, 14, 11, 12}

b) Circular Convolution of two given sequences using matrix method.

c) Linear convolution from circular convolution with zero padding.

x(n)= [1 2 1 2 1 2]
h(n)= [1 2 3 4]
M= length(x) + length(h)-1
= 6 +4 -1
=9
x(n) = [1 2 12 12 0 0 0]
h(n)= [1 2 34 0 0 0 0 0]
y(n)= x(n).h(n)

$$y(n)= \begin{bmatrix} 1 & 0 & 0 & 0 & 2 & 1 & 1 & 2 & 1 \\ 2 & 1 & 0 & 0 & 0 & 2 & 2 & 1 & 2 \\ 1 & 2 & 1 & 0 & 0 & 0 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 0 & 0 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 0 & 0 & 2 & 1 \\ 2 & 1 & 2 & 1 & 2 & 1 & 0 & 0 & 2 \\ 0 & 2 & 1 & 2 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 2 & 2 & 1 & 0 \end{bmatrix} . \begin{vmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

y(n)= [1 4 8 14 16 14 15 10 8]

4.6 MATLAB Program:

a) Circular Convolution of two given sequences using summation formula.

```
clc;
clear all;
close all;
xn=input('enter the first sequence x(n)=');
hn=input('enter the second sequence h(n)=');
l1=length(xn);
l2=length(hn);
N=max(l1,l2);
xn=[xn,zeros(1,N-l1)];
hn=[hn,zeros(1,N-l2)];
for n=0:N-1; y(n+1)=0; for k=0:N-1
i=mod((n-k),N);
```

```
y(n+1)=y(n+1)+hn(k+1)*xn(i+1);
end;
end;
disp('circular convolution output');
disp(y);
subplot(2,2,1);
stem(xn);
xlabel('n');
ylabel('x(n)');
title('plot of h(n)');
subplot(2,2,2);
stem(hn);
xlabel('n');
ylabel('h(n)');
title('plot of h(n)');
subplot(2,1,2);
stem(y);
xlabel('n');
ylabel('y(n)');
title('circular convolution output');
```

b) Circular Convolution of two given sequences using matrix method.

```
close all;
clear all;
x1=input('Enter the 1st sequence x[n] = ');
h1=input('Enter the 2nd sequence h[n] = ');
N1=length(x1);
N2=length(h1);
N=max(N1,N2);
if(N1>N2)
h1=[h1,zeros(1,N1-N2)];
else
x1=[x1,zeros(1,N2-N1)];
end;
```

```
x=transpose(x1); h=transpose(h1); temp=h;
for i=1:N-1;
temp=circshift(temp,1);
h=horzcat(h,temp);
end;
h
x
y=h*x ;
disp('Circular convolved output y[n] = ');
y
subplot(3,1,1);
stem(x1); xlabel('N-->');
ylabel('Amplitude-->');
title('1st input sequence x[n] ');
subplot(3,1,2);
stem(h1); xlabel('N-->');
ylabel('Amplitude-->');
title('2nd input sequence h[n]');
subplot(3,1,3);
stem(y); xlabel('N-->');
ylabel('Amplitude-->');
title('Circular convolved output y[n]');
```

c) Linear convolution from circular convolution with zero padding.

```
clc;
close all;
clear all;
x1=input('enter the first sequence=');
x2=input('enter the second sequence=');
n=input('enter the no of points of the dft=');
subplot(3,1,1);
stem(x1,'filled');
title('plot of first sequence');
subplot(3,1,2);
```

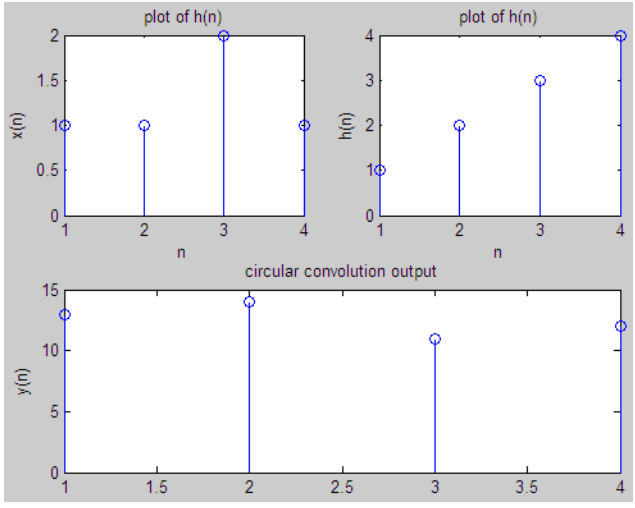
```
stem(x2,'filled');
title('plot of second sequence');
n1=length(x1);
n2=length(x2);
m=n1+n2-1;
x=[x1 zeros(1,n2-1)];
y=[x2 zeros(1,n1-1)];
x_fft=fft(x,m);
y_fft=fft(y,m);
dft_xy=x_fft.*y_fft
y=ifft(dft_xy,m);
disp('the circular convolution result is');
disp(y);
subplot(3,1,3);
stem(y,'filled');
title('plot of circularly convoluted sequence');
```

4.7 Output Waveforms

a) Circular Convolution of two given sequences using summation formula.

Input:
enter the first sequence x(n)=[1 1 2 1]
enter the second sequence h(n)=[1 2 3 4]

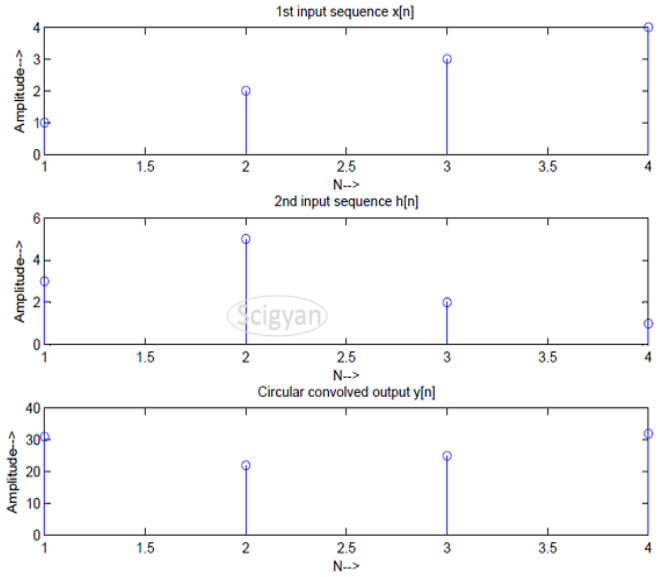
Output:
circular convolution output
13 14 11 12



b) Circular Convolution of two given sequences using matrix method.

Output

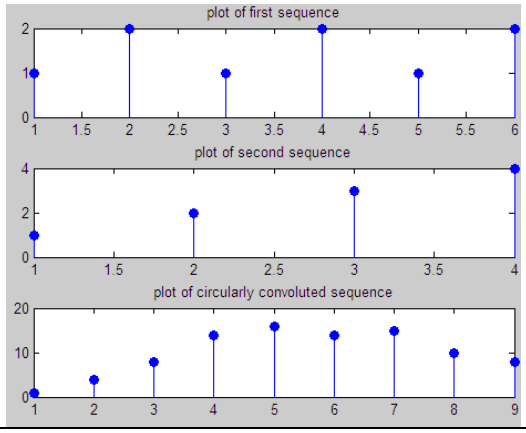
Enter the 1st sequence x[n] = [1 2 3 4]
Enter the 2nd sequence h[n] = [3 5 2 1]
h =
3 1 2 5
5 3 1 2
2 5 3 1
1 2 5 3
x =
1
2
3
4
Circular convolved output y[n] =
y =
31
22
25
32



c) Linear convolution from circular convolution with zero padding.

Input:
enter the first sequence=[1 2 1 2 1 2]
enter the second sequence=[1 2 3 4]
enter the no of points of the dft=9

Output:
the circular convolution result is
1.0000 4.0000 8.0000 14.0000 16.0000 14.0000 15.0000 10.0000 8.0000



4.8 Results

4.9 Conclusion

Experiment 5

5.0 Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum.

5.1 Aim

To compute N-point DFT of a given sequence and to plot magnitude and phase spectrum.

5.2 Theory

Discrete Fourier Transform is a powerful computation tool which allows us to evaluate the Fourier Transform X (e^{jw}) on a digital computer or specially designed digital hardware. Since X(e^{j}) is continuous and periodic, the DFT is obtained by sampling one period of the Fourier Transform at a finite number of frequency points. Apart from determining the frequency content of a signal, DFT is used to perform linear filtering operations in the frequency domain.

The sequence of N complex numbersx_0... x_{N-1} is transformed into the sequence of *N* complex numbers X_0... X_{N-1} by the DFT according to the formula:

$$X(k) = \sum_{n=0}^{N-1} x(n) e - j2\pi nk/N \qquad k = 0,1, \ldots. N\text{-}1$$

5.3 Algorithm

1. Enter the number of points N
2. Enter the input sequence elements x[n]
3. Create a vector for sample index n
4. Calculate DFT using built in function FFT
5. Plot the magnitude and phase spectrum

5.4 MATLAB Implementation

MATLAB has an inbuilt function 'FFT' which computes the Discrete Fourier transform. FFT(X) is the discrete Fourier transform (DFT) of vector X. For length N input vector x, the DFT is a length N vector X, with elements N. FFT(X,N) is the N-point FFT, padded with zeros if X has less than N points and truncated if it has more. The magnitude spectrum is computed using the function ABS Absolute value. ABS(X) is the absolute value of the elements of X. When X is complex, ABS(X) is the complex modulus (magnitude) of the elements of X. The phase spectrum is computed using the function ANGLE Phase angle. ANGLE (H) returns the phase angles, in radians, of a matrix with complex elements.

5.5 MATLAB Program: Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum.

```
clc;
close all;
clear all;
x=input('enter the sequence= ');
N=input('enter the no of dft points= ');
xk=fft(x,N);
disp('N point DFT of the sequence is=')
disp(xk);
n=0:1:N-1;
figure(1);
stem(n,abs(xk));
disp('magnitude of the sequence is=')
disp(abs(xk));
xlabel('k');
ylabel('|xk|');
title('magnitude spectrum');
figure(2);
stem(angle(xk));
disp('phase value of the sequence is=')
disp(angle(xk));
xlabel('k');
ylabel('angle (xk)');
title('phase spectrum');
figure(3);
n1=0:1:length(x)-1;
stem(n1,x);
xlabel('n');
ylabel('x[n]');
title('original signal');
```

5.6 Calculation

EXAMPLE:

Let us assume the input sequence x[n] = [1 1 0 0]

We have,

$$X(k) = \sum_{n=0}^{N-1} x(n) e - j2\pi nk/N \qquad k = 0,1, \ldots N-1$$

For k = 0,

$$X(0) = \sum_{n=0}^{3} x(n)$$

X(0) = x(0) + x(1) + x(2) + x(3)

X(0) = 1+1+0+0 = 2

For k = 1,

$$X(1) = \sum_{n=0}^{3} x(n) e - j\pi n/2$$

X(1) = 1 – j

For k = 2

$$X(2) = \sum_{n=0}^{3} x(n) e - j\pi n$$

X (2) = x(0) + x(1) e^{-j} + x(2) e^{-j2} + x(3) e^{-j3}

X (2) = 1 + cos – jsin

X (2) = 1-1 = 0

For k = 3,

$$X(3) = \sum_{n=0}^{3} x(n) e - j3n\pi/2$$

X(3) = x (0) + x (1) $e^{-j3/2}$ + x (2) e^{-j3} + x (3) $e^{-j9/2}$

X (3) = 1 + cos (3 /2) - jsin (3 /2)

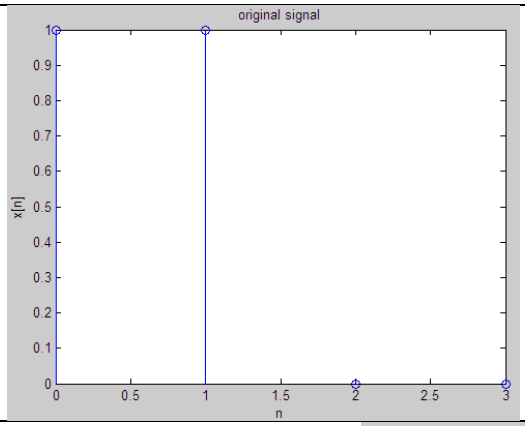
X (3) = 1 + j

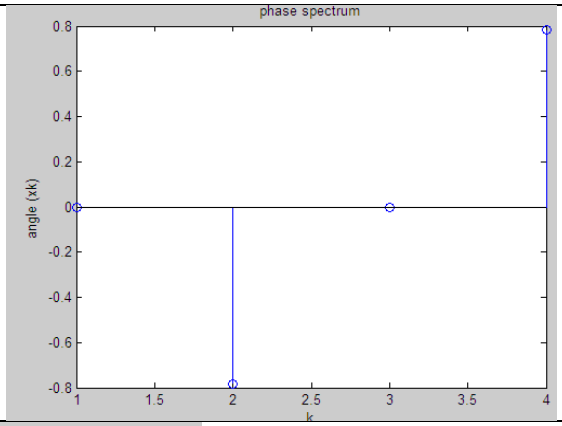
The DFT of the given sequence is,

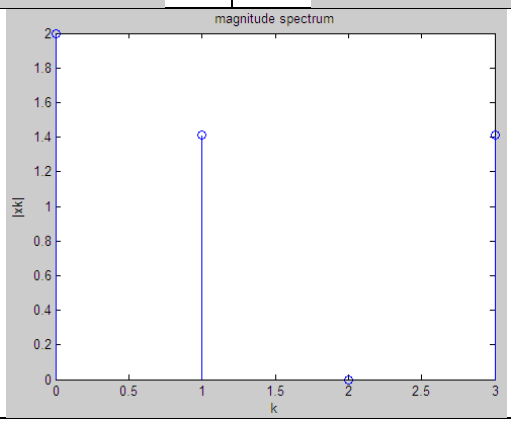
X (k) = {2, 1-j, 0, 1+j}

5.7 Output Waveforms

Input:
enter the sequence= [1 1 0 0]
enter the no of dft points= 4
Output:
N point DFT of the sequence is=
2.0000 1.0000 - 1.0000i 0 1.0000 + 1.0000i
Magnitude of the sequence is=
2.0000 1.4142 0 1.4142
Phase value of the sequence is=
0 -0.7854 0 0.7854







5.8 Results

5.9 Conclusion

Experiment 6

6.0 Linear and circular convolution by DFT and IDFT method

6.1 Aim

To write MATLAB program to verify linear and circular convolution by DFT and IDFT method.

6.2 Theory

Circular convolution is another way of finding the convolution sum of two input signals. It resembles the linear convolution, except that the sample values of one of the input signals is folded and right shifted before the convolution sum is found. Also note that circular convolution could also be found by taking the DFT of the two input signals and finding the product of the two frequency domain signals. The Inverse DFT of the product would give the output of the signal in the time domain which is the circular convolution output. The two input signals could have been of varying sample lengths. But we take the DFT of higher point, which ever signals levels to. For e.g. If one of the signal is of length 256 and the other spans 51 samples, then we could only take 256 point DFT. So the output of IDFT would be containing 256 samples instead of 306 samples, which follows N1+N2 – 1 where N1 & N2 are the lengths 256 and 51 respectively of the two inputs. Thus the output which should have been 306 samples long is fitted into 256 samples. The 256 points end up being a distorted version of the correct signal. This process is called circular convolution. Circular convolution is explained using the following example.

The two sequences are x1 (n) = {2,1,2,1}

x2 (n) = {1,2,3,4 }

Each sequence consists of four nonzero points. For purpose of illustrating the operations involved in circular convolution it is desirable to graph each sequence as points on a circle. Thus the sequences x1 (n) and x2 (n) are graphed as illustrated in the fig. We note that the sequences are graphed in a counterclockwise direction on a circle. This establishes the reference direction in rotating one of sequences relative to the other. Now, y (m) is obtained by circularly convolving x (n) with h (n).

6.3 Algorithm

1. Give input sequence x[n].
2. Give impulse response sequence h[n].
3. Find the Circular Convolution and linear convolution y[n] using the DFT method.
4. Plot x[n],h[n],y[n].

6.4 Calculation

Circular Convolution

x[n]= [1 2 3 4] y[n]= [3 5 2 1]

$$x(k)= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & 1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 10 \\ -2+2j \\ -2 \\ -2-2j \end{bmatrix}$$

$$h(k)= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & 1 & -j \end{bmatrix} \begin{bmatrix} 3 \\ 5 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 11 \\ 1-4j \\ -1 \\ 1+4j \end{bmatrix}$$

x(n)= x(k) * h(k)

$$y(n)= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & 1 & -j \end{bmatrix} \begin{bmatrix} 3 \\ 5 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 11 \\ 1-4j \\ -1 \\ 1+4j \end{bmatrix}$$

y1(n)= 110+6+1 0j-2 -6+10j =128/4=31

y2(n)= 110+6j+10 (-1) -2 -6j + 10(-1)=88/4=22

y3(n)= 110-6-10j+2-6+10j=100/4=25

y4(n)=110-6j+10-2+6j+10=128/4=32

$$y(n)= \begin{bmatrix} 31 \\ 22 \\ 25 \\ 32 \end{bmatrix}$$

6.5 MATLAB Program: linear convolution using DFT and IDFT

```
clc;
close all;
clear all;
x1=input('enter the first sequence=');
x2=input('enter the second sequence=');
n=input('enter the no of points of the dft=');
subplot(3,1,1);
stem(x1,'filled');
title('plot of first sequence');
subplot(3,1,2);
stem(x2,'filled');
title('plot of second sequence');
n1=length(x1);
n2=length(x2);
m=n1+n2-1;
x=[x1 zeros(1,n2-1)];
y=[x2 zeros(1,n1-1)];
x_fft=fft(x,m);
y_fft=fft(y,m);
dft_xy=x_fft.*y_fft
y=ifft(dft_xy,m);
disp('the linear convolution result is');
disp(y);
subplot(3,1,3);
stem(y,'filled');
title('plot of linearly convoluted sequence');
```

6.6 MATLAB Program: circular convolution using DFT and IDFT

```
clc;
close all;
clear all;
x=input('enter input x(n)= ');
m=length(x)
h=input('enter input h(n)= ');
n=length(h)
subplot(3,2,1);
stem(x);
title('input sequence x(n)');
xlabel('----->n');
ylabel('----->amplitude');
grid;
subplot(3,1,2);
stem(h);
title('input sequence h(n)');
xlabel('----->n');
ylabel('----->amplitude');
grid;
disp('circular convolution of x(n) &h(n) ');
y1=fft(x,n);
y2=fft(h,n);
y3=y1.*y2;
y=ifft(y3,n);
disp(y);
subplot(3,1,3);
stem(y);
title('circular convoluted output');
xlabel('----->n');
ylabel('----->amplitude');
grid;
```

6.7 Output Waveforms

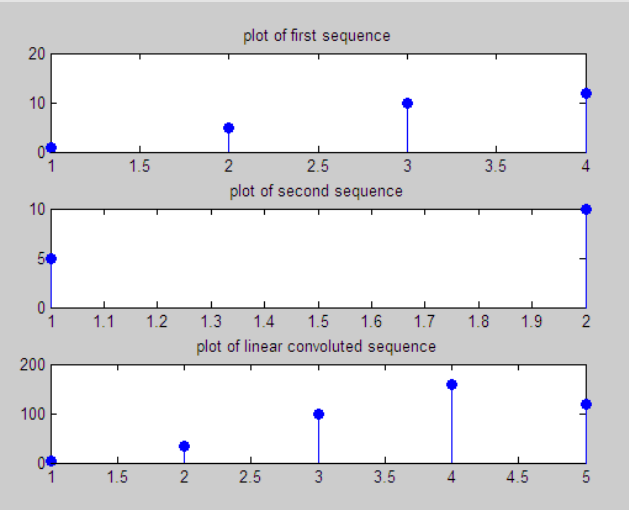
A) For linear convolution:

Input:

enter the first sequence=[1 5 10 12]
enter the second sequence=[5 10]
enter the no of points of the dft=5

Output:

the linear convolution result is
5.0000 35.0000 100.0000 160.0000 120.0000



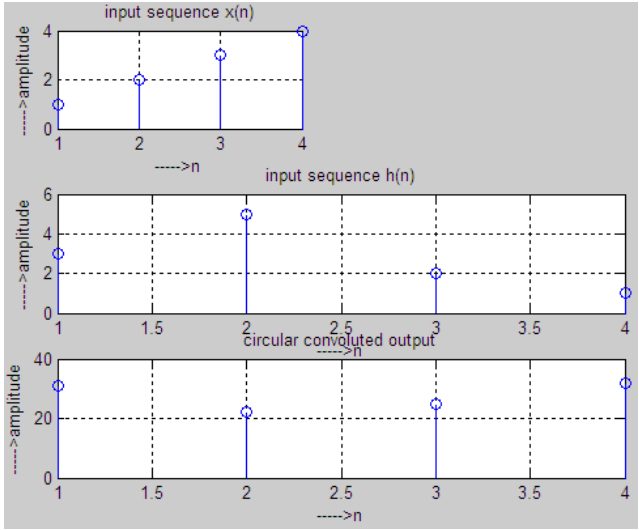
B) For circular convolution:

Input:

enter input x(n)= [1 2 3 4]
m = 4
enter input h(n)= [3 5 2 1]
n = 4

Output:

circular convolution of x(n) & h(n)
31 22 25 32



6.8 Results

6.9 Conclusion

Experiment 7

7.0 Solving a given difference equation

7.1 Aim

To obtain the impulse response/step response of a system described by the given difference equation**.**

7.2 Theory

- A difference equation with constant coefficients describes a LTI system. For example the difference equation **y[n] + 0.8y[n-2] + 0.6y[n-3] = x[n] + 0.7x[n-1] + 0.5x[n-2]** describes a LTI system of order 3. The coefficients 0.8, 0.7, etc are all constant i.e., they are not functions of time (n). The difference equation y[n]+0.3ny[n-1]=x[n] describes a time varying system as the coefficient 0.3n is not constant.
- The difference equation can be solved to obtain y[n], the output for a given input x[n] by rearranging as y[n] = x[n] + 0.7x[n-1]+0.5x[n-2]- 0.8y[n-2]- 0.6y[n-3] and solving.
- The output depends on the input x[n]

- With **x[n]= δ[n],** an impulse, the computed output y[n] is the **impulse response**.
- If **x[n]=u[n]**, a **step response** is obtained.
- If **x[n] = cos(wn)** is a sinusoidal sequence, a **steady state response** is obtained (wherein y[n] is of the same frequency as x[n], with only an amplitude gain and phase shift-refer Fig.7.3).
- Similarly for any arbitrary sequence of x[n], the corresponding output response y[n] is computed.

- The difference equation containing past samples of output, i.e., y[n-1], y[n-2], etc leads to a recursive system, whose impulse response is of infinite duration (IIR). For such systems the impulse response is computed for a large value of n, say n=100 (to approximate n=∞). The MATLAB function filter is used to compute the impulse response/ step response/ response to any given x[n]. Note: The filter function evaluates the convolution of an infinite sequence (IIR) and x[n], which is not possible with conv function (remember conv requires both the sequences to be finite).
- The difference equation having only y[n] and present and past samples of input (x[n], x[n-k]), represents a system whose impulse response is of finite duration (FIR). The response of FIR systems can be obtained by both the 'conv' and 'filter' functions. The filter function results in a response whose length is equal to that of the input x[n], whereas the output sequence from conv function is of a longer length (xlength + hlength-1).

7.3 MATLAB Implementation

MATLAB has an inbuilt function '***filter***' to solve difference equations numerically, given the input and difference equation coefficients (b, a).

y=filter (b, a, x)

Where x is the input sequence, y is the output sequence which is of same length as x.

7.4 Algorithm

1. Input the two sequences as a and b representing the coefficients of y and x.
2. If IIR response, then input the length of the response required (say 100, which can be made constant).
3. Compute the output response using the 'filter' command.
4. Plot the input sequence & impulse response (and also step response, etc if required).

7.5 Calculation

The difference equation is given by

$y(n) = x(n)+0.7x(n-1)+0.\ 5x(n-2)-0.8y(n-2)-0.6y(n-3)$

$y(z)+0.8z^{-2}+0.6z^{-3}\ y(z)=x(z)+0.7z^{-1}\ x(z) + 0.5\ z^{-2}\ x(z)$

$y(z)\ [1+0.8z^{-2}+0.6z^{-3}=x(z)+(1+0.7z^{-1}+0.5z^{-2})]$

$H(z)= \frac{y(z)}{x(z)} = \frac{(1+0.7z^{-1}+0.5z^{-2})}{(1+0.8z^{-1}+0.6z^{-3})}$

$$
\begin{array}{l|l}
 & 1+0.7z^{-1}-0.3z^{-2}-1.67\,z^{-3} \\
\hline
1+0.8z^{-1}+0.6z^{-3} & 1+0.7z^{-1}+0.5z^{2} \\
 & 1+0.8z^{-2}+0.6z^{3} \\
 & \hline
 & 0.7z^{-1}-0.3z^{-2}-6.6z^{-3} \\
 & -0.72z^{-1}+0.56z^{-3}+0.42z^{-4} \\
 & \hline
 & -0.32z^{-2}-1.16z^{-3}-0.42z^{-7} \\
 & 0.3z^{-2}-0.24z^{-4}-0.18z^{-5} \\
 & \hline
 & -1.16z^{-3}-0.18z^{-4}+0.18z^{-5} \\
 & -1.16z^{3}-0.0325z^{-5}-0.62z^{-6} \\
 & \hline
 & -0.18z^{-4}-0.212z^{5}-0.02z^{-6}
\end{array}
$$

$H(z)= 1+0.7z^{-1}-0.3z^{-2}-1.167z^{-3}$

h(h)= [1 0.7 -0.3 -1.167]

7.6 MATLAB Program: Solving a given difference equation

```
clc;
close all;
clear all;
N=input('enter the length of response= ');
b=input('enter the coefficient of x(n)...= ');
a=input('enter the coefficient of y(n)...= ');
%to find the impulse response
figure(1);
X=[1,zeros(1,N-1)];
n=0:1:N-1;
h=filter(b,a,X);
disp('impulse response of filter= ');
disp(h);
subplot(2,1,1);
stem(n,X);
title('impulse input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title('impulse response');
xlabel('n');
ylabel('h(n)');
%to find the step response
figure(2);
X=[ones(1,N)];
n=0:1:N-1;
h=filter(b,a,X);
disp('step response of filter= ');
disp(h);
subplot(2,1,1);
stem(n,X);
title('step input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title('step response');
xlabel('n');
ylabel('h(n)');
%to find the exponential response
figure(3);
n=0:1:N-1;
X=2.^n;
h=filter(b,a,X);
disp('exponential response of filter= ');
disp(h);
subplot(2,1,1);
stem(n,X);
title('exponnetial input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title('expontial response');
xlabel('n');
ylabel('h(n)');
%to find the steady response
figure(4);
n=0:1:N-1;
X=cos(0.5*pi*n);
h=filter(b,a,X);
disp('steady state response of filter= ');
disp(h);
subplot(2,1,1);
stem(n,X);
title('steady input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title(' steady response');
xlabel('n');
ylabel('h(n)');
```

7.7 Output Waveforms

Input:

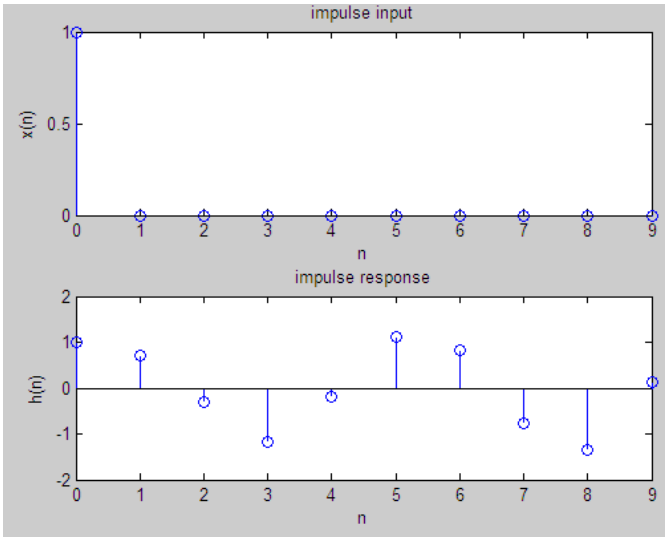
enter the length of response= 10
enter the coefficient of x(n)...= [1 0.7 0.5]
enter the coefficient of y(n)...= [1 0 0.8 0.6]

Output:

Impulse response of filter=

1.0000 0.7000 -0.3000 -1.1600 -0.1800

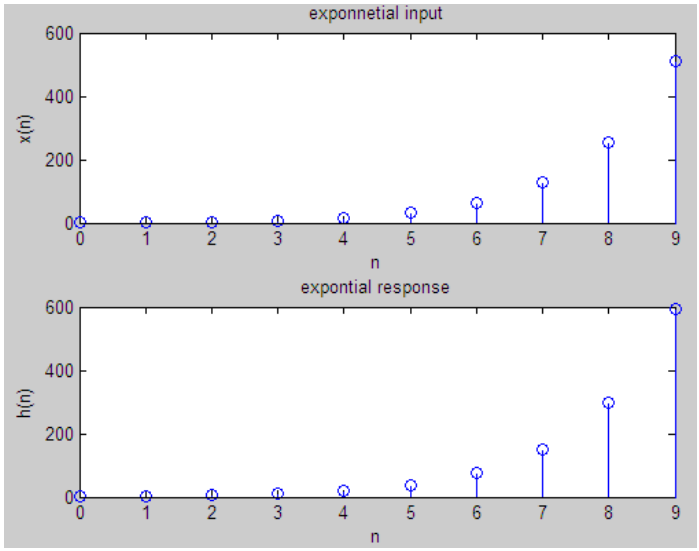
1.1080 0.8400 -0.7784 -1.3368 0.1187



Exponential response of filter= 1.0000 2.7000

5.1000 9.0400 17.9000 36.9080 74.6560

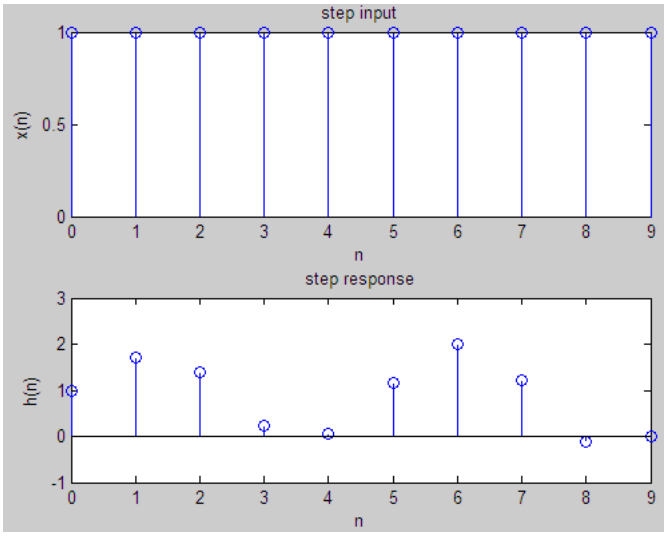
148.5336 295.7304 591.5795



Step response of filter=

1.0000 1.7000 1.4000 0.2400 0.0600

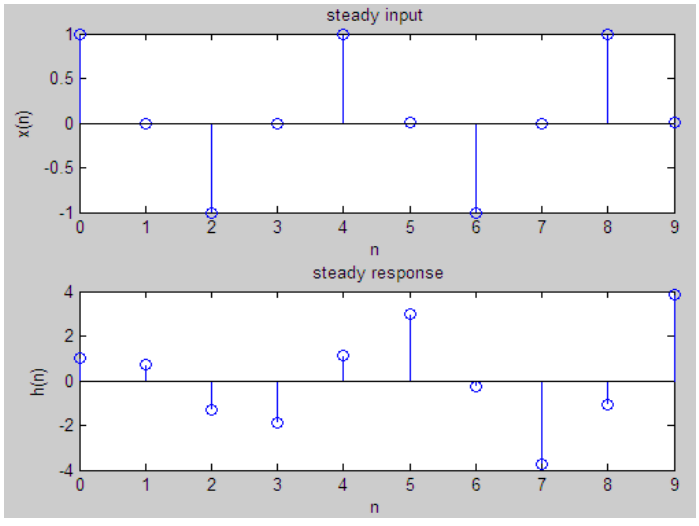
1.1680 2.0080 1.2296 -0.1072 0.0115



Steady state response of filter=

1.0000 0.7000 -1.3000 -1.8600 1.1200

2.9680 -0.2800 -3.7464 -1.0568 3.8651



7.8 Result

7.9 Conclusion

Experiment 8

8.0 Calculation of DFT and IDFT by FFT

8.1 Aim

To write a MATLAB program for computation of DFT and IDFT using Direct and FFT method

8.2 Theory

DFT:

Discrete Fourier Transform (DFT) is used for performing frequency analysis of discrete time signals. DFT gives a discrete frequency domain representation whereas the other transforms are continuous in frequency domain. The N point DFT of discrete time signal x[n] is given by the equation.

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}} ; \quad k = 0,1,2,..N-1$$

The inverse DFT allows us to recover the sequence x[n] from the frequency samples

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n] e^{\frac{j2\pi kn}{N}} ; \quad n = 0,1,2,..N-1$$

FFT:

A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers. Evaluating the sums of DFT directly would take O(N 2) arithmetical operations. An FFT is an algorithm to compute the same result in only O(N log N) operations. In general, such algorithms depend upon the factorization of N, but there are FFTs with O(N log N) complexity for all N, even for prime N. Since the inverse DFT is the same as the DFT, but with the opposite sign in the exponent and a 1/N factor, any FFT algorithm can easily be adapted for it as well.

8.3 Algorithm

1. Get the input sequence
2. Find the DFT of the input sequence using direct equation of DFT.
3. Find the IDFT using the direct equation.
4. Find the FFT of the input sequence using MATLAB function.
5. Find the IFFT of the input sequence using MATLAB function.
6. Display the above outputs using stem function

8.4 Calculation

x(n)=[1 1 1 1 1 1 0 0]

this equation is formulated in the matrix form by equation

$X_N = [Wn]\, x_N$

with N=8 x8=[W8]x8

$$x8= \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} \quad \text{and} \quad x8= \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} & -j & \frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} & -1 & \frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} & j & \frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & \frac{-1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} & j & \frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} & -1 & \frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} & j & -\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} & -j & \frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} & -1 & \frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} & j & \frac{-1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & \frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} & j & -\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} & -1 & -\frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} & -j & \frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1+1+1+1+1+1 \\ 1+\frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}}-j\frac{-1}{\sqrt{2}}-j\frac{1}{\sqrt{2}}-1-\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}}+0+0 \\ 1-\frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}}+j+\frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}}-1+\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}} \\ 1-j-1+j+1-j+0+0 \\ 1-1+1-1+1-1+0+0 \\ 1-\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}}-j+\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}}-1+\frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}}+0+0 \\ 1-j-1-j+1+j+0+0 \\ 1-\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}}+j\frac{1}{\sqrt{2}}+j-\frac{1}{\sqrt{1}}+j\frac{1}{\sqrt{2}}-1-\frac{1}{\sqrt{2}}-j\frac{1}{\sqrt{2}}+0+0 \end{bmatrix}$$

$$= \begin{bmatrix} 6 \\ (\frac{-2+\sqrt{2}}{2})\,j - \frac{1}{\sqrt{2}} \\ (\frac{2-\sqrt{2}}{2})\,j \\ 1-j \\ 0 \\ (\frac{-2+\sqrt{2}}{2})\,j + \frac{1}{\sqrt{2}} \\ 1+j \\ (\frac{2+\sqrt{2}}{2}) - \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} xR(0) \\ xR(1) \\ xR(2) \\ xR(3) \\ xR(4) \\ xR(5) \\ xR(6) \\ xR(7) \end{bmatrix} = \begin{bmatrix} 6 \\ -\frac{1}{\sqrt{2}} \\ 0 \\ 1 \\ 0 \\ \frac{1}{\sqrt{2}} \\ 1 \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \text{ and } \begin{bmatrix} 0 \\ (\frac{-2+\sqrt{2}}{2})\,j \\ (\frac{2-\sqrt{2}}{2})\,j \\ -1 \\ 0 \\ (\frac{-2+\sqrt{2}}{2})\,j \\ j \\ (\frac{2+\sqrt{2}}{2})\,j \end{bmatrix} = \begin{bmatrix} x1(0) \\ x1(1) \\ x2(2) \\ x2(3) \\ x2(4) \\ x2(5) \\ x2(6) \\ x2(7) \end{bmatrix}$$

= 6 -0.7071-1.7071i 1.0000-1.0000i 0.707i+0.2929i 0 0.7071-0.292i 1.0000+1.0000i -0.7071+1.7071i

8.5 MATLAB Program: Calculation of DFT and IDFT by FFT

```
%direct dft
clc;
close all;
clear all;
xn=input('enter input');
N=length(xn);
n=0:N-1;
k=0:N-1;
wn=exp((-1i*2*pi*n'*k)/N);
xf=wn*xn'
subplot(2,1,1);
figure (1);
stem(abs(xf));
title('dft magnitude response');
ylabel('magnitude');
xlabel('frequence');
%direct idft
wN=exp((1i*2*pi*n'*k)/N);
pn=wN*xf/N
subplot(2,1,2);
stem(abs(pn));
title('idft magnitude response');
ylabel('magnitude');
xlabel('time');
%fft mathod
figure(2);
xp=fft(xn,N)
subplot(2,1,1);
stem(abs(xp));
title('fft magnitude response');
ylabel('magnitude');
xlabel('frequency');
%ifft method
xw=ifft(xp,N)
subplot(2,1,2);
stem(abs(xw));
title('ifft magnitude response');
ylabel('magnitude')
xlabel('time');
```

8.6 Output Waveforms

Input:
enter input [1 1 1 1 1 1 0 0]

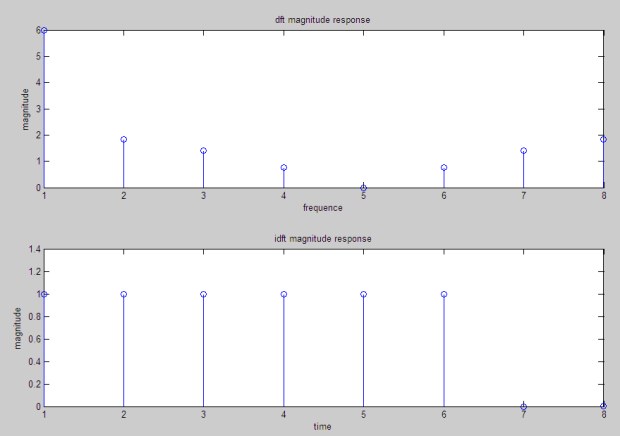
Output:

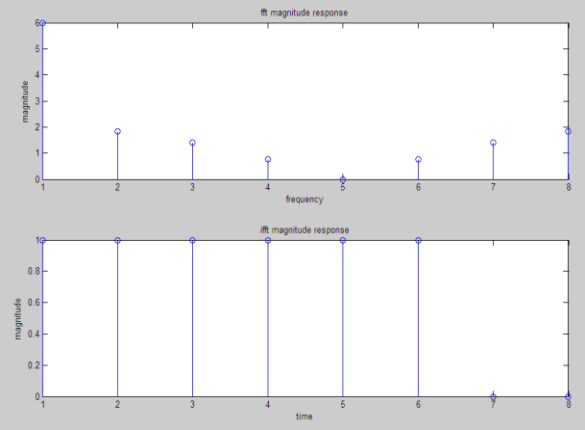
xf= 6.0000 -0.7071 - 1.7071i 1.0000 - 1.0000i 0.7071 + 0.2929i 0 - 0.0000i
0.7071 - 0.2929i 1.0000 + 1.0000i -0.7071 + 1.7071i

pn = 1.0000 - 0.0000i 1.0000 1.0000 - 0.0000i 1.0000 - 0.0000i 1.0000 + 0.0000i
1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

xp = 6.0000 -0.7071 - 1.7071i 1.0000 - 1.0000i 0.7071 + 0.2929i 0
0.7071 - 0.2929i 1.0000 + 1.0000i -0.7071 + 1.7071i

xw = 1 1 1 1 1 1 0 0





8.7 Results

8.8 Conclusion

Experiment 9

9.0 Design and implementation of IIR filter to meet given specifications.

9.1 Aim

Design and implementation of IIR filter to meet given specifications.

9.2 Theory

Basically digital filter is a linear time-invariant discrete time system.

Infinite Impulse Response (IIR) filter: IIR filters are of recursive type, whereby the present output sample depends on the present input, past input samples and output samples. The impulse response h(n) for a realizable filter is,h(n) = 0 for n 0. And for stability, it must satisfy the condition,

$$\sum_{n=0}^{\infty} |h(n)| < \infty$$

There are two methods of stating the specifications as illustrated in previous program. In the first program, the given specifications are directly converted to digital form and the designed filter is also implemented. In the last two programs the Butterworth and chebyshev filters are designed using bilinear transformation (for theory verification).

Method I: Given the order N, cutoff frequency fc, sampling frequency fs and the IIR filter type (Butterworth, cheby1, cheby2).

- Step 1: Compute the digital cut-off frequency Wc (in the range $-\pi < Wc < \pi$, with π corresponding to fs/2) for fc and fs in Hz. For example let fc=400Hz, fs=8000Hz

  $$Wc = 2*\pi* fc / fs = 2* \pi * 400/8000 = 0.1* \pi \text{ radians}$$

  For MATLAB the Normalized cut-off frequency is in the range 0 and 1, where 1 corresponds to fs/2 (i.e., fmax)). Hence to use the MATLAB commands

  $$wc = fc / (fs/2) = 400/(8000/2) = 0.1$$

  Note: if the cut off frequency is in radians then the normalized frequency is computed as wc = Wc / π
- Step 2: Compute the Impulse Response [b,a] coefficients of the required IIR filter and the response type (low pass, band pass, etc) using the appropriate butter, cheby1, cheby2 command. For example given a Butterworth filter, order N=2, and a high pass response, the coefficients [b,a] of the filter are computed using the MATLAB inbuilt command ‘butter’ as [b,a] =butter(N, wc , 'high');

Method 2:

Given the pass band (Wp in radians) and Stop band edge (Ws in radians) frequencies, Pass band ripple Rp and stop band attenuation As.

- Step 1: Since the frequencies are in radians divide by π to obtain normalized frequencies to get Wp=Wp/pi and ws=Ws/pi

  If the frequencies are in Hz (note: in this case the sampling frequency should be given), then obtain normalized frequencies as wp=fp/(fs/2), ws=fstop/(fs/2), where fp, fstop and fs are the passband, stop band and sampling frequencies in Hz
- Step 2: Compute the order and cut off frequency as

  [N, wc] = BUTTORD(wp, ws, Rp, Rs)
- Step 3: Compute the Impulse Response [b,a] coefficients of the required IIR filter and the response type as [b,a] =butter(N, wc , 'high');

IMPLEMENTATION OF THE IIR FILTER

1. Once the coefficients of the IIR filter [b,a] are obtained, the next step is to simulate an input sequence x[n], say input of 100, 200 & 400 Hz (with sampling frequency of fs), each of 20/30 points. Choose the frequencies such that they are >, < and = to fc.
2. Filter the input sequence x[n] with Impulse Response, to obtain the output of the filter y[n] using the 'filter' command.
3. Infer the working of the filter (low pass/ high pass, etc).

9.3 MATLAB Implementation

- 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 pass band and has at least Rs dB of attenuation in the stop band. Wp and Ws are the pass band and stop band edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample). For example,

  Low pass: Wp = .1, Ws = .2 High pass: Wp = .2, Ws = .1

  Band pass: Wp = [.2 .7], Ws = [.1 .8] Band stop: Wp = [.1 .8], Ws = [.2 .7]
- BUTTORD also returns Wn, the Butterworth natural frequency (or, the "3 dB frequency") to use with BUTTER to achieve the specifications. [N, Wn] = BUTTORD (Wp, Ws, Rp, Rs,'s') does the computation for an analog filter, in which case Wp and Ws are in radians/second. When Rp is chosen as 3 dB, the Wn in BUTTER is equal to Wp in BUTTORD.
- **BUTTER** Butterworth digital and analog filter design. [B,A] = BUTTER (N,Wn) designs an Nth order low pass 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. If Wn is a two-element vector, Wn = [W1 W2], BUTTER returns an order 2N band pass filter with pass band W1 < W < W2. [B,A] = BUTTER (N, Wn, 'high') designs a high pass filter. [B,A] = BUTTER (N, Wn, 'stop') is a band stop filter if Wn = [W1 W2]. BUTTER (N, Wn, 's'), BUTTER(N, Wn, 'high','s') and BUTTER(N, Wn, 'stop', 's') design analog Butterworth filters. In this case, Wn is in [rad/s] and it can be greater than 1.0.

9.4 Algorithm

1. Get the order of the filter
2. Find the filter coefficients
3. Plot the magnitude response

9.5 Example with Calculation

1. Let's design an analog Butterworth low pass filter.
2. Steps to design an analog Butterworth low pass filter.
3. Get the pass band and stop band edge frequencies
4. Get the pass band and stop band ripples
5. Get the sampling frequency
6. From the given specifications find the order of the filter N.
7. Round off it to the next higher integer.
8. Find the transfer function H(s) for c = 1rad/sec for the value of N.
9. Calculate the value of cutoff frequency c
10. Find the transfer function Ha(s) for the above value of c by substituting s (s/ c) in H(s).

Design: step 1:

$$w_p = \frac{2\pi f_p}{F_s} = \frac{2\pi * 500}{2000} = 0.5\pi\ rad$$

$$w_p = \frac{2\pi f_s}{F_s} = \frac{2\pi * 750}{2000} = 0.75\pi\ rad$$

Step 2: T=1

$$\Omega_p = \frac{2}{T}\tan\frac{w_p}{2} \qquad \Omega_s = \frac{2}{T}\tan\frac{w_s}{2}$$

$$\Omega_p = 2\tan\frac{0.5\pi}{2} \qquad \Omega_s = 2\tan\frac{0.75\pi}{2}$$

$$\Omega_p = 2\frac{rad}{sec} \qquad \Omega_s = 4.828\frac{rad}{sec}$$

Step 3: order of filter

$$N \geq \frac{log\frac{10^{0.1Ap}-1}{10^{0.1As}-1}}{2log\frac{\Omega_p}{\Omega_s}} \qquad N \geq \frac{log\frac{10^{0.801}-1}{10^{1.5}-1}}{2log\frac{2}{4.828}}$$

$$N \geq 1.941, \qquad so \quad N = 2$$

Step 4: cut off frequency

$$\Omega_c = \frac{\Omega_s}{(10^{0.1As}-1)^{\frac{1}{2N}}}$$

$$\Omega_c = 2.052\frac{rad}{sec}$$

Step 5: poles

$$s_k = \pm\Omega_c \left|(N+2K+1)\frac{\pi}{2N}\right.$$

Where K=0 to N-1

Therefore $s_0 = -1.45 + j1.45$

$$s_1 = -1.45 - j1.45$$

$$H_a(s) = \frac{{\Omega_c}^2}{(s-s_0)(s-s_1)} = \frac{{\Omega_c}^2}{(s+1.45-j1.45)(s+1.45+j1.45)}$$

$$H_a(s) = \frac{4.2107}{s^2+2.9s+4.205}$$

Step 6: conversion of analog to digital filter using bilinear transformation

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

$$H_a(s) = \frac{4.2107}{\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)^2 + (2.9)\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) + 4.205}$$

$$H_a(s) = \frac{0.30065 + 0.30065z^2 + 0.6012z}{z^2 + 0.0292z + 0.17174}$$

Let's design an analog Chebyshev filter.

Step 1:

$$w_p = \frac{2\pi f_s}{F_s} = \frac{2\pi * 500}{4000} = 0.25\pi \text{ rad}$$

Step 2: T=1

$$\Omega_p = \frac{2}{T}\tan\frac{w_p}{2} \qquad \Omega_s = \frac{2}{T}\tan\frac{w_s}{2}$$

$$\Omega_p = 2\tan\frac{0.05\pi}{2} \qquad \Omega_s = 2\tan\frac{0.25\pi}{2}$$

$$\Omega_p = 0.1574\frac{rad}{sec} \qquad \Omega_s = 0.8284\frac{rad}{sec}$$

Step 3: order of filter

$$N \geq \frac{cosh^{-1}\sqrt{\frac{10^{0.1As}-1}{10^{0.1Ap}-1}}}{cosh^{-1}\frac{\Omega_s}{\Omega_p}} \qquad N \geq \frac{cosh^{-1}\sqrt{\frac{10^{0.1(20)}-1}{10^{0.1(2)}-1}}}{cosh^{-1}\frac{0.8284}{0.1574}}$$

$$N \geq 1.3, \qquad so \quad N = 2$$

Step 4: cut off frequency

$$\Omega_c = \Omega_p = 0.1574\frac{rad}{sec}$$

Step 5: poles

$$s_k = \sigma_k + j\Omega_k$$

$$\sigma_k = -\sinh[\frac{1}{N}\sinh^{-1}\frac{1}{\epsilon}]\sin[(\frac{2k-1}{2N})\pi]$$

$$\Omega_k = cosh[\frac{1}{N}\sinh^{-1}\frac{1}{\epsilon}]\cos[(\frac{2k-1}{2N})\pi]$$

$$\epsilon = \sqrt{10^{0.1As}-1} = \sqrt{0.584} = 0.7641$$

$$\sigma_k = -\sinh[\frac{1}{N}\sinh^{-1}\frac{1}{0.7641}]\sin[(\frac{2k-1}{2N})\pi]$$

$$\sigma_1 = -0.4016 \qquad \Omega_1 = 0.813$$

$$\sigma_2 = 0.4016 \qquad \Omega_2 = -0.813$$

Therefore $s_1 = -0.4016 + j0.813$

$$s_2 = -0.4016 - j0.813$$

$$H_n(s) = \frac{K}{(s-s_1)(s-s_2)} = \frac{K}{(s-0.4016+j0.813)(s--0.4016-j0.813)}$$

$$H_n(s) = \frac{K}{s^2+0.8032s+0.82218}$$

$$K = \frac{b_0}{\sqrt{1+\epsilon^2}} \qquad therefore\ K = 0.65226$$

Step 6: analog to digital conversion

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

$$s = \frac{0.00378 + 0.00755z^{-1} + 0.00378z^{-2}}{1 - 1.8626z^{-1} + 0.8816z^{-2}}$$

9.6 MATLAB Program: Design and Implementation of IIR filter to meet given specifications. (Butterworth Filter)

1. Butterworth LPF

```
%the specifications
clc;
clear all;
close all;
alphap=3;
alphas=15;
fp=500;
fs=750;
f=2000;
omp=2*fp/f;
oms=2*fs/f;
%to find cut off frequency & order of the filter
[N,Wn]=buttord(omp,oms,alphap,alphas)
disp('order of the filter n =');
disp(N);
disp('cut off frequency Wn= ');
disp(Wn);
```

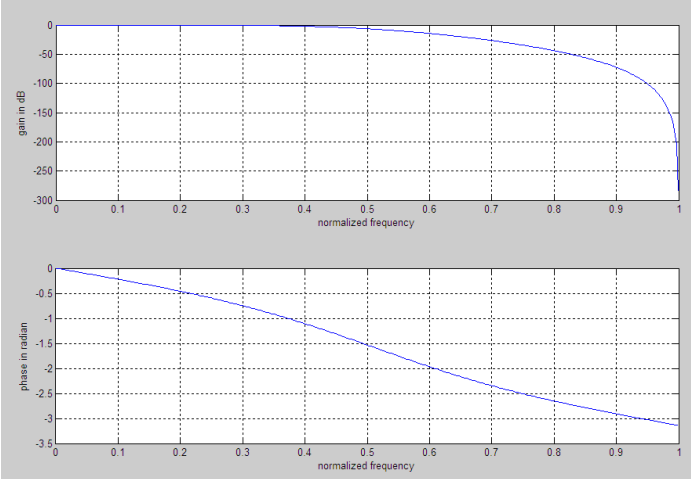
```
%system function of the filter
[b,a]=butter(N,Wn)
w=0:0.01:pi;
[h,om]=freqz(b,a,w,'whole');
m=abs(h);
an=angle(h);
subplot(2,1,1);
plot(om/pi,20*log(m));
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(om/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
% to convert analog filter to digital filter
% using bilinear transformation
[bz,az]=bilinear(b,a,f);
```

Output:

```
N = 2
Wn = 0.5083
order of the filter n = 2
cut off frequency Wn=  0.5083
```

b = 0.3005 0.6011 0.3005

a = 1.0000 0.0304 0.1717



2. Butterworth HPF

```
%the specifications
clc;
clear all;
close all;
alphap=3;
alphas=15;
fp=500;
fs=750;
f=2000;
omp=2*fp/f;
oms=2*fs/f;
%to find cut off frequency & order of the filter
[N,Wn]=buttord(omp,oms,alphap,alphas)
disp('order of the filter n =');
disp(N);
disp('cut off frequency Wn= ');
disp(Wn);
```

```
%system function of the filter
[b,a]=butter(N,Wn,'high')
w=0:0.01:pi;
[h,om]=freqz(b,a,w,'whole');
m=abs(h);
an=angle(h);
subplot(2,1,1);
plot(om/pi,20*log(m));
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(om/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
% to convert analog filter to digital filter
% using bilinear transformation
[bz,az]=bilinear(b,a,f);
```

Output:

N = 2

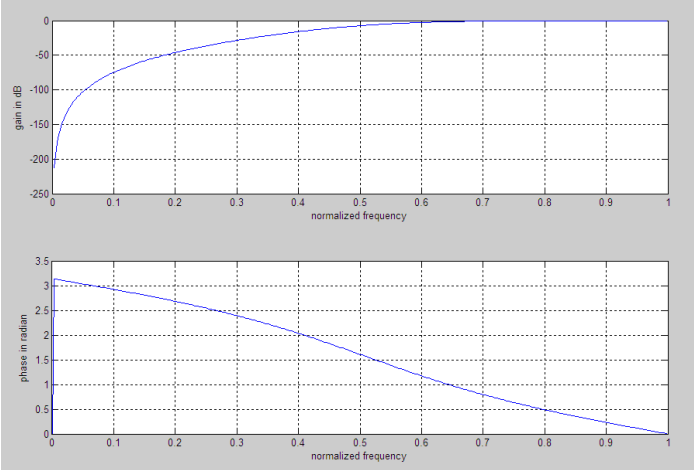
Wn = 0.5083

order of the filter n = 2

cut off frequency Wn= 0.5083

b = 0.2853 -0.5707 0.2853

a = 1.0000 0.0304 0.1717



3. Butterworth BPF

```
clear all;
alphap=2;
alphas=20;
wp=[0.2*pi,0.4*pi];
ws=[0.1*pi,0.5*pi];
[n,wn]=buttord(wp/pi,ws/pi,alphap,alphas)
[b,a]=butter(n,wn)
w=0:0.01:pi;
[h,ph]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(2,1,1);
plot(ph/pi,m);
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(ph/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
```

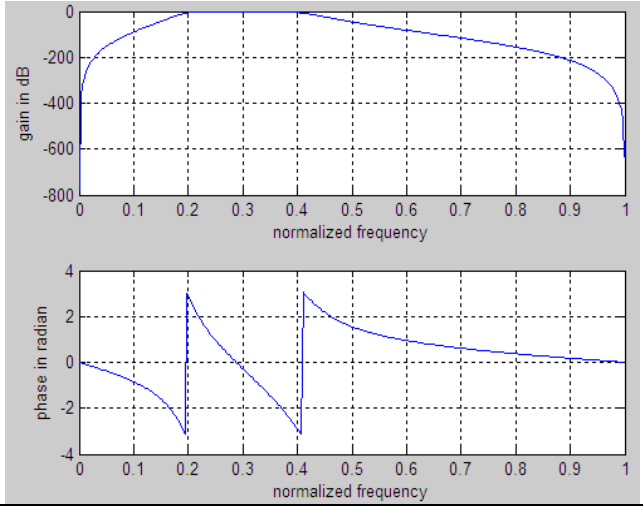
Output:

n = 4

wn = 0.1950 0.4082

b = 0.0060 0 -0.0240 0 0.0359 0 -0.0240 0 0.0060

a = 1.0000 -3.8710 7.9699 -10.6417 10.0781 -6.8167 3.2579 -1.0044 0.1670



4. Butterworth BSF

```
clear all;
alphap=2;
alphas=20;
wp=[0.2*pi,0.4*pi];
ws=[0.1*pi,0.5*pi];
[n,wn]=buttord(wp/pi,ws/pi,alphap,alphas)
[b,a]=butter(n,wn,'stop')
w=0:0.01:pi;
[h,ph]=freqz(b,a,w);
m=20*log(abs(h));
 an=angle(h);
subplot(2,1,1);
plot(ph/pi,m);
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(ph/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
```

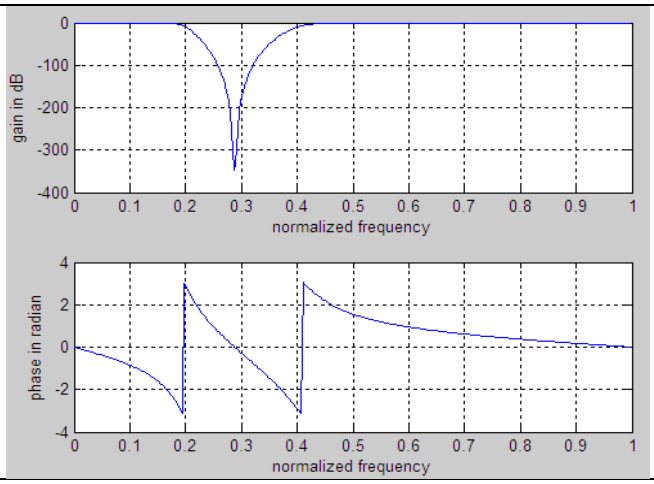
Output:

n = 4

wn = 0.1950 0.4082

b = 0.4086 -2.0201 5.3798 -9.1468 10.8962 -9.1468 5.3798 -2.0201 0.4086

a = 1.0000 -3.8710 7.9699 -10.6417 10.0781 -6.8167 3.2579 -1.0044 0.1670



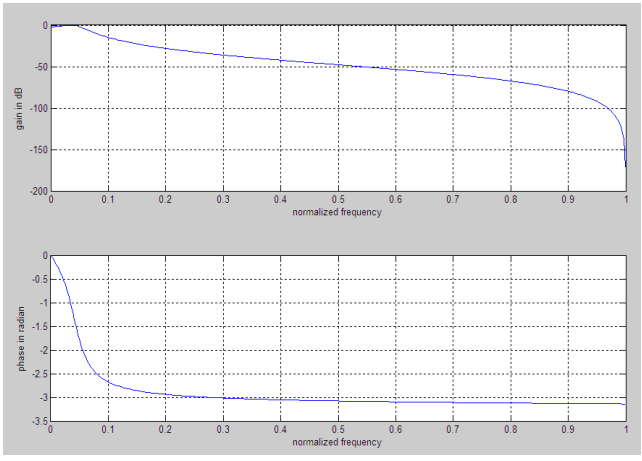
9.7 MATLAB Program: Design and Implementation of IIR filter to meet given specifications. (Chebyshev Filter)

1. Chebyshev type-I LPF

```
clear all;
alphap=2;
alphas=20;
wp=0.05*pi;
ws=0.25*pi;
[n,wn]=cheb1ord(wp/pi,ws/pi,alphap,alphas)
[b,a]=cheby1(n,alphap,wn)
w=0:0.01:pi;
[h,ph]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(2,1,1);
plot(ph/pi,m);
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(ph/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
```

Output:

```
N = 2
Wn = 0.5083
order of the filter n = 2
cut off frequency Wn=  0.5083
b = 0.2853   -0.5707    0.2853
a = 1.0000    0.0304    0.1717
n = 2
wn = 0.0500
b = 0.0038    0.0076    0.0038
a = 1.0000   -1.8625    0.8816
```

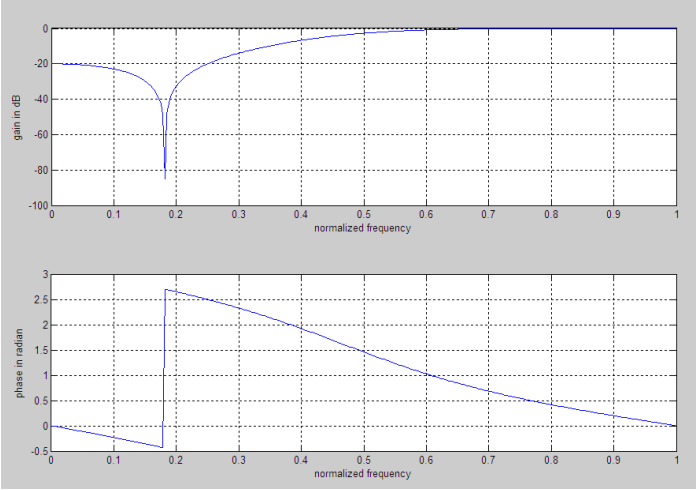


2. Chebyshev type- II HPF

```
clear all;
alphap=2;
alphas=20;
wp=0.05*pi;
ws=0.25*pi;
[n,wn]=cheb2ord(wp/pi,ws/pi,alphap,alphas)
[b,a]=cheby2(n,alphas,wn, 'high')
w=0:0.01:pi;
[h,ph]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(2,1,1);
plot(ph/pi,m);
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(ph/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
```

Output:

```
N = 2
Wn = 0.5083
order of the filter n = 2
cut off frequency Wn=  0.5083
b = 0.2853   -0.5707    0.2853
a = 1.0000    0.0304    0.1717
n = 2
wn = 0.0500
b = 0.0038    0.0076    0.0038
a = 1.0000   -1.8625    0.8816
n = 2
wn = 0.2500
b = 0.1047   -0.1024    0.1047
a = 1.0000   -1.5055    0.6125
n = 2
wn = 0.2500
b = 0.3502   -0.5897    0.3502
a = 1.0000   -0.0917    0.1984
```

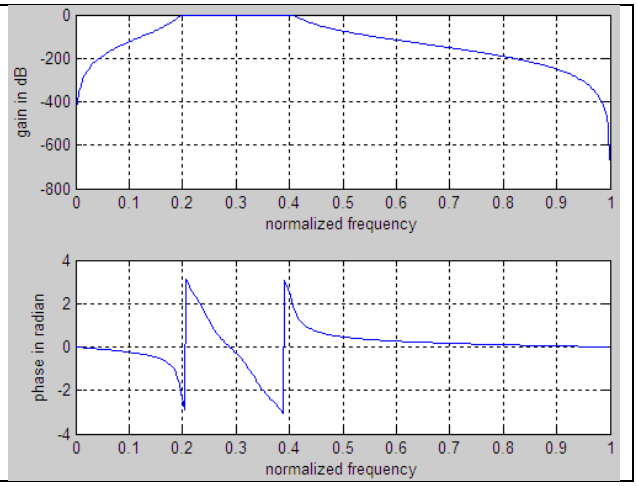


3. Chebyshev type- I BPF

```
clear all;
alphap=3;
alphas=20;
wp=[0.2*pi,0.4*pi];
ws=[0.1*pi,0.5*pi];
[n,wn]=buttord(wp/pi,ws/pi,alphap,alphas)
[b,a]=cheby1(n,alphap,wn)
w=0:0.01:pi;
[h,ph]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(2,1,1);
plot(ph/pi,m);
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(ph/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
```

Output:

```
n =     4
wn =    0.1950    0.4082
b =     0.0017         0   -0.0067         0    0.0101
0   -0.0067         0    0.0017
a =     1.0000   -4.3974   10.5070  -16.3549
18.1728  -14.5121    8.2671   -3.0636    0.6200
```



4. Chebyshev type- II BSF

```
clear all;
alphap=2;
alphas=20;
wp=[0.2*pi,0.4*pi];
ws=[0.1*pi,0.5*pi];
[n,wn]=cheb2ord(wp/pi,ws/pi,alphap,alphas)
[b,a]=cheby2(n,alphas,wn,'stop')
w=0:0.01:pi;
[h,ph]=freqz(b,a,w);
m=20*log(abs(h));
an=angle(h);
subplot(2,1,1);
plot(ph/pi,m);
grid;
ylabel('gain in dB');
xlabel('normalized frequency');
subplot(2,1,2);
plot(ph/pi,an);
grid;
ylabel('phase in radian');
xlabel('normalized frequency');
```

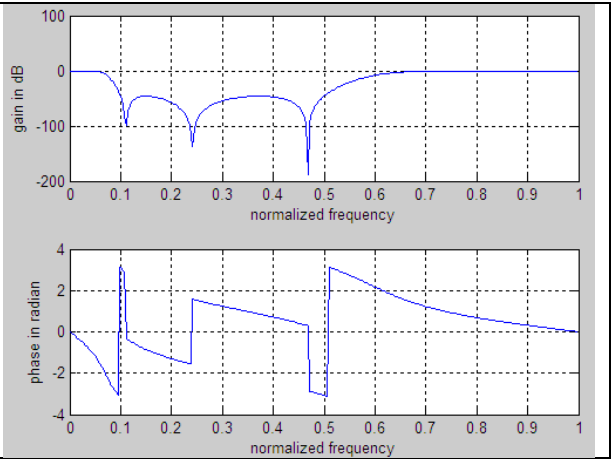
Output:

n = 3

wn = 0.1000 0.5000

b = 0.2537 -0.8969 1.6246 -1.9332 1.6246 -0.8969 0.2537

a = 1.0000 -2.1790 1.8039 -1.1676 0.9185 -0.3803 0.0343



9.8 Output of Butterworth filter

Enter the Pass band edge frequency in Hz = 500

Enter the stop band frequency in Hz = 750

Enter the sampling frequency in Hz = 2000

Enter the pass band ripple n db = 3.01

Enter the stop band attenuation in db = 15

Order of the filter N = 2

Normalized cutoff frequency = 2.052

9.9 Output of Chebyshev filter

Enter the Pass band edge frequency in Hz = 100

Enter the stop band frequency in Hz = 500

Enter the sampling frequency in Hz = 4000

Enter the pass band ripple n db = 2

Enter the stop band attenuation in db = 20

Order of the filter N = 2

Normalized cutoff frequency = 0.1574

9.10 Results

9.11 Conclusion

Experiment 10

10.0 Design and implementation of FIR filter to meet given specifications (using different window techniques).

10.1 Aim

Design and implementation of FIR filter to meet given specifications (low pass filter using hamming window).

10.2 Theory

There are two types of systems – Digital filters (perform signal filtering in time domain) and spectrum analyzers (provide signal representation in the frequency domain). The design of a digital filter is carried out in 3 steps- specifications, approximations and implementation.

DESIGNING AN FIR FILTER (using window method):

Method I: Given the order N, cutoff frequency fc, sampling frequency fs and the window.

- Step 1: Compute the digital cut-off frequency Wc (in the range $-\pi < Wc < \pi$, with π corresponding to fs/2) for fc and fs in Hz. For example let fc=400Hz, fs=8000Hz Wc = 2*π* fc / fs = 2* π * 400/8000 = 0.1* π radians. For MATLAB the Normalized cut-off frequency is in the range 0 and 1, where 1 corresponds to fs/2 (i.e., fmax)). Hence to use the MATLAB commands. wc = fc / (fs/2) = 400/(8000/2) = 0.1

Note: if the cut off frequency is in radians then the normalized frequency is computed as wc = Wc / π

- Step 2: Compute the Impulse Response h(n) of the required FIR filter using the given Window type and the response type (low pass, band pass, etc). For example given a rectangular window, order N=20, and a high pass response, the coefficients (i.e., h[n] samples) of the filter are computed using the MATLAB inbuilt command 'fir1' as h =fir1(N, wc , 'high', boxcar(N+1)); Note: In theory we would have calculated h[n]=hd[n]×w[n], where hd[n] is the desired impulse response (low pass/ high pass, etc given by the sinc function) and w[n] is the window coefficients. We can also plot the window shape as stem(boxcar(N)). Plot the frequency response of the designed filter h(n) using the freqz function and observe the type of response (low pass / high pass /band pass).

Method 2:

Given the pass band (wp in radians) and Stop band edge (ws in radians) frequencies, Pass band ripple Rp and stopband attenuation As.

- Step 1: Select the window depending on the stopband attenuation required. Generally if As>40 dB, choose Hamming window. (Refer table)
- Step 2: Compute the order N based on the edge frequencies as

Transition bandwidth = tb=ws-wp;

N=ceil (6.6*pi/tb);

- Step 3: Compute the digital cut-off frequency Wc as

Wc=(wp+ws)/2

Now compute the normalized frequency in the range 0 to 1 for MATLAB as

wc=Wc/pi;

Note: In step 2 if frequencies are in Hz, then obtain radian frequencies (for computation of tb and N) as wp=2*pi*fp/fs, ws=2*pi*fstop/fs, where fp, fstop and fs are the passband, stop band and sampling frequencies in Hz

- Step 4: Compute the Impulse Response h(n) of the required FIR filter using N, selected window, type of response(low/high,etc) using 'fir1' as in step 2 of method 1.

IMPLEMENTATION OF THE FIR FILTER

1. Once the coefficients of the FIR filter h[n] are obtained, the next step is to simulate an input sequence x[n], say input of 100, 200 & 400 Hz (with sampling frequency of fs), each of 20/30 points. Choose the frequencies such that they are >, < and = to fc.
2. Convolve input sequence x[n] with Impulse Response, i.e., x (n)*h (n) to obtain the output of the filter y[n]. We can also use the 'filter' command.
3. Infer the working of the filter (low pass/ high pass, etc).

10.3 Algorithm

1. Get the sampling frequency
2. Get the pass band frequency
3. Get the stop band frequency
4. Get the pass band ripple and stop band attenuation
5. Select the window suitable for stop band attenuation
6. Calculate the order N based on transition width
7. Find the N window coefficients
8. Find the impulse response of h[n]
9. Verify the frequency response of h[n]

10.4 MATLAB Implementation

FIR1 Function

B = FIR1(N, Wn) designs an N^{th} order low pass FIR digital filter and returns the filter coefficients in length N+1 vector B. The cut-off frequency Wn must be between 0 < Wn < 1.0, with 1.0 corresponding to half the sample rate. The filter B is real and has linear phase, i.e., even symmetric coefficients obeying B (k) = B(N+2-k), k = 1,2,...,N+1.

If Wn is a two-element vector, Wn = [W1 W2], FIR1 returns an order N band pass filter with pass band W1 < W < W2. B = FIR1 (N, Wn, 'high') designs a high pass filter. B = FIR1 (N, Wn, 'stop') is a band stop filter if Wn = [W1 W2]. If Wn is a multi-element vector,

Wn = [W_1 W_2 W_3 W_4 W_5 ... W_N], FIR1 returns an order N multiband filter with bands

$$0 < W < W1, W1 < W < W_2 ... W_N < W < 1.$$

FREQZ Digital filters 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 whose numerator and denominator coefficients are in vectors B and A. The frequency response is evaluated at N points equally spaced around the upper half of the unit circle. If N isn't specified, it defaults to 512. For FIR filter enter A=1 and B = h[n] coefficients. Appropriately choose N as 128, 256, etc

10.5 Design Method

Here we design a lowpass filter using hamming window. Hamming window function is given by,

$$w_H(n) = 0.54 + 0.46 \cos((2\ n)/(N-1)) \quad ; -(N-1)/2 \leq n \leq (N-1)/2$$
$$= 0 \quad ; \text{Otherwise}$$

The frequency response of Hamming window is,

$W_H(e^{jw}) = 0.54[(\sin(w_N/2))/(\sin(w/2)) + 0.23[\sin(w_N/2 - N/N - 1)/\sin(w/2 - /N - 1)] + 0.23[\sin(w_N/2 + N/N - 1)/\sin(w/2 + /N - 1)]$

Commonly used window function characteristics

Window Name	Transition Approximate	Width $\Delta\omega$ Exact values	Min. Stop band Attenuation	Matlab Command
Rectangular	$\frac{4\Pi}{M}$	$\frac{1.8\Pi}{M}$	21db	B = FIR1(N,Wn,boxcar)
Bartlett	$\frac{8\Pi}{M}$	$\frac{6.1\Pi}{M}$	25db	B = FIR1(N,Wn,bartlett)
Hanning	$\frac{8\Pi}{M}$	$\frac{6.2\Pi}{M}$	44db	B = FIR1(N,Wn,hanning)
Hamming	$\frac{8\Pi}{M}$	$\frac{6.6\Pi}{M}$	53db	B= FIR1(N,W_n)
Blackman	$\frac{12\Pi}{M}$	$\frac{11\Pi}{M}$	74db	B = FIR1(N,Wn,blackman)

10.6 MATLAB Program: Design and Implementation of FIR filter using windows Techniques.

1. To plot frequency response of band pass filter using Hamming window

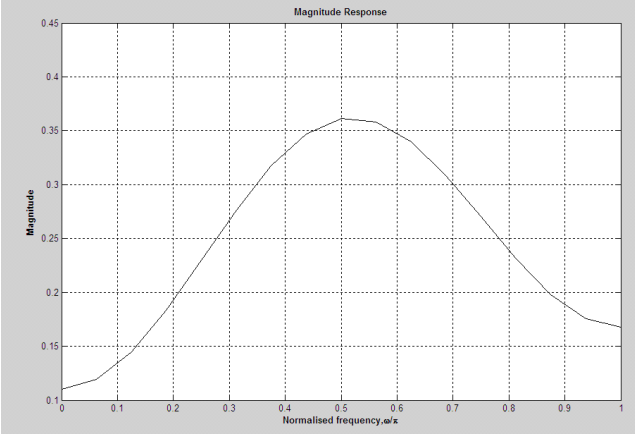
```
clc;
close all;
clear all;
wc1=.4*pi;
wc2=.65*pi;
N=7;
hd=zeros(1,N);
a=(N-1)/2;
hna=(wc2-wc1)/pi;

k=1 : 1 : ((N-1)/2);
n=k-1-((N-1)/2);
w_han(k)=.5-.5*cos(2*pi*(k-1)/(N-1));
hd(k)=(sin(wc2*n)-sin(wc1*n))./(pi*n);

for  s=1 :length(k)
hn(s)=hd(s)*w_han(s);
end
 hn=[hn hna]
a=(N-1)/2;
w=0 : pi/16 : pi;
Hw1=hna*exp(-j*w*a);
Hw2=0;

for m=1:1:a
    Hw3= hn(m)*((exp(j*w*(1-m)))+ (exp(-j*w*(1-m+2*a))));
    Hw2=Hw2+Hw3;
end
Hw=Hw2+Hw1;
H_mag=abs(Hw)
plot(w/pi,H_mag,'k');
grid;
title('Magnitude Response','fontweight','b');
xlabel('Normalised frequency,\omega/\pi','fontweight','b');
ylabel('Magnitude','fontweight','b');
```

Output:

1. Frequency response of band pass filter using Hamming window

hn = 0 -0.0556 -0.0143 0.2500

H_mag = 0.1102 0.1192 0.1449 0.1836 0.2297 0.2766 0.3176 0.3471 0.3612 0.3583 0.3396 0.3085 0.2703 0.2313 0.1979 0.1754 0.1675



2. To plot frequency response of low pass filter using rectangular window.

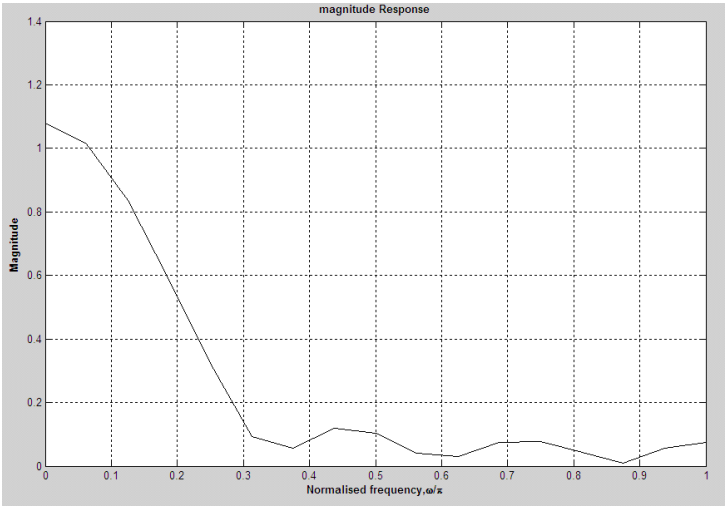
```
clc;
close all;
clear all;
wc=.2*pi;
N=7;
hd=zeros(1,N);
a=(N-1)/2;
hna=wc/pi;
k=1:1:((N-1)/2);
n=k-1-((N-1)/2);
hd(k)=(sin(wc*n))./(pi*n);
hn(k)=hd(k);
hn=[hn hna];
a=(N-1)/2;
w=0:pi/16 : pi;
Hw1=hna*exp(-j*w*a);
Hw2=0;
for m=1:1:a
```

```
    Hw3=hn(m)*((exp(j*w*(1-m)))+(exp(-j*w*(1-m+2*a))));
    Hw2=Hw2+Hw3;
end
Hw=Hw2+Hw1
H_mag=abs(Hw)
plot(w/pi,H_mag,'k');
grid;
title('magnitude Response','fontweight','b');
xlabel('Normalised frequency,\omega/\pi','fontweight','b');
ylabel('Magnitude','fontweight','b');
```

2. Frequency response of low pass filter using rectangular window.

Hw = 1.0787 0.8435 - 0.5636i 0.3203 - 0.7733i -0.1146 - 0.5763i -0.2276 - 0.2276i
-0.0923 - 0.0184i 0.0530 - 0.0219i 0.0660 - 0.0988i 0.0000 - 0.1027i -0.0225 - 0.0337i
0.0270 + 0.0112i 0.0728 - 0.0145i 0.0552 - 0.0552i 0.0086 - 0.0432i 0.0034 + 0.0082i
0.0458 + 0.0306i 0.0733 + 0.0000i

H_mag = 1.0787 1.0145 0.8370 0.5876 0.3219 0.0941 0.0573 0.1188 0.1027
0.0406 0.0292 0.0742 0.0781 0.0441 0.0089 0.0551 0.0733



3. To plot frequency response of high pass filter using Hamming window

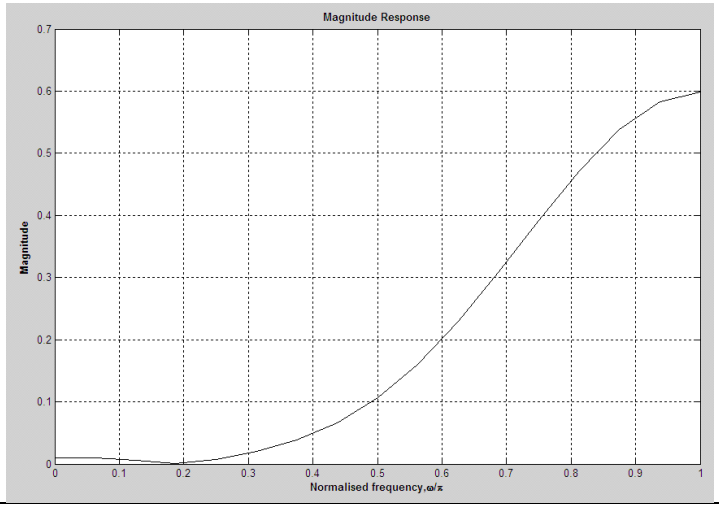
```
clc;
close all;
clear all;
wc=.8*pi;
N=7;
hd=zeros(1,N);
a=(N-1)/2;
hna=1-(wc/pi);
 k=1 : 1 : ((N-1)/2);
n=k-1-((N-1)/2);
w_ham(k)=.54-.46*cos(2*pi*(k-1)/(N-1));
hd(k)=(-sin(wc*n))./(pi*n);
for  s=1 :length(k)
hn(s)=hd(s)*w_ham(s);
end
hn=[hn hna]
a=(N-1)/2;
w=0 : pi/16 : pi;
Hw1=hna*exp(-j*w*a);
Hw2=0;
for m=1:1:a
    Hw3= hn(m)*((exp(j*w*(1-m)))+ (exp(-j*w*(1-m+2*a))));
    Hw2=Hw2+Hw3;
end
Hw=Hw2+Hw1;
H_mag=abs(Hw)
plot(w/pi,H_mag,'k');
grid;
title('Magnitude Response','fontweight','b');
xlabel('Normalised frequency,\omega/\pi','fontweight','b');
ylabel('Magnitude','fontweight','b');
```

Output:

3. To plot frequency response of high pass filter using Hamming window

hn = -0.0081 0.0469 -0.1441 0.2000

H_mag = 0.0104 0.0093 0.0060 0.0005 0.0077 0.0198 0.0383 0.0661 0.1062
0.1605 0.2290 0.3083 0.3923 0.4723 0.5387 0.5827 0.5981



4. To plot frequency response of band stop filter using rectangular window

```
clc;
close all;
clear all;
wc1=.4*pi;
wc2=.65*pi;
N=7;
 hd=zeros(1,N);
a=(N-1)/2;
hna=1-((wc2-wc1)/pi);
 k=1 : 1 : ((N-1)/2);
n=k-1-((N-1)/2);
hd(k)=(sin(wc1*n)-sin(wc2*n))./(pi*n);
hn(k)=hd(k);
hn=[hn hna]
a=(N-1)/2;
w=0 : pi/16 : pi;
Hw1=hna*exp(-j*w*a);
Hw2=0;
```

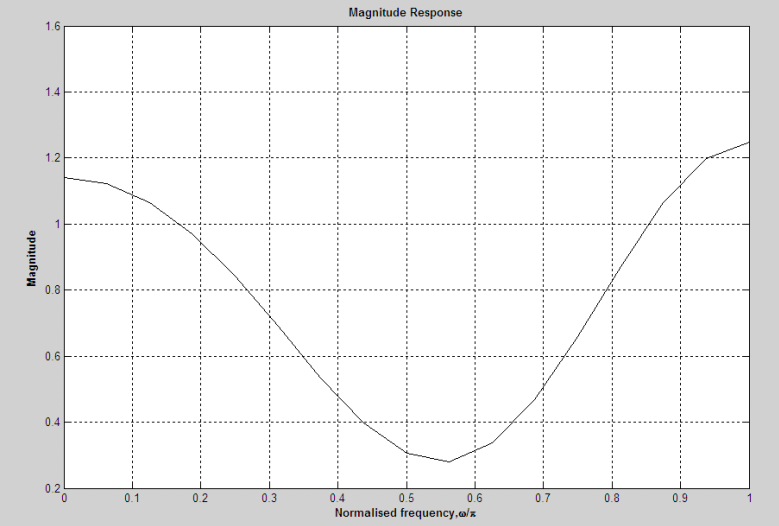
```
 for m=1:1:a
     Hw3= hn(m)*((exp(j*w*(1-m)))+ (exp(-j*w*(1-m+2*a))));
     Hw2=Hw2+Hw3;
end
Hw=Hw2+Hw1;
H_mag=abs(Hw)
plot(w/pi,H_mag,'k');
grid;
title('Magnitude Response','fontweight','b');
xlabel('Normalised frequency,\omega/\pi','fontweight','b');
ylabel('Magnitude','fontweight','b');
```

Output:

4. To plot frequency response of band stop filter using rectangular window

hn = -0.0458 0.2223 0.0191 0.7500

H_mag = 1.1413 1.1222 1.0647 0.9698 0.8418 0.6909 0.5348 0.3975 0.3054
0.2809 0.3364 0.4688 0.6582 0.8705 1.0641 1.1994 1.2479



10.8 Results

10.9 Conclusion

Experiment 11

11.0 Design and implement FIR filter using frequency sampling method.

11.1 Aim

To Design and implement FIR filter using frequency sampling method.

11.2 Theory

A finite impulse response (FIR) filter is discrete linear time invariant system whose output is based on the weighted summation of a finite number of input. An FIR transversal filter structure can be obtained directly from the equation for discrete time convolution

$$y(n)=\sum_{k=0}^{N-1} x(k)h(n-k) \quad 0<n<n-1$$

In this equation x(k) and y(n) represents the input to & output from the filter at time n h(n-k) is the transversal filter coefficient at time n these coefficients are generated by using FDS.

FIR filter is a finite impulse response filter order of the filter should be specified Infinite response in truncated to get finite impulse response placing a window of finite length. This type of available are rectangular barter, Hamming, Hanning Blackman window etc. This FIR filter is an zero filter.

11.3 Algorithm

1. Get the sampling frequency
2. Get the passband & stopband frequency
3. Get the passband ripple & stopband attenuation
4. Calculate the order N based on transition width
5. Find the N window coefficient
6. Verify the frequency response of h(n)

11.4 MATLAB Implementation

The impulse response of filter hn is find out using ifft function.

H= freqz(hn,1,w) returns the n complex frequency response vector H of the filter whose coefficients are in vector hn.

11.5 Calculation

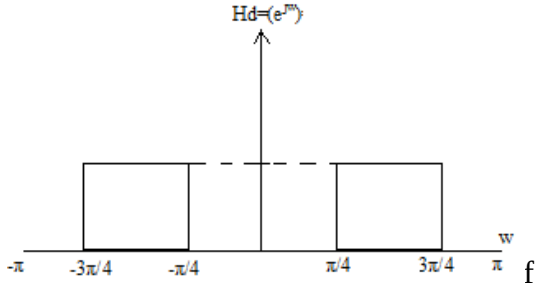
Design an ideal bandpass filter with a frequency response

$Hd=(e^{jw})$= 1 for $\pi/4 <= |w| <= 3\pi/4$

0 otherwise

Find the values of h(n) for N=11 and plot the frequency response .

The ideal frequency response of the filter shown in fig



$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})\cdot e^{j\omega n}$$

$$= \frac{1}{2\pi}\left[\int_{-3\pi/4}^{-\pi/4} e^{j\omega n}\,d\omega + \int_{\pi/4}^{3\pi/4} e^{j\omega n}\,d\omega\right]$$

$$= \frac{1}{2\pi jn}\left[e^{-j\pi n/4} - e^{j3\pi n/4} + e^{j3\pi n/4} - e^{j\pi n/4}\right]$$

$$= \frac{1}{\pi n}\left[\sin\frac{3\pi}{4}n - \sin\frac{\pi}{4}n\right] \quad -\infty \le n \le \infty$$

Truncating hd(n) to 11 samples we have
h(n)= hd(n) for |n|<= 5
0 otherwise
The filter coefficients are symmetrical about n=0 satisfying the condition h(n)=h(-n)
For n=0

$$h(0) = \frac{1}{2\pi}\left[\int_{-3\pi/4}^{-\pi/4} d\omega + \int_{\pi/\omega}^{3\pi/\omega} d\omega\right]$$

$$= \frac{1}{2\pi}\left[-\frac{\pi}{4} + \frac{3\pi}{4} + \frac{3\pi}{4} - \frac{\pi}{4}\right] = \frac{1}{2} = 0.5$$

$$h(1) = h(-1) = 0$$

$$h(2) = h(-2) = \frac{\sin 3\pi/2 - \sin\pi/2}{2\pi} = \frac{-2}{2\pi} = -0.3183$$

$$h(3) = h(-3) = \frac{\sin 9\pi/4 - \sin 3\pi/4}{2\pi} = 0$$

$$h(4) = h(-4) = \sin 3\pi - \sin\pi/4\pi = 0$$

$$h(5) = h(-5) = \sin 15\pi/4 - \sin 5\pi/4\,/5\pi = 0$$

The transfer function of the filter is

$$H(z) = h(0) + \sum_{n=1}^{N-1/2}\left[h(n)\cdot(z^n + z^{-n})\right]$$

$$= 0.5 - 0.3183\,(z^2 + z^{-2})$$

The transfer function of the realizable filter is

$H'(z) = z^{-5}\left[0.5 - 0.3183\left(z^2 + z^{-2}\right)\right]$

$= -0.318 z^{-3} + 0.5 z^{-5} - 0.318 z^{-7}$

$h(0) = h(10) = h(1) = h(9) = h(2) = h(8) = h(4) = h(6) = 0$

$h(3) = h(7) = -0.3183$

$h(5) = 0.5$

$H'(z) = z^{-5}\left[0.5 - 0.3183\left(z^2 + z^{-2}\right)\right]$

$= -0.318 z^{-3} + 0.5 z^{-5} - 0.318 z^{-7}$

$h(0) = h(10) = h(1) = h(9) = h(2) = h(8) = h(4) = h(6) = 0$

$h(3) = h(7) = -0.3183$

$h(5) = 0.5$

$h(0) = h\left[\frac{N-1}{2}\right] = h(5) = 0.5$

$a(n) = 2h\left(\frac{N-1}{2} - n\right)$

$a(1) = 2h(5-1) = 2h(4) = 0$

$a(2) = 2h(5-2) = 2h(3) = 0.6366$

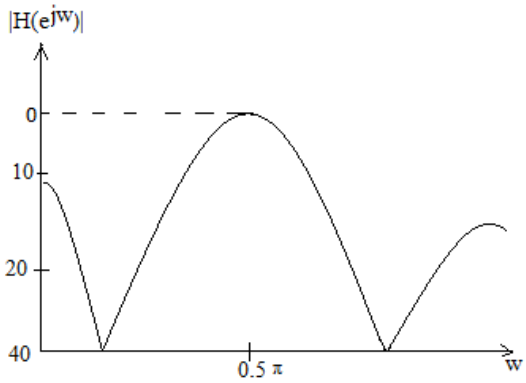
$a(3) = 2h(5-3) = 2h(2) = 0$

$a(4) = 2h(5-4) = 0$

$a(5) = 0$

$H(e^{jw}) = 0.5 - 0.636 \cos 2w$

w in degree	**0**	**20**	**30**	**45**	**60**	**75**	**90**	**105**	**120**	**135**	**150**	**160**	**180**
$\overline{H}(e^{jw})$	-0.1366	0.012	-0.1817	0.5	0.818	1.05	1.1366	1.05	0.818	0.5	0.1817	0.012	-0.1366
in db	-17.3	-38.17	-14.8	-6.02	-1.74	0.4346	1.11	0.4346	-1.74	-6.05	-14.8	-38.17	-17.3



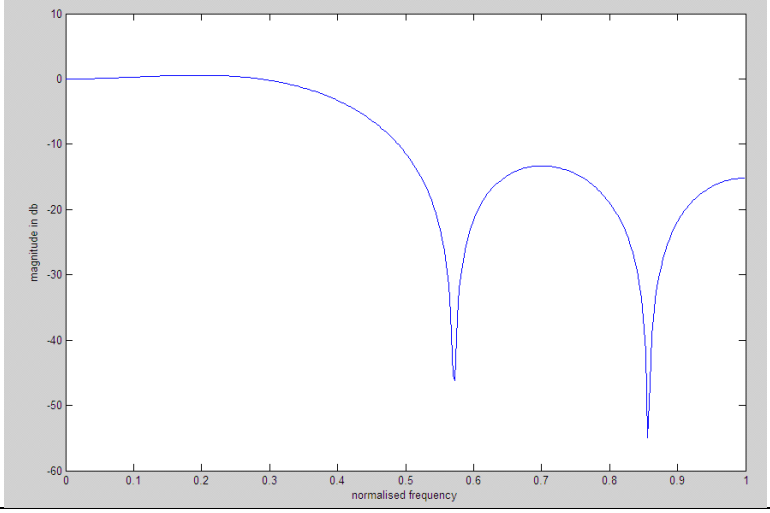
11.6 MATLAB Program: To design and implement FIR filter using frequency sampling method.

```
%sampling method
clc;
clear all;
close all;
N=input('enter the order of the filter N=');
alpha=(N-1)/2;
Hrk=[ones(1,2),zeros(1,4),ones(1,1)];
k1=0:(N-1)/2;
k2=(N+1)/2:N-1;
theetak=[(-alpha*(2*pi)/N)*k1,(alpha*(2*pi)/N)*(N-k2)];
Hk=Hrk.*(exp(i*theetak));
w=0:0.01:pi;
hn=real(ifft(Hk,N))
H=freqz(hn,1,w);
plot(w/pi,20*log10(abs(H)))
ylabel('magnitude in db');
xlabel('normalised frequency');
```

11.7 Output

Input:
enter the order of the filter N=7

Output:
hn = -0.1146 0.0793 0.3210 0.4286 0.3210 0.0793 -0.1146



11.8 Results

11.9 Conclusion

Experiment 12

12.0 To obtain realization of IIR & FIR filters.

12.1 Aim

To obtain realization of IIR & FIR filters.

12.2 Theory

A digital filter transfer can be realized in a variety of ways. There are two types of realization
1. Recursive 2. Non Recursive.

1. For recursive realization the current output y(n) is a function of past outputs, past & present inputs. This form corresponds to an Infinite Impulse Response (IIR) digital filter. In this section we discuss this type of realization.

2. For non-recursive realization current output sample y(n) is a function of only past and present inputs. This form corresponds to a Finite Impulse Response (FIR) digital filter.

- IIR filter can be realized in many forms. They are
- Direct form –I realization
- Direct form –II realization
- Transposed direct form realization
- Cascade form realization
- Parallel form realization
- Lattice-ladder structure

12.3 MATLAB Implementation

The structure are implemented using MATLAB inbuilt functions residuez and tf.

12.4 Calculation

1. Realize the system with difference equation
$y(n)= \frac{3}{4}y(n-1)- \frac{1}{8}y(n-2)+x(n)+ \frac{1}{3}x(n-1)$ is cascade form
$\rightarrow y(n)= \frac{3}{4}y(n-1)- \frac{1}{8}y(n-2)+x(n)+ \frac{1}{3}x(n-1)$

$$Y(z)-\frac{3}{4}z^{-1}Y(z)+\frac{1}{8}z^{-2}Y(z)=X(z)+\frac{1}{3}z^{-1}X(z)$$

$$H(z)=\frac{Y(z)}{X(z)}=\frac{1+\frac{1}{3}z^{-1}}{1-\frac{3}{4}z^{-1}+\frac{1}{8}z^{-2}}$$

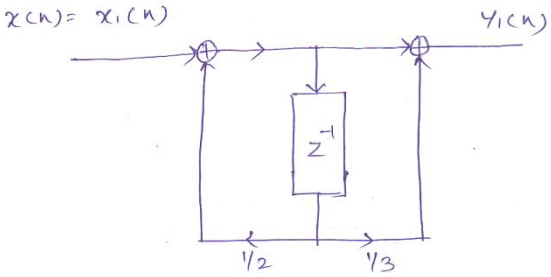
$$=\frac{1+3z^{-1}}{(1-\frac{1}{2}z^{-1})(1-\frac{1}{4}z^{-1})}$$

$$=H_1(z)\,H_2(z)$$

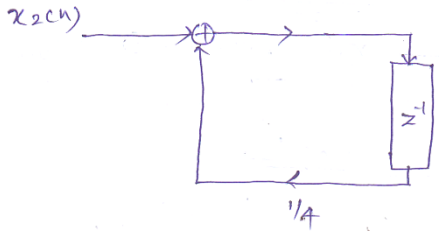
Here

$$H_1(z) = \frac{1+\frac{1}{3}z^{-1}}{1-\frac{1}{2}z^{-1}} \quad \& \quad H2(z) = \frac{1}{1-\frac{1}{4}z^{-1}}$$

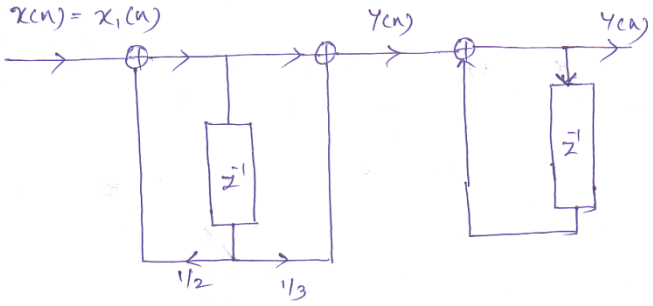
$H_{1(z)}$ can be realized in direct form II as



Similarly H_{2z} can be realized in direct form II as



Cascading the realization of H1(z) & H2(z) we have



2. Realize the system given by difference equation

y(n)=-0.1(n-1)+0.72(n-2)_0.7x(n)-0.252x(n-2) in parallel form

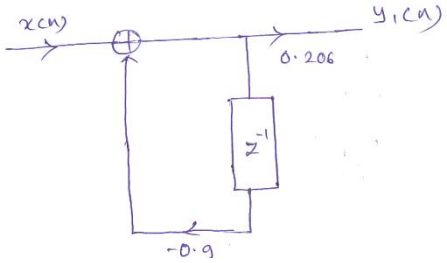
$$H(z) = \frac{0.7 - 0.252z^{-2}}{1+0.1z^{-1}-0.72z^{-2}}$$

$$= 0.35 + \frac{0.35 - 0.035z^{-1}}{1+0.1z^{-1}-0.72z^{-2}}$$

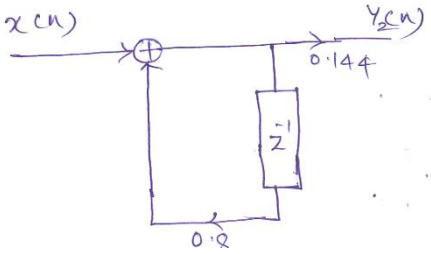
$$= 0.35 + \frac{0.206}{1+0.9z^{-1}} + \frac{0.144}{1-0.8z^{-1}}$$

$$= C + H_1(z) + H_2(z)$$

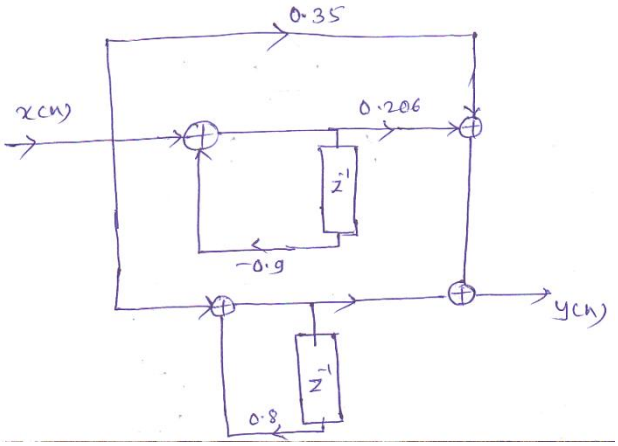
H1(z) can be realized in direct form II as



H2(z) can be realized in direct form II as



Now the realization of H(z) is shown below



12.5 MATLAB Program: To obtain the parallel form realization

```
%realization of IIR parallel structure
num=input('enter the coefficients of numerator=');
den=input('enter the coefficients of denominator=');
[r1 p1 k1]=residuez(num,den);
disp('parallel form')
disp('residues are')
disp(r1)
disp('poles are at=')
disp(p1)
disp('constant value')
disp(k1)
```

1. Realization of IIR parallel structure

Input

enter the coefficients of numerator=[3 8 12 7 2 -2]

enter the coefficients of denominator=[16 24 24 14 5 1]

Output

parallel form

residues are

-0.4219 + 0.6201i

-0.4219 - 0.6201i

2.3438 0.3437 - 2.5079i

0.3437 + 2.5079i

poles are at=

-0.2500 + 0.6614i

-0.2500 - 0.6614i

-0.5000

-0.2500 + 0.4330i

-0.2500 - 0.4330i

constant value

-2

12.6 MATLAB Program: To obtain the cascade form realization

```
%realization of FIR cascade structure
num=input('enter the coefficients of numerator=');
den=input('enter the coefficients of denominator=');
[b,a]=eqtflength(num, den);
[z, p, k]=tf2zp(b,a);
sos=zp2sos(z,p,k)
```

12.7 Output

2. Realization of IIR cascade structure

Input:

enter the coefficients of numerator=[2 10 23 34 31 16 4]

enter the coefficients of denominator=[1]

Output:

sos =

2.0000	6.0000	4.0000	1.0000	0	0
1.0000	1.0000	2.0000	1.0000	0	0
1.0000	1.0000	0.5000	1.0000	0	0

12.8 Result

12.9 Conclusion