METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY
(Affiliated to Osmania University & Approved by AICTE, New Delhi)

Estd:2008

LABORATORY MANUAL

DIGITAL SIGNAL PROCESSING LABORATORY

B.E, VIII Semester (CBCS): 2021-22

NAME: _____

ROLL NO:_____

BRANCH:_____

SEM:_____

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Empower youth- Architects of Future World

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

_____

—

VISION

To produce ethical, socially conscious and innovative professionals who would contribute to sustainable technological development of the society.

MISSION

To impart quality engineering education with latest technological developments and interdisciplinary skills to make students succeed in professional practice.

To encourage research culture among faculty and students by establishing state of art laboratories and exposing them to modern industrial and organizational practices.

To inculcate humane qualities like environmental consciousness, leadership, social values, professional ethics and engage in independent and lifelong learning for sustainable contribution to the society.

DEPARTMENT
OF
ELECTRICAL AND ELECTRONICS ENGINEERING

LABORATORY MANUAL

DIGITAL SIGNAL PROCESSING LABORATORY

Prepared
By

Mrs. A. Archana,

Assistant Professor

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

VISION

To become a reputed centre for imparting quality education in Electrical and Electronics Engineering with human values, ethics and social responsibility.

MISSION

- To impart fundamental knowledge of Electrical, Electronics and Computational Technology.

- To develop professional skills through hands-on experience aligned to industry needs.

- To undertake research in sunrise areas of Electrical and Electronics Engineering.

- To motivate and facilitate individual and team activities to enhance personality skills.

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES

BE-Electrical Engineering graduates shall be able to:

- **PEO1**. Utilize domain knowledge required for analyzing and resolving practical Electrical Engineering problems.

- **PEO2**.Willing to undertake inter-disciplinary projects, demonstrate the professional skills and flair for investigation.
- **PEO3**. Imbibe the state of the art technologies in the ever transforming technical scenario.

- **PEO4**. Exhibit social and professional ethics for sustainable development of the society.

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF ELECTRICAL AND ELCTRONICS ENGINEERING

PROGRAM OUTCOMES

Engineering Graduates will have ability to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of electrical and electronics engineering problems.

- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex electrical and electronics engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

- **PO3. Design/development of solutions:** Design solutions for complex electrical and electronics engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex electrical and electronics engineering activities with an understanding of the limitations.

- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional electrical and electronics engineering practice.

- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- **PO.8 Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the electrical and electronics engineering practice.

- **PO9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

- **PO12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

At the end of BE program Electrical and Electronics Engineering graduates will be able to:

- **PSO1**.Provide effective solutions in the fields of Power Electronics, Power Systems and Electrical Machines using MATLAB/MULTISIM.
- **PSO2.** Design and Develop various Electrical and Electronics Systems, particularly Renewable Energy Systems.
- **PSO3.** Demonstrate the overall knowledge and contribute for the betterment of the society.

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

I.   PREREQUISITE(S):

Level	Credits	Semester	Prerequisites
UG	1	VI	**DIGITAL SIGNAL PROCESSING**

II.   SCHEME OF INSTRUCTIONS

Lectures	Tutorials	Practicals	Credits
0	0	2	1

III. SCHEME OF EVALUATION & GRADING

S. No	Component	Duration	Maximum Marks
	Continuous Internal Evaluation (CIE)		
1.	Internal Examination – I and II	1 hour each	25
	CIE (Total)		**25**
2.	**Semester End Examination** (University Examination)	3 hours	**50**
		TOTAL	**75**

%Marks Range	>=90	80 to < 90	70 to < 80	60 to < 70	50 to < 60	40 to < 50	< 40	Absent
Grade	S	A	B	C	D	E	F	Ab
Grade Point	10	9	8	7	6	5	0	-

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

COURSE OUTCOMES

After completing this course the student will be able to:

CO No.	Course Outcome	Taxonomy Level
C463.1	Develop code to generate basic waves	**Apply**
C463.2	Develop code perform basic operations on them	**Apply**
C463.3	Develop code to obtain linear and circular convolution	**Apply**
C463.4	Develop code to obtain DFT and FFT.	**Apply**
C463.5	Develop code to design FIR filters.	**Apply**
C463.6	Develop code to design IIR filters	**Apply**

MAPPING OF COs WITH POs & PSOs

Correlation Level: High – 3; Medium – 2; Low – 1

PO / CO	PO 1	PO 2	PO 3	P O4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS 01	PS O2	PS O3
C463.1	3	3	2	2	3	-	-	2	3	2	-	-	3	2	2
C463.2	3	3	2	2	3	-	-	2	3	2	-	-	3	2	2
C463.3	3	3	2	2	3	-	-	2	3	2	-	-	3	2	2
C463.4	3	3	2	2	3	-	-	2	3	2	-	-	3	2	2
C463.5	3	3	2	2	3	-	-	2	3	2	-	-	3	2	2
C463.6	3	3	2	2	3	-	-	2	3	2	-	-	3	2	2
C463	3	3	2	2	3	-	-	2	3	2	-	-	3	2	2

LABORATORY CODE OF CONDUCT

1. Students should report to the labs concerned as per the scheduled time table.

2. Students, who report late to the labs will not be permitted to perform the experiment scheduled for the day.

3. Students to bring a 100 pages note book to enter the readings /observations while performing the experiment.

4. After completion of the experiment, certification of the staff in-charge concerned, in the observation book is necessary.

5. Staff member in-charge shall evaluate for 25 marks, each experiment, based on continuous evaluation which will be entered in the continuous internal evaluation sheet.

6. The record of observations, along with the detailed procedure of the experiment performed in the immediate previous session should be submitted for certification by the staff member in-charge.

7. Not more than three students in a group would be permitted to perform the experiment on the equipment-based lab set up. However only one student is permitted per computer system for computer-based labs.

8. The group-wise division made at the start of the semester should be adhered to, and no mix up with any other group would be allowed.

9. The components required, pertaining to the experiment should be collected from the stores in-charge, after duly filling in the requisition form / log register.

10. After the completion of the experiment, students should disconnect the setup made by them, and return all the components / instruments taken for the purpose, in order.

11. Any damage of the equipment or burn-out of components will be charged at cost as a penalty or the total group of students would be dismissed from the lab for the semester/year.

12. Students should be present in the lab for the total time duration, as scheduled.

13. Students are required to prepare thoroughly, before coming to Laboratory to perform the experiment.

14. Procedure sheets / data sheets provided to the students, if any, should be maintained neatly and returned after the completion of the experiment.

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

DOS AND DON'TS IN THE LABORATORY

<u>**Do's**</u>

1. Remove your shoes or wear foot socks before you enter the lab.
2. Always keep quiet. Be considerate to other lab users.
3. Report any problems with the computer to the person in charge.
4. Shut down the computer properly.

<u>**Don'ts**</u>

1. Do not bring any food or drinks in the computer room.
2. Do not touch any part of the computer with wet hands.
3. Do not hit the keys on the computer too hard.
4. Don't damage, remove, or disconnect any labels, parts, cables or equipment.
5. Do not install or download any software or modify or delete any system files on any lab computers.
6. If you leave the lab, do not leave your personal belongings unattended.

<u>**Before Leaving Lab:**</u>

- Place the stools properly
- Turn off the power to computers
- Please check the laboratory notice board regularly for updates

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

CONTENTS

Sl. No.	Name of Experiment
1	Generation of different discrete signal sequences and waveforms.
2	Basic operations on Discrete Time Signals
3	DFT Computation and FFT Algorithms.
4	Verification of Convolution Theorem.
5	Verification of Sampling Theorem.
6	Design of Butterworth and Chebyshev LP and HP filters.
7	Design of LPF using Rectangular, Hamming and Kaiser Windows.
8	To perform linear and circular convolution for the given sequences.
9	Design and implementation of FIR and IIR filter.
10	Computation of DFT using DIT and DIF algorithm.
Additional Experiments	
11	Design of LPF using Blackman Window
12	Design of HPF using Hamming Window

Introduction to MATLAB

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.

MATLAB is developed by Math Works.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and Fortran; analyze data; develop algorithms; and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots and performing numerical methods.

MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

Features of MATLAB

Following are the basic features of MATLAB:

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality and maintainability and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

Uses of MATLAB

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- Signal Processing and Communications
- Image and Video Processing
- Control Systems
- Test and Measurement
- Computational Finance
- Computational Biology

The MATLAB environment

The MATLAB environment (on most computer systems) consists of menus, buttons and a writing area similar to an ordinary word processor. There are plenty of help functions that can be used. The writing area that is appeared when MATLAB is started is called the *command window*. In this window, the commands to MATLAB can be given. For example, if a program written in MATLAB has to be run, the program has to be typed in the command window by its name at the prompt. The command window is also useful in using MATLAB as a scientific calculator or as a graphing tool. If longer programs are to be written then, it more convenient to write the program code in a separate window, and then run it in the command window.

In the command window, a prompt that looks like >> is visible. Commands can be typed immediately after this prompt. Once the command are typed, press <enter> for MATLAB to perform. If *a command* that MATLAB is running has to be *interrupted,* type <ctrl> + <c>.

The commands typed in the command window are stored by MATLAB and can be viewed in the *Command History* window. To repeat a command that is already used, simply double-click on the command in the history window, or use the <up arrow> at the command prompt to iterate through the commands that is used until the command desired to repeat is reached.

LIBRARY FUNCTIONS

clc:

clc clears the command window and homes the cursor.

clear all:

clear: clear variables and functions from memory. clear removes all variables

from the workspace. clear variables does the same thing.

close all:

close: close figure. close, by itself, closes the current figure window.

close all: closes all the open figure windows.

exp:

exp: exponential.

exp(x) is the exponential of the elements of x, e to the x.

input:

input prompt for user input.

r = input('how many apples') gives the user the prompt in the text string and then waits for input from the keyboard. the input can be any matlab expression, which is evaluated, using the variables in the current workspace, and the result returned in r. if the user presses the return key without entering anything, input returns an empty matrix.

linspace:

linspace linearly spaced vector.

linspace(x1, x2) generates a row vector of 100 linearly equally spaced points between x1 and x2.

rand:

the rand function generates arrays of random numbers whose elements are uniformly distributed in the interval (0,1).

ones:

ones(n) is an n-by-n matrix of ones.
ones(m,n) or ones([m,n]) is an m-by-n matrix of ones.

zeros:

zeros(n) is an n-by-n matrix of zeros.

zeros(m,n) or zeros([m,n]) is an m-by-n matrix of zeros

plot:

plot linear plot.

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, whichever line up.

subplot:

subplot create axes in tiled positions.

h = subplot(m,n,p), or subplot(mnp), breaks the figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axis handle. the axes are counted along the top row of the figure window, then the second row, etc.

stem:

stem 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.

stem(x,y) plots the data sequence y at the values specified in x.

title:

title graph title.

title('text') adds text at the top of the current axis.

xlabel:

xlabel x-axis label.

xlabel('text') adds text beside the x-axis on the current axis.

ylabel:

ylabel y-axis label.

ylabel('text') adds text beside the y-axis on the current axis.

fprintf:

write formatted data to file. The special formats \n,\r,\t,\b,\f can be used to produce linefeed, carriage return, tab, backspace, and form feed characters respectively.

Expt. No. 1. Generation of different discrete signal sequences and waveforms

AIM: Write a MATLAB program to plot basic sequences like Unit Impulse Sequence, Unit Step Sequence, Ramp Sequence and Exponential Sequence and to plot different waveforms like Triangular pulse, Rectangular pulse, Periodic sine, Saw tooth and Square wave.

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

PROGRAM:

```
%WAVE FORM GENERATION
%CT SIGNAL
%UNIT IMPULSE
clc;
clear all;
close all;
t1=-3:1:3;
x1=[0,0,0,1,0,0,0];
subplot(2,3,1);
plot(t1,x1);
xlabel('time');
ylabel('Amplitude');
title('Unit impulse signal');
%UNIT STEP SIGNAL
t2=-5:1:25;
x2=[zeros(1,5),ones(1,26)];
subplot(2,3,2);
plot(t2,x2);
xlabel('time');
ylabel('Amplitude');
title('Unit step signal');
%EXPONENTIAL SIGNAL
a=input('Enter the value of a:');
t3=-10:1:20;
x3=exp(-1*a*t3);
subplot(2,3,3);
plot(t3,x3);
xlabel('time');
ylabel('Amplitude');
title('Exponential signal');
%UNIT RAMP SIGNAL
t4=-10:1:20;
x4=t4;
subplot(2,3,4);
plot(t4,x4);
xlabel('time');
```

```matlab
ylabel('Amplitude');
title('Unit ramp signal');
%SINUSOIDAL SIGNAL
A=input('Enter the amplitude:');
f=input('Enter the frequency:');
t5=-10:1:20;
x5=A*sin(2*pi*f*t5);
subplot(2,3,5);
plot(t5,x5)
xlabel('time');
ylabel('Amplitude');
title('Sinusoidal signal');
%RANDOM SIGNAL
t6=-10:1:20;
x6=rand(1,31);
subplot(2,3,6);
plot(t6,x6);
xlabel('time');
ylabel('Amplitude');
title('Random signal');
%WAVE FORM GENERATION
%DT SIGNAL
%UNIT IMPULSE
clc;
clear all;
close all;
n1=-3:1:3;
x1=[0,0,0,1,0,0,0];
subplot(2,3,1);
stem(n1,x1);
xlabel('time');
ylabel('Amplitude');
title('Unit impulse signal');
%UNIT STEP SIGNAL
n2=-5:1:25;
x2=[zeros(1,5),ones(1,26)];
subplot(2,3,2);
stem(n2,x2);
xlabel('time');
ylabel('Amplitude');
title('Unit step signal');
%EXPONENTIAL SIGNAL
a=input('Enter the value of a:');
n3=-10:1:20;
x3=power(a,n3);
subplot(2,3,3);
stem(n3,x3);
xlabel('time');
ylabel('Amplitude');
```
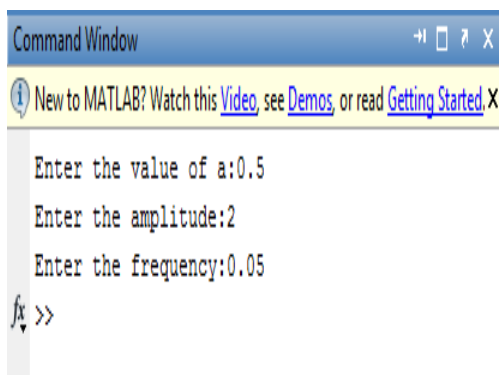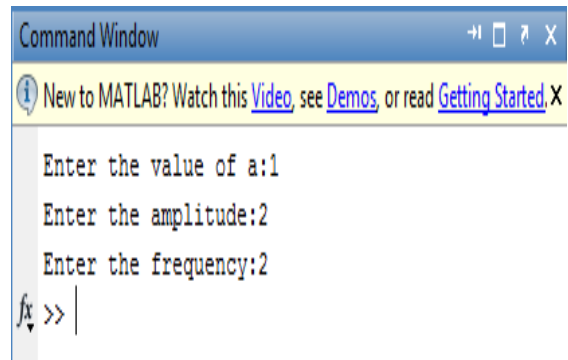
```matlab
title('Exponential signal');
%UNIT RAMP SIGNAL
n4=-10:1:20;
x4=n4;
subplot(2,3,4);
stem(n4,x4);
xlabel('time');
ylabel('Amplitude');
title('Unit ramp signal');
%SINUSOIDAL SIGNAL
A=input('Enter the amplitude:');
f=input('Enter the frequency:');
n5=-10:1:20;
x5=A*sin(2*pi*f*n5);
subplot(2,3,5);
stem(n5,x5);
xlabel('time');
ylabel('Amplitude');
title('Sinusoidal signal');
%RANDOM SIGNAL
n6=-10:1:20;
x6=rand(1,31);
subplot(2,3,6);
stem(n6,x6);
xlabel('time');
ylabel('Amplitude');
title('Random signal');
```

CONTINUOUS TIME:

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started. X

```
Enter the value of a:0.5
Enter the amplitude:2
Enter the frequency:0.05
fx >>
```

DISCRETE TIME:

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started. X

```
Enter the value of a:1
Enter the amplitude:2
Enter the frequency:2
fx >>
```

OUTPUT WAVEFORM
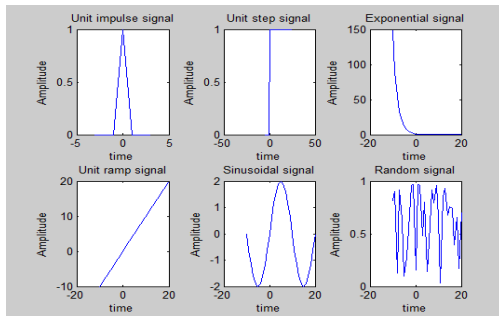
(CONTINUOUS TIME):   **(DISCRETE TIME):**
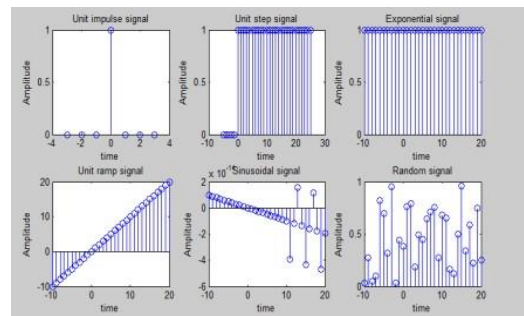
RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. Classify continuous signals.
5. Classify discrete signals?
6. Distinguish between discrete signal and digital signal.

Expt. No. 2. Basic operations on Discrete Time Signals

AIM: Write a program in MATLAB to study the basic operations on the Discrete – time signals. (Operation on dependent variable (amplitude manipulation) and Operation on independent variable (time manipulation).

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

PROGRAM

%%

clear all;

close all;

clc;

%operations on the amplitude of signal

 x=input('Enter input sequence:');

a=input('Enter amplification factor:');

 b=input('Enter attenuation factor:');

 c=input('Enter amplitude reversal factor:');

 y1=a*x;

y2=b*x;

 y3=c*x;

 n=length(x);

 subplot(2,2,1);

stem(0:n-1,x);

xlabel('time');

ylabel('amplitude');

title('Input signal');

subplot(2,2,2);

```
stem(0:n-1,y1);
 xlabel('time');
ylabel('Amplitude');
title('Amplified signal');
subplot(2,2,3);
stem(0:n-1,y2);
xlabel('time');
ylabel('Amplitude');
 title('Attenuated signal');
 subplot(2,2,4);
stem(0:n-1,y3);
xlabel('time');
ylabel('Amplitude');
 title('Amplitude reversal signal');
 %scalar addition
d=input('Input the scalar to be added:');
y4=d+x;
 figure(2);
stem(0:n-1,y4);
 xlabel('time');
ylabel('Amplitude');
title('Scalar addition signal');
%%
```

```matlab
%Time manipulations:
clear all;
 close all;
clc;
%Operations on the independent variable
 %Time shifting of the independent variable
x=input('Enter the input sequence:');
n0=input('Enter the +ve shift:');
n1=input('Enter the -ve shift:');
 l=length(x);
subplot(2,2,1);
 stem(0:l-1,x);
xlabel('time');
ylabel('Amplitude');
title('Input signal');
i=n0:(l+n0-1);
 j=n1:(l+n1-1);
subplot(2,2,2);
stem(i,x);
 xlabel('time');
ylabel('Amplitude');
 title('Positive shifted signal');
subplot(2,2,3);
stem(j,x);
xlabel('time');
```

```matlab
ylabel('Amplitude');

title('Negative shifted signal');

%Time reversal subplot(2,2,4);

stem(-1*(0:l-1),x);

xlabel('time');

ylabel('Amplitude');

title('Time reversal signal');

%%

%Arithmetic Operations:

clear all;

close all;

clc;

%Arithmetic operations on signals

%Addition and multiplication of two signals

x1=input('Enter the sequence of first signal:');

x2=input('Enter the sequence of second signal:');

l1=length(x1);

l2=length(x2);

subplot(2,2,1);

stem(0:l1-1,x1);

xlabel('time');

ylabel('Amplitude');

title('Input sequence 1');

subplot(2,2,2);

stem(0:l2-1,x2);
```

```matlab
xlabel('time');

ylabel('Amplitude');

title('Input sequence 2');

 if l1>l2

 l3=l1-l2;

x2=[x2,zeros(1,l3)];

y1=x1+x2;

subplot(2,2,3);

 stem(0:l1-1,y1);

xlabel('time');

ylabel('Amplitude');

title('Addition of two signals');

y2=x1.*x2; subplot(2,2,4);

stem(0:l1-1,y2);

xlabel('time');

ylabel('Amplitude');

 title('Multiplication of two signals');

end

 if l2>l1

l3=l2-l1;

x1=[x1,zeros(1,l3)];

 y1=x1+x2; subplot(2,2,3);

 stem(0:l2-1,y1);

 xlabel('time');

 ylabel('Amplitude');

title('Addition of two signals');
```

```matlab
 y2=x1.*x2;
subplot(2,2,4);
stem(0:l2-1,y2);
 xlabel('time');
ylabel('Amplitude');
title('Multiplication of two signals');
 else
y1=x1+x2;
subplot(2,2,3);
stem(0:l1-1,y1);
 xlabel('time');
ylabel('Amplitude');
title('Addition of two signals');
y2=x1.*x2;
 subplot(2,2,4);
 stem(0:l1-1,y2);
 xlabel('time');
ylabel('Amplitude');
 title('Multiplication of two signals');
 end
%%
```

Operations on the amplitude of signal :   **Time shifting of the independent variable :**

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started. ×

    Enter input sequence:[1,2,3]
    Enter amplification factor:2
    Enter attenuation factor:0.5
    Enter amplitude reversal factor:-1
    Input the scalar to be added:2
fx >>
```

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started. ×

    Enter the input sequence:[1,2,3]
    Enter the +ve shift:2
    Enter the -ve shift:4
fx >>
```

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started. ×

    Enter the sequence of first signal:[1,2,3,4]
    Enter the sequence of second signal:[2,4,1]
fx >>
```

Addition and multiplication of two signals :

OUTPUT WAVEFORMS:

Operation on the amplitude of signal:
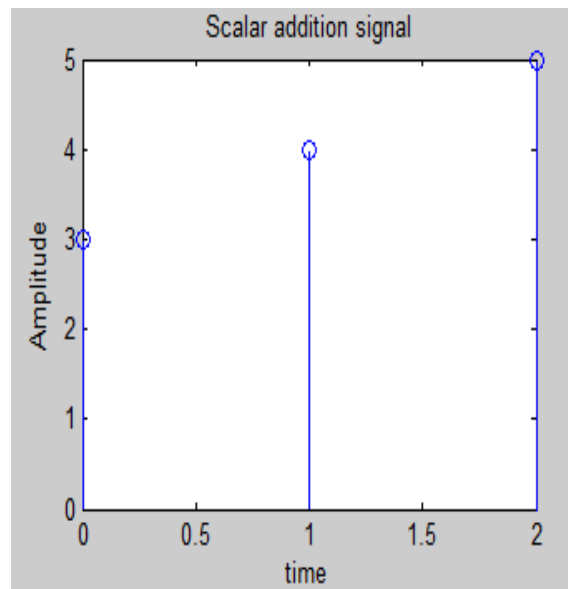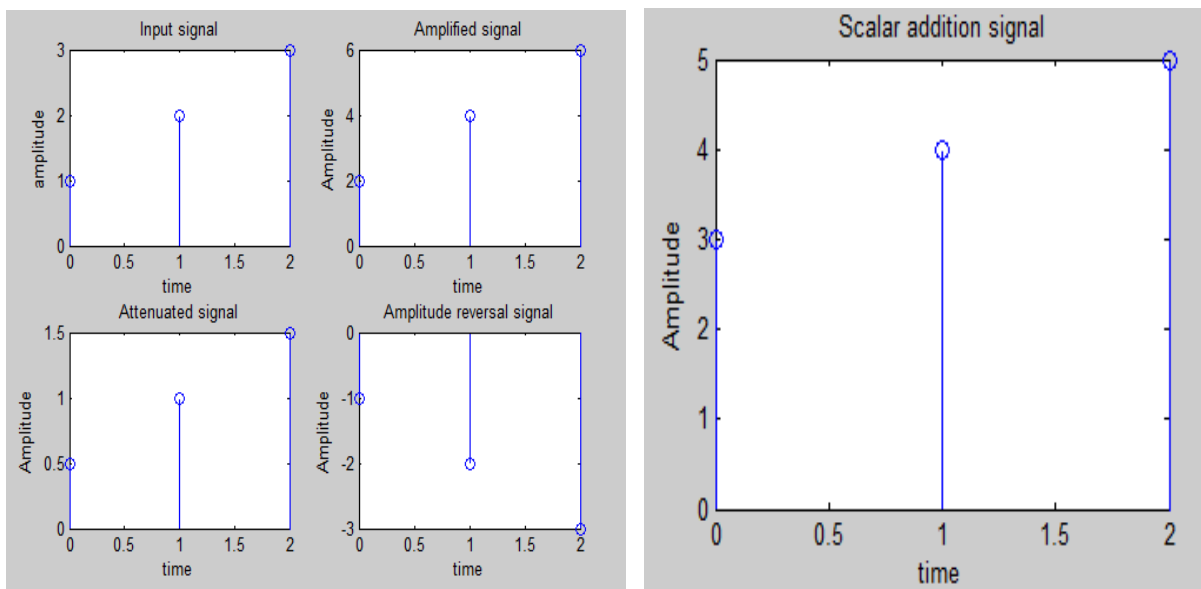
Time shifting of the independent variable: **Addition and multiplication of two signals:**
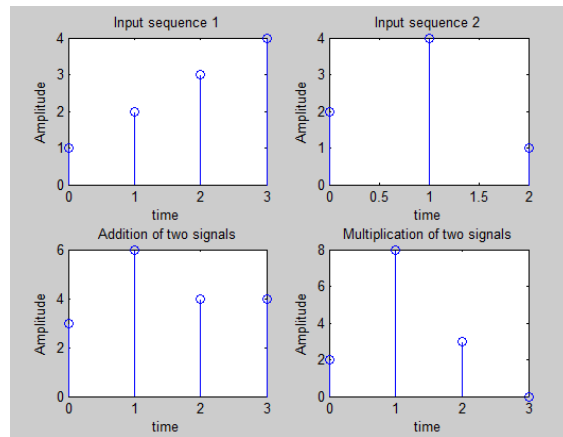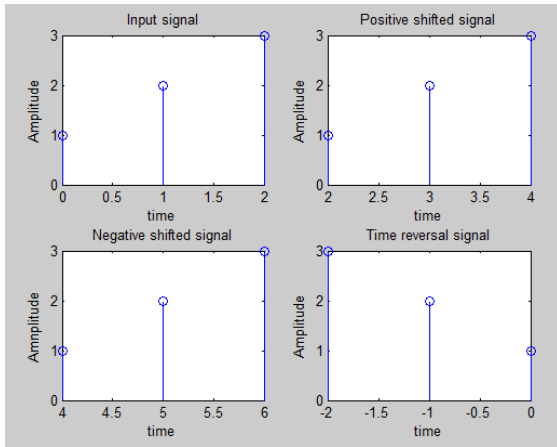
RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. How to add two discrete signals?
5. What is scalar multiplication?
6. Explain about time shifting property?

Expt. No. 3. DFT and IDFT Computation & FFT Computation

AIM: Write a MATLAB program to compute the DFT and IDFT for a discrete signal.

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

PROGRAM:

```
%DFT FFT
 clc;
close all;
clear all;
xn=input('Enter inputs:');
 N=length(xn);
 in=menu('DFT/IDFT or FFT/IFFT','DFT','FFT');
 if(in==1)
n=0:N-1;
k=0:N-1;
wn=exp((-1i*2*pi*k'*n)/N);
X=wn*xn';
display(X);
Wn=exp((1i*2*pi*k'*n)/N);
 x_n=Wn*X/N;
 display(x_n);
 else
X=fft(xn);
display(X);
 x_n=ifft(X);
display(x_n);
end
```

OUTPUT:

DFT and IDFT:

```
Enter inputs:[1:8]

X =

   36.0000 + 0.0000i
   -4.0000 + 9.6569i
   -4.0000 + 4.0000i
   -4.0000 + 1.6569i
   -4.0000 - 0.0000i
   -4.0000 - 1.6569i
   -4.0000 - 4.0000i
   -4.0000 - 9.6569i


x_n =

    1.0000 - 0.0000i
    2.0000 + 0.0000i
    3.0000 - 0.0000i
    4.0000 - 0.0000i
    5.0000 - 0.0000i
    6.0000 + 0.0000i
    7.0000 + 0.0000i
    8.0000 - 0.0000i
```

FFT and IFFT:

```
Enter inputs:[1:8]

X =

  Columns 1 through 3

  36.0000 + 0.0000i  -4.0000 + 9.6569i  -4.0000 + 4.0000i

  Columns 4 through 6

  -4.0000 + 1.6569i  -4.0000 + 0.0000i  -4.0000 - 1.6569i

  Columns 7 through 8

  -4.0000 - 4.0000i  -4.0000 - 9.6569i


x_n =

     1     2     3     4     5     6     7     8
```

RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What is DFT?
5. Why to perform DFT and IDFT on sequences?
6. What is the advantage of FFT?

Expt. No. 4. Verification of Convolution Theorem

AIM: Write a MATLAB program to determine the convolution of two sequences using linear and circular convolution methods.

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

PROGRAM:

```
%Convolution Theorem
 clear all;
close all;
 clc;
a=input('Enter first input:');
 b=input('Enter second input:');
input=menu('menu','linear convolution','linear usin circular','circular convolution');
 if(input==1)
 h=conv(a,b);
display('The result of Linear Convolution is:');
display(h);
plot(h);
stem(h)
 if (input==2)
X=fft([a zeros(1,length(b)-1)]);
display(X);
Y=fft([b zeros(1,length(a)-1)]);
 display(Y); h=ifft(X.*Y);
 display('The result of linear using circular is:');
display(h);
```

```
plot(h);

stem(h);

else

if(length(a)==length(b))

 X=fft(a);

display(X);

Y=fft(b);

 display(Y);

h=ifft(X.*Y);

display('The result of circular Convolution is:');

display(h);

 plot(h);

 stem(h);

else

display('circular Convolution cannot be performed');

end

end
```

OUTPUT:

```
Enter  first  input:[1:3]
Enter  second  input:[1  1  1]
```

LINEAR CONVOLUTION:

The result of Linear Convolution is:

h =

        1   3   6   5   3
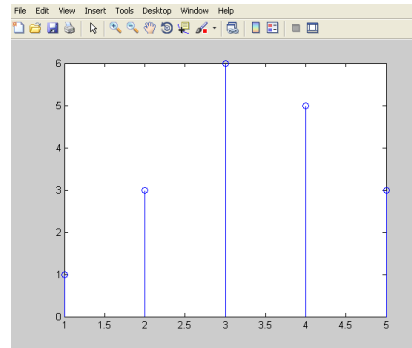
LINEAR USING CIRCULAR:

```
Enter first input:[1 2 3]
Enter second input:[1 1 1]

X =

  Columns 1 through 3

   6.0000 + 0.0000i  -0.8090 - 3.6655i   0.3090 + 1.6776i

  Columns 4 through 5

   0.3090 - 1.6776i  -0.8090 + 3.6655i


Y =

  Columns 1 through 3

   3.0000 + 0.0000i   0.5000 - 1.5388i   0.5000 + 0.3633i

  Columns 4 through 5

   0.5000 - 0.3633i   0.5000 + 1.5388i

The result of linear using circular is:
```

h =

    1.0000   3.0000   6.0000   5.0000   3.0000
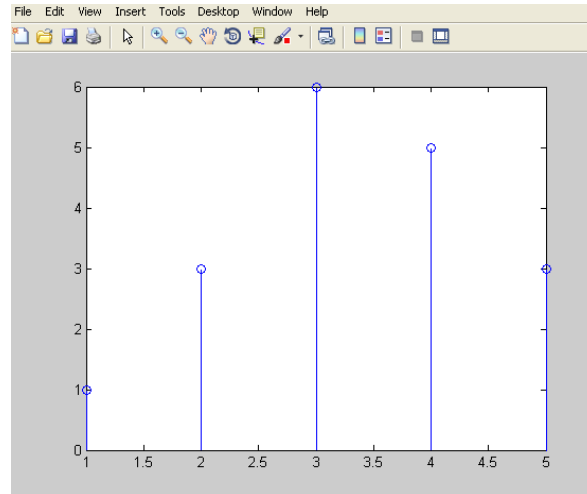
CIRCULAR CONVOLUTION:

Enter first input:[1 2 3]
Enter second input:[1 1 1]

X =

   6.0000 + 0.0000i -1.5000 + 0.8660i -1.5000 - 0.8660i

Y =

     3   0   0

The result of circular Convolution is:

h =

     6   6   6
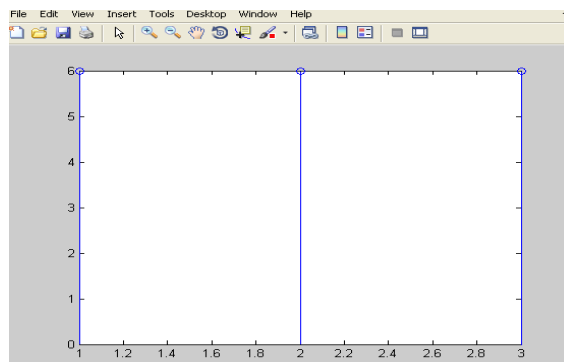
RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What is convolution?
5. Distinguish between linear and circular convolution?

Expt. No. 5. Verification of Sampling Theorem

AIM: Write a MATLAB program to verify sampling theorem.

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

PROGRAM:

```
%sampling theorem

clear all;

 close all;

clc;

tfinal=0.05;

t=0:0.0005:tfinal;

fd=input('Enter the fundemental analog signal frequency:');

xt=sin(2*pi*fd*t);

 choice=menu('menu','Under Sampling','Nyquist Sampling','Over Sampling');

if(choice==1)

 fs=1.3*fd;

 n=0:1/fs:tfinal;

xn=sin(2*pi*n*fd);

plot(t,xt,'b',n,xn,'r*-');

title('Under Sampled Plot');

xlabel('Time');

ylabel('Amplitude');

 legend('Analog','Discrete');

elseif(choice==2)

fs=3*fd;
```

```
n=0:1/fs:tfinal;

xn=sin(2*pi*n*fd);

 plot(t,xt,'b',n,xn,'r*-');

 title('Nyquist Sampled Plot');

xlabel('Time');

ylabel('Amplitude');

 legend('Analog','Discrete');

 else

fs=10*fd;

n=0:1/fs:tfinal;

 xn=sin(2*pi*n*fd);

 plot(t,xt,'b',n,xn,'r*-');

title('Over Sampled Plot');

xlabel('Time');

 ylabel('Amplitude');

legend('Analog','Discrete');

 end
```

OUTPUT:

Enter the fundemental analog signal frequency:20
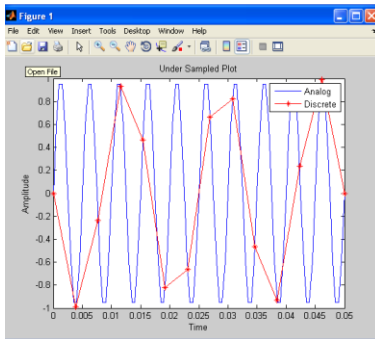
Under Sampled:	**Nyquist Sampled(For sine wave):**	**Over Sampled:**
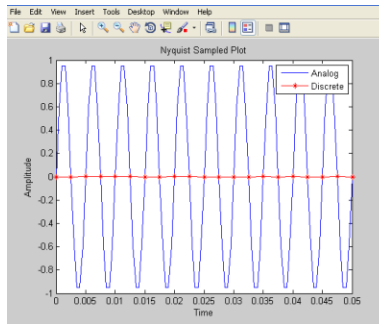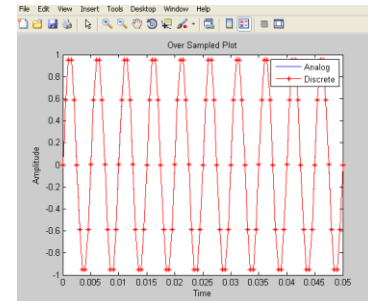
RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What is sampling?
5. What is the need for sampling?

Expt. No. 6. Design of Butterworth and Chebyshev LP and HP filters

AIM: Write a MATLAB program to verify the Butterworth and chebyshev analog LP and HP filters.

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

PROGRAM :

%Design of Butterworth and Chebyshev LPF and HPF

```
clc;
close all;
clear all;
disp('Enter IIR filter specifications');
rp=input('Enter pass band ripple:');
rs=input('Enter stop band ripple:');
fp=input('Enter pass band frequency:');
fs=input('Enter stop band frequency:');
F=input('Enter sampling frequency:');
wp=(fp)/F;
ws=(fs)/F;
c=menu('Enter choice','Butterworth LP','Butterworth HP','Chebyshev LP','Chebyshev HP');
if(c==1)
    [n,wc]=buttord(wp,ws,rp,rs);
    display(n);
    display(wc);
    [b,a]=butter(n,wc,'low');
    display(b);
    displayelseIse
  if(c==2)
    [n,wc]=buttord(wp,ws,rp,rs);
    display(n);
```

```
    display(wc);
    [b,a]=butter(n,wc,'high'); display(b);

display(a); elseif(c==3)

    [n,wc]=cheb1ord(wp,ws,rp,rs);

    display(n);

    display(wc);

    [b,a]=cheby1(n,rp,wc,'low');

    display(b);

display(a); elseif(c==4)

    [n,wc]=cheb1ord(wp,ws,rp,rs);

    display(n);

    display(wc);

    [b,a]=cheby1(n,rp,wc,'high');

    display(b);

display(a); end

freqz(b,a,1024,2*F);

title('nfilter');

disp('Enter any key to get H(z)');

pause; [bz,az]=impinvar(b,a,2*F);

disp(az);

disp(bz);
```

OUTPUT:

```
Enter  IIR filter specifications
Enter  pass band ripple:3
Enter  stop band ripple:60
Enter  pass band frequency:150
Enter  stop band frequency:40
Enter  sampling frequency:500
```

MENU
Enter choice

Butterworth LP

Butterworth HP

Chebyshev LP

Chebyshev HP
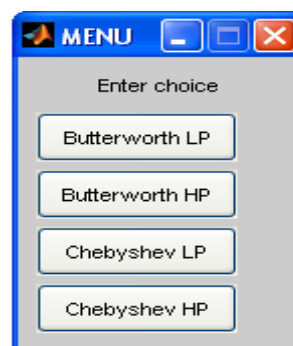
Butterworth Low Pass Filter:


```
n =

     5

wc =

   0.2967


b =

   0.0066    0.0331    0.0662    0.0662    0.0331    0.0066


a =

   1.0000   -2.0092    2.0609   -1.1366    0.3392   -0.0423

Enter any key to get H(z)
   1.0000   -5.0020   10.0080  -10.0121    5.0080   -1.0020

   1.0e-03 *

   0.0531   -0.2188    0.3447   -0.2519    0.0796   -0.0066
```
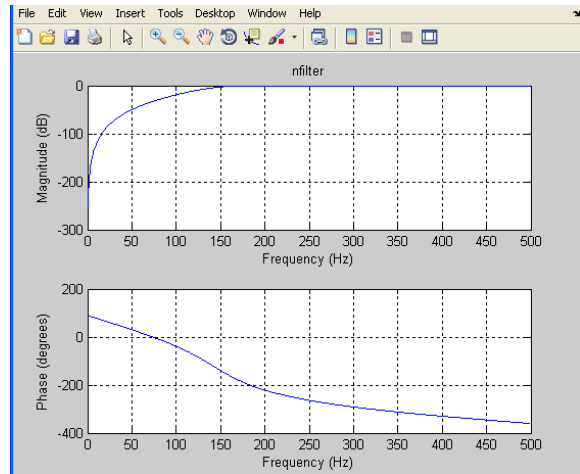
Butterworth High Pass Filter:
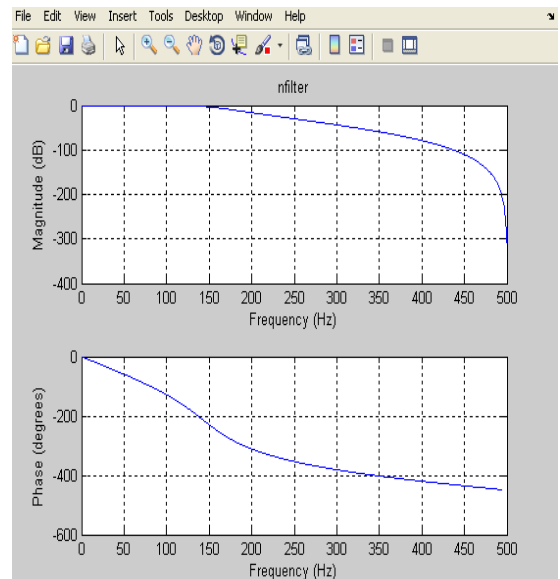
```
n =

    5

wc =

    0.2967

b =

    0.2059   -1.0294    2.0588   -2.0588    1.0294   -0.2059

a =

    1.0000   -2.0092    2.0609   -1.1366    0.3392   -0.0423

Enter any key to get H(z)
    1.0000   -5.0020   10.0080  -10.0121    5.0080   -1.0020

   -0.0004    0.0014   -0.0016    0.0004    0.0004   -0.0002
```

Chebyshev LowPass Filter:

```
n =

    4

wc =

    0.3000

b =

    0.0051    0.0203    0.0304    0.0203    0.0051

a =

    1.0000   -2.6649    3.2814   -2.0817    0.5798

Enter any key to get H(z)
    1.0000   -4.0027    6.0080   -4.0080    1.0027

   1.0e-03 *

    0.0389   -0.1217    0.1318   -0.0541    0.0051
```
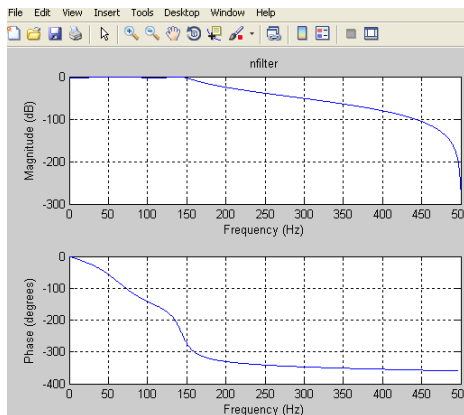

Chebyshev High Pass Filter:

```
n =

    4

wc =

    0.3000

b =

    0.1508   -0.6031    0.9046   -0.6031    0.1508

a =

    1.0000   -0.8386    1.0339   -0.2163    0.3185

Enter any key to get H(z)
    1.0000   -4.0008    6.0025   -4.0025    1.0008

   1.0e-03 *

   -0.3259    0.8274   -0.5264   -0.1261    0.1509
```
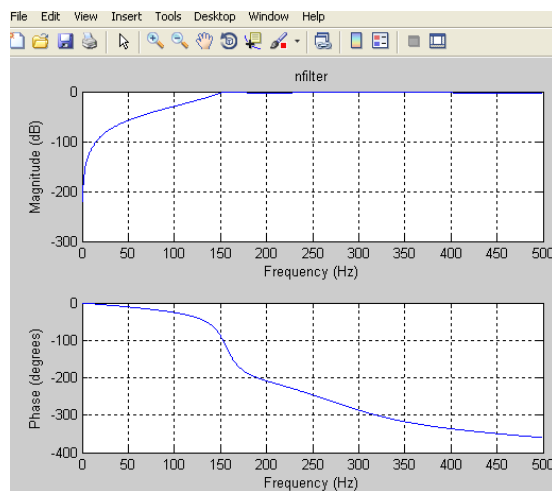
RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What are various filters available in DSP?
5. What is the need for filters?
6. Distinguish between LPF and HPF?

Expt. No. 7. Design of FIR LP filter using Rectangular, Hamming and Kaiser Windows

AIM: Write a MATLAB program to verify the FIR LP filter using Rectangular, Hamming and Kaiser windows.

APPARATUS:
1. PC with Windows OS.
2. MATLAB tool.

PROGRAM:

```
% FIR LP filter design using windows
 clc;
close all;
clear all;
N=input('Enter the order of the filter:');
Fc=input('Enter cut off frequency:');
Fs=input('Enter sampling frequency:');
input=menu('window selection','Rectangular','Hamming','Kaiser');
wc=Fc/Fs;
   if(input==1) wn=rectwin(N);
   elseif(input==2)
      wn=hamming(N);
else wn=kaiser(N); end
b=fir1(N-1,wc,wn);
freqz(b,1,512)
```

OUTPUT:

```
Enter the order of the filter:48
Enter cut off frequency:500
Enter sampling frequency:5000
```
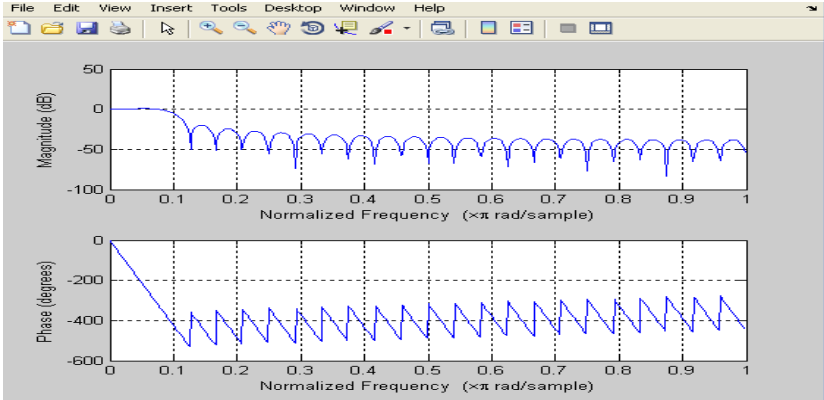
MENU

window selection

Rectangular

Hamming

Kaiser

Rectangular Window:
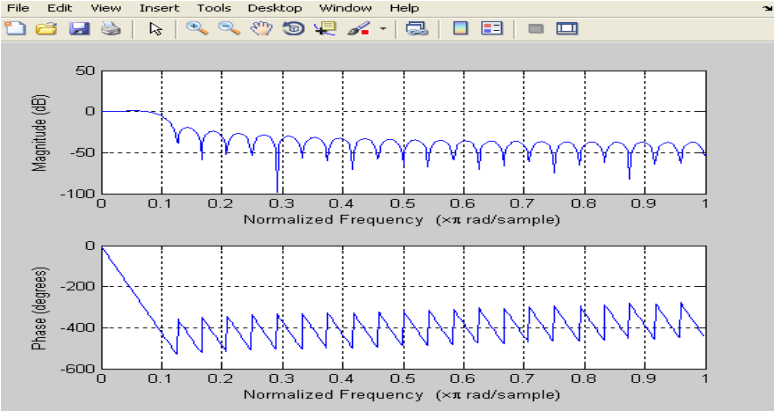
Hamming Window:
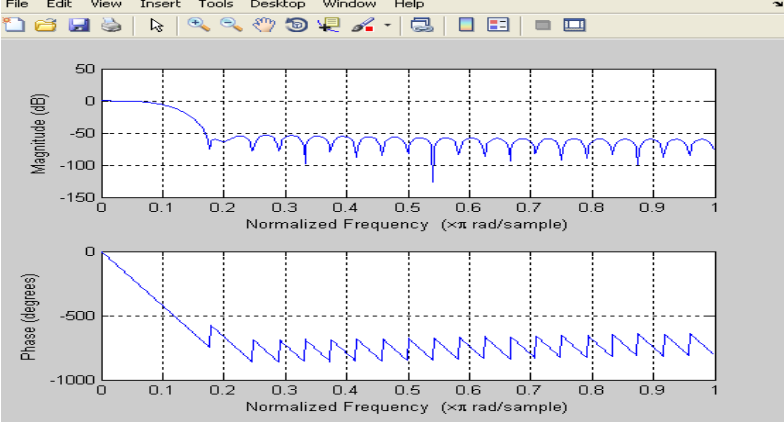
Kaiser Window:

RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What are various filters available in DSP?
5. What is the need for filters?
6. Distinguish between LPF and HPF?

EXPERIMENTS ON DSP TRAINER KIT
INTRODUCTION

TMS320C6745 DSP320 Features:

The DSP features the TMS320C6745 DSP320, a 375 MHz device delivering up to 3648 million instructions per second (MIPs) and 2736 MFLOPS. This DSP generation is designed for applications that require high precision accuracy. The C6745 is based on the TMS320C6000 DSP platform designed to needs of high performing high-precision applications such as pro-audio, medical and diagnostic.

Other hardware features of the TMS320C6745 DSK board include:

- Embedded JTAG supported via USB
- TLV320AIC23B programmable stereo codec
- Two 3.5mm audio jacks for microphone and speaker
- Expansion for port connector for plug-in modules
- Power supply : +5V, ±12V, GND
- 8 DIP switches for inputs
- 8 LED indication for output
- Provision for manual Reset
- 4*4 LED matrix
- Noise generator : White noise generator
- : Amplitude 0 ~ 5Vpp
- 20*2 character LCD display.
- 2 No. 7 segment displays.
- RTC interface : I2C based RTC section
- Phone keypad : 0 to 9 digits and *, # characters

Software -Code Composer Studio v5™ DSP320

- A complete Integrated Development Environment (IDE), an efficient optimizing C/C++ compiler assembler, linker, debugger, an a advanced editor with Code Maestro™ technology for faster code creation, data visualization, a profiler and a flexible project manager DSP/BIOS™ real-time

kernel Target error recovery software DSP320 diagnostic tool "Plug-in" ability for third-party software for additional functionality.

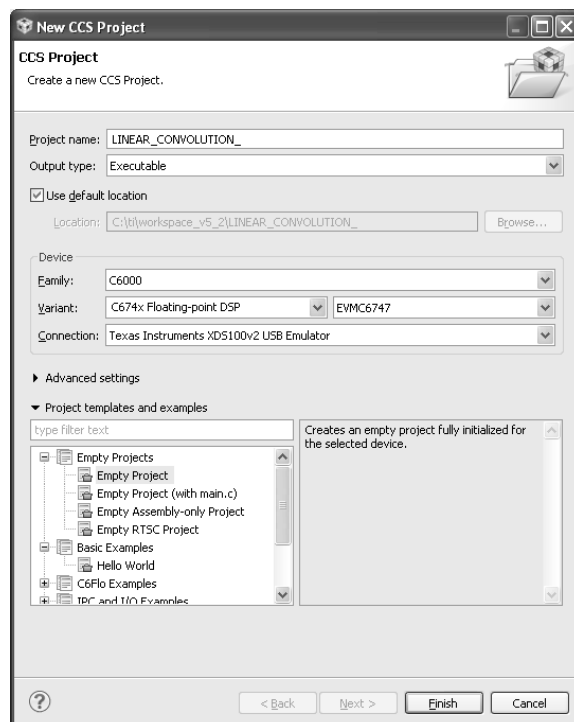- **PROCEDURE TO WORK ON CODE COMPOSER STUDIO:**

1. Double click on Code Composer Studio v5 icon which is on desktop.

2. It will open the workspace window as follows.
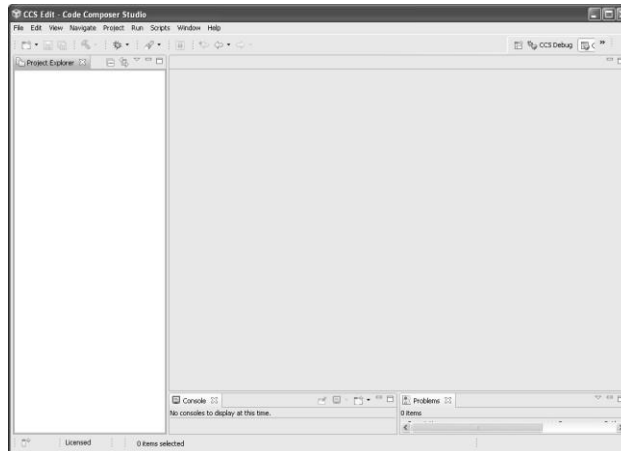
To create a new project,

- Go to **Project>New CCS Project**.

    Following window should appear

o Give name to project say "YOUR ROLL NO" with location to save project or use default location.

o Check that output type must be **Executable**.

o Device family is **C6000**.

o The Variant is **C674x Floating-point DSP** and next to this you have to select **EVMC6747**. In device connection, select **Texas Instruments XDS100v2 USB Emulator**. In project templates and example section, you have to select an **Empty project**.

o Click on 'Finish' button. It starts creating an empty project.

o After finishing project creation. Empty Project will appear in left window of software, as shown below.
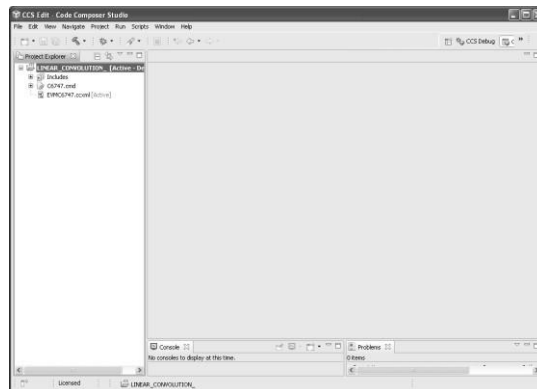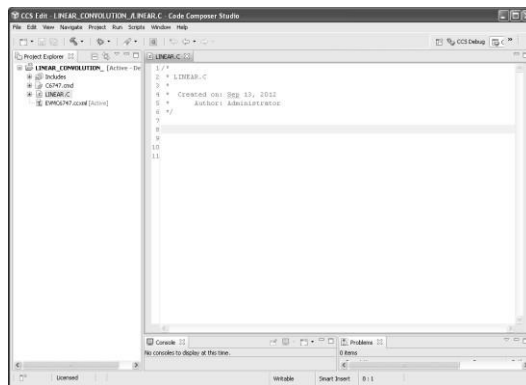

o Create a source file. Go to File>New>source file.
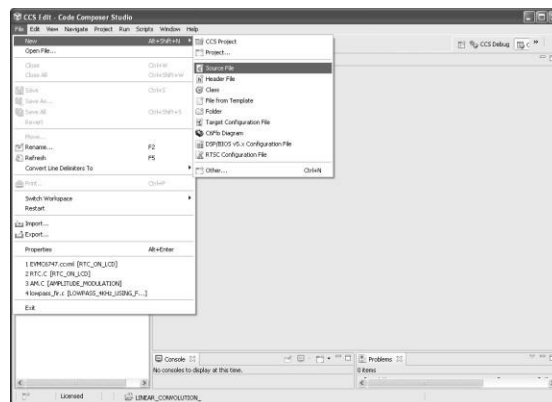
The new source file popup window should appear containing **source folder**, type in Linear convolution. In **Source file** section have same file name with say "**linear convolution**" with extension **.c**, Select template as a **Default C++ source template** as shown below:
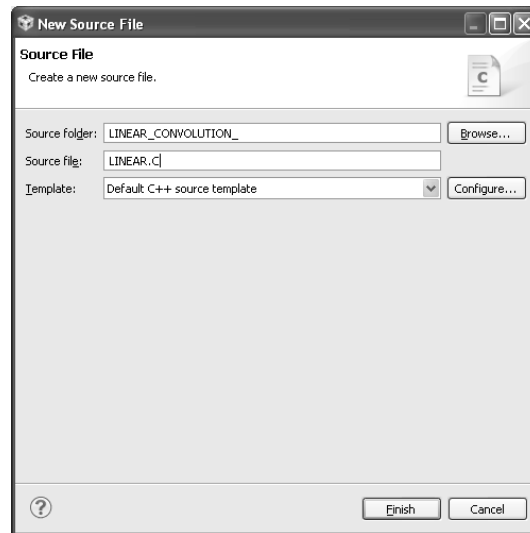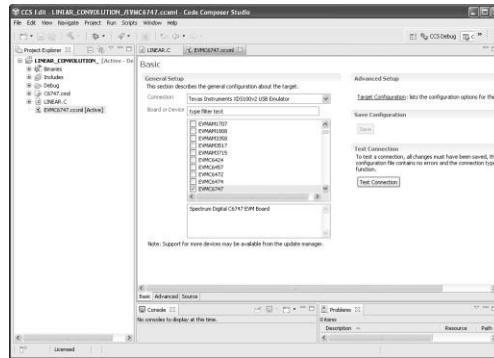
Click on 'Finish' button. It will create a new source file.
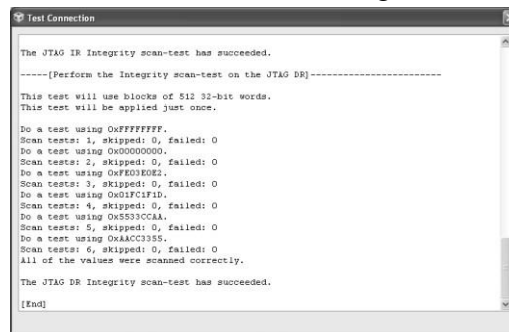

you can write your code and save it.
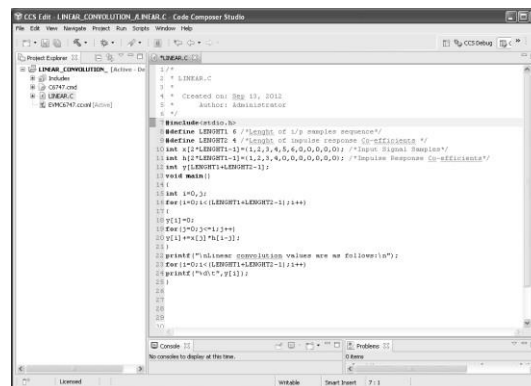
Check the connection with board, double click on EVMC6747.ccxml so following window appear.
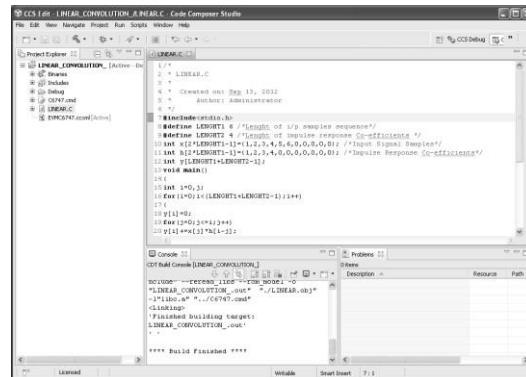
click on Test connection in above window, so testing window will appear as follows

The window appear with the message the connection is correct, than you can close
this window and proceed further

Build your program from Project>Build ALL. Following console window will
appear. It shows if any error present or not.

Debug the program. For that double click on Debug icon shown in above window or
press F11. So debug process starts as follows. double click on Debug icon shown in
above window or press F11 So debug process starts as follows.
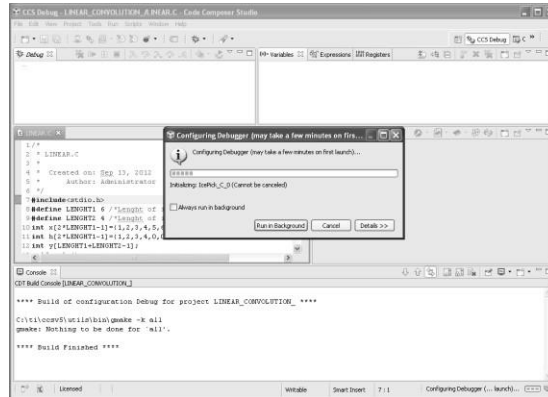
After that loading of program is completed that is shown in following in the window
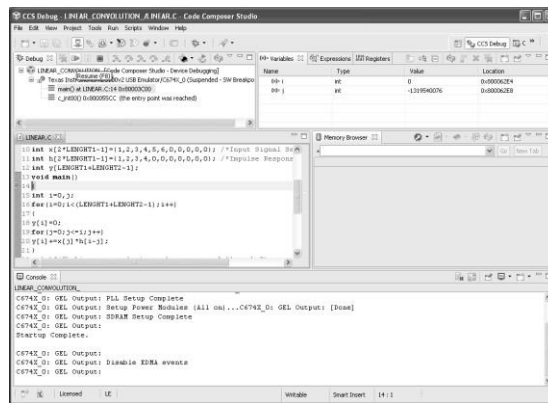
Run program by clicking on RUN
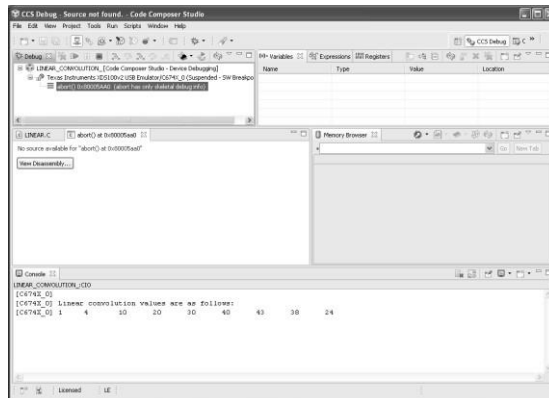
After completion of all program, close debug session by clicking on disconnect icon shown in above window. It will come to previous edit window. After completion of one program close project by, right click on project name and delete as shown below.
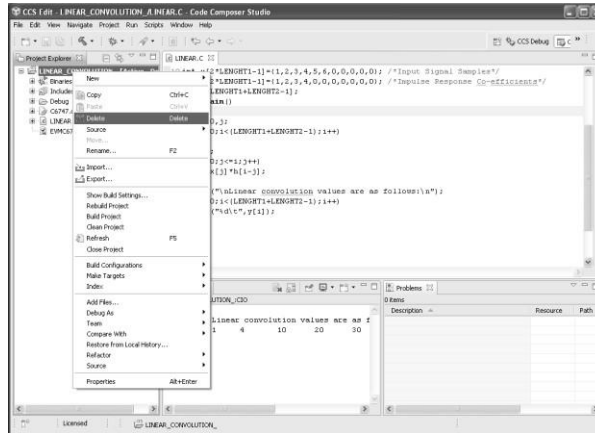
HARDWARE SETTING

Connect the DSP- 320 to Your PC

1. Connect the supplied USB cable to your PC or laptop.

2. If you plan to connect a microphone, speaker, function generator, DSO, or expansion card these must be plugged in properly before you connect power to the DSP320 board.

3. JTAG cable must be connected to PC and kit before power ON the DSP320 board. The required driver for the emulator is automatically installed by computer.

4. Connect the power supply to DSP320 and switch it ON.

Expt. No. 8. To perform Linear & Circular Convolution for the given sequences

AIM: To perform linear and circular convolution for the given sequences using DSK code composer studio.

APPARATUS:

1. TMS 320C6745 DSO 320 Kit
2. RS232 serial cable Power cord
3. Operating system: Windows XP
4. Software: ccs studio V3.1

PROGRAM:

Linear Convolution:

```
#include<stdio.h>
#define length1 6
#define length2 4
int x[2*length1-1]={1,2,3,4,5,6,0,0,0,0,0}
int h[2*length1-1]={1,2,3,4,0,0,0,0,0,0,0}
inty[length1+length2-1];
void main()
{
int i=0;j;
for(i=0;i<( length1+length2-1);i++)
{y[i]=0;
for(j=0;j<=I;j++)
y[i]+=x[j]*h[i-j];
}
printf("\nLinear convolution values are as follows:\n");
for(i=0;i< (length1+length2-1);i++) printf("%d\t",y[i]);
}
```

x[n]={1,2,3,4,5,6}

h[n]{1,2,3,4}

y[k]={1,4,10,20,30,40,43,38,24}

PROGRAM:

Circular Convolution:

```
#include<stdio.h>
Int m,n,x[30],h[30],i,j,temp[30],k,x2[30],a[30],y[30];
Void main ()
{
printf("\nEnter the length of the first sequence:\n");
scanf("%d",&m);
printf("\nEnter the length of the second sequence:\n");
scanf("%d",&n);
printf("\nEnter the first sequence:\n");
for(i=0;i<m;i++)
scanf("%d",&x[i]);
printf("\nEnter the second sequence:\n");
for(j=0;j<n;j++)
scanf("%d",&h[j]);
if(m-n!=0)
{if(m>n)
{
for(i=n;i<m;i++)
   h[i]=0;
   N=m;
   }
   For(i=m;i<n;i++)
   x[i]=0;
   m=n;
   }
```

```
y[0]=0
a[0]=h[0];
for(j=1;j<n;j++)
a[j]=h[n-j];
for(i=0;i<n;i++)
y[0]+=x[i]*a[i];
for(k=1;k<n;k++)
{
y[k]=0;
for(j=1;j<n;j++)
x2[j]=a[j-1];
x2[0]=a[n-1];
for(i=0;i<n;i++)
{
a[i]=x2[i];
y[k]+=x[i]*x2[i];
}
}
printf("The circular convolution is\n")
for(i=0;i<n;i++)
printf("%d\t",y[i]);
}
```

OUTPUT:

x[n]={2,1,2,1}, h[n]={1,2,3,4}

RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What is convolution?
5. Distinguish between linear and circular convolution?

Expt. No. 9. Design and implementation of FIR and IIR filter

AIM: To design FIR Low pass filter using Rectangular window, Triangular window, Kaiser window, FIR High pass filter using Rectangular window, Triangular window, Kaiser window,IIR Low pass filter using Butterworth filter & Chebyshev filters, IIR High pass filter using Butterworth filter & Chebyshev filters.

APPARATUS:

1. TMS 320C6745 DSO 320 Kit
2. RS232 serial cable Power cord
3. Operating system: Windows XP
4. Software: ccs studio V3.1
5. CRO

PROCEDURE:

Note down the code

Connect CRO to the **LINE OUT.**

Connect a Signal Generator to the **LINE IN.**

Switch on the Signal Generator with a sine wave of frequency 100 Hz. and

Vp-p=1.0v & vary the frequency.

 1. FOR LOW PASS FILTER:

 a. For FIR low pass rectangular window (cutoff 500Hz)

Open Code Composer Studio, make sure the DSP kit is turned on.

Import program using 'Project_import Existing ccs Eclipse project.

Which is saved in DVD at following location

PATH: DSP320_PROGRAMS\ FIR_LP_RECT_500Hz

Then debug and run the program

 b. For FIR low pass triangular window (cutoff 1000Hz)

Open Code Composer Studio, make sure the DSP kit is turned on.

Import program using 'Project_import Existing ccs Eclipse project.

Which is saved in DVD at following location

PATH: DSP320_PROGRAMS\ FIR_LP_TRIAN_1000Hz

Then debug and run the program

 c. For FIR low pass kaiser window (cutoff 1500Hz)

· Open Code Composer Studio, make sure the DSP kit is turned on.

· Import program using 'Project_import Existing ccs Eclipse

project. Which is saved in DVD at following location

PATH: DSP320_PROGRAMS\ FIR_LP_KAISER_1500Hz

Then debug and run the program

2. FOR HIGH PASS FILTER:

a. For FIR high pass rectangular window (cufoff 400Hz)

· Open Code Composer Studio, make sure the DSP kit is turned on.

· Import program using 'Project_import Existing ccs Eclipse

project. Which is saved in DVD at following location

PATH: DSP320_PROGRAMS\ FIR_ HP_RECT_400Hz

Then debug and run the program

b. For FIR high pass triangular window (cutoff 800Hz)

· Open Code Composer Studio, make sure the DSP kit is turned on.

· Import program using 'Project_import Existing ccs Eclipse

project. This is saved in DVD at following location

DSP320: TMS 320XXXX DSP Trainer V-12.0

- 72 - Embedded & DSP

PATH: DSP320_PROGRAMS\ FIR_ HP_TRIAN_800Hz

Then debug and run the program

c. For FIR high pass triangular window (cutoff 1200Hz)

· Open Code Composer Studio, make sure the DSP kit is turned on.

· Import program using 'Project_import Existing ccs Eclipse

project. Which is saved in DVD at following location

PATH: DSP320_PROGRAMS\ FIR_ HP_KAISER_1200Hz

Then debug and run the program

OUTPUT

· As we applied sine input through line in, the output will appear as per the filter type on

DSO.

· Output will decrease after the cutoff frequency for low pass filter

· Output will appear after cutoff frequency for high pass filter.

· Output will appear in between specified band of frequency for band pass filter.

PROCEDURE:

Note down the code :

For IIR filter design

· Connect CRO to **LINE OUT.**

· Connect a Signal Generator to the **LINE IN.**

DSP320 : TMS 320XXXX DSP Trainer V-12.0

- 78 - Embedded & DSP

· Switch on the Signal Generator with a sine wave of frequency 100 Hz. and

Vp-p=1.0V & vary frequency.

1. FOR IIR LOW PASS FILTER

a. For IIR low pass Butterworth filter (cutoff 800Hz)

· Open Code Composer Studio, make sure the DSP kit is turned on.

· Import program using 'Project_import Existing ccs Eclipse

project. Which is saved in DVD at following location

PATH: DSP320_PROGRAMS\ IIR_LP_BUTTER_800Hz.

Then debug and run the program

b. For IIR low pass chebyshev filter (cutoff 1000Hz)

· Open Code Composer Studio, make sure the DSP kit is turned on.

· Import program using 'Project_import Existing ccs Eclipse

project. Which is saved in DVD at following location

PATH: DSP320_PROGRAMS\ IIR_LP_CHEBY_1000Hz.

Then debug and run the program

2. FOR IIR HIGH PASS FILTER

a. For IIR high pass Butterworth filter (cutoff 2500Hz)

· Open Code Composer Studio; make sure the DSP kit is turned on.

· Import program using 'Project import Existing ccs Eclipse

project. This is saved in DVD at following location

PATH: DSP320_PROGRAMS\ IIR_HP_BUTTER_2500Hz.

Then debug and run the program

b. For IIR high pass chebyshev filter (cutoff 1000Hz)

· Open Code Composer Studio, make sure the DSP kit is turned on.

· Import program using 'Project_import Existing ccs Eclipse

project. This is saved in DVD at following location

PATH: DSP320_PROGRAMS\ IIR_HP_CHEBY_1000Hz.

Then debug and run the program

OUTPUT:

· Output will decreasing after the cutoff frequency for low pass filter.

· Output will appear at the cutoff frequency for high pass filter.

· Output will appear in between specified band.

RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What are various filters available in DSP?
5. What is the need for filters?
6. Distinguish between LPF and HPF?

Expt. No. 10. Computation Of DFT using DIT and DIF

AIM: To find DFT of the given sequence using DITFFT and DIFFFT for the given sequence using DSK code composer studio.

APPARATUS:

1. TMS 320C6745 DSO 320 Kit
2. RS232 serial cable Power cord
3. Operating system: Windows XP
4. Software: ccs studio V3.1

PROGRAM:

DIT-FFT:

```
#include<stdio.h>
//#include<conio.h>
#include<math.h>
#define PI 3.14
typedef struct
{
float real,imag;
}com;
void main()
{
com xx[8],x[8],temp[8],temp1[8],y[8],a[8],b[8],w[4];
int i,j=0;//loop counter variables
//printf("Enter the no of pionts of FFT==");
//scanf("%d", &PTS);
printf("\nEnter values==");
for(i=0;i<8;i++)
{
scanf("%f",&xx[i].real);
scanf("%f",&xx[i].imag);
```

```c
    }
    j=0;
    for(i=0;i<8;i=i+2)
    {
    x[j].real=xx[i].real;
    x[j].imag=xx[i].imag;
    x[j+1].real=xx[i+4].real;
    x[j+1].imag=xx[i+4].imag;
    if(i==2)
    i=-1;
    j=j+2;
    }
    for(i=0;i<4;i++)
    {
    w[i].real=cos(2*PI*i/8);
    w[i].imag=-sin(2*PI*i/8);
    }
    for(i=0;i<8;i=i+2)
    {
    temp[i].real=x[i].real+x[i+1].real;
    temp[i].imag=x[i].imag+x[i+1].imag;
    temp[i+1].real=x[i].real-x[i+1].real;
    temp[i+1].imag=x[i].imag-x[i+1].imag;
    }
    for(i=2;i<8;i=3*i)
    {
    a[i].real=temp[i].real*w[0].real-temp[i].imag*w[0].imag;
    a[i].imag=temp[i].real*w[0].imag+temp[i].imag*w[0].real;
    a[i+1].real=temp[i+1].real*w[2].real-temp[i+1].imag*w[2].imag;
    a[i+1].imag=temp[i+1].real*w[2].imag+temp[i+1].imag*w[2].real;
    temp[i].real=a[i].real;
    temp[i].imag=a[i].imag;
```

```c
temp[i+1].real=a[i+1].real;

temp[i+1].imag=a[i+1].imag;

}
for(i=0;i<6;i++)

{

temp1[i].real=temp[i].real+temp[i+2].real;

temp1[i].imag=temp[i].imag+temp[i+2].imag;

temp1[i+2].real=temp[i].real-temp[i+2].real;

temp1[i+2].imag=temp[i].imag-temp[i+2].imag;

if(i==1)

i=3;

}

for(i=4;i<8;i++)

{

b[i].real=temp1[i].real*w[i-4].real-temp1[i].imag*w[i-4].imag;

b[i].imag=temp1[i].real*w[i-4].imag+temp1[i].imag*w[i-4].real;

temp1[i].real=b[i].real;

temp1[i].imag=b[i].imag;

}

for(i=0;i<4;i++)

{

y[i].real=temp1[i].real+temp1[i+4].real;

y[i].imag=temp1[i].imag+temp1[i+4].imag;

y[i+4].real=temp1[i].real-temp1[i+4].real;

y[i+4].imag=temp1[i].imag-temp1[i+4].imag;

}

printf("\nDFT values==\n");

for(i=0;i<8;i++)

{

printf("\nF(%d)=(%0.1f)+j(%0.1f)\n",i,y[i].real,y[i].imag);

}

}
```

DIF-FFT:

```c
#include<stdio.h>
#include<math.h>
#define PI 3.14
typedef struct
{
float real,imag;
}com;
void main()
{
com x[8],y[8],w[4],temp,temp1[8],temp2[8],temp3[8];
int i,j=0;
printf("\nEnter the values of x(n)\n");
for(i=0;i<8;i++)
scanf("%f%f",&x[i].real,&x[i].imag);
for(i=0;i<4;i++)
{
w[i].real=cos(2*PI*i/8);
w[i].imag=-sin(2*PI*i/8);
}
for(i=0;i<4;i++)
{
temp1[i].real=x[i].real+x[i+4].real;
temp1[i].imag=x[i].imag+x[i+4].imag;
temp.real=x[i].real-x[i+4].real;
temp.imag=x[i].imag-x[i+4].imag;
temp1[i+4].real=w[i].real*temp.real-w[i].imag*temp.imag;
temp1[i+4].imag=w[i].real*temp.imag+w[i].imag*temp.real;
}
for(i=0;i<6;i++)
{
```

```c
temp2[i].real=temp1[i].real+temp1[i+2].real;
temp2[i].imag=temp1[i].imag+temp1[i+2].imag;


temp.real=temp1[i].real-temp1[i+2].real;
temp.imag=temp1[i].imag-temp1[i+2].imag;
temp2[i+2].real=w[j].real*temp.real-w[j].imag*temp.imag;
temp2[i+2].imag=w[j].real*temp.imag+w[j].imag*temp.real;
if(j==2)
j=-2;
j=j+2;
if(i==1)
i=3; }
for(i=0;i<8;i=i+2)
{ temp3[i].real=temp2[i].real+temp2[i+1].real;
temp3[i].imag=temp2[i].imag+temp2[i+1].imag;
temp3[i+1].real=temp2[i].real-temp2[i+1].real;
temp3[i+1].imag=temp2[i].imag-temp2[i+1].imag;
}
printf("\n\nDFT Values are===\n");
j=0;
for(i=0;i<8;i=i+2)
{ y[j].real=temp3[i].real;
y[j].imag=temp3[i].imag;
y[j+1].real=temp3[i+4].real;
y[j+1].imag=temp3[i+4].imag;
if(i==2)
i=-1;
j=j+2; }
for(i=0;i<8;i++)
{
printf("\nF(%d)=(%0.1f)+j(%0.1f)\n",i,y[i].real,y[i].imag);
```

}

}

RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What is DFT?
5. Why to perform DFT and IDFT on sequences?
6. What is the advantage of FFT?

Expt. No. 11. Design of LPF using Blackman Window

AIM: Write a MATLAB program to verify the FIR LP filter using Blackman window.

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

PROGRAM:

```
% FIR LP filter design using Blackman windows


clc;
close all;
clear all;
rp = input('Enter the Pass Band Ripple: ');
rs = input('Enter the Stop Band Ripple: ');
fp = input('Enter the Pass Band Frequency: ');
fs = input('Enter the Stop Band Frequency: ');
f = input('Enter the Sampling Frequency: ');
wp = 2 * fp /f;
ws = 2 * fa/f;
num = - 20 * log10( sqrt(rp*rs))- 13;
den = 14.6 * (fs-fp)/f;
n = ceil (nurrr/ den);
n1 = n+l;

if (rem(n,2)~=0);

n1 = n;
n = n-1;
end

y = blackman (n1);

% LOW PASS FILTER
b = fir1(n,wp,y);
[h,o] = freqz(b,1,256);
m = 20 * log10(abs(h));
subplot(2,2,1) ;
plot (o/pi,m) ;
title(' ***** BLACKMAN WINDOW *****');
ylabel('Gain indb------>');
xlabel('(a) Normalised Frequency------->');
```
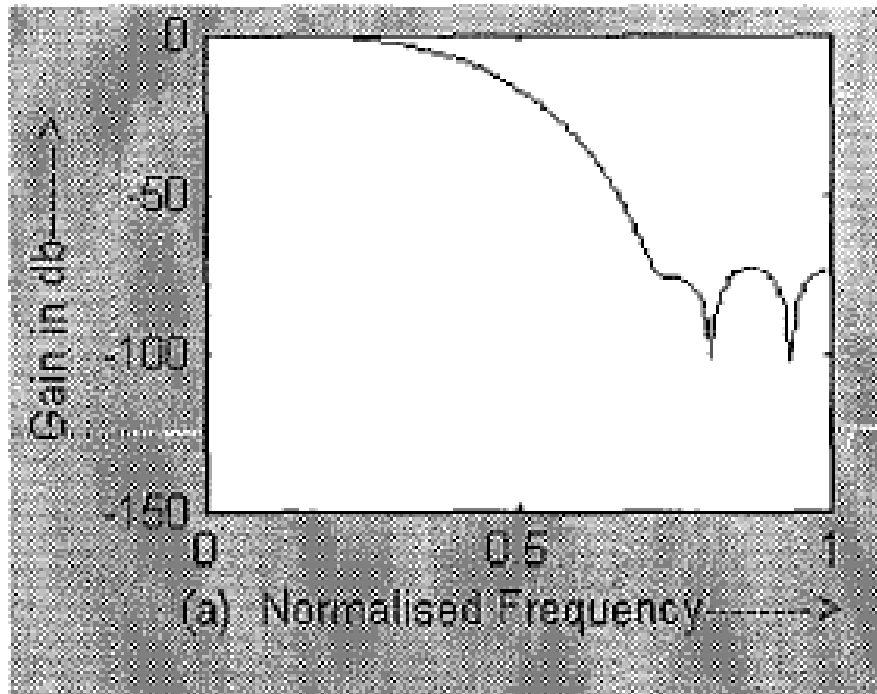
OUTPUT:

Enter the Pass Band Ripple: 0.05
Enter the Stop Band Ripple: 0.04
Enter the Pass Band Frequency: 1500
Enter the Stop Band Frequency: 2000
Enter the Sampling Frequency: 8000

(a) Normalised Frequency----->

RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What are various filters available in DSP?
5. What is the need for filters?
6. Distinguish between LPF and HPF?

Expt. No. 12. Design of HPF using Hamming Window

AIM: Write a MATLAB program to verify the FIR HP filter using Hamming window.

APPARATUS:

1. PC with Windows OS.
2. MATLAB tool.

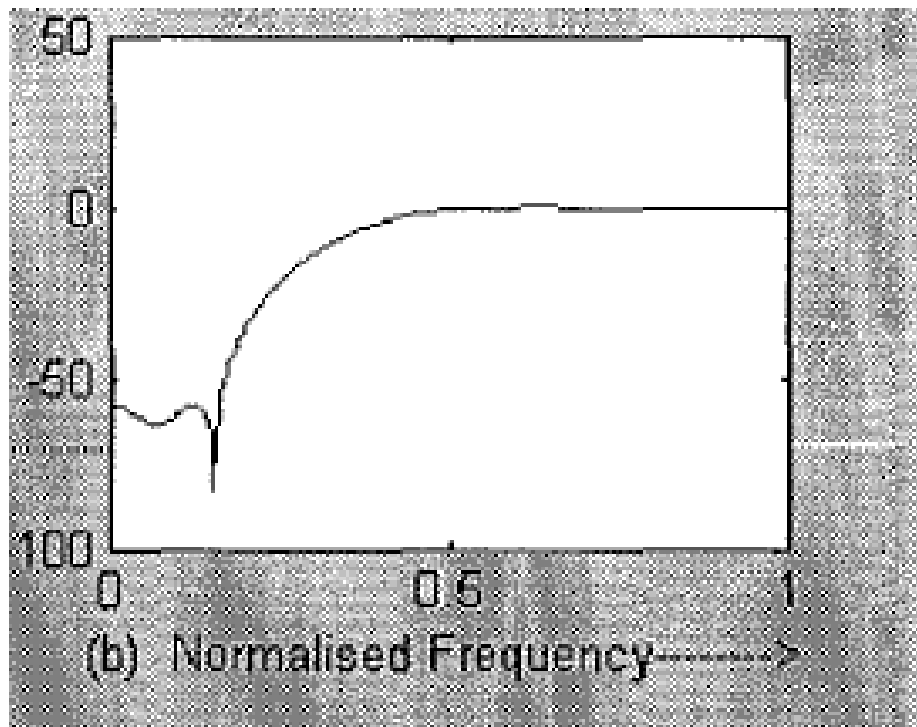**PROGRAM:**

```
% FIR HP filter design using Hamming window

clc;
close all;
clear all;
rp = input('Enter the Pass Band Ripple: ');
rs = input('Enter the Stop Band Ripple: ');
fp = input('Enter the Pass Band Frequency: ');
fs = input('Enter the Stop Band Frequency: ');
f = input('Enter the Sampling Frequency: ');
wp = 2 * fpjf;
ws = 2 * fsjf;
num = - 20 * log10( sqrt(rp*rs))- 13;
den = 14.6 * (fs-fp)jf;
n = ceil (numjden);
n1 = n+1;
if (rem(n,2)-=0) ;
n1 = n;
n = n-1;
end
y = hamming (n1);

% HIGH PASS FILTER
b = firl(n,wp,'high',y);
[h,o] = freqz(b,1,256);
m = 20*log10(abs(h));
subplot(2,2,2) .plotto Zpi.m] ;
ylabel('Gain in db------>');

xlabel(' (b) Normalised Frequency------>'];
```

OUTPUT:

Enter the Pass Band Ripple: 0.05
Enter the Stop Band Ripple: 0.04
Enter the Pass Band Frequency: 1500
Enter the Stop Band Frequency: 2000
Enter the Sampling Frequency: 8000

(b) Normalised Frequency--------->

RESULT:

VIVA QUESTIONS:

1. What does MATLAB stand for?
2. What is the importance of MATLAB tool?
3. What is the difference between continuous signal and discrete signal?
4. What are various filters available in DSP?
5. What is the need for filters?
6. Distinguish between LPF and HPF?