DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

COURSE MATERIALS

ECL 333:DIGITAL SIGNAL PROCESSING LABORATORY

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- ♦ Established in: 2002
- ♦ Course offered : B.Tech in Electronics and Communication Engineering

  M.Tech in VLSI
- ♦ Approved by AICTE New Delhi and Accredited by NAAC
- ♦ Affiliated to the University of Dr. A P J Abdul Kalam Technological University.

DEPARTMENT VISION

Providing Universal Communicative Electronics Engineers with corporate and social relevance towards sustainable developments through quality education.

DEPARTMENT MISSION

1) Imparting Quality education by providing excellent teaching, learning environment.

2) Transforming and adopting students in this knowledgeable era, where the electronic gadgets (things) are getting obsolete in short span.

3) To initiate multi-disciplinary activities to students at earliest and apply in their respective fields of interest later.

4) Promoting leading edge Research & Development through collaboration with academia & industry.

PROGRAMME EDUCATIONAL OBJECTIVES

PEOI. To prepare students to excel in postgraduate programmes or to succeed in industry / technical profession through global, rigorous education and prepare the students to practice and innovate recent fields in the specified program/ industry environment.

PEO2. To provide students with a solid foundation in mathematical, Scientific and engineering fundamentals required to solve engineering problems and to have strong practical knowledge required to design and test the system.

PEO3. To train students with good scientific and engineering breadth so as to comprehend, analyze, design, and create novel products and solutions for the real life problems.

PEO4. To provide student with an academic environment aware of excellence, effective communication skills, leadership, multidisciplinary approach, written ethical codes and the life-long learning needed for a successful professional career.

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

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

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality
System Software Tools and Efficient Web Design Models with a focus on performance optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software
products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

.

ECL333	DIGITAL SIGNAL PROCESSING LABORATORY	CATEGORY	L	T	P	CREDIT
		PCC	0	0	3	2

Preamble:

- The following experiments are designed to make the student do real time DSP computing.
- Dedicated DSP hardware (such as TI or Analog Devices development/evaluation boards) will be used for realization.

Prerequisites:

- ECT 303 Digital Signal Processing
- EST 102 Programming in C

Course Outcomes: The student will be able to

CO 1	Simulate digital signals.
CO 2	verify the properties of DFT computationally
CO 3	Familiarize the DSP hardware and interface with computer.
CO 4	Implement LTI systems with linear convolution.
CO 5	Implement FFT and IFFT and use it on real time signals.
CO 6	Implement FIR low pass filter.
CO 7	Implement real time LTI systems with block convolution and FFT.

Mapping of Course Outcomes with Program Outcomes

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12
CO1	3	3	1	2	3	0	0	0	3	0	0	1
CO2	3	3	1	2	3	0	0	0	3	0	0	1
CO3	3	3	3	2	3	0	0	0	3	1	0	1
CO4	3	3	1	2	3	0	0	0	3	0	0	1
CO5	3	3	1	1	3	0	0	0	0	0	0	1
CO6	3	3	1	1	3	0	0	0	0	0	0	1
CO7	3	3	1	3	3	0	0	0	3	3	0	0

Assessment Pattern

Mark Distribution:

Total Mark	CIE	ESE
150	50	100

Continuous Internal Evaluation Pattern:

Each experiment will be evaluated out of 50 credits continuously as

Attribute	Mark
Attendance	15
Continuous assessment	30
Internal Test (Immediately before the second series test)	30

End Semester Examination Pattern: The following guidelines should be followed regarding award of marks

Attribute	Mark
Preliminary work	15
Implementing the work/ Conducting the experiment	10
Performance, result and inference (usage of equipments and trouble shooting)	25
Viva voce	20
Record	5

Course Level Assessment Questions

CO1-Simulation of Signals

1. Write a Python/MATLAB/SCILAB function to generate a rectangular pulse.

2. Write a Python/MATLAB/SCILAB function to generate a triangular pulse.

CO2-Verfication of the Properties of DFT

1. Write a Python/MATLAB/SCILAB function to compute the N -point DFT

matrix and plot its real and imaginary parts.

2. Write a Python/MATLAB/SCILAB function to verify Parseval’s theorem for $N = 1024$.

CO3-Familarization of DSP Hardware

1. Write a C function to control the output LEDs with input switches.
2. Write a C function to connect the analog input port to the output port and test with a microphone.

CO4-LTI System with Linear Convolution

1. Write a function to compute the linear convolution and download to the hardware target and test with some signals.

CO5-FFT Computation

1. Write and download a function to compute N point FFT to the DSP hardware target and test it on real time signal.
2. Write a C function to compute IFFT with FFT function and test in on DSP hardware.

CO6-Implementation of FIR Filter

1. Design and implement an FIR low pass filter for a cut off frequency of 0.1π and test it with an AF signal generator.

CO7-LTI Systems by Block Convolution

1. Implement an overlap add block convolution for speech signals on DSP target.

List of Experiments

(All experiments are mandatory.)

Experiment 1. Simulation of Signals Simulate the following signals using Python/ Scilab/MATLAB.

1. Unit impulse signal
2. Unit pulse signal
3. Unit ramp signal
4. Bipolar pulse
5. Triangular signal

Experiment 2. Verification of the Properties of DFT

- Generate and appreciate a DFT matrix.
    1. Write a function that returns the N point DFT matrix $\mathbf{V_N}$ for a given N.
    2. Plot its real and imaginary parts of $\mathbf{V_N}$ as images using *matshow* or *imshow* commands (in Python) for $N = 16$, $N = 64$ and $N = 1024$
    3. Compute the DFTs of 16 point, 64 point and 1024 point random sequences using the above matrices.
    4. Observe the time of computations for $N = 2^\gamma$ for $2 \leq \gamma \leq 18$ (You may use the *time* module in Python).
    5. Use some iterations to plot the times of computation against γ. Plot and understand this curve. Plot the times of computation for the *fft* function over this curve and appreciate the computational saving with FFT.
- Circular Convolution.
    1. Write a python function *circcon.py* that returns the circular convoluton of an N_1 point sequence and an N_2 point sequence given at the input. The easiest way is to convert a linear convolution into circular convolution with $N = max(N_1, N_2)$.
- Parseval's Theorem

  For the complex random sequences $x_1[n]$ and $x_2[n]$,

$$\sum_{n=0}^{N-1} x_1[n]x_2^*[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_1[k]X_2^*[k]$$

1. Generate two random complex sequences of say 5000 values.
2. Prove the theorem for these signals.

Experiment 3. Familarization of DSP Hardware

1. Familiarization of the code composer studio (in the case of TI hard- ware) or Visual DSP (in the case of Analog Devices hardware) or any equivalent cross compiler for DSP programming.
2. Familiarization of the analog and digital input and output ports of the DSP board.
3. Generation and cross compilation and execution of the C code to con- nect the input digital switches to the output LEDs.
4. Generation and cross compilation and execution of the C code to con- nect the input analog port to the output. Connect a microphone, speak into it and observe the output electrical signal on a DSO and store it.
5. Document the work.

Experiment 4. Linear convolution

1. Write a C function for the linear convolution of two arrays.
2. The arrays may be kept in different files and downloaded to the DSP hardware.
3. Store the result as a file and observe the output.
4. Document the work.

Experiment 5. FFT of signals

1. Write a C function for N - point FFT.
2. Connect a precision signal generator and apply 1 mV , 1 kHz sinusoid at the analog port.
3. Apply the FFT on the input signal with appropriate window size and observe the result.
4. Connect microphone to the analog port and read in real time speech.
5. Observe and store the FFT values.
6. Document the work.

Experiment 6. IFFT with FFT

1. Use the FFT function in the previous experiment to compute the IFFT of the input signal.

2. Apply IFFT on the stored FFT values from the previous experiments and observe the reconstruction.

3. Document the work.

Experiment 7. FIR low pass filter

1. Use Python/scilab to implement the FIR filter response $h[n] = \frac{sin(\omega cn)}{\pi n}$ for a filter size $N = 50$, $\omega_c = 0.1\pi$ and $\omega_c = 0.3\pi$.

2. Realize the hamming($w_H\ [n]$) and kaiser ($w_K[n]$) windows.

3. Compute $h[n]w[n]$ in both cases and store as file.

4. Observe the low pass response in the simulator.

5. Download the filter on to the DSP target board and test with 1 mV sinusoid from a signal generator connected to the analog port.

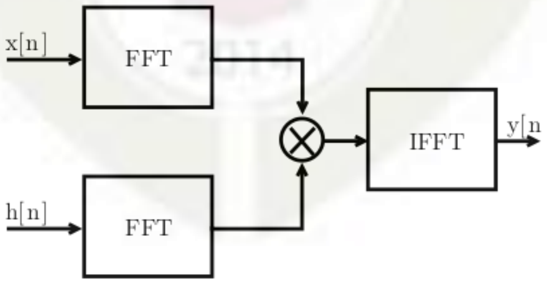
6. Test the operation of the filters with speech signals.

7. Document the work.

Experiment 8. Overlap Save Block Convolution

1. Use the file of filter coefficients From the previos experiment.

2. Realize the system shown below for the input speech signal $x[n]$.



3. Segment the signal values into blocks of length $N = 2000$. Pad the last

block with zeros, if necessary.

4. Implement the *overlap save* block convolution method

5. Document the work.

Experiment 9. Overlap Add Block Convolution

1. Use the file of filter coefficients from the previous experiment.

2. Realize the system shown in the previous experiment for the input speech signal $x[n]$.

3. Segment the signal values into blocks of length $N = 2000$. Pad the last block with zeros, if necessary.

4. Implement the *overlap add* block convolution method

5. Document the work.

Schedule of Experiments: Every experiment should be completed in three hours.

Textbooks

1. Vinay K. Ingle, John G. Proakis, “Digital Signal Processing Using MATLAB.”

2. Allen B. Downey, “Think DSP: Digital Signal Processing using Python.”

3. Rulph Chassaing, “DSP Applications Using C and the TMS320C6x DSK (Topics in Digital Signal Processing)”

TABLE OF CONTENTS

EXP.No.	Topic	Page No.
0	FAMILIARISATION WITH MATLAB	2
1	GENERATION OF WAVEFORMS (CONTINUOUS AND DISCRETE)	5
2	LINEAR CONVOLUTION, CIRCULAR CONVOLUTION AND LINEAR CONVOLUTION USING CIRCULAR CONVOLUTION.	9
3	TO FIND THE DFT AND IDFT FOR THE GIVEN INPUT SEQUENCE.	15
4	TO FIND FFT AND IFFT FOR THE GIVEN INPUT SEQUENCE.	18
5	GENERATION OF AM, FM & PWM WAVEFORMS AND THEIR SPECTRUM.	21
6	FIR FILTER (LOW-PASS, HIGH-PASS AND BAND-PASS)DESIGN (WINDOW METHOD).	26
7	IIR FILTER (LOW-PASS, HIGH-PASS AND BAND-PASS)DESIGN (BUTTERWORTH AND CHEBYCHEV).	30
	Experiments on Digital Signal Processor/ DSP kits	43
8	GENERATION OF SINE WAVE AND STANDARD TEST SIGNALS.	47
9	CONVOLUTION : LINEAR AND CIRCULAR	54
10	REAL TIME FIR FILTER IMPLEMENTATION (LOW-PASS, HIGH-PASS AND BAND-PASS) BY INPUTTING A SIGNAL FROM THE SIGNAL GENERATOR	61
11	REAL TIME IIR FILTER IMPLEMENTATION (LOW-PASS, HIGH-PASS AND BAND-PASS) BY INPUTTING A SIGNAL FROM THE SIGNAL GENERATOR	64
12	SAMPLING OF ANALOG SIGNAL AND STUDY OF ALIASING.	67

FAMILIARISATION WITH MATLAB

Aim: To familiarize with MATLAB software, general functions and signal processing toolbox functions.

The name MATLAB stands for MATrix LABoratory produced by Math works Inc., USA. It is a matrix-based powerful software package for scientific and engineering computation and visualization. Complex numerical problems can be solved in a fraction of the time that required with other high level languages. It provides an interactive environment with hundreds of built -in –functions for technical computation, graphics and animation. In addition to built-in-functions, user can create his own functions. MATLAB offers several optional toolboxes, such as signal processing, control systems, neural networks etc. It is command driven software and has online help facility.

MATLAB has three basic windows normally; command window, graphics window and edit window.

Command window is characterized by the prompt '>>'.All commands and the ready to run program filename can be typed here. Graphic window gives the display of the figures as the result of the program. Edit window is to create program files with an extension .m.

Some important commands in MATLAB

Help	**List topics on which help is available**
Help command name	**Provides help on the topic selected**
Demo	**runs the demo program**
Who	**Lists variables currently in the workspace**
Whos	**Lists variables currently in the workspace with their size**
Clear	**clears the workspace, all the variables are removed**
Clear x,y,z	**Clears only variables x,y,z**
Quit	**Quits MATLAB**

Some of the frequently used built-in-functions in Signal Processing Toolbox

filter(b.a.x) Syntax of this function is Y = filter(b.a.x) It filters the data in vector x with the filter described by vectors a and b to create the filtered data y.

fft (x) It is the DFT of vector x

ifft (x) It is the DFT of vector x

conv (a,b) Syntax of this function is C = conv (a,b) It convolves vectors a and b. The resulting vector is of Length, Length (a) + Length (b)-1

deconv(b,a) Syntax of this function is [q,r] = deconv(b,a) It deconvolves vector q and the remainder in vector r such that
b = conv(a,q)+r

butter(N,Wn) designs an Nth order lowpass digital
Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and a (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

buttord(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Butterworth filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, Normalized from 0 to 1, (where 1 corresponds to pi rad/sec)

Cheby1(N,R,Wn) designs an Nth order lowpass digital Chebyshev filter with R decibels of peak-to-peak ripple in the passband. CHEBY1 returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

Cheby1(N,R,Wn,'high') designs a highpass filter.

Cheb1ord(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Chebyshev Type I filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample)

cheby2(N,R,Wn) designs an Nth order lowpass digital Chebyshev filter with the stopband ripple R decibels down andstopband edge frequency Wn. CHEBY2 returns the filter coefficients in length N+1 vectors B (numerator) and A . The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

cheb2ord(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Chebyshev Type II filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies.

abs(x) It gives the absolute value of the elements of x. When x is complex, abs(x) is the complex modulus (magnitude) of the elements of x.

angle(H) It returns the phase angles of a matrix with complex elements in radians.

freqz(b,a,N) Syntax of this function is [h,w] = freqz(b,a,N) returns the Npoint frequency vector w in radians and the N-point complex frequency response vector h of the filter b/a.

stem(y) It plots the data sequence y aa stems from the x axis terminated with circles for the data value.

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

plot(x,y) It 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.

title(‘text’) It adds text at the top of the current axis.

xlabel(‘text’) It adds text beside the x-axis on the current axis.

ylabel(‘text’) It adds text beside the y-axis on the current axis

GENERATION OF WAVEFORMS

Experiment No: - 01

AIM: - To write a MATLAB program to common continues and discrete time signals

PROCEDURE:-

□ Open MATLAB
□ Open new M-file
□ Type the program
□ Save in current directory
□ Compile and Run the program
□ for the output see command window\ Figure window

ALGORITHM:-

□ Get the amplitude and frequency of the signal
□ Use „sin‟, ‟cos ,‟square‟ matlab built in functions
□ Using „plot‟ function plot the signal
□ Using „stem‟ function plot the signal

MATLAB CODE:-

Continous Time Signals

```
clc;
clear all;
close all;
t=0:0.001:1;
f=input('Enter the value of frequency');
a=input('Enter the value of amplitude');
subplot(3,3,1);
y=a*sin(2*pi*f*t); %sine wave
plot(t,y,);
xlabel('time');
ylabel('amplitude');
title('sine wave')
grid on;
subplot(3,3,2);
z=a*cos(2*pi*f*t); %cos wave
plot(t,z);
xlabel('time');
ylabel('amplitude');
title('cosine wave')
grid on;
subplot(3,3,3);
s=a*square(2*pi*f*t); %square wave
plot(t,s);
xlabel('time');
ylabel('amplitude');
```

```
title('square wave')
grid  on;
subplot(3,3,4);
plot(t,t); %ramp signal
xlabel('time');
ylabel('amplitude');
title('ramp wave')
grid on;
subplot(3,3,5);
plot(t,a,'r'); %unit step wave
xlabel('time');
ylabel('amplitude');
title('unit step wave')
grid on;
x=a*exp(-t);
subplot(3,3,6);
plot(t,x); %exponential wave
xlabel('Time');
ylabel('Amplitude');
title('Exponentially Decaying Signal');
grid on;
```

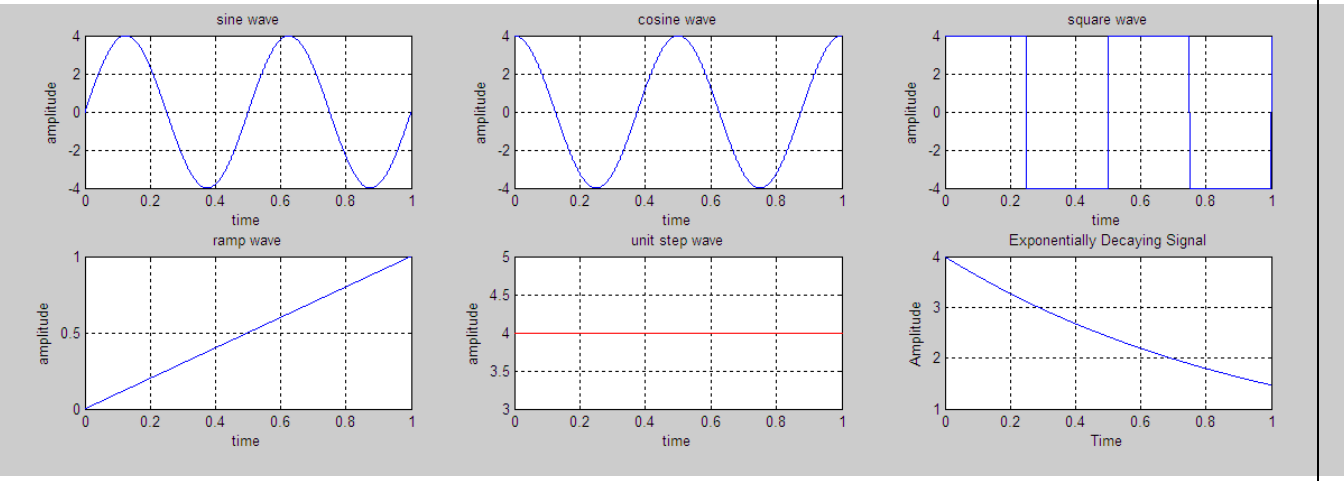
Discrete Time Signal

```
clc;
clear all;
close all;
n=0:1:50;
f=input('Enter the value of frequency');
a=input('Enter the value of amplitude');
N=input('Enter the length of unit step');
subplot(3,3,1);
y=a*sin(2*pi*f*n);
stem(n,y,'r');
xlabel('time');
ylabel('amplitude');
title('sine wave')
grid on;
subplot(3,3,2);
z=a*cos(2*pi*f*n);
stem(n,z);
xlabel('time');
ylabel('amplitude');
title('cosine wave')
grid on;
subplot(3,3,3);
s=a*square(2*pi*f*n);
stem(n,s);
xlabel('time');
ylabel('amplitude');
title('square wave')
grid on;
subplot(3,3,4);
stem(n,n);
```

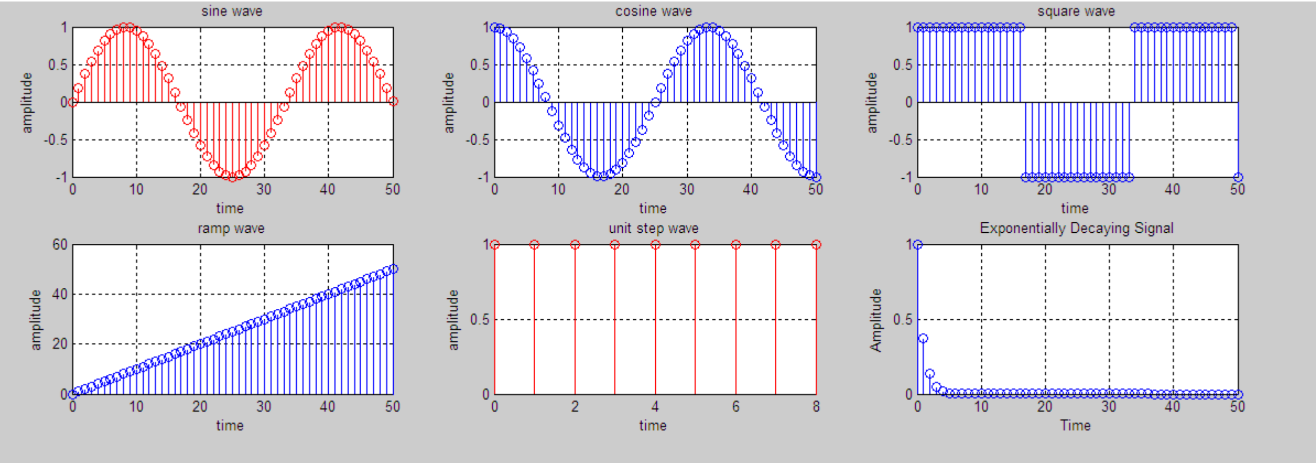
```
xlabel('time');
ylabel('amplitude');
title('ramp wave')
grid on;
x=0:N-1;
d=ones(1,N);
subplot(3,3,5);
stem(x,d,'r');
xlabel('time');
ylabel('amplitude');
title('unit step wave')
grid on;
x=a*exp(-t);
subplot(3,3,6);
stem(t,x);
xlabel('Time');
ylabel('Amplitude');
title('Exponentially Decaying Signal');
grid on;
```

FIGURE:-

Continuous Time Signal



Discrete Time Signal



SAMPLE INPUT:-

Continuous Time Signal

```
Enter the value of frequency2
Enter the value of amplitude1
```

Discrete Time Signal

```
Enter the value of frequency 0.03
Enter the value of amplitude 1
Enter the length of unit step 9
```

RESULTS: - Thus the generation of continuous and discrete time signals using matlab was verified.

GENERATION OF LINEAR CONVOLUTION, CIRCULAR CONVOLUTION, LINEAR CONVOLUTION USING CIRCULAR CONVOLUTION

Experiment No: - 02

AIM: - To write a MATLAB program to compute linear convolution, Circular Convolution and linear convolution using circular convolution of two given sequences

PROCEDURE:-

□ Open MATLAB
□ Open new M-file
□ Type the program
□ Save in current directory
□ Compile and Run the program
□ For the output see command window\ Figure window

a) Linear convolution using function

ALGORITHM:-

□ Read the input sequence x[n] ,and plot
□ Read the impulse sequence h[n] , and plot
□ Use the matlab function „conv‟or dft or loop
□ Convolve the two sequence and plot the result

MATLAB CODE:-

```
clc;
x=input('Enter the sequence 1:');
h=input('Enter the sequence 2:');
y=conv(x,h);
subplot(3,1,1);
stem(x);
ylabel('Amplitude->');
xlabel('N');
subplot(3,1,2);
stem(h);
ylabel('Amplitude->');
xlabel('N');
subplot(3,1,3);
stem(y);
ylabel('Amplitude->');
xlabel('N');
```

b)Linear convolution using DFT

MATLAB CODE:-

```
clc;
n1=input('enter the length of sequence');
x=input('Enter the sequence 1:');
n2=input('enter the length of sequence');
h=input('Enter the sequence 2:');
x=[x,zeros(1,n2-1)];
h=[h,zeros(1,n1-1)];
X=fft(x);
H=fft(h);
Y=X.*H;
y=ifft(Y);
subplot(311);
stem(x);
subplot(312);
stem(h);
title('using DFT');
subplot(313);
stem(y)
```

c)Linear convolution without using Function

MATLAB CODE:-

```
close all
clear all
x=input('Enter x: ')
h=input('Enter h: ')
m=length(x);
n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)];
for i=1:n+m-1
Y(i)=0;
for j=1:m
if(i-j+1>0)
Y(i)=Y(i)+X(j)*H(i-j+1);
else
end
end
end
Y
stem(Y);
ylabel('Y[n]');
xlabel('---- >n');
title('Convolution of Two Signals without conv function');
```

ALGORITHM:- Circular Convolution

□ Read the input sequence x1[n], and plot
□ Read the input sequence x2[n], and plot
□ Use the user defined matlab function „crconc"
□ convolve the two sequences and plot the result

a) Circular convolution using DFT

MATLAB CODE:-

```
clc;
n1=input('enter the length of sequence');
x=input('Enter the sequence 1:');
h=input('Enter the sequence 2:');
X=fft(x);
H=fft(h);
Y=X.*H;
y=ifft(Y);
subplot(311);
stem(x);
title('sequence1')
subplot(312);
stem(h);
title('sequence2');
subplot(313);
stem(y);
title('using dft');
```

b) Circular convolution without using Function

MATLAB CODE:-

```
clc;
x=input('Enter the sequence 1:');
h=input('Enter the sequence 2:');
n1=length(x);
n2=length(h);
N=max(n1,n2);
x=[x,zeros(1,N-n1)];
h=[h,zeros(1,N-n2)];
for n=0:N-1
y(n+1)=0;
for i=0:N-1
   j=mod(n-i,N);
   y(n+1)=y(n+1)+x(i+1)*h(j+1);
end
end
display(y)
subplot(131);
stem(x);
title('firstsequence');
subplot(132);
```

```
stem(h);
title('second sequence');
subplot(133);
stem(y);
title('circular convolution');
```

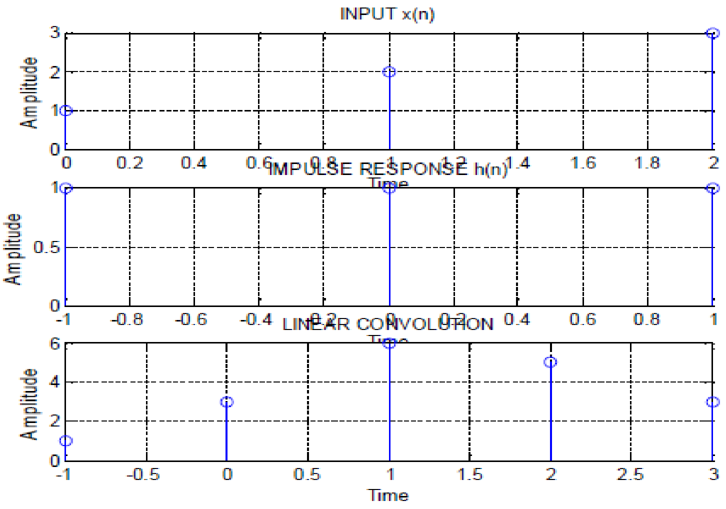
ALGORITHM:- Linear Convolution using Circular convolution

- □ Read the input sequence x1[n], and plot
- □ Read the input sequence x2[n], and plot
- □ Use the user defined matlab functions
- □ convolve the two sequences and plot the result

MATLAB CODE:-

```
clear all;close all;
x1=input('Enter the sequence 1: ');
x2=input('Enter the sequence 2: ');
subplot(311);
stem(x1);
title('Input sequence1');
subplot(312);
stem(x2);
title('Input sequence2');
N1=numel(x1);
N2=numel(x2);
x1=[x1 zeros(1,N2-1)];
x2=[x2 zeros(1,N1-1)];
y=cconv(x1,x2);
subplot(313);
stem(y);
title('Lin Using Circular');
```

FIGURE: Linear Convolution

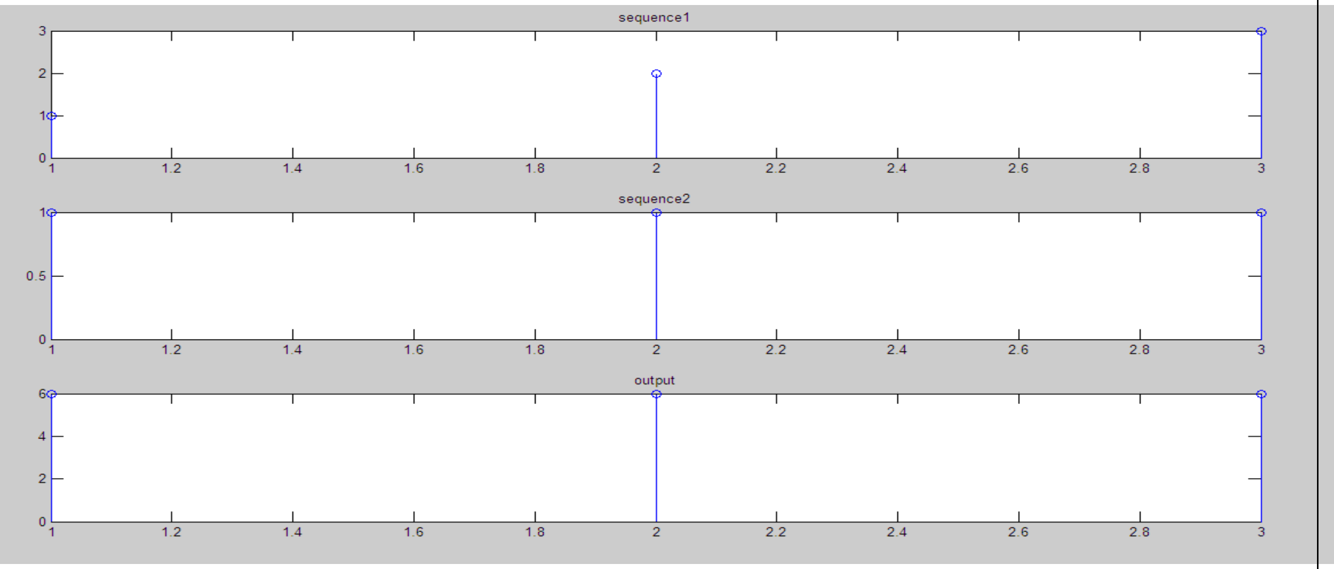


SAMPLE INPUT:-

Enter the co-efficient of x(n)=[1 2 3]
Enter the co-efficient of h(n)=[1 1 1]

```
y= 1    3         6        5          3
```

FIGURE: Circular Convolution



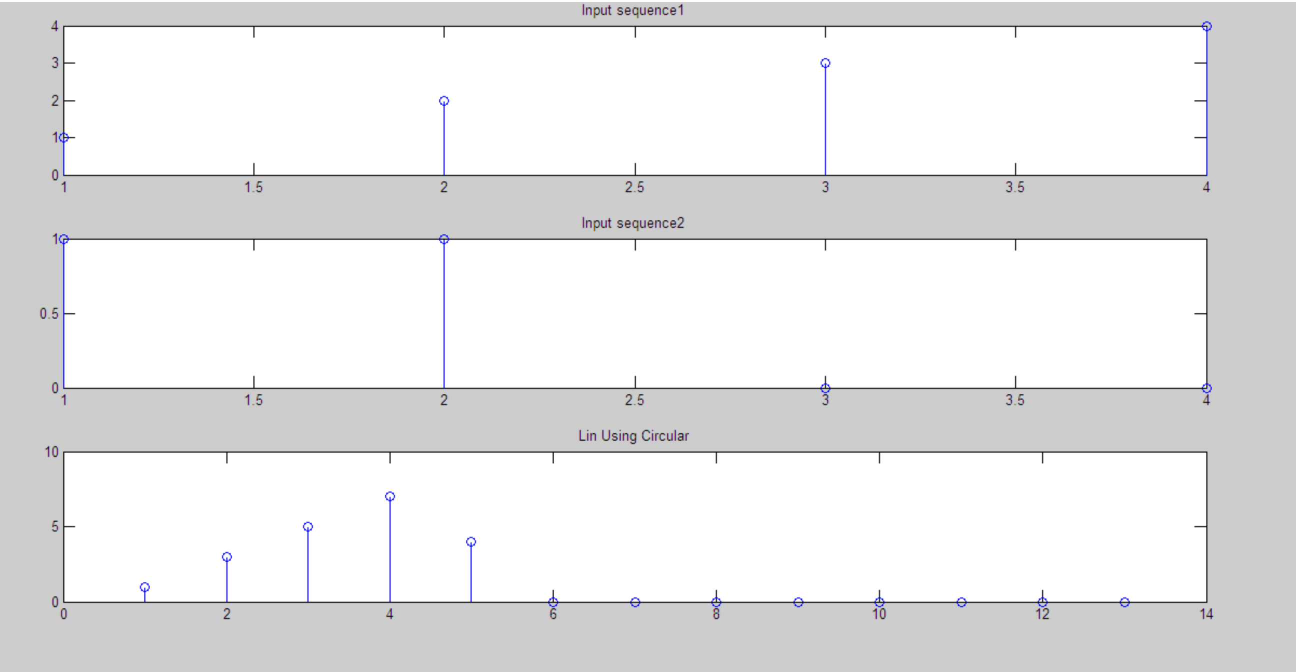
SAMPLE INPUT:-

Enter the co-efficient of x(n)=[1 2 3]

Enter the co-efficient of h(n)=[1 1 1]

```
y = 6     6     6
```

FIGURE:



SAMPLE INPUT:-

Enter the co-efficient of x(n)=[1 2 3 4]

Enter the co-efficient of h(n)=[1 1 0 0]

RESULTS: - Thus the program for linear convolution, circular convolution and linear convolution using circular convolution is written using MATLAB and verified.

GENERATION OF DISCRETE FOURIER TRANSFORM AND INVERSE DISCRETE FOURIER TRANSFORM

Experiment No: - 03

AIM: - TO write a MATLAB program to find the DFT and IDFT of a sequence

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

ALGORITHM:-

- Enter the input sequence x[n]
- Enter the length of sequence
- Use the matlab function „fft‟ for DFT
- Use the matlab function „ifft‟ for IDFT
- Plot the input and output sequence

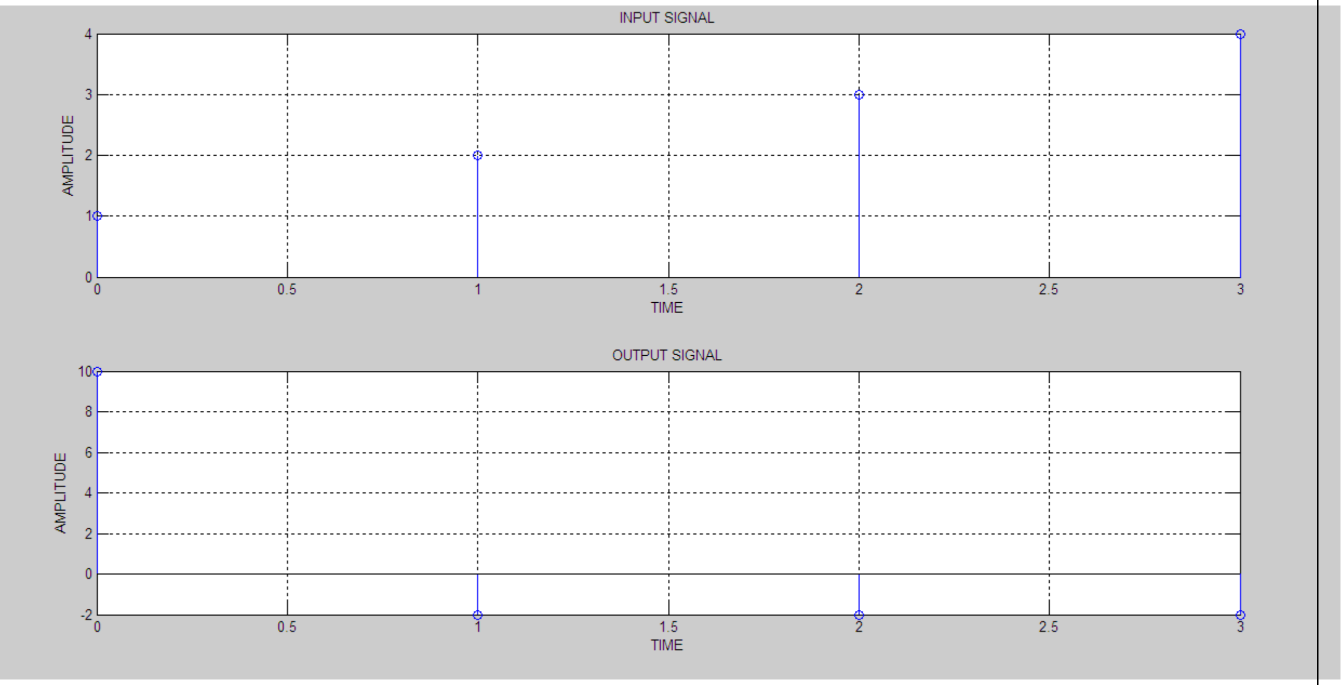
MATLAB CODE:- DFT

```
clc;
clear all;
close all;
N=input('Enter the value of N');
x=input('Enter the input sequence X(n):');
t=0:N-1;
subplot(2,1,1);
stem(t,x);
xlabel('TIME');
ylabel('AMPLITUDE');
title('INPUT SIGNAL');
grid on;
y=fft(x,N)
subplot(2,1,2);
stem(t,y);
xlabel('TIME');
ylabel('AMPLITUDE');
title('OUTPUT SIGNAL');
grid on;
```

MATLAB CODE:- IDFT

```
clc;
clear all;
close all;
N=input('Enter the value of N=');
y=input('Enter the sequence y[n]=');
t=0:N-1;
subplot(2,1,1);
stem(t,y);
xlabel('TIME');
ylabel('AMPLITUDE');
title('INPUT SIGNAL');
grid on;
x=ifft(y,N)
subplot(2,1,2);
stem(t,x);
xlabel('TIME');
ylabel('AMPLITUDE');
title('OUTPUT SIGNAL');
grid on;;
```

FIGURE: DFT



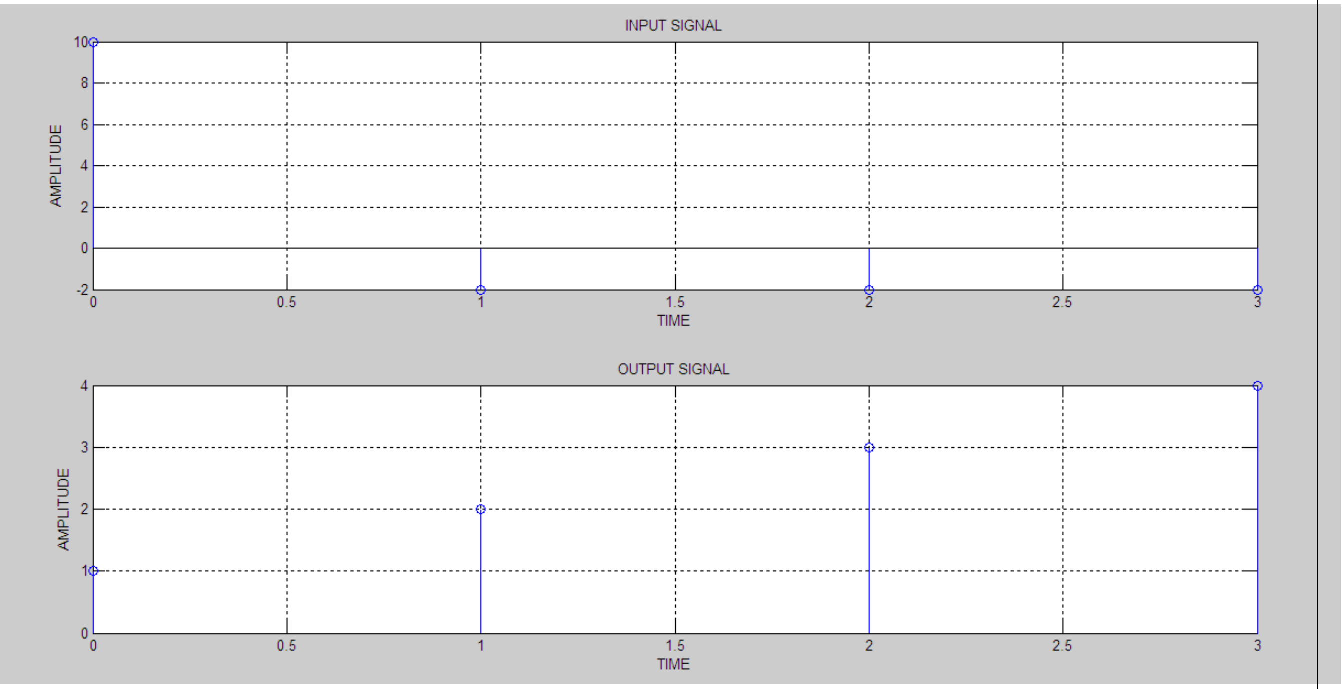
SAMPLE INPUT:-

Enter the value of N 4

Enter the input sequence X(n):[1 2 3 4]

```
y = 10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i
```

FIGURE: IFFT



SAMPLE INPUT:-

Enter the value of N=4

Enter the sequence y[n]=[10 -2+2i -2 -2-2i]

x =1 2 3 4

RESULTS: - Thus the program for DFT and IDFT is written using MATLAB and verified.

GENERATION OF FAST FOURIER TRANSFORM AND INVERSE FAST FOURIER TRANSFORM

Experiment No: - 04

AIM: - To write a MATLAB program to find the FFT and IFFT of a sequence

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

ALGORITHM:-

- Enter the input sequence x[n]
- Enter the length of sequence
- Use the matlab function „fft‟ for FFT
- Use the matlab function „ifft‟ for IFFT
- Plot the input and output sequence

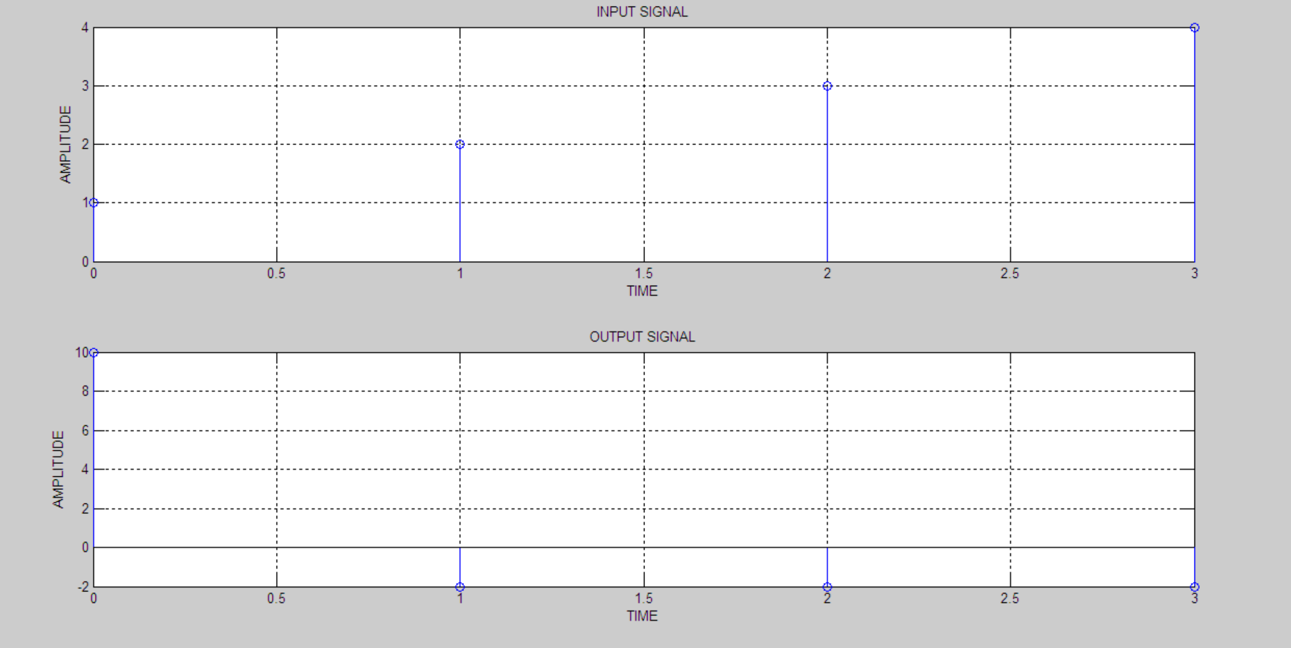
MATLAB CODE:- FFT

```
clc;
clear all;
close all;
N=input('Enter the value of N which is a factor of 2');
x=input('Enter the input sequence x(n):');
t=0:N-1;
subplot(2,1,1);
stem(t,x);
xlabel('TIME');
ylabel('AMPLITUDE');
title('INPUT SIGNAL');
grid on;
x=[x,zeros(1,N-length(x))];
for k=1:N
y(k)=0;
for n= 1:N
y(k)=y(k) + x(n)*exp(-1i*2*pi*(k-1)*(n-1))/N);
subplot(2,1,2);
stem(t,y);
xlabel('TIME');
ylabel('AMPLITUDE');
title('OUTPUT SIGNAL');
grid on;
```

MATLAB CODE:- IFFT

```
clc;
clear all;
close all;
N=input('Enter the value of N=');
y=input('Enter the sequence y[n]=');
t=0:N-1;
subplot(2,1,1);
stem(t,y);
xlabel('TIME');
ylabel('AMPLITUDE');
title('INPUT SIGNAL');
grid on;
y=[y,zeros(1,N-length(y))];
for k=1:N
x(k)=0;
for n= 1:N
x(k)=x(k) + y(n)*exp(1i*2*pi*(k-1)*(n-1))/N);
subplot(2,1,2);
stem(t,x);
xlabel('TIME');
ylabel('AMPLITUDE');
title('OUTPUT SIGNAL');
grid on;;
```

FIGURE: FFT



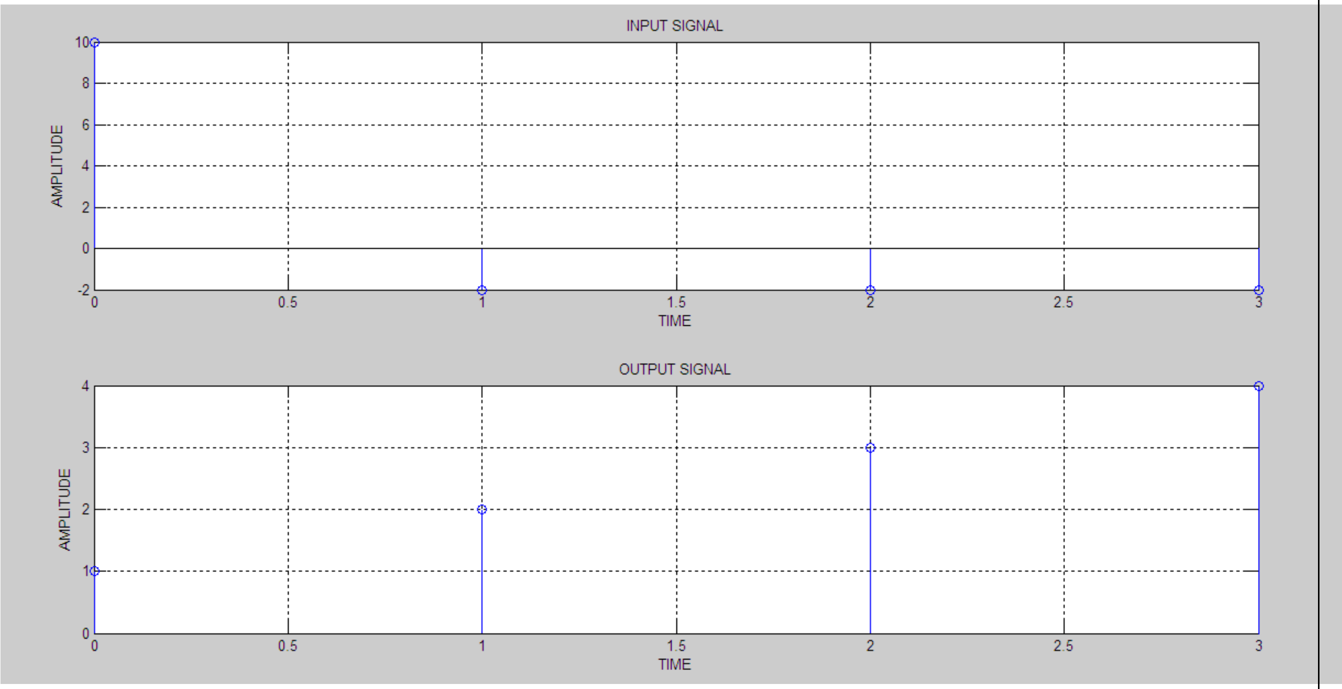
SAMPLE INPUT:-

Enter the value of N 4

Enter the input sequence x(n):[1 2 3 4]

y = 10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i

FIGURE: IFFT



SAMPLE INPUT:-

Enter the value of N=4

Enter the sequence y[n]=[10 -2+2i -2 -2-2i]

x =1 2 3 4

RESULTS: - Thus the program for FFT and IFFT is written using MATLAB and verified.

GENERATION OF AM, FM AND PWM WAVE

Experiment No: - 05

AIM: - To write MATLAB program to generating AM, FM and PWM wave

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- for the output see command window\ Figure window

ALGORITHM:-

- Get the amplitude and frequency of the signal
- Use „sin" matlab built in functions
- Using „plot" function plot the signal

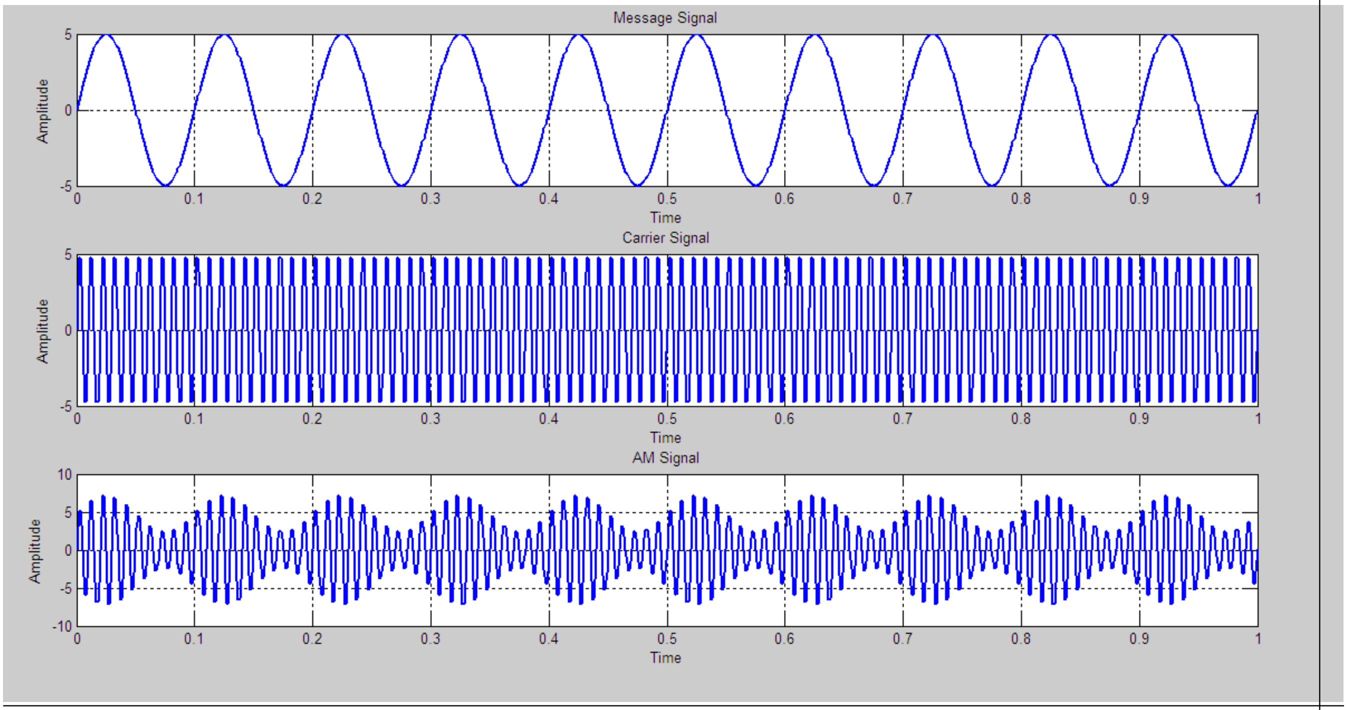
(a) AM WAVE

MATLAB CODE:-

```
clc;
clear all;
close all;
t=0:0.001:1;
set(0,'defaultlinelinewidth',2);
A=5;
fm=input('Message frequency=');
fc=input('Carrier frequency=');
mi=input('Modulation Index=');

Sm=A*sin(2*pi*fm*t);
subplot(3,1,1);
plot(t,Sm);
xlabel('Time');
ylabel('Amplitude');
title('Message Signal');
grid on;

Sc=A*sin(2*pi*fc*t);
subplot(3,1,2);
plot(t,Sc);
xlabel('Time');
ylabel('Amplitude');
title('Carrier Signal');
grid on;

Sfm=(A+mi*Sm).*sin(2*pi*fc*t);
subplot(3,1,3);
plot(t,Sfm);
xlabel('Time');
```

```
ylabel('Amplitude');
title('AM Signal');
grid on;
```

FIGURE:-



SAMPLE INPUT:-

```
Enter the value of message frequency 10
Enter the value of carrier frequency 100
Enter the value of modulation index 0.5
```

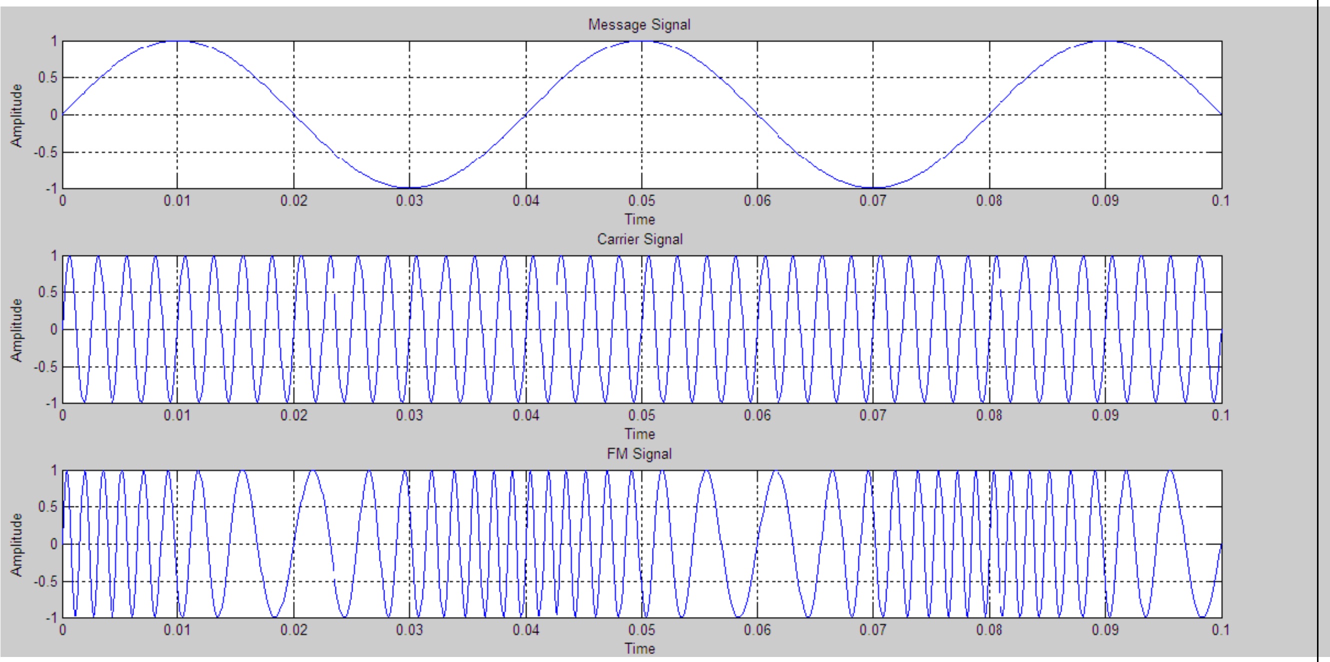
(b) FM WAVE

MATLAB CODE:-

```
clc;
clear all;
close all;
fm=input('Message Frequency=');
fc=input('Carrier Frequency=');
mi=input('Modulation Index=');
t=0:0.0001:0.1;
m=sin(2*pi*fm*t);
subplot(3,1,1);
plot(t,m);
xlabel('Time');
ylabel('Amplitude');
title('Message Signal');
grid on;

c=sin(2*pi*fc*t);
subplot(3,1,2);
```

```
plot(t,c);
xlabel('Time');
ylabel('Amplitude');
title('Carrier Signal');
grid on;

y=sin(2*pi*fc*t+(mi.*sin(2*pi*fm*t)));
subplot(3,1,3);
plot(t,y);
xlabel('Time');
ylabel('Amplitude');
title('FM Signal');
grid on;
```

FIGURE:-



SAMPLE INPUT:-

Enter the value of message frequency 25
Enter the value of carrier frequency 400
Enter the value of modulation index 10

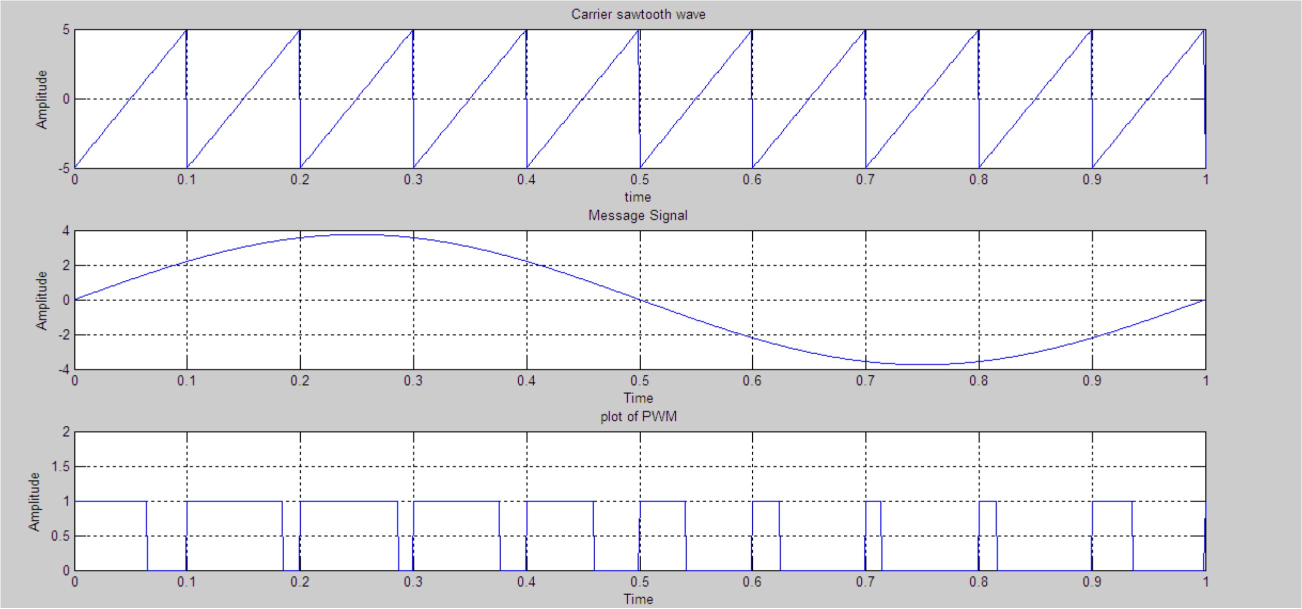
(c) PWM WAVE

MATLAB CODE:-

```
clc;
clear all;
close all;
F2=input('Message frequency=');
F1=input('Carrier Sawtooth frequency=');
A=5;
t=0:0.001:1;
c=A.*sawtooth(2*pi*F1*t);
```

```
subplot(3,1,1);
plot(t,c);
xlabel('time');
ylabel('Amplitude');
title('Carrier sawtooth wave');
grid on;
m=0.75*A.*sin(2*pi*F2*t);
subplot(3,1,2);
plot(t,m);
xlabel('Time');
ylabel('Amplitude');
title('Message Signal');
grid on;

n=length(c);
for i=1:n
if (m(i)>=c(i))
    pwm(i)=1;
else
    pwm(i)=0;
end
end
subplot(3,1,3);
plot(t,pwm);
xlabel('Time');
ylabel('Amplitude');
title('plot of PWM');
axis([0 1 0 2]);
grid on;
```

FIGURE:-



SAMPLE INPUT:-

Message frequency=1
Carrier Saw tooth frequency=10

RESULTS: - Thus the program for FM, AM and PWM is written using MATLAB and verified.

GENERATION OF FIR FILTER USING WINDOWS. (RECTANGULAR, HANNING AND HAMMING WINDOW)

Experiment No: - 06

AIM: - To write a MATLAB program to plot magnitude response and phase response of digital FIR filter using windows.

PROCEDURE:-

□ Open MATLAB
□ Open new M-file
□ Type the program
□ Save in current directory
□ Compile and Run the program
□ For the output see command window\ Figure window

ALGORITHM:-

□ Get the order of the filter
□ Get the cut off frequency
□ Use „fir1 “& corresponding window functions to compute the filter coefficient
□ Draw the magnitude and phase response

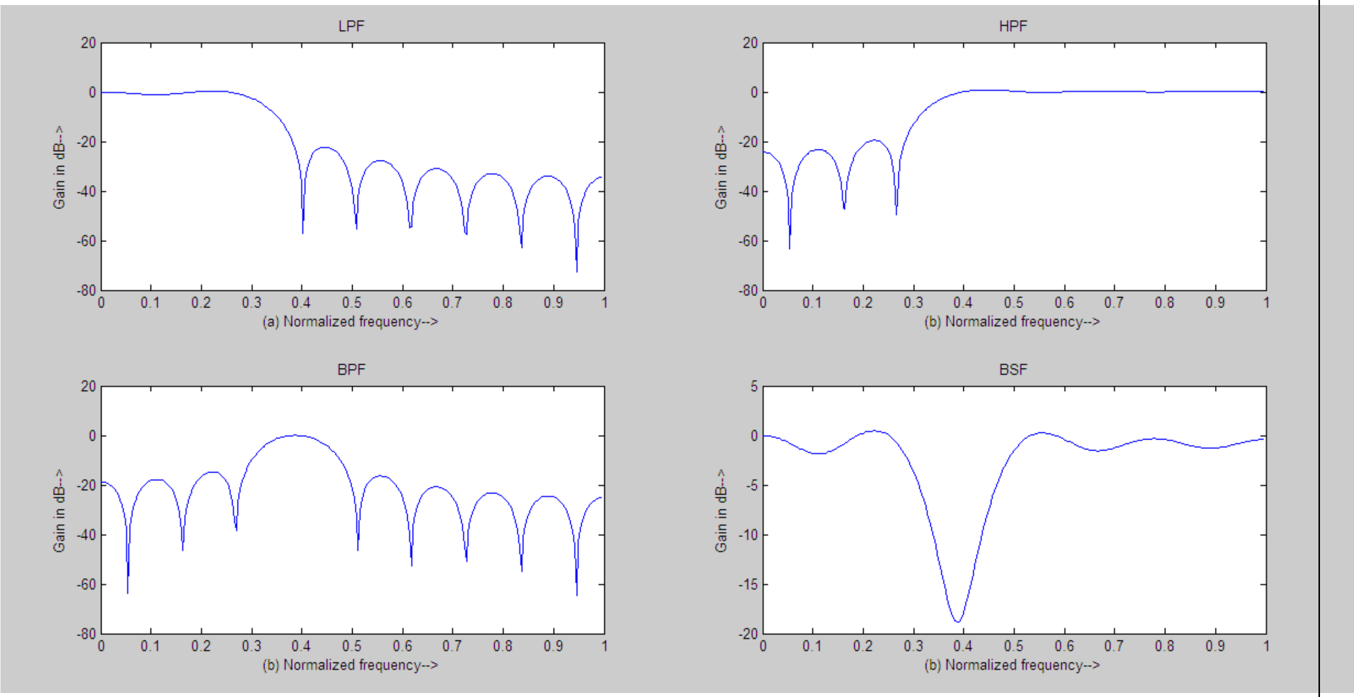
MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input('enter passband ripple');
rs=input('enter the stopband ripple');
fp=input('enter passband freq');
fs=input('enter stopband freq');
f=input('enter sampling freq ');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2)~=0)
n1=n;
n=n-1;
end
c=input('enter your choice of window function 1. rectangular 2.hanning
3.hamming: \n ');
if(c==1)
y=rectwin(n1);
disp('Rectangular window filter response');
end
if(c==2)
y=hanning(n1);
disp('hanning window filter response');
end
if(c==3)
```

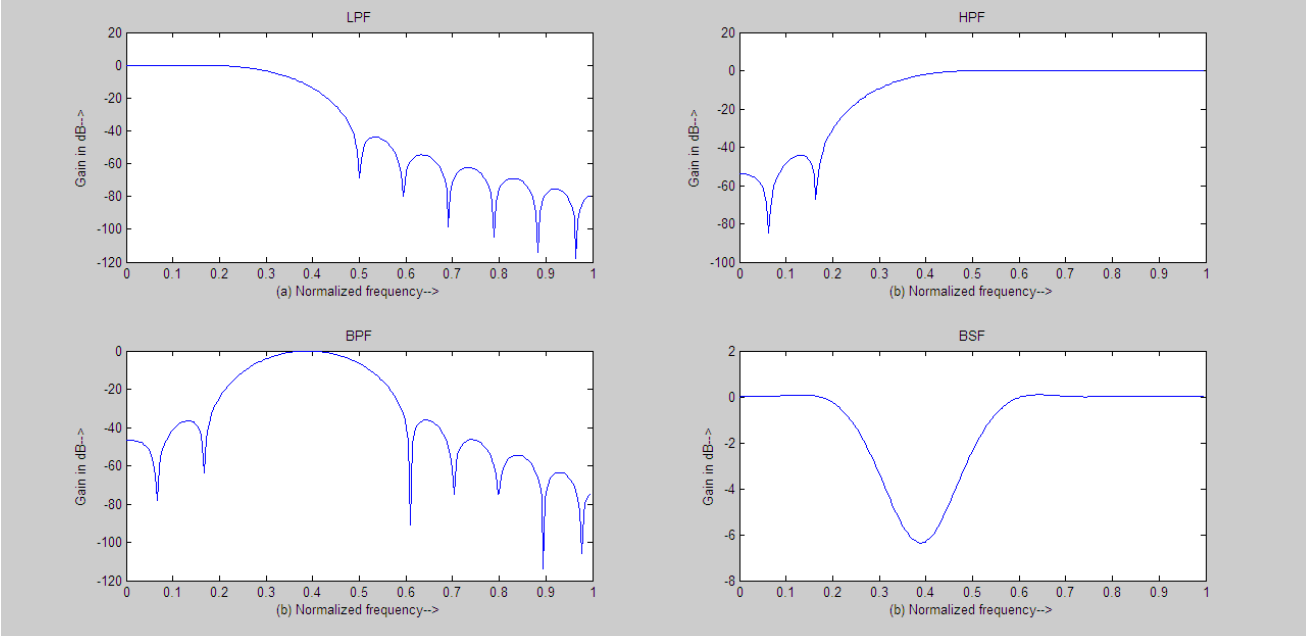
```
y=hamming(n1);
disp('hamming window filter response');
end


%LPF
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('LPF');
ylabel('Gain in dB-->');
xlabel('(a) Normalized frequency-->');

%HPF
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);plot(o/pi,m);
title('HPF');
ylabel('Gain in dB-->');
xlabel('(b) Normalized frequency-->');

%BPF
wn=[wp,ws];
b=fir1(n,wn,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);plot(o/pi,m);
title('BPF');
ylabel('Gain in dB-->');
xlabel('(b) Normalized frequency-->');


%BSF
wn=[wp,ws];
b=fir1(n,wn,'stop',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);plot(o/pi,m);
title('BSF');
ylabel('Gain in dB-->');
xlabel('(b) Normalized frequency-->');
```

FIGURES:-

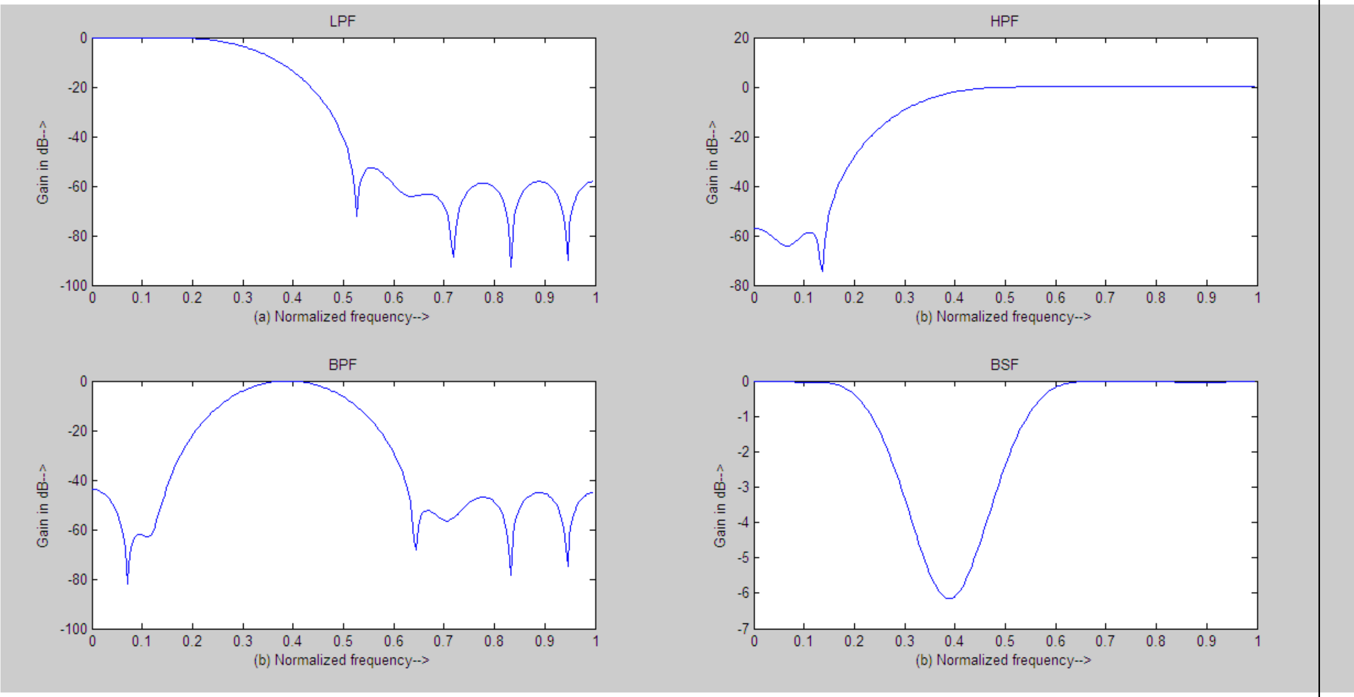
1) USING RECTANGULAR WINDOW



2) USING HANNING WINDOW



3) USING HAMMING WINDOW



SAMPLE INPUTS:-

Enter the value of pass band ripple:0.05
Enter the value of stop band ripple:0.04
Enter the value of pass band frequency:1500
Enter the value of stop band frequency:2000
Enter the value of sampling frequency:9000

RESULTS: - Thus the magnitude response of FIR filter using all types of windows was verified.

IIR FILTER DESIGN

Experiment No: - 07

AIM: - To write a MATLAB program to plot magnitude response and phase response of digital Butter worth and Chebychev filters.

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

ALGORITHM:- BUTTERWORTH FILTER

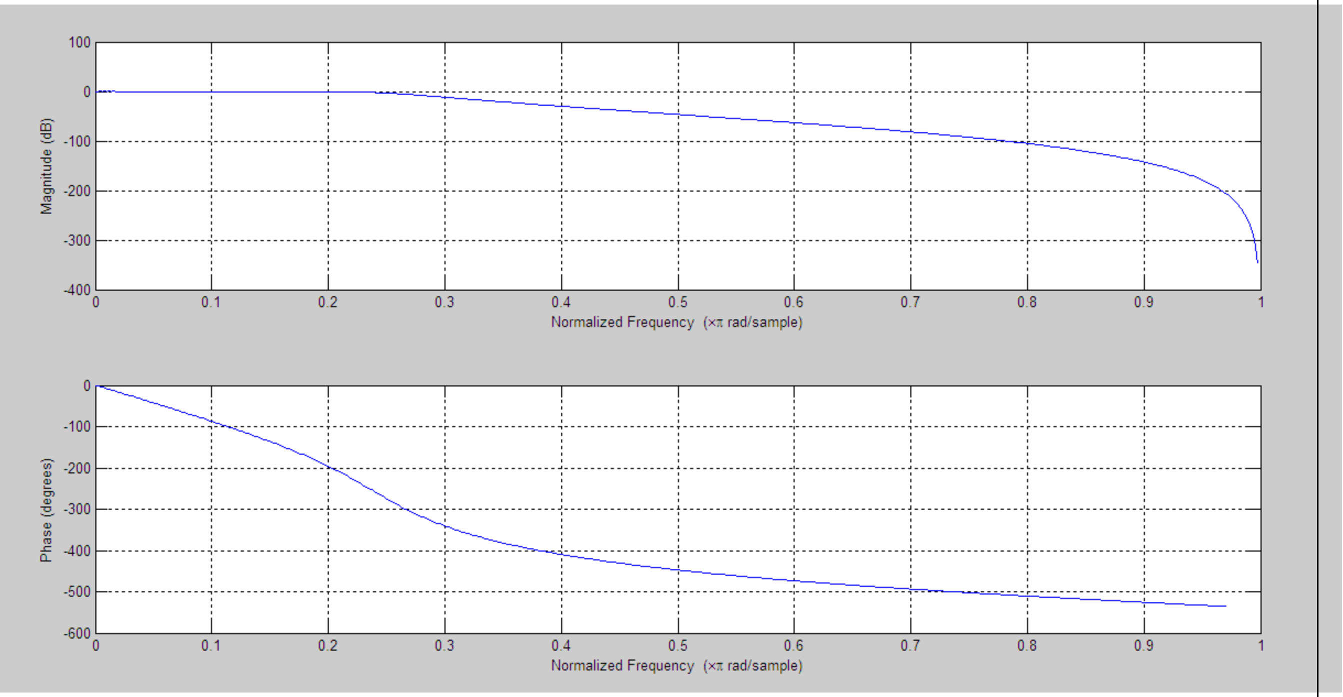
- Get the passband and stopband ripples
- Get the passband and stopband edge frequencies
- Calculate the order of the filter using „ buttord ‟ function
- Find the filter coefficients, using „butter‟ function
- Draw the magnitude and phase response

(i) BUTTERWORTH LOW PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input('enter the passband attenuation:');
rs=input('enter the stop band attenuation:');
wp=input('enter the pass band frequency:');
ws=input('enter the stop band frequency:');
[N,wn]=buttord(wp/pi,ws/pi,rp,rs);
[b,a]=butter(N,wn);
freqz(b,a)
```

FIGURE:-



SAMPLE INPUT:-

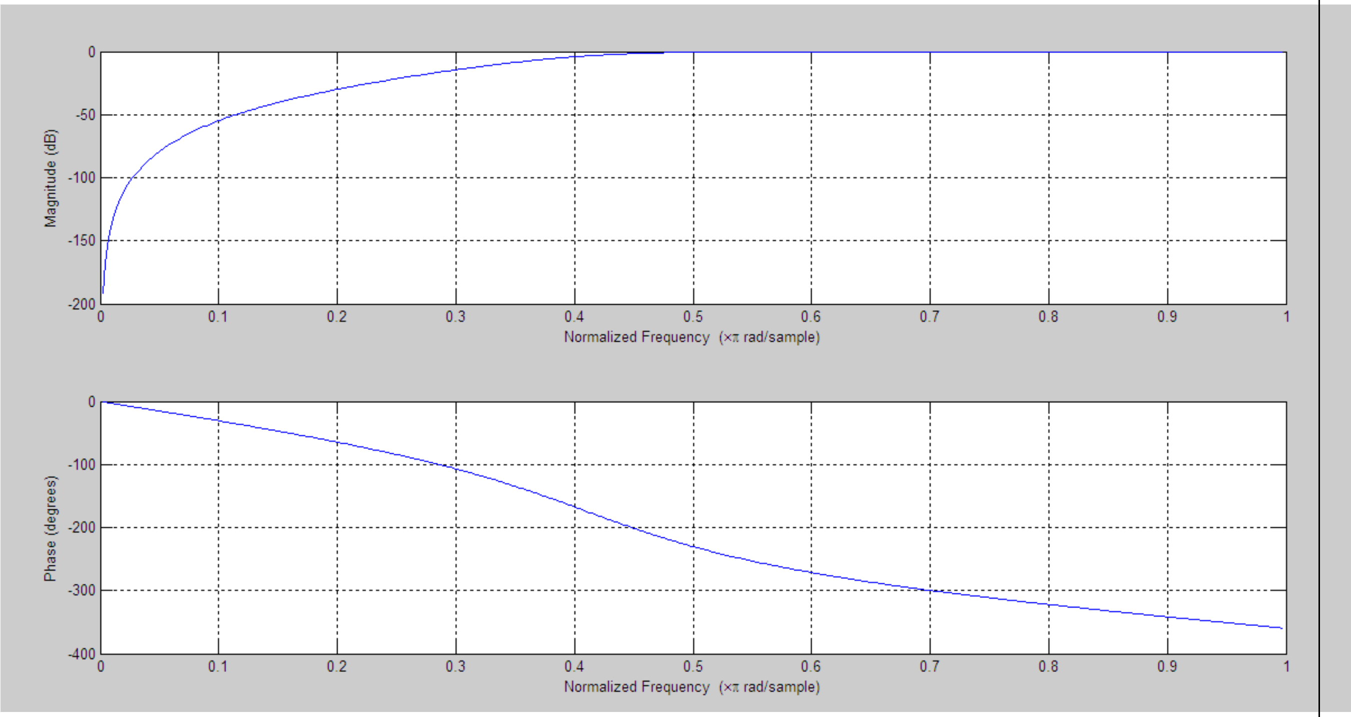
Enter the passband attenuation:0.4
Enter the stop band attenuation:30
Enter the pass band frequency:0.2*pi
Enter the stop band frequency:0.4*pi

(ii) BUTTERWORTH HIGH PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input ('Enter the pass band attenuation:');
rs=input ('Enter the stop band attenuation:');
wp=input ('Enter the pass band frequency:');
ws=input ('Enter the stop band frequency:');
[N,wn]=buttord(wp/pi,ws/pi,rp,rs);
[b,a]=butter(N,wn,'high');
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

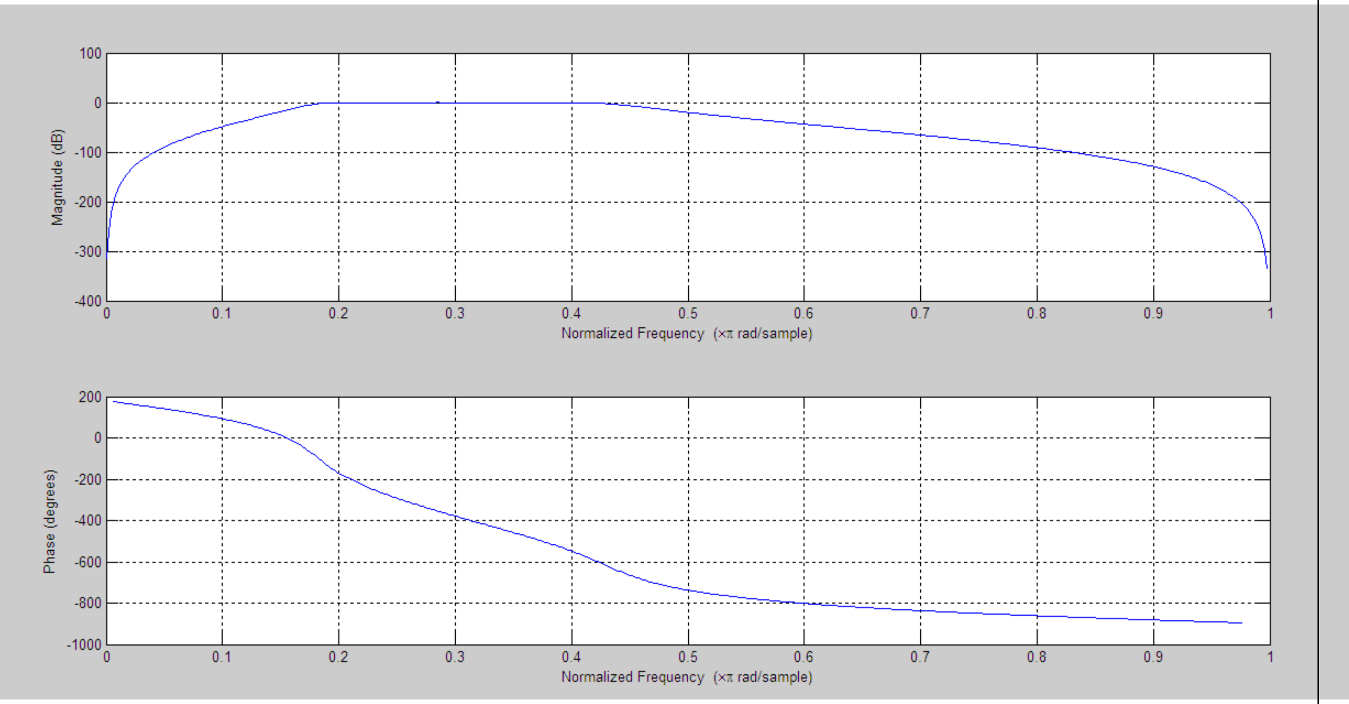
Enter the pass band attenuation:0.4
Enter the stop band attenuation:30
Enter the pass band frequency:0.6*pi
Enter the stop band frequency:0.2*pi

(iii) BUTTERWORTH BAND PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input('enter the passband attenuation:');
rs=input('enter the stop band attenuation:');
wp=input('enter the pass band frequency:');
ws=input('enter the stop band frequency:');
[N,wn]=buttord(wp/pi,ws/pi,rp,rs);
[b,a]=butter(N,wn);
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

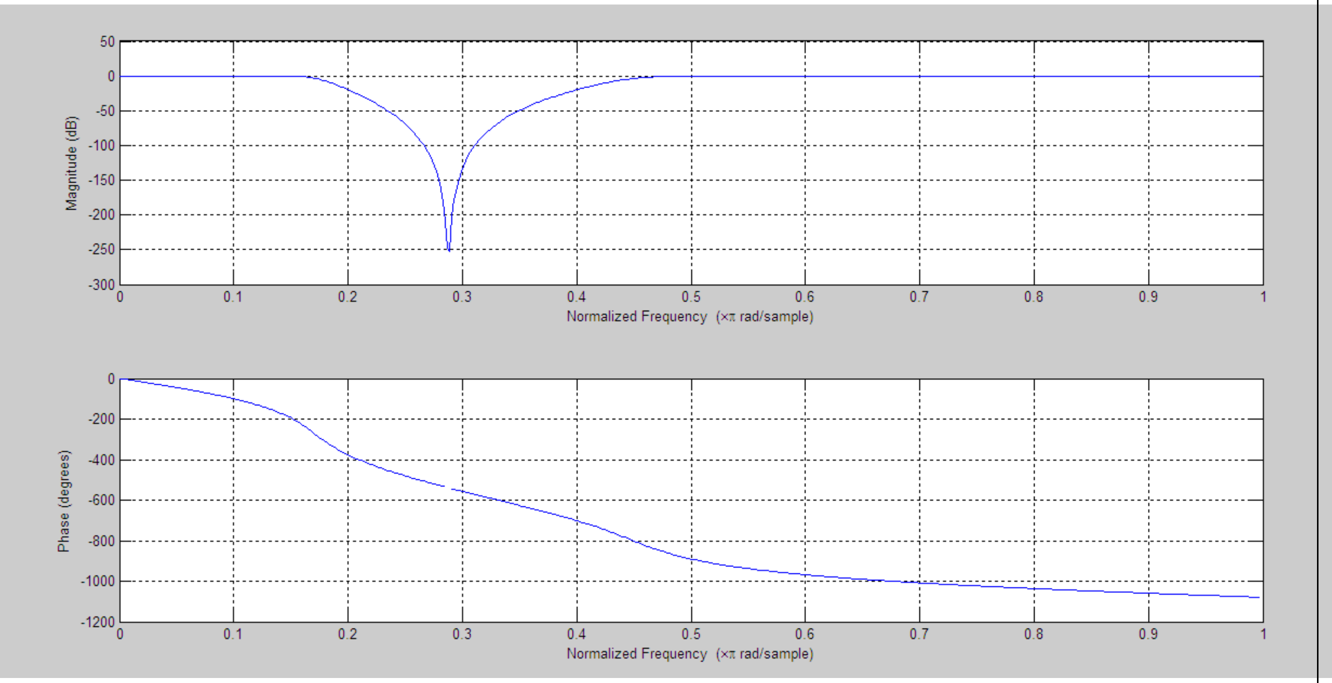
Enter the passband attenuation:0.2
Enter the stop band attenuation:20
Enter the pass band frequency:[0.2*pi,0.4*pi]
Enter the stop band frequency: [0.1*pi, 0.5*pi]

(iv) BUTTERWORTH BAND STOP FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input('enter the passband attenuation:');
rs=input('enter the stop band attenuation:');
wp=input('enter the pass band frequency:');
ws=input('enter the stop band frequency:');
[N,wn]=buttord(wp/pi,ws/pi,rp,rs);
[b,a]=butter(N,wn,‟stop‟);
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

Enter the passband attenuation:0.2
Enter the stop band attenuation:20
Enter the pass band frequency:[0.1*pi,0.5*pi]
Enter the stop band frequency:[0.2*pi,0.4*pi]

ALGORITHM:- CHEBYSHEV FILTERS

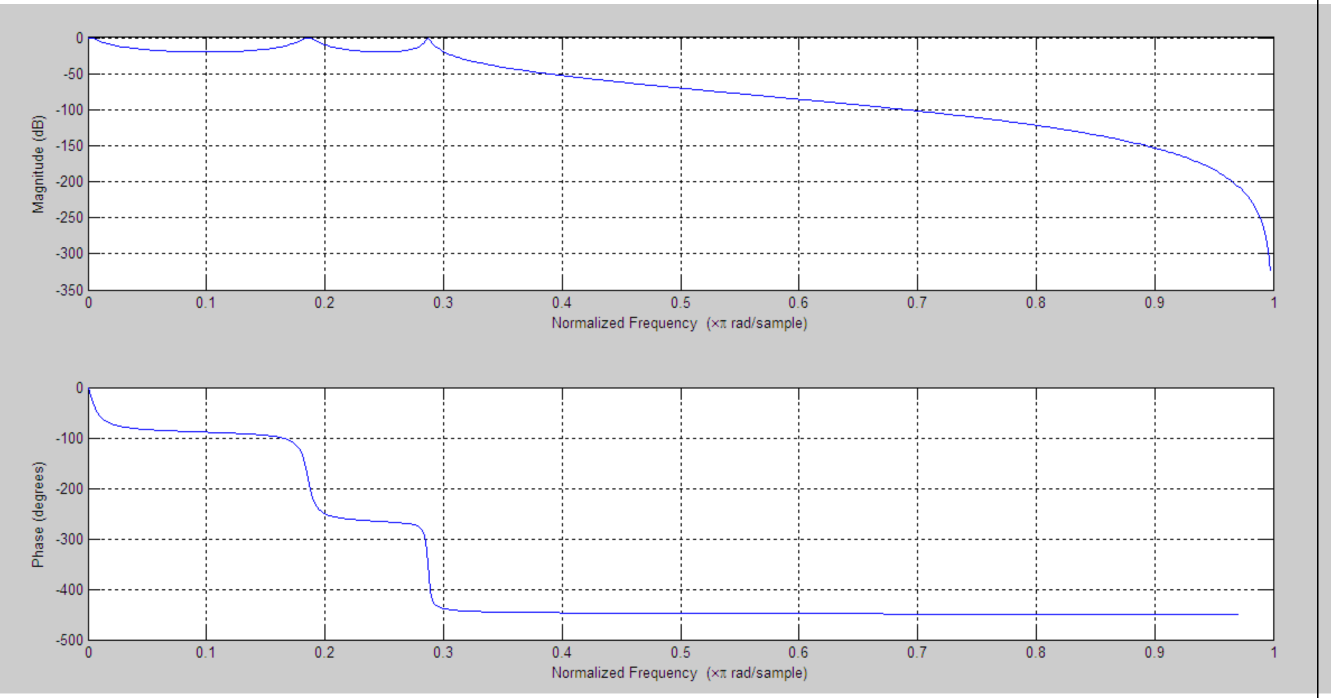
□ Get the passband and stopband ripples
□ Get the passband and stopband edge frequencies
□ Calculate the order of the filter using „ cheb1ord ‟ function
□ Find the filter coefficients, using „cheby1‟ function
□ Draw the magnitude and phase response

(i) CHEBYSHEV(TYPE-1) LOW PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input ('Enter the pass band attenuation:');
rs=input ('Enter the stop band attenuation:');
wp=input ('Enter the pass band frequency:');
ws=input ('Enter the stop band frequency:');
[N,wn]=cheb1ord(wp/pi,ws/pi,rp,rs);
[b,a]=cheby1(N,rp,wn);
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

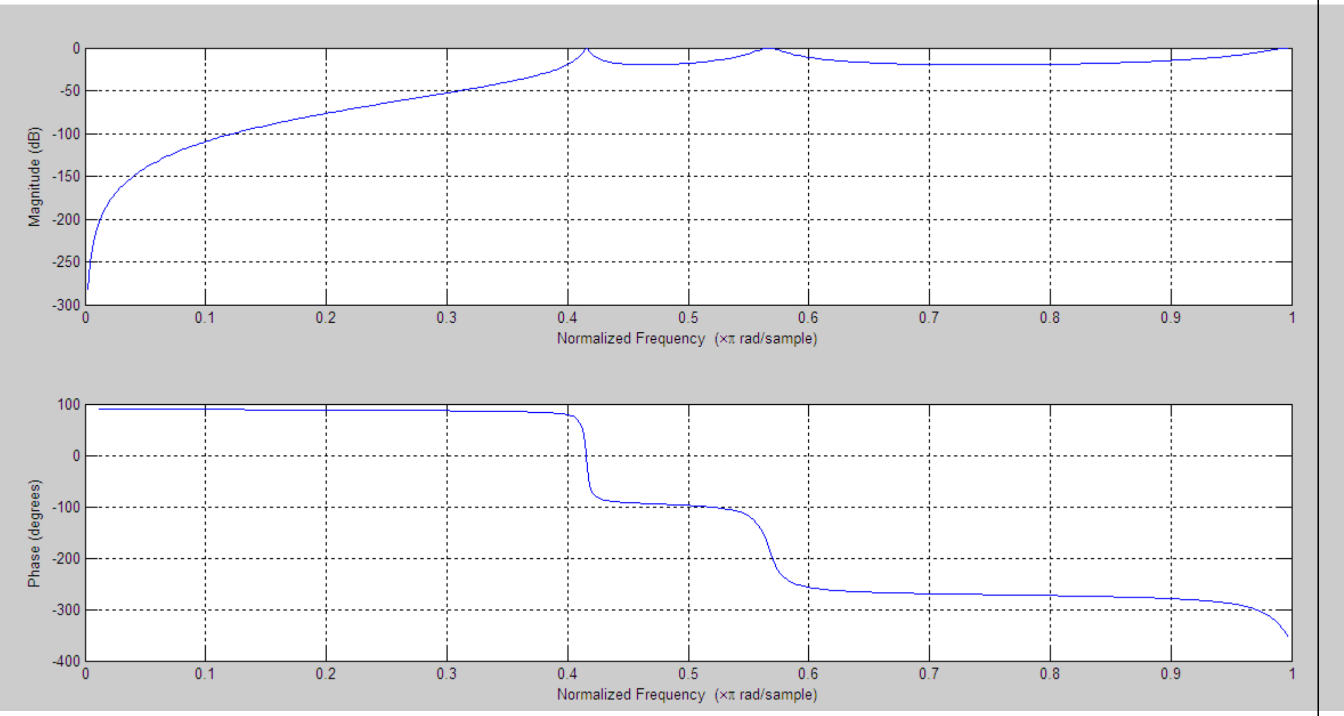
```
Enter the pass band attenuation:20
Enter the stop band attenuation:50
Enter the pass band frequency:0.3*pi
Enter the stop band frequency:0.4*pi
```

(ii) CHEBYSHEV(TYPE-1)HIGH PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input ('Enter the pass band attenuation:');
rs=input ('Enter the stop band attenuation:');
wp=input ('Enter the pass band frequency:');
ws=input ('Enter the stop band frequency:');
[N,wn]=cheb1ord(wp/pi,ws/pi,rp,rs);
[b,a]=cheby1(N,rp,wn,'high');
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

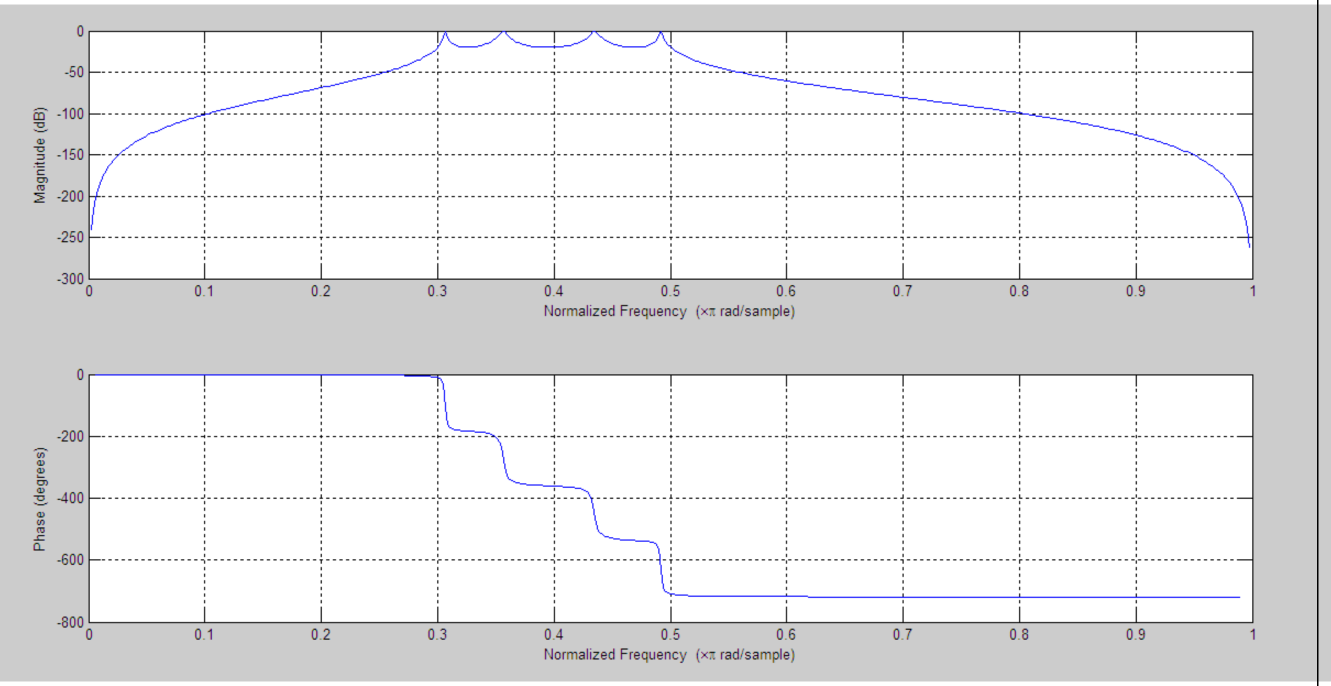
Enter the pass band attenuation:20
Enter the stop band attenuation:50
Enter the pass band frequency:0.4*pi
Enter the stop band frequency:0.3*pi

(iii) CHEBYSHEV(TYPE-1) BAND PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input ('Enter the pass band attenuation:');
rs=input ('Enter the stop band attenuation:');
wp=input ('Enter the pass band frequency:');
ws=input ('Enter the stop band frequency:');
[N,wn]=cheb1ord(wp/pi,ws/pi,rp,rs);
[b,a]=cheby1(N,rp,wn);
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

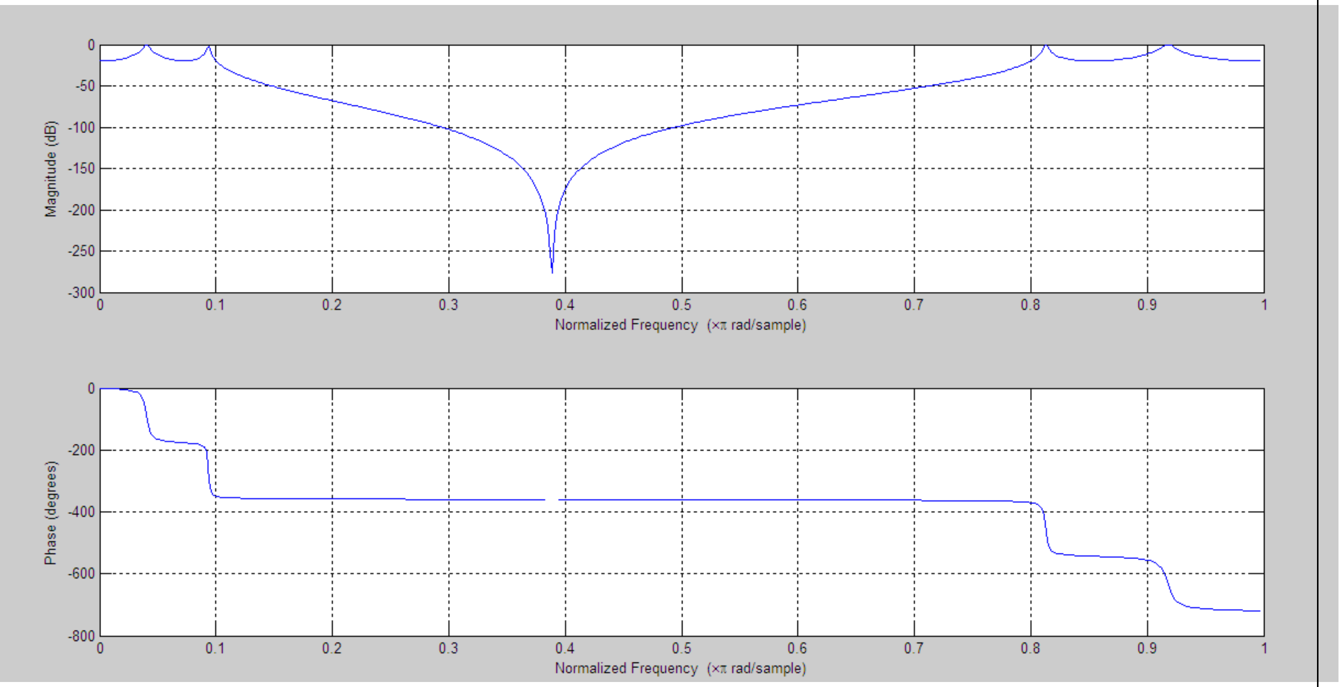
```
Enter the pass band attenuation:20
Enter the stop band attenuation:98
Enter the pass band frequency:[0.3*pi,0.5*pi]
Enter the stop band frequency:[0.1*pi,0.8*pi]
```

(iv) CHEBYSHEV(TYPE-1)BAND STOP FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input('Enter the pass band attenuation:');
rs=input('Enter the stop band attenuation:');
wp=input('Enter the pass band frequency:');
ws=input('Enter the stop band frequency:');
[N,wn]=cheb1ord(wp/pi,ws/pi,rp,rs);
[b,a]=cheby1(N,rp,wn,'stop');
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

Enter the pass band attenuation:20
Enter the stop band attenuation:98
Enter the pass band frequency:[0.1*pi,0.8*pi]
Enter the stop band frequency:[0.3*pi,0.5*pi]

ALGORITHM:- CHEBYSHEV TYPE 2 FILTERS

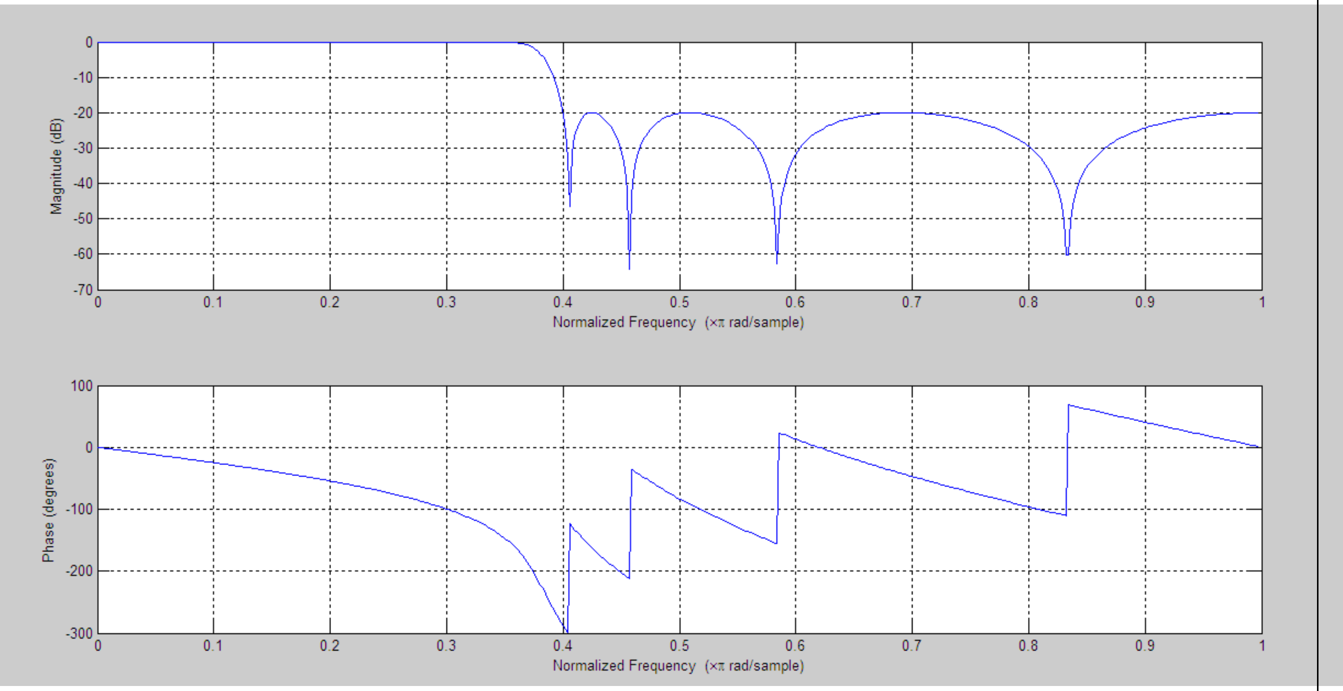
□ Get the passband and stopband ripples
□ Get the passband and stopband edge frequencies
□ Calculate the order of the filter using „ cheb2ord ‟ function
□ Find the filter coefficients, using „cheby2‟ function
□ Draw the magnitude and phase response

(i) CHEBYSHEV(TYPE-2) LOW PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input ('Enter the pass band attenuation:');
rs=input ('Enter the stop band attenuation:');
wp=input ('Enter the pass band frequency:');
ws=input ('Enter the stop band frequency:');
[N,wn]=cheb2ord(wp/pi,ws/pi,rp,rs);
[b,a]=cheby2(N,rp,wn);
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

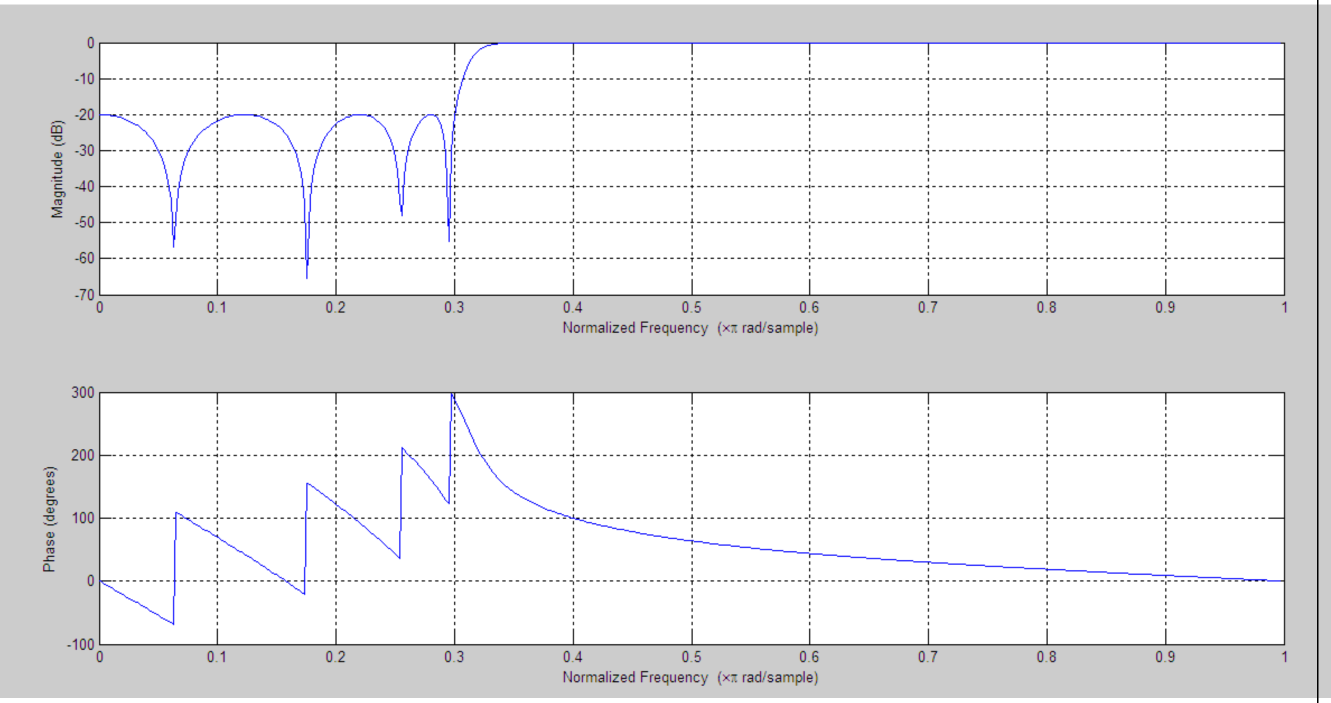
Enter the pass band attenuation:20
Enter the stop band attenuation:70
Enter the pass band frequency:0.3*pi
Enter the stop band frequency:0.4*pi

(ii) CHEBYSHEV (TYPE-2) HIGH PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input ('Enter the pass band attenuation:');
rs=input ('Enter the stop band attenuation:');
wp=input ('Enter the pass band frequency:');
ws=input ('Enter the stop band frequency:');
[N,wn]=cheb2ord(wp/pi,ws/pi,rp,rs);
[b,a]=cheby2(N,rp,wn,'high');
freqz(b,a)
```

FIGURE:-



SAMPLE INPUT:-

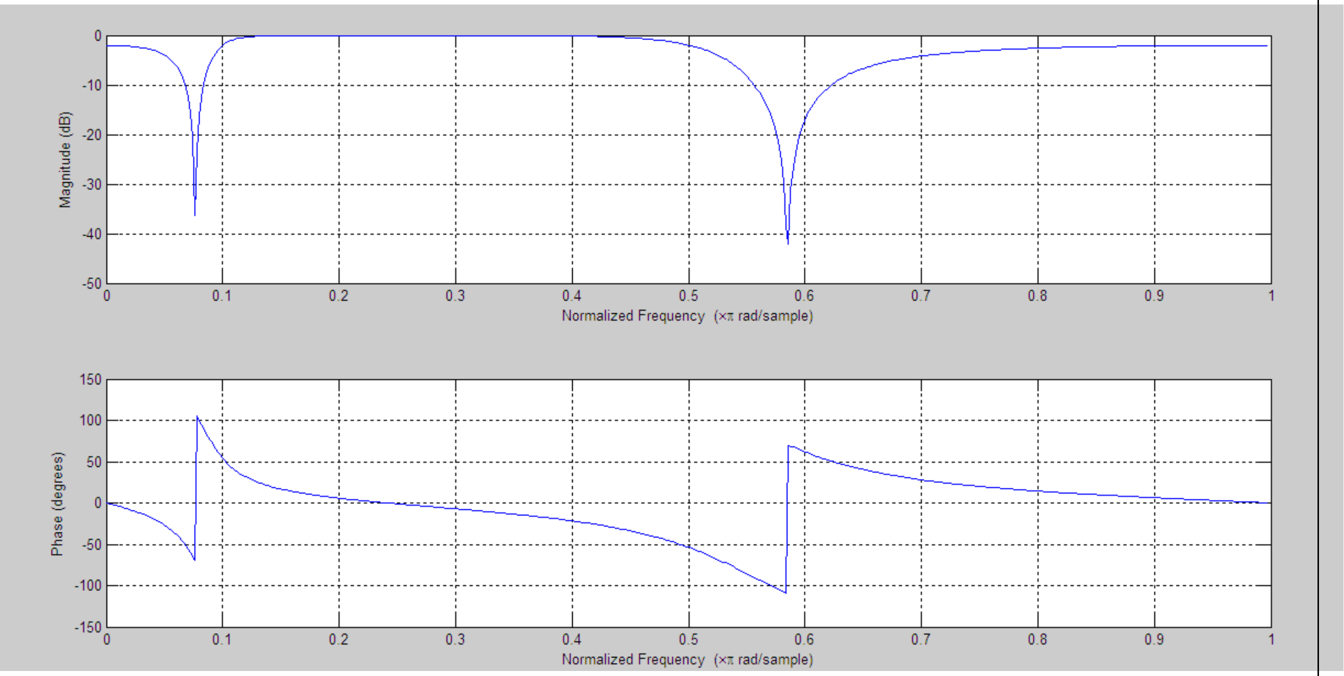
```
Enter the pass band attenuation:20
Enter the stop band attenuation:70
Enter the pass band frequency:0.4*pi
Enter the stop band frequency:0.3*pi
```

(iii) CHEBYSHEV (TYPE-2) BAND PASS FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input ('Enter the pass band attenuation:');
rs=input ('Enter the stop band attenuation:');
wp=input ('Enter the pass band frequency:');
ws=input ('Enter the stop band frequency:');
[N,wn]=cheb2ord(wp/pi,ws/pi,rp,rs);
[b,a]=cheby2(N,rp,wn);
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

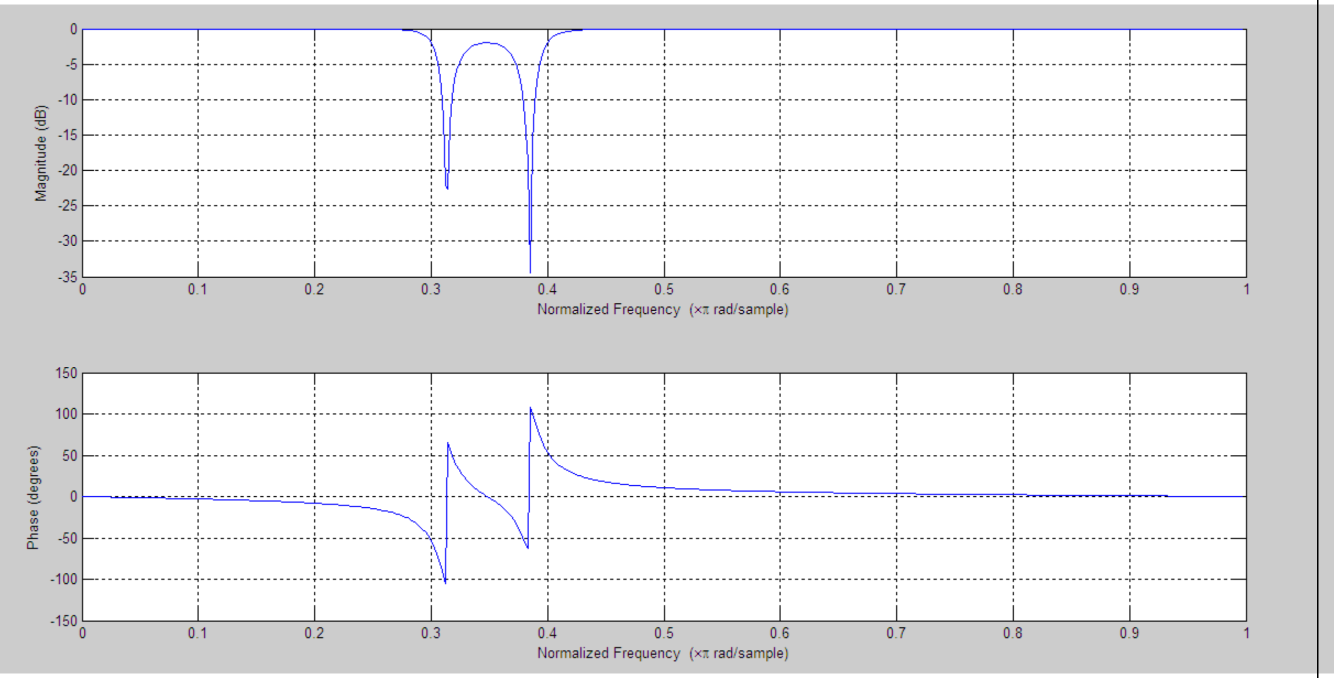
Enter the pass band attenuation:2
Enter the stop band attenuation:20
Enter the pass band frequency:[0.3*pi,0.4*pi]
Enter the stop band frequency:[0.1*pi,0.5*pi]

(iv) CHEBYSHEV (TYPE-2) BAND STOP FILTER

MATLAB CODE:-

```
clc;
clear all;
close all;
rp=input('Enter the pass band attenuation:');
rs=input('Enter the stop band attenuation:');
wp=input('Enter the pass band frequency:');
ws=input('Enter the stop band frequency:');
[N,wn]=cheb2ord (wp/pi,ws/pi,rp,rs);
[b,a]=cheby2(N,rp,wn,'stop');
freqz(b,a);
```

FIGURE:-



SAMPLE INPUT:-

Enter the pass band attenuation:2
Enter the stop band attenuation:20
Enter the pass band frequency:[0.1*pi,0.5*pi]
Enter the stop band frequency:[0.3*pi,0.4*pi]

RESULTS: - Thus the Amplitude response and phase response of Butterworth and chebyshev type 1 and type 2 filters were verified.

INTRODUCTION TO THE TMS 320C6713

The C6713 DSK is a low-cost standalone development platform that enables users to evaluate and develop applications for the TI C67xx DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market.

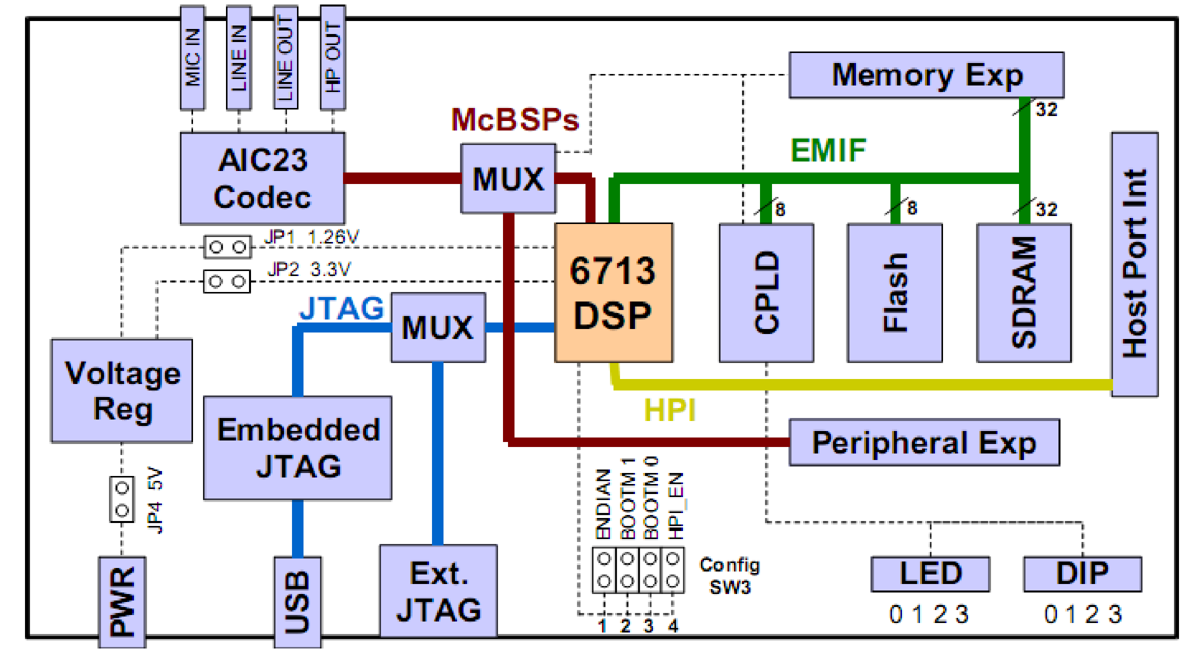


Figure 1-1, Block Diagram C6713 DSK

The DSK comes with a full compliment of on-board devices that suit a wide variety of application environments. Key features include:

- A Texas Instruments TMS320C6713 DSP operating at 225 MHz.
- An AIC23 stereo codec
- 16 Mbytes of synchronous DRAM
- 512 Kbytes of non-volatile Flash memory (256 Kbytes usable in default configuration)
- 4 user accessible LEDs and DIP switches

- Software board configuration through registers implemented in CPLD
- Configurable boot options
- Standard expansion connectors for daughter card use
- JTAG emulation through on-board JTAG emulator with USB host interface or external emulator
- Single voltage power supply (+5V)

FUNCTIONAL OVERVIEW OF THE TMS 320C6713 DSK

The DSP on the 6713 DSK interfaces to on-board peripherals through a 32-bit wide EMIF (External Memory InterFace). The SDRAM, Flash and CPLD are all connected to the bus. EMIF signals are also connected daughter card expansion connectors which are used for third party add-in boards.

The DSP interfaces to analog audio signals through an on-board AIC23 codec and four 3.5 mm audio jacks (microphone input, line input, line output, and headphone output). The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP0 is used to send commands to the codec control interface while McBSP1 is used for digital audio data. McBSP0 and McBSP1 can be re-routed to the expansion connectors in software.

A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD has a register based user interface that lets the user configure the board by reading and writing to its registers.

The DSK includes 4 LEDs and a 4 position DIP switch as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers.

An included 5V external power supply is used to power the board. On-board switching voltage regulators provide the +1.26V DSP core voltage and +3.3V I/O supplies. The board is held in reset until these supplies are within operating specifications.

Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

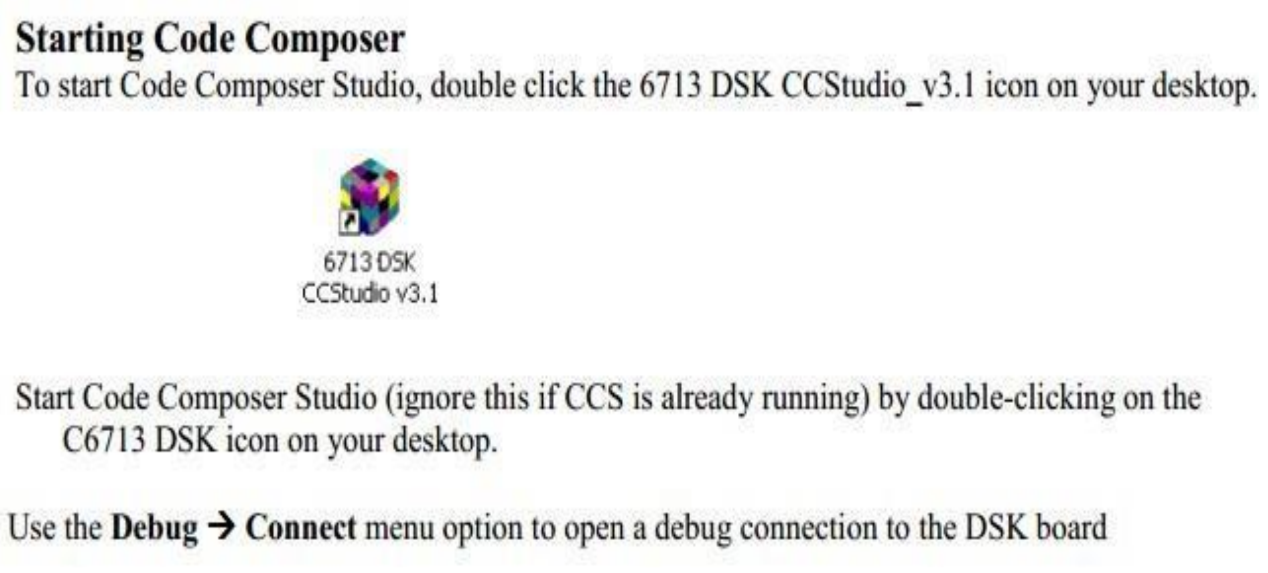
CODE COMPOSER STUDIO

The Code Composer Studio (CCS) provides an integrated development environment (IDE) to incorporate the software tools. CCS includes tools for code generation, such as a C compiler, an assembler, and a linker. It has graphical capabilities and supports real-time debugging. It provides an easy-to-use software tool to build and debug programs.

The C compiler compiles a C source program with extension .c to produce an assembly source file with extension*.asm*. The assembler assembles an*.asm* source file to produce a machine language object file with extension*.obj*. The linker combines object files and object libraries as input to produce an executable file with extension*.out*. This executable file represents a linked common object file format (COFF), popular in Unix-based systems and adopted by several makers of digital signal processors. This executable file can be loaded and run directly on the C6713 processor.

To create an application project, one can “add” the appropriate files to the project. Compiler/linker options can readily be specified. A number of debugging features are available, including setting breakpoints and watching variables, viewing memory, registers, and mixed C and assembly code, graphing results, and monitoring execution time. One can step through a program in different ways (step into, or over, or out).

Real-time analysis can be performed using real-time data exchange (RTDX) associated with DSP/BIOS .RTDX allows for data exchange between the host and the target and analysis in real time without stopping the target. Key statistics and performance can be monitored in real time. Through the Joint Team Action Group (JTAG), communication with on-chip emulation support occurs to control and monitor program execution. The C6713 DSK board includes a JTAG emulator interface.



Starting Code Composer

To start Code Composer Studio, double click the 6713 DSK CCStudio_v3.1 icon on your desktop.

Start Code Composer Studio (ignore this if CCS is already running) by double-clicking on the C6713 DSK icon on your desktop.

Use the **Debug → Connect** menu option to open a debug connection to the DSK board

GENERATION OF SINE WAVE AND STANDARD TEST SIGNALS.

Experiment No. 8

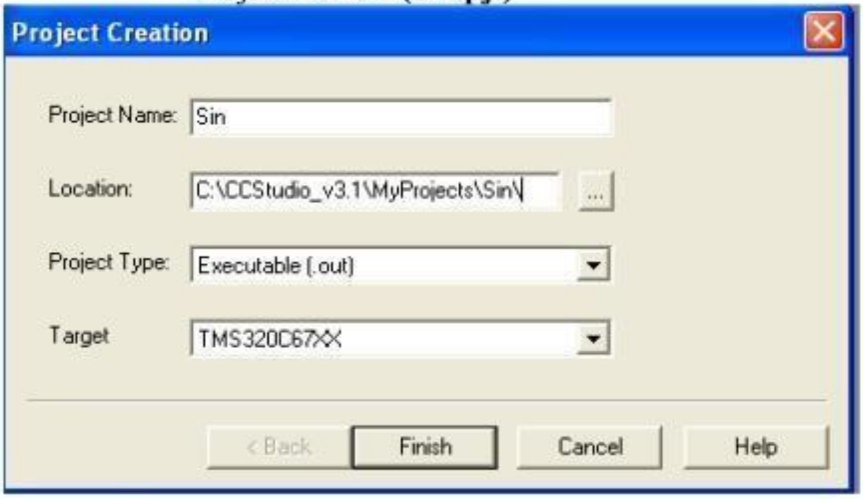
AIM: To generate a Sine waveform.

Procedure to work on Code Composer Studio

Generation of sine waveform

1. To create a New Project

Project → New **(Sin.pjt)**



Project Name: Sin

Location: C:\CCStudio_v3.1\MyProjects\Sin\

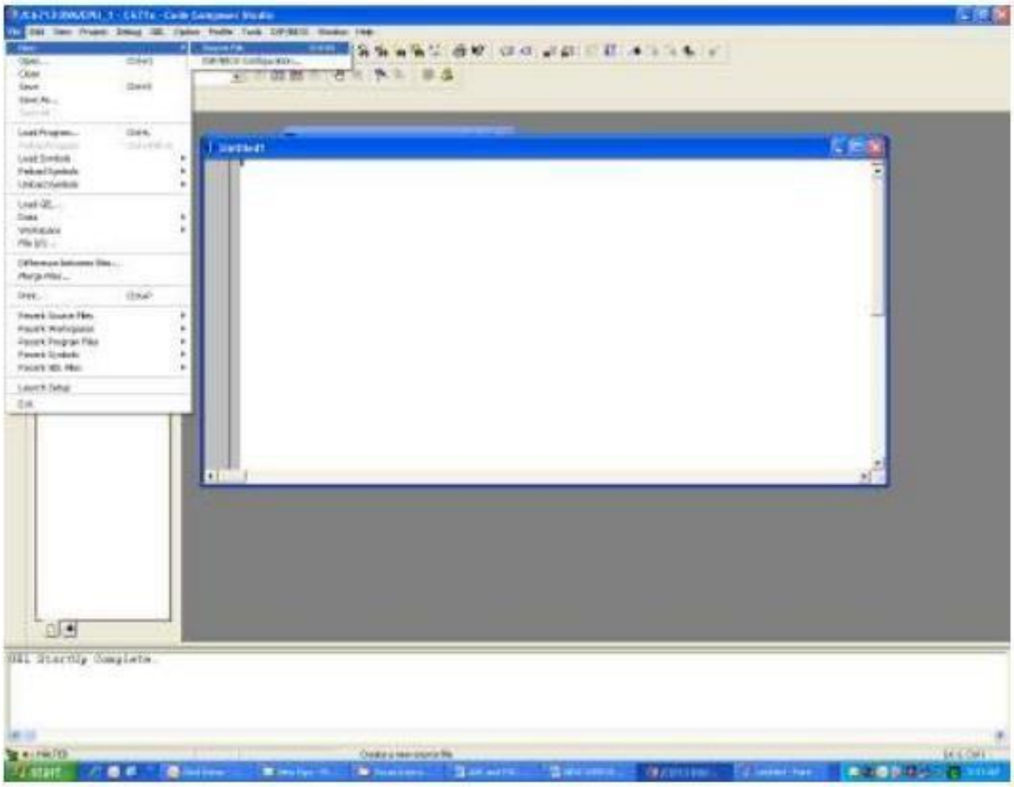
Project Type: Executable (.out)

Target TMS320C67XX

2. To Create a Source file

File → New→ source file.

Files of type → C Source File (*.c, *.ccc)

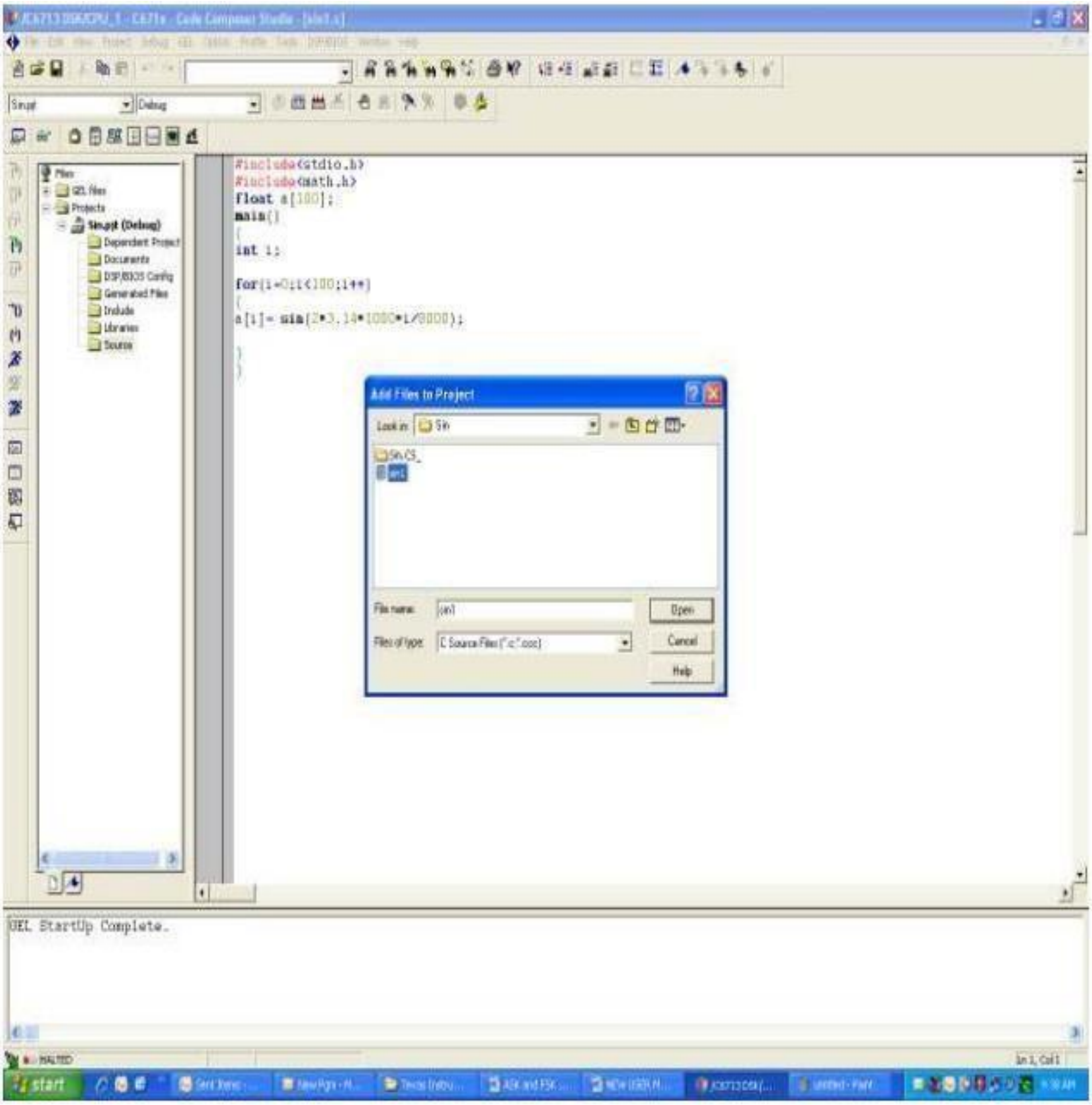


Type the code in editor window.
Save the file in project folder. (Eg: **Sin.c**).

Important note: Save your source code with preferred language extension. For ASM codes save the file as code(.asm) For C and C++ codes code (*.c,*.cpp) respectively.

3. To Add Source files to Project

Project → Add files to Project → Sin.c
Files of type → C Source File (*.c, *.ccc)



```
#include<stdio.h>
#include<math.h>
float a[100];
main()
{
int i;

for(i=0;i<100;i++)
{
a[i]= sin(2*3.14*1000*i/8000);

}
}
```

GEL StartUp Complete.

4. To Add rts6700.lib file .

The run-time-support object libraries do not contain functions involving signals and locale issues. They do contain the following:

- ISO C/C++ standard library
- C I/O library
- Low-level support functions that provide I/O to the host operating system
- Intrinsic arithmetic routines
- System startup routine, _c_int00
- Functions and macros that allow C/C++ to access specific instructionsrts6700.lib

Run-Time-Support (RTS) object libraries for use with little-endian C/C++ code

Project → Add files to Project →rts6700.lib
Files of type → Object and library files (*.o*, *.l*)

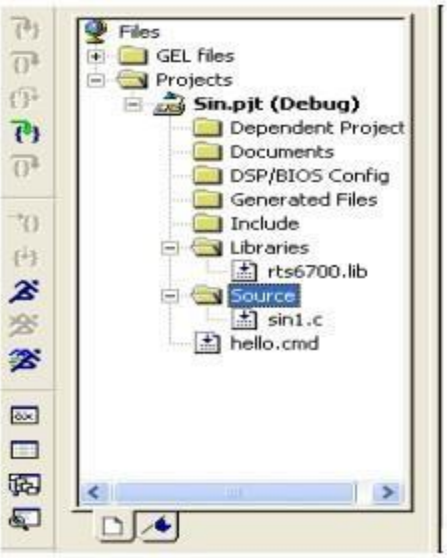
Path: C:\CCStudio_v3.1\c6000\cgtools\lib\rts6700.lib

5. Add Command and linking file

Linker command files allow you to put linking information in a file; this is useful when you invoke the linker often with the same information. Linker command files are also useful because they allow you to **use the MEMORY and SECTIONS directives** to customize your application.

Project → Add files to Project →hello.cmd
Files of type → Linker command file (*.cmd, *.lcf)

Path: C:\CCStudio_v3.1\tutorial\dsk6713\hello1\hello.cmd



5. To Compile:

Project → Compile File

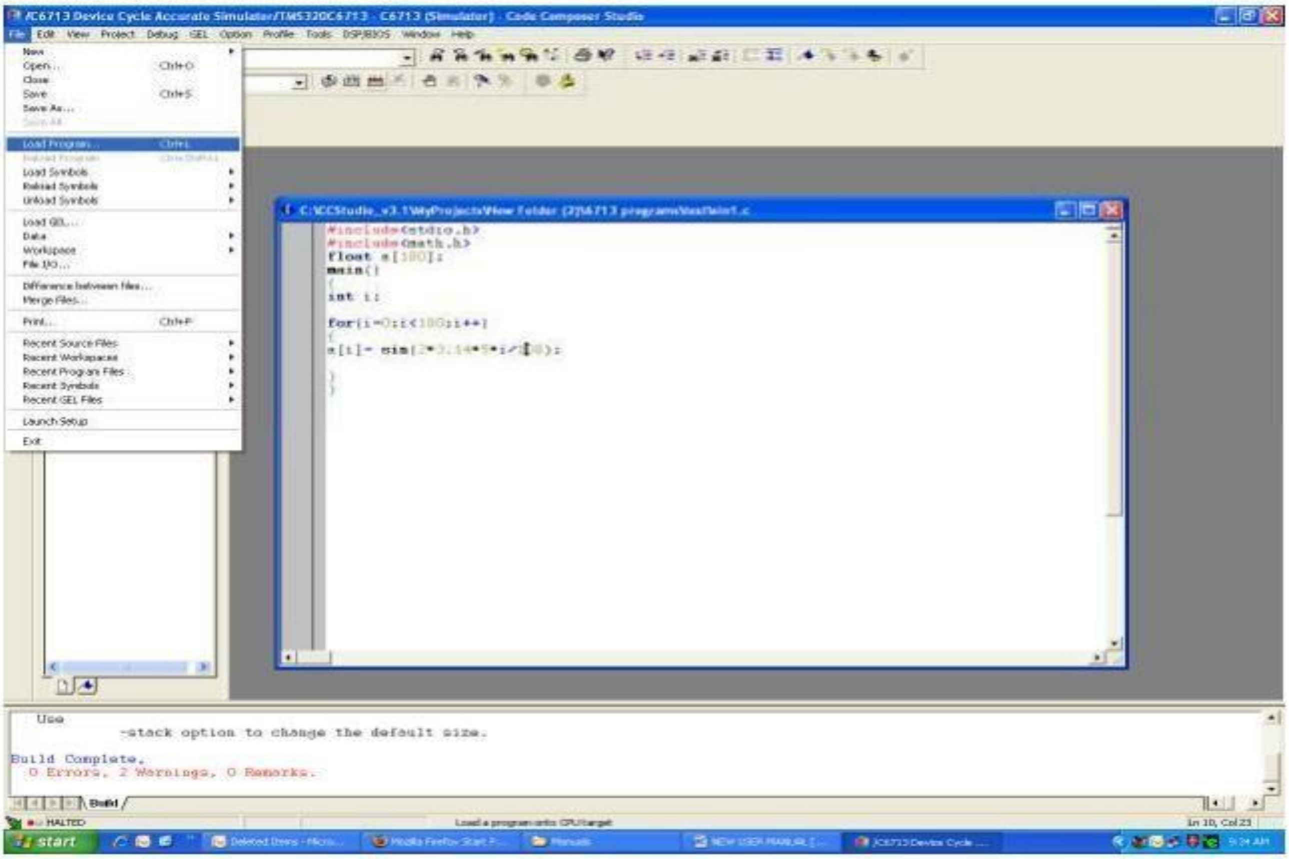
6. To build or Link:

Project → build,

Which will create the final executable **(.out)** file.(Eg. Sin. out).

7. Procedure to Load and Run program:

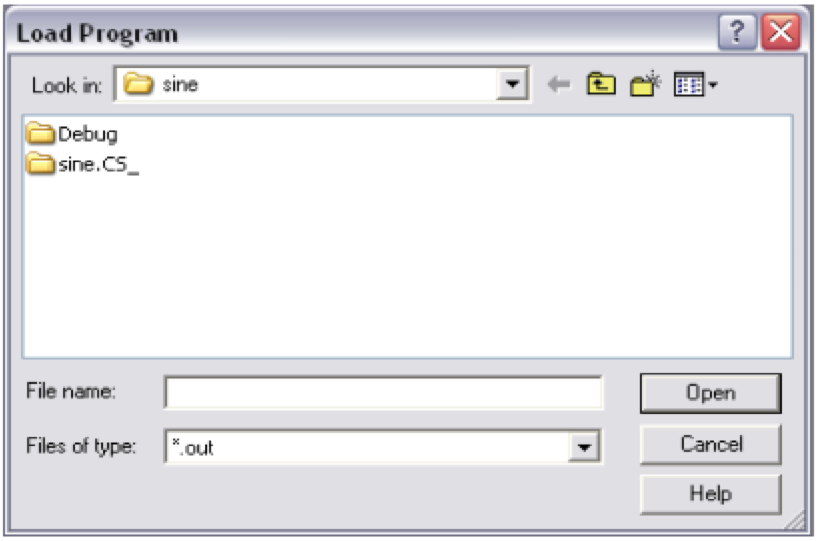
Load program to DSK:


/C6713 Device Cycle Accurate Simulator/TMS320C6713 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
New
Open... Ctrl+O
Close
Save Ctrl+S
Save As...
Save All
Load Program... Ctrl+L
Reload Program Ctrl+Shift+L
Load Symbols
Reload Symbols
Unload Symbols
Load GEL...
Data
Workspace
File I/O...
Difference between files...
Merge Files...
Print... Ctrl+P
Recent Source Files
Recent Workspaces
Recent Program Files
Recent Symbols
Recent GEL Files
Launch Setup
Exit
C:\CCStudio_v3.1\MyProjects\New Folder (2)\6713 programs\test\sin1.c
Build
HALTED
Load a program onto CPU/target
Ln 10, Col 23
start

```
#include<stdio.h>
#include<math.h>
float s[100];
main()
{
int i;

for(i=0;i<100;i++)
{
s[i]= sin(2*3.14*5*i/100);

}
}
```

```
Use
        -stack option to change the default size.

Build Complete,
  0 Errors, 2 Warnings, 0 Remarks.
```

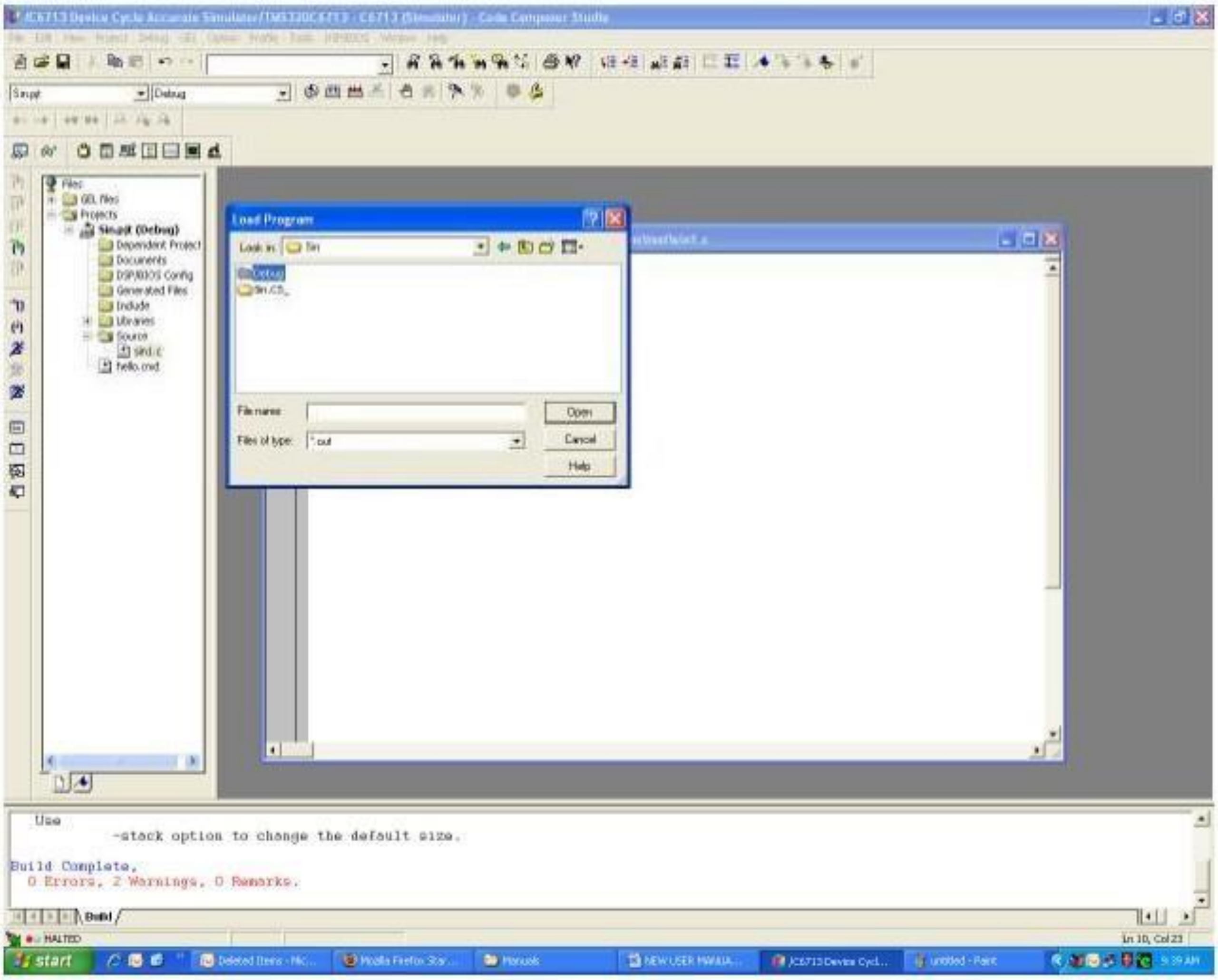
File → Load program → Sin. Out
From **Debug** folder



Look in: sine

File name:

Files of type: *.out



```
  Use
          -stack option to change the default size.

Build Complete,
  0 Errors, 2 Warnings, 0 Remarks.
```

8. To execute project:

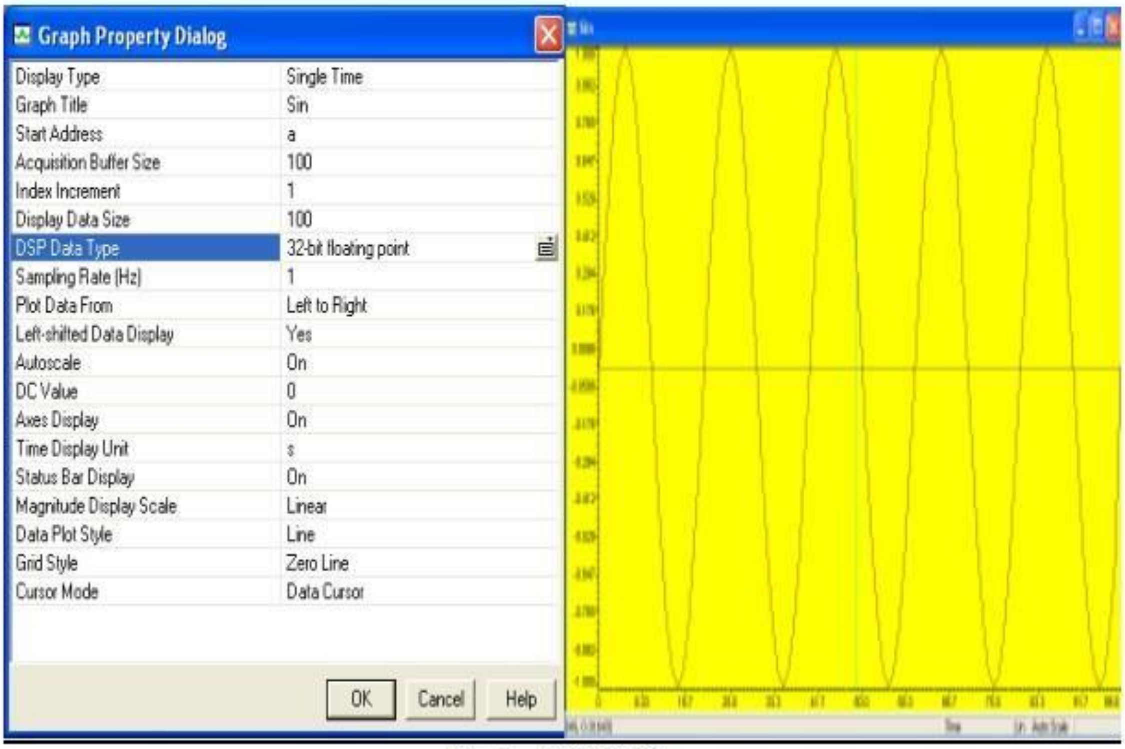
Debug → Run.

Sin.c

```
#include<stdio.h>
#include<math.h>
float a[100];
main()
{
 int i;
for(i=0;i<99;i++)
{
a[i]= sin(2*3.14*5*i/100);
printf("%f",a[i]);
}
}
```

Graph Plotting

1. Goto CCS tool bar **View – Graph –Time and frequency**.
   (In any graph the Start address is the important parameter to be checked.)
   Refer fig.1.for the graph properties



Graph Property Dialog

Display Type	Single Time
Graph Title	Sin
Start Address	a
Acquisition Buffer Size	100
Index Increment	1
Display Data Size	100
DSP Data Type	32-bit floating point
Sampling Rate (Hz)	1
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Time Display Unit	s
Status Bar Display	On
Magnitude Display Scale	Linear
Data Plot Style	Line
Grid Style	Zero Line
Cursor Mode	Data Cursor

Fig. 1 OUTPUT

To Perform Single Step Debugging:

1. Keep the cursor on the on to the line from where u want to start single step debugging. (eg: set a break point on to first line int i=0; of your project.)

   To set **break point** select icon from tool bar menu.

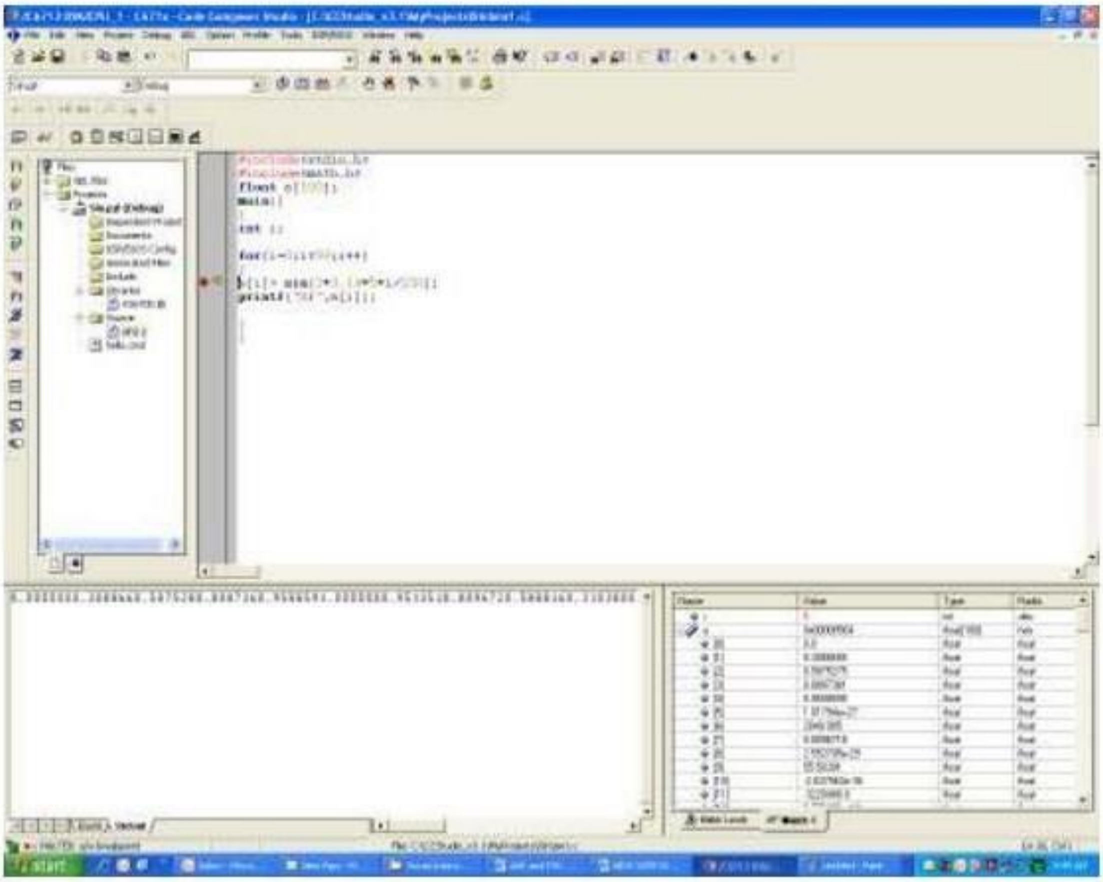
2. Load the Sin**. out** file onto the target.

3. Go to **view → Watch window**.

4. **Debug → Run**.

5. Execution should halt at break point.

6. Press F10 for step over. See the changes happening in the watch window.



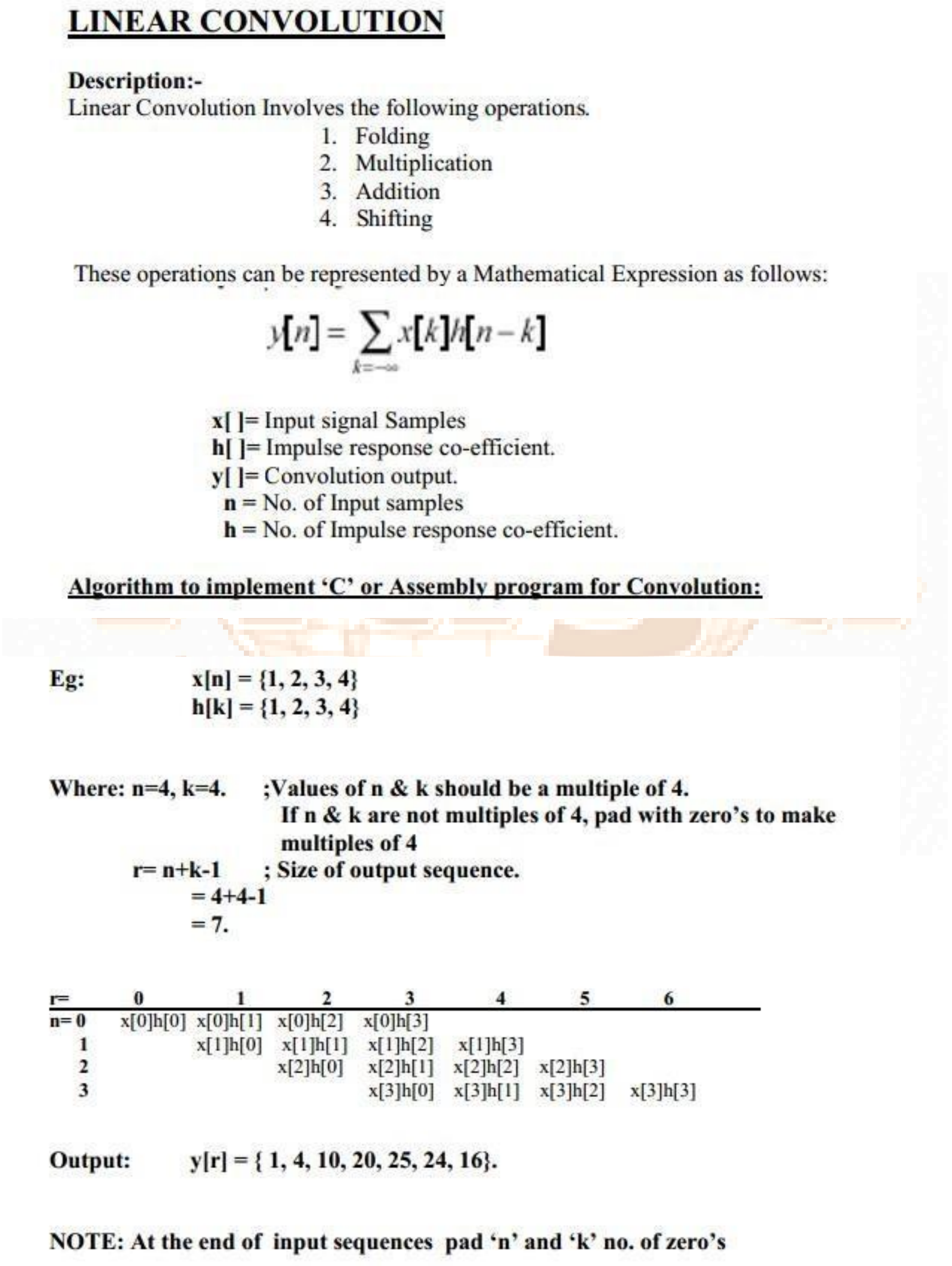
```
#include<stdio.h>
#include<math.h>
float a[100];
main()
{
int i;

for(i=0;i<99;i++)
{
a[i]= sin([illegible]);
printf("%f",a[i]);
}
```

7. Similarly go to view & select CPU registers to view the changes happening in CPU registers.
8. Repeat steps 2 to 6.

CONVOLUTION : LINEAR AND CIRCULAR

Experiment No. 09

AIM: To find linear and circular convolution



LINEAR CONVOLUTION

Description:-

Linear Convolution Involves the following operations.

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

These operations can be represented by a Mathematical Expression as follows:

$$y[n] = \sum_{k=-\infty} x[k]h[n-k]$$

x[ ]= Input signal Samples
h[ ]= Impulse response co-efficient.
y[ ]= Convolution output.
n = No. of Input samples
h = No. of Impulse response co-efficient.

Algorithm to implement 'C' or Assembly program for Convolution:

Eg: **x[n] = {1, 2, 3, 4}**
h[k] = {1, 2, 3, 4}

Where: n=4, k=4. ;Values of n & k should be a multiple of 4.
If n & k are not multiples of 4, pad with zero's to make multiples of 4
r= n+k-1 ; Size of output sequence.
= 4+4-1
= 7.

r=	0	1	2	3	4	5	6
n= 0	x[0]h[0]	x[0]h[1]	x[0]h[2]	x[0]h[3]			
1		x[1]h[0]	x[1]h[1]	x[1]h[2]	x[1]h[3]		
2			x[2]h[0]	x[2]h[1]	x[2]h[2]	x[2]h[3]	
3				x[3]h[0]	x[3]h[1]	x[3]h[2]	x[3]h[3]

Output: y[r] = { 1, 4, 10, 20, 25, 24, 16}.

NOTE: At the end of input sequences pad 'n' and 'k' no. of zero's

C PROGRAM TO IMPLEMENT LINEAR CONVOLUTION

conv.c:

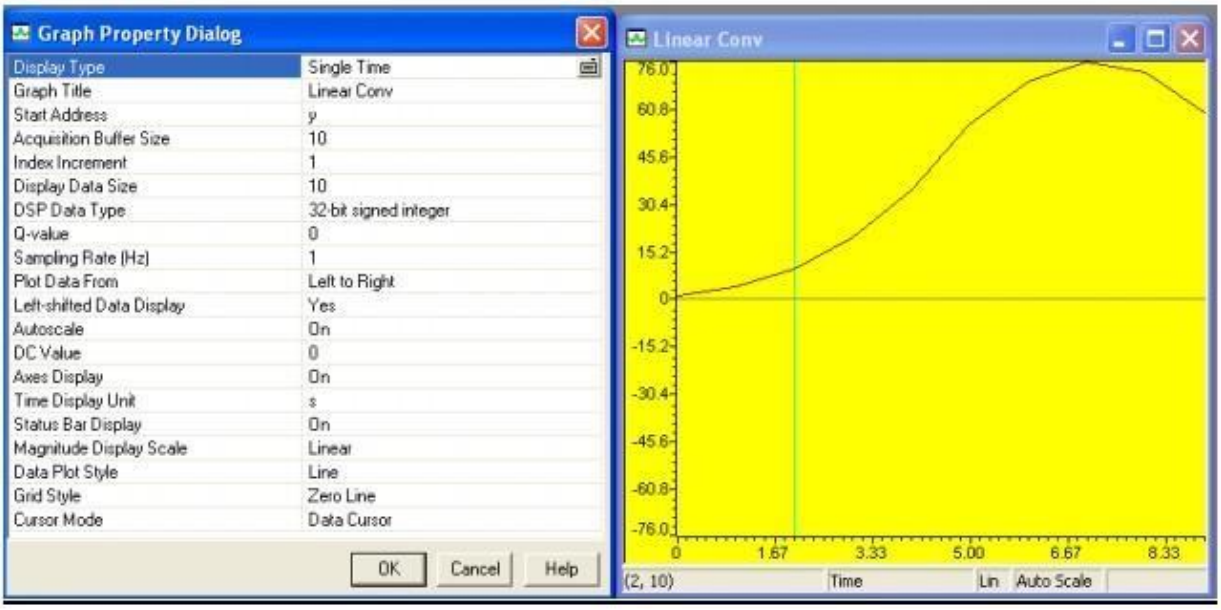
```
 /* prg to implement linear convolution */
#include<stdio.h>
#include<math.h>
int y[20];

main()
{       int m=6;                                        /*Lenght of i/p samples sequence*/
        int n=6;                       /*Lenght of impulse response Co-efficients */
        int i=0,j;
        int x[15]={1,2,3,4,5,6,0,0,0,0,0,0};   /*Input Signal Samples*/
        int h[15]={1,2,3,4,5,6,0,0,0,0,0,0};   /*Impulse Response Co-efficients*/

        for(i=0;i<m+n-1;i++)
        {
        y[i]=0;
        for(j=0;j<=i;j++)

                y[i]+=x[j]*h[i-j];
        }
        for(i=0;i<m+n-1;i++)
        printf("%d\n",y[i]);
}
```

PROCEDURE:

- Open Code Composer Studio, make sure the DSP kit is turned on.
- Use the **Debug → Connect** menu option to open a debug connection to the DSK board
- Start a new project using 'Project-new ' pull down menu, save it in a separate directory(C:\CCStudio_v3.1\myprojects) with name **lconv.pjt.**
- Add the source files **conv.c**
- to the project using 'Project→add files to project' pull down menu.
- Add the linker command file **hello.cmd** .
  (Path: C:\CCStudio_v3.1\tutorial\dsk6713\hello1\hello.cmd)
- Add the run time support library file **rts6700.lib**
  (Path: C:\CCStudio_v3.1\c6000\cgtools\lib\rts6700.lib)
- Compile the program using the '**Project-compile**' pull down menu or by clicking the shortcut icon on the left side of program window.
- Build the program using the '**Project-Build**' pull down menu or by clicking the shortcut icon on the left side of program window.
- Load the program(lconv.out) in program memory of DSP chip using the '**File-load program**' pull down menu.

- To View output graphically
  Select **view → graph → time and frequency**.

Configure the graphical window as shown below



Graph Property Dialog

Display Type	Single Time
Graph Title	Linear Conv
Start Address	y
Acquisition Buffer Size	10
Index Increment	1
Display Data Size	10
DSP Data Type	32-bit signed integer
Q-value	0
Sampling Rate (Hz)	1
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	On
DC Value	0
Axes Display	On
Time Display Unit	s
Status Bar Display	On
Magnitude Display Scale	Linear
Data Plot Style	Line
Grid Style	Zero Line
Cursor Mode	Data Cursor

CIRCULAR CONVOLUTION

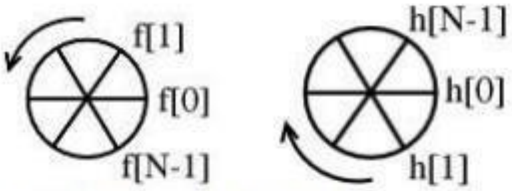
Description:-

Steps for Cyclic Convolution

Steps for cyclic convolution are the same as the usual convolution, except all index calculations are done "mod N" = "on the wheel"

Steps for Cyclic Convolution

Step1: “Plot f[m] and h[−m]



Subfigure 1.1 Subfigure 1.2

Step 2: "Spin" h[−m] *n* times Anti Clock Wise (counter-clockwise) to get h[n-m]
(i.e. Simply rotate the sequence, *h*[*n*], clockwise by *n* steps)

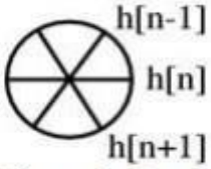
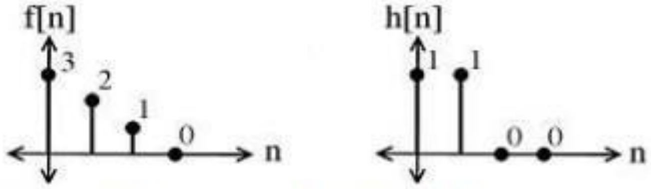


Figure 2: Step 2

Step 3: Point wise multiply the $f[m]$ wheel and the $h[n-m]$ wheel. Sum=$y[n]$

Step 4: Repeat for all $0 \le n \le N-1$

Example 1: Convolve (n = 4)



Subfigure 3.1 Subfigure 3.2

Figure 3: Two discrete-time signals to be convolved.

- $h[-m]$ =

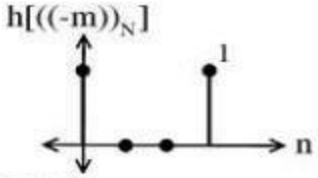


Figure 4

Multiply $f[m]$ and sum to yield: $y[0]$ =3

- $h[1-m]$

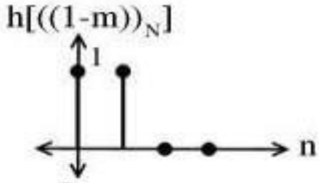


Figure 5

Multiply $f[m]$ and sum to yield: $y[1]$ =5

- $h[2-m]$

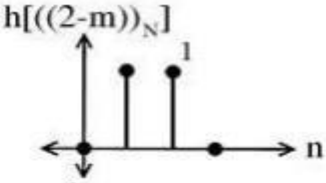


Figure 6

Multiply $f[m]$ and sum to yield: $y[2]=3$

- $h[3-m]$

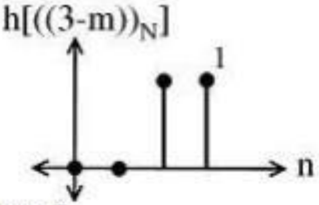


Figure 7

Multiply $f[m]$ and sum to yield: $y[3]=1$

C Program to Implement Circular Convolution

```
#include<stdio.h>
#include<math.h>
int m,n,x[30],h[30],y[30],i,j,temp[30],k,x2[30],a[30];
void main()
{
 printf("  enter the length of the first sequence\n");
 scanf("%d",&m);
printf("  enter the length of the second sequence\n");
scanf("%d",&n);
printf("  enter the first sequence\n");
for(i=0;i<m;i++)
scanf("%d",&x[i]);
printf("  enter the second sequence\n");
for(j=0;j<n;j++)
scanf("%d",&h[j]);
if(m-n!=0)                    /*If length of both sequences are not equal*/
{
      if(m>n)                    /* Pad the smaller sequence with zero*/
      {
      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++)                                 /*folding h(n) to h(-n)*/
a[j]=h[n-j];
  /*Circular convolution*/
for(i=0;i<n;i++)
 y[0]+=x[i]*a[i];
for(k=1;k<n;k++)
{
 y[k]=0;
 /*circular shift*/
 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];
  }
}
/*displaying the result*/
printf("  the circular convolution is\n");
for(i=0;i<n;i++)
printf("%d \t",y[i]);

}
```

IN PUT:

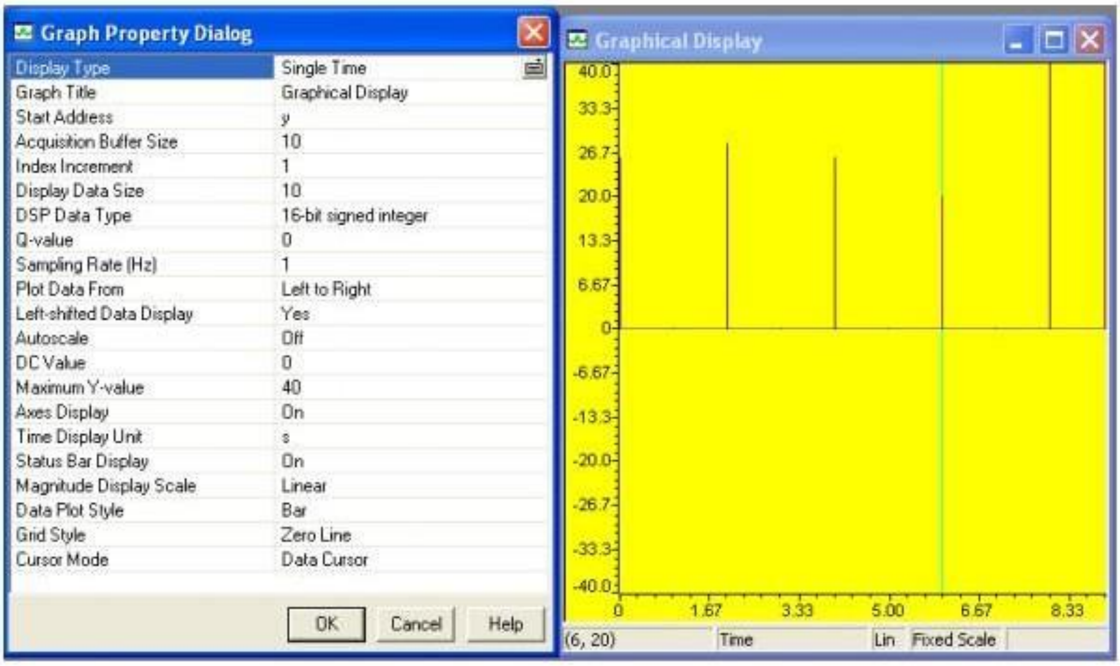
Eg: **x[4]={1, 2, 3,4}**

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

OUT PUT **y[4]={26, 28, 26,20}**

PROCEDURE:

- Open Code Composer Studio; make sure the DSP kit is turned on.
- Start a new project using 'Project-new ' pull down menu, save it in a separate directory(C:\CCStudio_v3.1\myprojects) with name **cir conv.pjt.**
- Add the source files **Circular Convolution**
- to the project using 'Project→add files to project' pull down menu.
- Add the linker command file hello.cmd.
  (Path: C:\CCStudio_v3.1\tutorial\dsk6713\hello1\hello.cmd)
- Add the run time support library file **rts6700.lib**
  (Path: C:\CCStudio_v3.1\c6000\cgtools\lib\rts6700.lib)
- Compile the program using the '**Project-compile**' pull down menu or by clicking the shortcut icon on the left side of program window.
- Build the program using the '**Project-Build**' pull down menu or by clicking the shortcut icon on the left side of program window.
- Load the program(lconv.out) in program memory of DSP chip using the '**File-load program**' pull down menu.



Display Type	Single Time
Graph Title	Graphical Display
Start Address	y
Acquisition Buffer Size	10
Index Increment	1
Display Data Size	10
DSP Data Type	16-bit signed integer
Q-value	0
Sampling Rate (Hz)	1
Plot Data From	Left to Right
Left-shifted Data Display	Yes
Autoscale	Off
DC Value	0
Maximum Y-value	40
Axes Display	On
Time Display Unit	s
Status Bar Display	On
Magnitude Display Scale	Linear
Data Plot Style	Bar
Grid Style	Zero Line
Cursor Mode	Data Cursor

REAL TIME FIR FILTER IMPLEMENTATION (LOW-PASS, HIGH-PASS AND BAND-PASS) BY INPUTTING A SIGNAL FROM THE SIGNAL GENERATOR

Experment No. 10

AIM: To implement FIR filter.

PROCEDURE:-

□ Connect Signal Generator in "Line in"
□ Connect CRO in "Line out"
□ Switch on DSK
□ Debug → connect
□ Create new project and give name as "FIR.pjt"
□ Select File → New → DSP/BIOS Configuration → "dsk6713.cdb" and save it as "xyz.cdb"
□ Add "xyz.cdb" to current project.
    Project → Add files to Project → xyz.cdb
□ Automatically three files are added in the generated file folder
    xyzcfg.cmd
    xyzcfg.562
    xyzcfg_c.c
□ Open File → New → Source file
□ Type the code in editor window. Save file in project folder (eg. Coder.c)
□ Project → Add files to project → Coder.c
□ Add the library file "dsk 6713bsl.lib" to current project.
    Path → C:\ccstudio_v3.1\c6000\dsk6713\lib\dsk6713bsl.lib
□ Copy header files "dsk 6713.h" and dsk6713_aic 23.h" from and paste it in current project folder
    C:\ccstudio_v3.1\c6000\dsk6713\include
□ Add the header file generated within xyzcfg_c.c to FIR.c
□ Compile
□ Build
□ Load Program (coder.out)
□ Debug → Run

CODE:

```
#include"xyzcfg.h"
#include"C:\ccstudio_v3.1\c6000\dsk6713\include\dsk6713.h"
#include "C:\ccstudio_v3.1\c6000\dsk 6713\include\dsk 6713_aic 23.h"
float filter_coeff[ ] = {-0.0001,-0.0003, -0.0004,0,0.0009,0.0019,0.0018,-0,-0.0034,-
0.0064,-0.0059,0,0.0096,0.0171,0.0152,-0,-0.0238,-0.0426,-0.0387,0,0.0711,0.1554,
0.2237,0.25, 0.2237,0.1554,0.0711,0,-0.0387,-0.0426,-0.0238,-0,0.0152,0.0101,
0.0096, 0, -0.0059, -0.0064, -0.0034, -0, 0.0018, 0.0019, 0.009, 0, -0.004, -0.0003, -
0.0001, -0, 0};
static short in_buffer[100];
DSK 6713_AIC23_Configconfig=
{
\0x0017, /*0DSK6713_AIC 23_LEFTINVOL Left line input channel volume*/
\0x0017, /*1DSK6713_AIC 23_RIGHTINVOL Right line input channel volume*/
\0x00d8, /*2DSK6713_AIC 23_LEFTHPVOL Left channel headphone volume*/
\0x00d8, /*3DSK6713_AIC 23_RIGHTHPVOL Right channel headphone volume*/
\0x0011, /*4DSK6713_AIC 23_ANAPATH Analog audio path control*/
\0x0000, /*5DSK6713_AIC 23_DIGPATH Digital audio path control*/
\0x0000, /*6DSK6713_AIC 23_POWER DOWN Power down control*/
\0x0043, /*7DSK6713_AIC 23_DIGIF Digital audio interface format*/
\0x0081, /*8DSK6713_AIC 23_SAMPLERATE Sample rate control*/
\0x0001, /*9DSK6713_AIC 23_DIGACT Digital Interface Activator*/
\};
/*main()-main code routine, initialize BSL*/
void main()
{
DSK6713_AIC23_CodecHandlehCodec;
Uint32 l_input,r_input,l_output,r_output;
/* Initialize the board support library, must be called first*/
DSK6713_init();
/*start the codec*/
hCodec= DSK6713_AIC23_opencodec(0,&config);
DSK6713_AIC23_setFreq(hCodec,1);
While(1)
{/*read a sample to the left channel*/
while(!DSK6713_AIC23_read(hCodec, &l_input));
/*read a sample to the right channel*/
while(!DSK6713_AIC23_read(hcodec,&r_input));
l_output= (Int16)FIR_FILTER(&filter_coeff,l_input);
r_output=(Int16)FIR_FILTER(&filter_coeff,r_input);
/*send a sample to the left channel*/
while(!DSK6713_AIC23_write(hCodec, l_output));
/*send a sample to the right channel*/
while(!DSK6713_AIC23_write(hcodec, r_output));
}
/*close the codec*/
DSK6713_AIC23_close codec(hcodec);
}
signed int FIR_FILTER(float*h,signedintx)
{
int i=0;
Signed long output=0;
in_buffer[0]=x; /*new input at buffer[0]*/
```

```
for(i=51; i>0; i--)
in_buffer[i]=in_buffer[i-1]; /*shuffle the buffer*/
for(i=0;i<51;i++)
output=output+h[i]*in_buffer[i];
return(output);
}
```

Observation

The connection from DSK board to the PC, signal generator and DSO is physically established. The frequency of the function generator is varied and the output signal are observed on the DSO. The attenuation caused by the filter is observed for different frequencies.

RESULT: Implemented FIR filter using code composer studio

REAL TIME IIR FILTER IMPLEMENTATION (LOW-PASS, HIGH-PASS AND BAND-PASS) BY INPUTTING A SIGNAL FROM THE SIGNAL GENERATOR

Experment No. 11

AIM: To implement IIR filter.

PROCEDURE:-

□ Connect Signal Generator in "Line in"
□ Connect CRO in "Line out"
□ Switch on DSK
□ Debug → connect
□ Create new project and give name as "IIR.pjt"
□ Select File → New → DSP/BIOS Configuration → "dsk6713.cdb"
and save it as "xyz.cdb"
□ Add "xyz.cdb" to current project.
Project → Add files to Project → xyz.cdb
□ Automatically three files are added in the generated file folder
xyzcfg.cmd
xyzcfg.562
xyzcfg_c.c
□ Open File → New → Source file
□ Type the code in editor window. Save file in project folder (eg. Coder.c)
□ Project → Add files to project → Coder.c
□ Add the library file "dsk 6713bsl.lib" to current project.
Path → C:\ccstudio_v3.1\c6000\dsk6713\lib\dsk6713bsl.lib
□ Copy header files "dsk 6713.h" and dsk6713_aic 23.h"
from and paste it in current project folder
C:\ccstudio_v3.1\c6000\dsk6713\include
□ Add the header file generated within xyzcfg_c.c to FIR.c
□ Compile
□ Build
□ Load Program (coder.out)
□ Debug → Run

CODE:

```
#include"xyzcfg.h"
#include"C:\ccstudio_v3.1\c6000\dsk6713\include\dsk6713.h"
#include "C:\ccstudio_v3.1\c6000\dsk 6713\include\dsk 6713_aic 23.h"
const signed int filter_coeff[ ] = {//12730,-12730,12730,2767,-18324,21137
/*HP2500*/
                                    //312,312,312,32767,-27943,24367 /*LP800*/
```

```
                                   //1455,1455,1455,32767,-23140,21735
/*LP2500*/
                                   //9268,-9268,9268,32767,-7395,18367
/*HP4000*/
                                   //7215,-7215,32767,5039,6171, /*HP7000*/};
/*codec configuration settings*/
DSK 6713_AIC23_Configconfig=
{
\0x0017, /*0DSK6713_AIC 23_LEFTINVOL Left line input channel volume*/
\0x0017, /*1DSK6713_AIC 23_RIGHTINVOL Right line input channel volume*/
\0x00d8, /*2DSK6713_AIC 23_LEFTHPVOL Left channel headphone volume*/
\0x00d8, /*3DSK6713_AIC 23_RIGHTHPVOL Right channel headphone volume*/
\0x0011, /*4DSK6713_AIC 23_ANAPATH Analog audio path control*/
\0x0000, /*5DSK6713_AIC 23_DIGPATH Digital audio path control*/
\0x0000, /*6DSK6713_AIC 23_POWER DOWN Power down control*/
\0x0043, /*7DSK6713_AIC 23_DIGIF Digital audio interface format*/
\0x0081, /*8DSK6713_AIC 23_SAMPLERATE Sample rate control*/
\0x0001, /*9DSK6713_AIC 23_DIGACT Digital Interface Activator*/
\};
/*main()-main code routine, initialize BSL and generate tone*/
void main()
{
DSK6713_AIC23_CodecHandlehCodec;
int l_input,r_input,l_output,r_output;
/* Initialize the board support library, must be called first*/
DSK6713_init();
/*start the codec*/
hCodec= DSK6713_AIC23_opencodec(0,&config);
DSK6713_AIC23_setFreq(hCodec,3);
While(1)
{/*read a sample to the left channel*/
while(!DSK6713_AIC23_read(hCodec, &l_input));
/*read a sample to the right channel*/
while(!DSK6713_AIC23_read(hcodec,&r_input));
l_output= IIR_FILTER(&filter_coeff,l_input);
r_output=IIR_FILTER(&filter_coeff,r_input);
/*send a sample to the left channel*/
while(!DSK6713_AIC23_write(hCodec, l_output));
/*send a sample to the right channel*/
while(!DSK6713_AIC23_write(hcodec, r_output));
}/*close the codec*/
DSK6713_AIC23_close codec(hcodec);
}
//Implementation of IIR filter
signed int IIR_FILTER(const signed int*h,signedintx1)
{
```

```
static signed int x[6]={0,0,0,0,0,0}; /*x(n),x(n-1),x(n-2) must be states*/
static signed int y[6]={0,0,0,0,0,0}; /*y(n),y(n-1),y(n-2) must be states*/
int temp=0
temp=(short int)x1; /*copy input to temp*/
x[0]=(signed int)temp; /*copy input to x[stages][0]*/
temp=((int)h[0]*x[0]); /*B0*x(n)*/
temp+=((int)h[1]*x[1]);/*B1/2*x(n-1)*/
temp+=((int)h[1]*x[1]);/*B1/2*x(n-1)*/
temp+=((int)h[2]*x[2]); /*B2*x(n-2)*/
temp-=((int)h[4]*y[1]);/*A1/2*y(n-1)*/
temp-=((int)h[4]*y[1]);/*A1/2*y(n-2)*/
temp-=((int)h[5]*y[2]);/*A2*y(n-2)*/
/*divide temp by coefficient [A0]*/
temp>>=15;
if(temp>32767)
{
temp=32767;
}
else if (temp<-32767)
{
temp=-32767;
}
y[0]=temp;
/*shuffle values along one place for next time*/
y[2]=y[1];/*y(n-2)=y(n-1)*/
y[1]=y[0];/*y(n-1)=y(n)*/
x[2]=x[1];/*x(n-2)=x(n-1)*/
x[1]=x[0];/*x(n-1)=x(n)*/
/*temp is used as input next time through*/
return(temp<<2);
}
```

Observation

The connection from DSK board to the PC, signal generator and DSO is physically established. The frequency of the function generator is varied and the output signal are observed on the DSO. The attenuation caused by the filter is observed for different frequencies.

RESULT: Implemented IIR filter using code composer studio

SAMPLING OF ANALOG SIGNAL AND STUDY OF ALIASING.

Experiment No. 12

AIM: To analyze the sampling of analog signal and study of aliasing.

PROCEDURE:-

□ Open MATLAB
□ Open new M-file
□ Type the program
□ Save in current directory
□ Compile and Run the program
□ For the output see command window\ Figure window

ALGORITHM:-

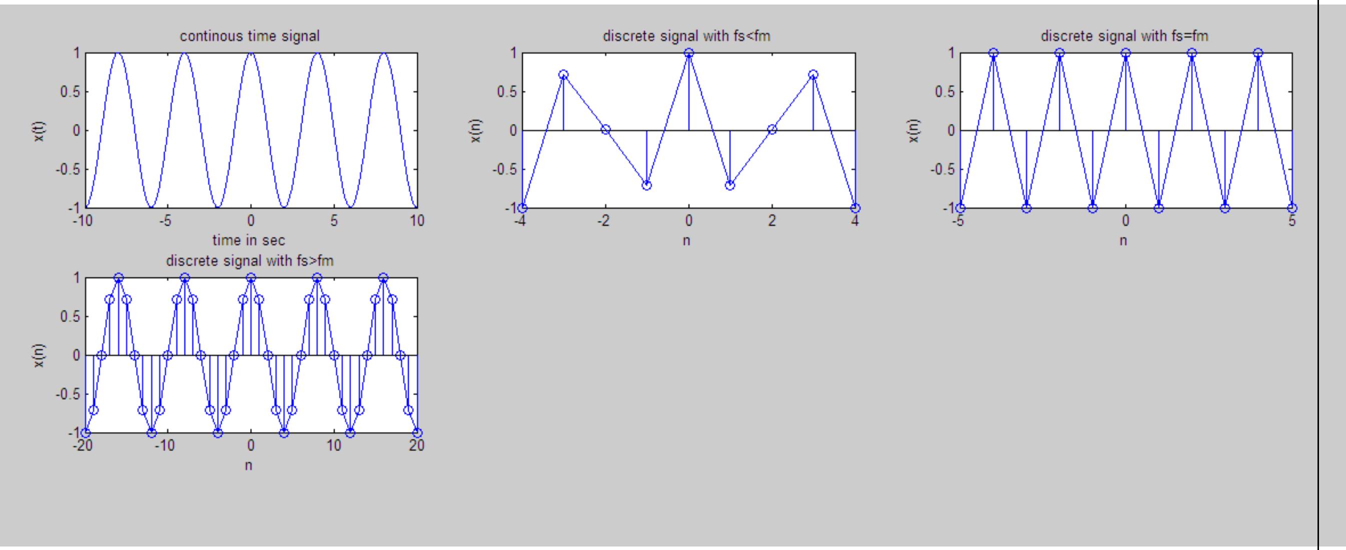
□ Get the amplitude and frequency of the signal
□ Use „sin‟ matlab built in functions
□ Using „plot‟ function plot the signal

MATLAB CODE:-

```
clc;
t=-10:.01:10;
T=4;
fm=1/T;
x=cos(2*pi*fm*t);
fs1=1.6*fm;
fs2=2*fm;
fs3=8*fm;
n1=-4:1:4;
xn1=cos(2*pi*n1*fm/fs1);
subplot(331);
plot(t,x);
xlabel('time in sec');
ylabel('x(t)');
title('continous time signal');

subplot(332);
stem(n1,xn1);
hold on;
subplot(332);
plot(n1,xn1);
xlabel('n');
ylabel('x(n)');
title('discrete signal with fs<fm');

n2=-5:1:5;
xn2=cos(2*pi*n2*fm/fs2);
subplot(333);
stem(n2,xn2);
```

```
hold on;
subplot(333);
plot(n2,xn2);
xlabel('n');
ylabel('x(n)');
title('discrete signal with fs=fm');

n3=-20:1:20;
xn3=cos(2*pi*n3*fm/fs3);
subplot(334);
stem(n3,xn3);
hold on;
subplot(334);
plot(n3,xn3);
xlabel('n');
ylabel('x(n)');
title('discrete signal with fs>fm');
```

FIGURE:



RESULTS: - Thus the program to find the concept of Aliasing is written using MATLAB and verified.

VIVA QUESTIONS AND ANSWERS

1. What is MATLAB?

   MATLAB(matrix laboratory) is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include: Data analysis, exploration, and visualization.

2. What are the applications of MATLAB?

   - Math and computation
   - Algorithm development
   - Modeling, simulation, and prototyping
   - Data analysis, exploration, and visualization
   - Scientific and engineering graphics
   - Application development, including Graphical User Interface building

3. State sampling theorem.

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

4. What is meant by Nyquist rate and Nyquist criteria?

   The **Nyquist frequency** should not be confused with the **Nyquist rate**, which is the minimum sampling **rate**that satisfies the **Nyquist** sampling **criterion** for a given signal or family of signals. The **Nyquist rate** is twice the maximum component **frequency** of the function being sampled.

5. Explain scaling and superposition properties of a system.

   OR

6. What is meant by linearity of a system and how it is related to scaling and superposition?

   The *scaling* property of linear systems states that scaling the input of a linear system (multiplying it by a constant gain factor) scales the output by the same factor. The *superposition* property of linear systems states that the response of a linear system to a sum of signals is the sum of the responses to each individual input signal. Another view is that the individual signals which have been summed at the input are processed independently inside the filter--they superimpose and do not interact. (The addition of two signals, sample by sample, is like converting stereo to mono by mixing the two channels together equally.)

7. What is impulse function?

OR

8. What is meant by impulse response?

In science and mathematics, the Dirac **delta function**, or δ **function**, is a generalized **function**, or distribution that was historically introduced by the physicist Paul Dirac for modelling the density of an idealized point mass or point charge, as a **function** that is equal to zero everywhere except for zero

9. What is energy signal? How to calculate energy of a signal?

Energy signals have values only in the limited time duration, a **signal** having only one square pulse is**energy signal**, A **signal** that decays exponentially has finite **energy**, so, it is also an **energy signal**, The power of an **energy signal** is zero, because of dividing finite **energy** by infinite time.

10. What is power signal? How to calculate power of a signal?

Those signals which have infinite energy and finite power known as power signal.

11. Differentiate between even and odd signals.

Even Signal:

A signal is referred to as an even if it is identical to its time-reversed counterparts;

x(t) = x(-t).

Odd Signal:

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

An odd signal must be 0 at t=0, in other words, odd signal passes the origin.

12. Explain time invariance property of a system.

A **time-invariant** (TIV) system has a **time**-dependent system function that is not a direct function of **time**. In the language of signal processing, this **property** can be satisfied if the transfer function of the system is not a direct function of **time** except as expressed by the input and output.

13. What is memory less system?

The output signal at each time depends only on the input at that time. Such **systems** are said to be **memoryless** because you do not have to remember previous values (or future values, for that matter) of the input in order to determine the current value of the output.

14. When a system is said to have memory?

The output signal at each time depends on the previous output. Such **systems** are said to be **memory** because you have to remember previous values (or future values, for that matter) of the input in order to determine the current value of the output.

15. What is meant by causality?

    A **causal system** (also known as a physical or nonanticipative**system**) is a **system** where the output depends on past and current inputs but not future inputs

16. Explain linear convolution and circular convolution.

    Linear convolution is the basic operation to calculate the output for any **linear** time invariant system given its input and its impulse response. **Circular convolution** is the same thing but considering that the support of the signal is periodic (as in a circle, hance the name).

17. What is the length of linear and circular convolutions if the two sequences are having the length n1 and n2?

    Linear Convolution N= n1 + n2 -1

    Circular Convolution N= max(n1,n2)

18. What are Fourier series and Fourier transform?

    The **Fourier series** is used to represent a periodic function by a discrete sum of complex exponentials, while the **Fourier transform** is then used to represent a general, nonperiodic function by a continuous superposition or integral of complex exponentials.

19. What are the advantages and special applications of Fourier transform, Fourier series, Z transform and Laplace transform?

    The Laplace and Fourier transforms are *continuous* (integral) transforms of continuous functions.

    The Laplace transform maps a function f(t)f(t) to a function F(s)F(s) of the complex variable *s*, where s=σ+jωs=σ+jω.

    Since the derivative f˙(t)=df(t)dtf˙(t)=df(t)dt maps to sF(s)sF(s), the Laplace transform of a linear differential equation is an algebraic equation. Thus, the Laplace transform is useful for, among other things, solving linear differential equations.

    If we set the real part of the complex variable *s* to zero, σ=0σ=0, the result is the Fourier transform F(jω)F(jω) which is essentially the *frequency domain representation* of f(t)f(t) (note that this is true only if for that value of σσ the formula to obtain the Laplace transform of f(t)f(t) exists, i.e., it does not go to infinity).

    The Z transform is essentially a discrete version of the Laplace transform and, thus, can be useful in solving *difference* equations, the discrete version of *differential*equations. The Z transform maps a sequence f[n]f[n] to a continuous function F(z)F(z) of the complex variable z=rejΩz=rejΩ.

If we set the magnitude of *z* to unity, r=1r=1, the result is the Discrete Time Fourier Transform (DTFT) F(jΩ)F(jΩ) which is essentially the frequency domain representation of f[n]f[n].

20. Differentiate between DTFT and DFT. Why it is advantageous to use DFT in computers rather than DTFT?

    In DTFT, frequency appears to be continuous. But, in DFT, frequency is discrete. This property is useful for computation in computers.

21. How to perform linear convolution using circular convolution?

    If two signals x (n) and y (n) are of length n1 and n2, then the linear convoluted output z (n) is of length n1+n2-1. Each of the input signals is padded with zeros to make it of length n1+n2-1. Then circular convolution is done on zero padded sequences to get the linear convolution of original input sequences x (n) and y (n).

22. What is meant by correlation?

    Correlation is the measure of similarity between two signal/waveforms. It compares the waveforms at different time instants.

23. What is auto-correlation?

    It is a measure of similarity of similarity of a signal/waveform with itself.

24. What is cross-correlation?

    In signal processing, **cross-correlation** is a measure of similarity of two series as a function of the displacement of one relative to the other. This is also known as a sliding dot product or sliding inner-product. It is commonly used for searching a long signal for a shorter, known feature.

25. What are the advantages of using autocorrelation and cross correlation properties in signal processing fields?

    In wireless communications we use cross correlation between a known preamble sequence and the received signal to detect the start of a transmission. It's also useful to look at the autocorrelation of the sequence when selecting/designing it. If it has a “peaked” autocorrelation, I.e. one large value and all the rest small, then it is a good sequence to use for transmission preamble because it produces a distinct peak in your detector.

26. How auto-correlation can be used to detect the presence of noise?

    The auto-correlation function can be used to detect repeats or periodicity in a signal. Here, we use the auto-correlation to assess the effect of fluctuations (noise) on a periodic signal.

    In absence of noise, the auto-correlation function oscillates with a constant amplitude and a maximum of 1. The period of the auto-correlation correspond to the period of the signal. In presence of noise, the envelop of the auto-correlation function decreases exponentially.

More important is the noise, faster is this decreasing. Use of the auto-correlation function to quantify the effect of noise on a periodic signal This phenomenon is also called "phase diffusion"

27. Differentiate between IIR filters and FIR filters.

Advantages	
FIR	IIR
▪ Stable ▪ Highly precise ▪ Finite duration impulse response ▪ Excellent phase response ▪ The word-size effect such as round-off noise and coefficient quantization errors are much less severe in FIR.	▪ cost lesser ▪ Faster computations ▪ Less hardware, computations ▪ Easier to design ▪ Lower order required

Disadvantages	
FIR	IIR
▪ Require higher order ▪ Increased hardware ▪ More computations ▪ Larger input and output delays ▪ Cost more	▪ Sensitive to data round off and cutoff ▪ Make become unstable ▪ Poor phase response

28. What is the procedure to design a digital Butterworth filter?

Design Steps of Butterworth Filter

1. Convert the filter specifications to their equivalents in the lowpass prototype frequency. .ε

2. From Ap determine the ripple factor

3. From As determine the filter order, N.

4. Determine the left-hand poles, using the equations given.

5. Construct the lowpass prototype filter transfer function.

6. Use the frequency transformation to convert the LP prototype filter to the given specifications.

29. What is the difference between Butterworth, Chebyshev I and Chebyshev II filters?

Magnitude response vs frequency curve: The magnitude response |H(jw)| of the butterworth filter decreases with increase in frequency from 0 to infinity, while the magnitude response of the Chebyshev filter fluctuates or show ripples in the passband and stopband depending on the type of the filter.

Width of Transition band: The width of the transition band is more in Butterworth filter compared to the Chebyshev filter.

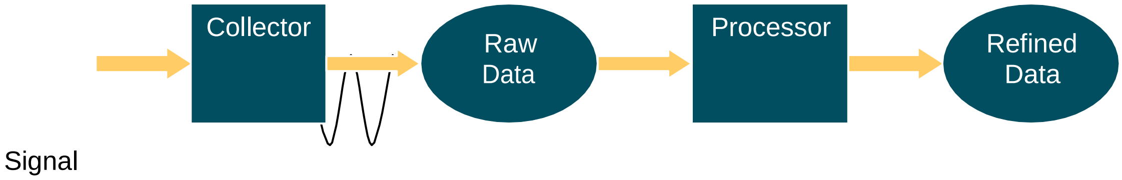
Location of the poles: The poles of a Butterworth filter lies only on a circle while that of the Chebyshev filter lies on an ellipse, which can be easily concluded on looking at the poles formula for both types of filters.

No. Of Components required for implementing the filter: The number of poles in Butterworth filter is more compared to that of the Chebyshev filter of same specifications, this means that the order of Butterworth filter is more than that of a Chebyshev filter. This fact can be used for practical implementation, since the number of components required to construct a filter of same specification can be reduced significantly.

30. What are difference equations and differential equations?

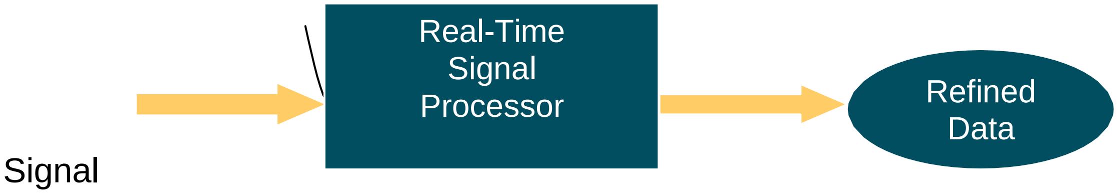
In the one-dimensional case, difference equations are the discrete-time analogue of differential equations. The same basic relationship holds in higher dimensions, but it's not quite so easy to state. Differential equation involves derivatives of function. Difference equation involves difference of terms in a sequence of numbers.

31. What is non real time processing?



32. What is meant by real time processing?

- Ability to collect, analyze, and modify signals in real-time
- Real-Time: As these signals are occurring
- We can analyze and process signals while collecting them, not at a later time.



33. What is a Digital Signal Processor (DSP)?

Microprocessor specifically designed to perform fast DSP operations (e.g., Fast Fourier Transforms, inner products, Multiply & Accumulate)

- Good at arithmetic operations (multiplication/division)

- Mostly programmed with Assembly and C through Integrated Development Environment (IDE)

34. Differentiate between RISC and CISC architectures.

RISC	Emphasis on software	Single-clock, reduced instruction only	large code size	Better C compilers
CISC	Emphasis on hardware	Includes multi-clock complex instructions	Small code sizes	Poor C compilers

35. Differentiate between General purpose MPU(Micro Processor Unit) and DSP Processor

MPU are built for a range of general-purpose functions such as:

Data manipulation

Math calculations

Control systems

They run large blocks of software

They are used in real-time and in unreal-time systems

DSPs are single-minded, dedicated to:

Perform mathematical calculations

Small blocks of software

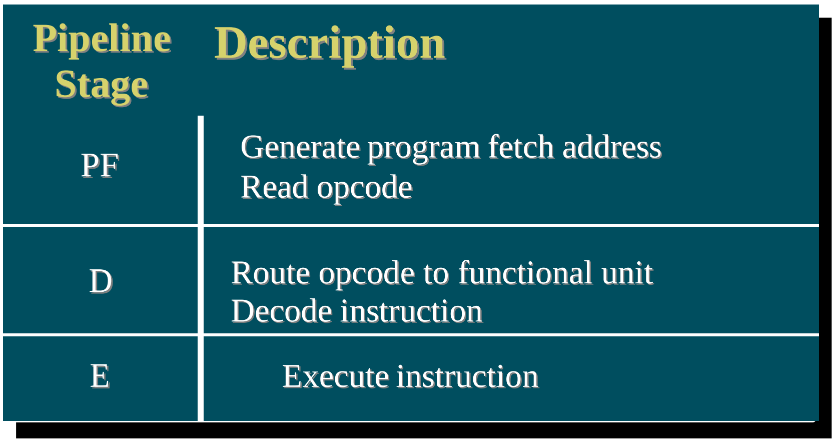
Have a predictable execution time

Real-time only

Could assist a general-purpose host MPU

Microprocessor	**DSP**
General purpose	Arithmetic
Fixed internal format	Varying internal format
Single memory access	Multiple memory access
General addressing mode	Special addressing mode
Very large external memory	Very large internal memory

36. What is pipelining?



Pipeline Stage	Description
PF	Generate program fetch address Read opcode
D	Route opcode to functional unit Decode instruction
E	Execute instruction

37. What is parallel processing?

A mode of operation in which a process is split into parts, which are executed simultaneously on different processors attached to the same computer.

38. What is MAC?

The multiply–accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator.

39. What is barrel shifter?

A **barrel shifter** is a digital circuit that can shift a data word by a specified number of bits without the use of any sequential logic, only pure combinatorial logic.

40. Differentiate between floating point DSP and fixed point DSP.

Fixed Point/Floating Point

- fixed point processor are :
    - i. cheaper
    - ii. smaller
    - iii. less power consuming
    - iv. Harder to program
        - 1. Watch for errors: truncation, overflow, rounding
    - v. Limited dynamic range
    - vi. Used in 95% of consumer products
- floating point processors
    - i. have larger accuracy
    - ii. are much easier to program
    - iii. can access larger memory
    - iv. It is harder to create an efficient program in C on a fixed point processors than on floating point processors

Floating Point

Applications

- Modems
- Digital Subscriber Line (DSL)
- Wireless Base stations
- Digital Imaging
- 3D Graphics
- Speech Recognition
- Voice over IP

Fixed Point

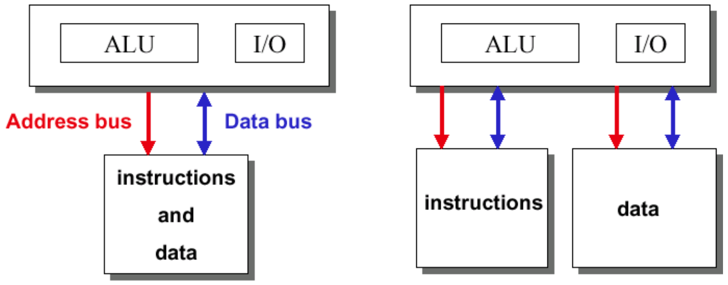
Applications

- Portable Products
- 2G, 2.5G and 3G Cell Phones
- Digital Audio Players
- Digital Still Cameras
- Voice Recognition
- Headsets
- Fingerprint Recognition

41. What is code composer studio?

Code Composer Studio (CCStudio or CCS) is an integrated development environment (IDE) to develop applications for Texas Instruments (TI) embedded processors.

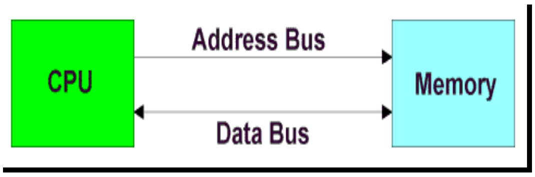
42. Explain Von-Neumann and Harvard architectures



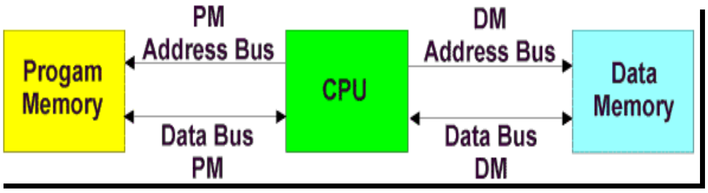
Von Neuman architecture
Area efficient but requires higher bus bandwidth because instructions and data must compete for memory.

Harvard architecture **was coined to describe machines with separate memories.**
Speed efficient: Increased parallelism.

- **Von Neumann Architecture** : Single memory shared by both the program instructions and data



- **Harvard Architecture :** Two separate memories, a program memory (PM) for instructions, and a data memory (DM) for data



43. What are Line-in, Line-out, Mic-in, Mic-out?

Line in will usually be stereo and about 10Kohm impedance.

Mic in is mono and about 600-1Kohm impedance and expecting SIGNIFICANTLY lower levels than the line, as it has a preamp to pick up the very low levels present in a microphone.

Microphone level is in the region of -60 dBV (0.001 volt) to -40 dBV (0.010 volt).

A mic-level or microphone-level signal is the voltage level that comes out of a microphone when someone speaks into it, typically just a few ten-thousandths of a volt. Of course, this voltage varies in response to changes in voice level and and in the talker-to-mic distance. But the signal is still quite small.

Line level is in the region of 0 dBV (1.000 volt).

A line-level signal is approximately one volt, or about 1,000 times greater than a mic-level signal. Connecting a microphone to a line-level input will result in almost no sound at all because the mic signal is so faint that the line input cannot hear it.

44. Define discrete time and digital signal.

Discrete time signal is continuous in amplitude and discrete in time, where Digital signal is discrete in time and amplitude.

45. Explain briefly, the various methods of representing discrete time signal

Graphical, Tabular, Sequence, Functional representation

46. Define sampling and aliasing.

Converting a continuous time signal into discrete time signal is called as Sampling, Aliasing is an effect that causes different signals to become indistinguishable.

47. What is Nyquist rate?

Its the sampling frequency which is equal to twice of Continuous time signal which has to be sampled.

48. State sampling theorem.

It states that , To reconstruct the continuous time signal from its Discrete time signal, The sampling frequency should be more than twice of continuous time signal frequency.

49. Express the discrete time signal x(n) as a summation of impulses.

$$\sum_{k=-\infty}^{\infty} \delta(n-k)x(k)$$

50. How will you classify the discrete time signals?

Causal and Non causal, Periodic and non periodic, even and odd, energy and power signals

51. When a discrete time signal is called periodic?

    If some set of samples repeats after a regular interval of time then its called as periodic.

52. What is discrete time system?

    If a system's excitation and responses are both discrete time signals then its called as discrete time system.

53. What is impulse response? Explain its significance.

    The response of a system when the excitation is Impulse signal is called as impulse response. it also called as Natural response, free forced response.

54. Write the expression for discrete convolution.

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

55. Classifying discrete time systems.

    Causal, Non causal, time variant, time invariant, Linear, non linear, stable and unstable system.

56. Define time invariant system.

    If a system's operation is independent of time then its time invariant, i.e delayed system response is equal to system's response for delayed input.

57. What is linear and nonlinear systems?

    If a system satisfies homogeneity principle and superposition principle then it is Linear. if not Non linear.

58. What is the importance of causality?

    causality states that system's response should depend on present and past inputs only not on the future inputs. so causal systems are realizable.

59. What is BIBO stability? What is the condition to be satisfied for stability?

    If a system's response is Bounded for Bounded excitation then its BIBO stable. For stable system the impulse response should be absolutely sum able .

60. What are FIR and IIR systems?

    FIR: system's impulse response contains finite no. of samples

    IIR: system's impulse response contains infinite no. of samples

61. What are recursive and non recursive systems? give examples?

    A Recursive system is one in which the output depend on it,s one or more past outputs while a non recursive is one in which output is independent of output.

    Ex: any system with feedback is Recursive , without feedback is non recursive.

62. Write the properties of linear convolution.

    1) x(n)*y(n)= y(n)*x(n)

2) [x(n)+y(n)]*z(n)=x(n)*z(n)+y(n)*z(n)

3) [x(n)*y(n)]*z(n) =x(n)*[y(n)*z(n)]

63. Define circular convolution.

Circular convolution is same as linear convolution but circular is for periodic signals.

64. What is the importance of linear and circular convolution in signals and systems?

Convolution is used to calculate a LTI system's response for given excitation.

65. How will you perform linear convolution via circular convolution?

Circular convolution with the length of linear convolution length (l+m-1) results linear convolution.

66. What is sectioned convolution? Why is it performed?

If any one of the given two sequences length is very high then we have to go for sectioned convolution.

67. What are the two methods of sectioned convolution?

1) Over lap-Add method. 2) Over lap save method.

68. Define cross correlation and auto-correlation?

Auto correlation is a measure of similarity between signals and its delayed version as a function of time delay.

Cross correlation is a measure of similarity between two signals as a function of time delay between them.

69. What are the properties of Coorelation?

1) $R_{12}(T) \neq R_{21}(T)$

2) $R_{12}(T)=R_{21}^{*}(-T)$

3) if $R_{12}(T)=0$, both signals are orthogonal to each other

4) Fourier transform of auto correlation gives energy spectral density.

DFT & FFT

70. Define DFT of a discrete time sequence?

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-i2\pi kn/N}$$

71. Define inverse DFT.

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1}X(k)\,W^{nk}$$

72. What is the relation between DTFT and DFT?

$$X[k]=X\left(e^{j\Omega}\right)\Big|_{\Omega=k\frac{2\pi}{N}},\quad 0\leq k\leq N-1$$

73. What is the drawback in Fourier transform and how is it overcome?

Fourier transform is that it is not truly realizable in practice but we can get closer, it is not applicable to all signals, so we go for Laplace in continuous , Z in discrete.

74. List any four properties of DFT

linearity Property : ax1(n)+bx2(n)→aX1(ω)+bX2(ω)

Duality Property: X(N)↔Nx[((−k))N]

Complex conjugate property: x∗(n)↔X∗((K))N=X∗(N−K)

Circular shift property: x(n)exp(j2ΠKn/N)↔X((K−L)

75. What is FFT, What it's importance?

FFT stands for Fast Fourier Transform, this is same as DFT but algorithm is different by FFT with in lees time we can compute Fourier transform compared to DFT.

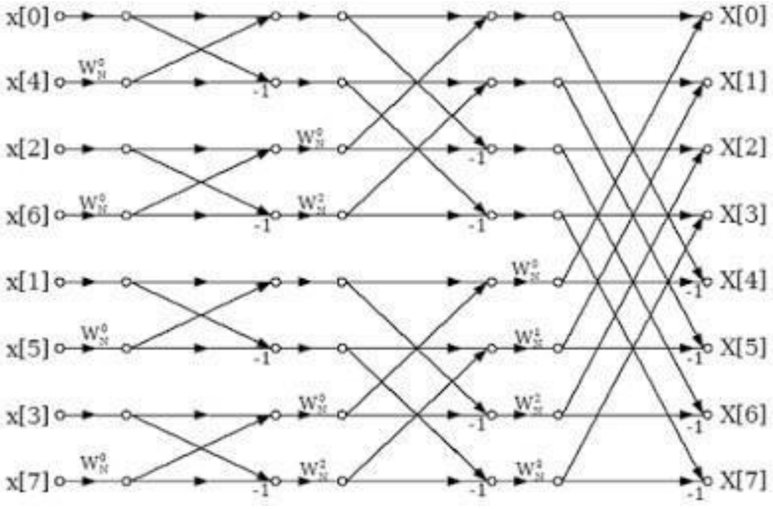
76. Compare FFT and DFT?

DFT and FFT both are used to represent a discrete time signal in frequency domain, But DFT procedure is formula based where FFT is algorithm based, FFT is more efficient and faster than DFT, i.e if a sequence contains N samples then to calculate DFT no. of multiplications and additions required are: N^2 , N(N-1) FFT no. of multiplications and additions required are : (N/2) $\log_2(N)$, N $\log_2(N)$

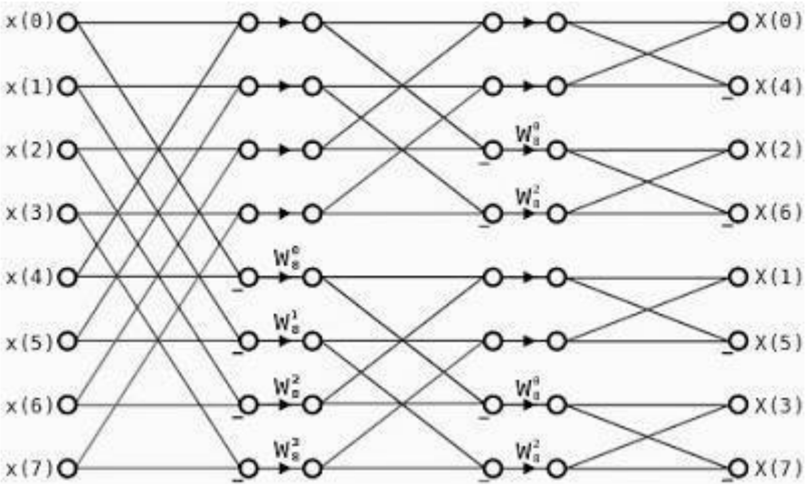
77. What are the various algorithms to calculate FFT?

Decimation In Time (DIT), Decimation In frequency (DIF)

78. Draw the DIT FFT structure with the length of 8?



79. Draw the DIF FFT structure with the length of 8?



80. What is phase factor or twiddle factor?

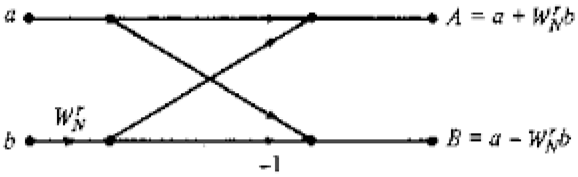
It is defined as $\mathbf{W_N = e^{-j2\pi/N}}$

81. What are the phase factors involved in all stages of computation in the 8-point DIT radix-2 FFT?

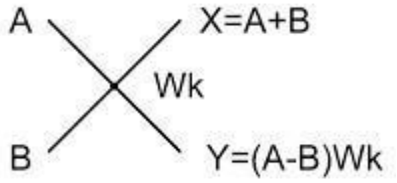
First stage: $\mathbf{W_8^0}$
Second stage: $\mathbf{W_8^0, W_8^2}$

Third stage: $\mathbf{W_8^0, W_8^1, W_8^2, W_8^3}$

82. Draw the basic butterfly diagram or flow graph of DIT radix-2 FFT?



83. Draw the basic butterfly diagram or flow graph of DIF radix-2 FFT?



84. What are the phase factors involved in all stages of computation in 8-point DIF radix-2 FFT?
First stage: $\mathbf{W_8^0, W_8^1, W_8^2, W_8^3}$
Second stage: $\mathbf{W_8^0, W_8^2}$

Third stage: $\mathbf{W_8^0}$

85. What is magnitude and phase spectrum?

Magnitude spectrum is the graph between Fourier transform magnitudes and frequency. Phase spectrum is the graph between Fourier transform phases and frequency.

References

[1] M. Abramowitz and I.A. Stegun, editors. Handbook of Mathematical Functions. Dover Publications, New York NY, 1972.

[2] S. Bagchi and S.K. Mitra. Nonuniform Discrete Fourier Transform and Its Signal Processing Applications. Kluwer, Boston MA, 1998.

[3] M.R. Bateman and B. Liu. An approach to programmable CTD filters using coefficients 0, +1, and −1. IEEE Trans. on Circuits and Systems, CAS-27:451-456, June 1980.

[4] J. Cioffi. A Multicarrier Primer. ANSI T1E1.4 Committee Contribution, Boca Raton FL, November 1991.

[5] A.C. Constantinides. Spectral transformations for digital filters. Proc. IEE (London), 117:1585–1590, August 1970.

[6] R.E. Crochiere and A.V. Oppenheim. Analysis of linear digital networks. Proc. IEEE, 62:581–595, April 1975.

[7] L. Gaszi. Explicit formulas for lattice wave digital filters. IEEE Trans. on Circuits & Systems, CAS-32:68–88, January 1985.

[8] A.H. Gray, Jr. and J. D. Markel. Digital lattice and ladder filter synthesis. IEEE Trans. on Audio and Electroacoustics, AU-21:491–500, December 1973.

[9] O. Herrmann, L.R. Rabiner, and D.S.K. Chan. Practical design rules for optimum finite impulse response lowpass digital filters. Bell System Tech. J., 52:769-799, 1973.

[10] L.B. Jackson. On the interaction of roundoff noise and dynamic range in digital filters. Bell System Technical Journal, 49:159–184, February 1970.

[11] L.B. Jackson. Digital Filters and Signal Processing. Kluwer, Boston MA, third edition, 1996.

[12] P. Jarske, Y. Neuvo, and S.K. Mitra. A simple approach to the design of FIR filters with variable characteristics. Signal Processing, 14:313–326, 1988.

[13] J.F. Kaiser. Nonrecursive digital filter design using the I0-sinh window function. Proc. 1974 IEEE International Symposium on Circuits and Systems, pages 20-23, San Francisco CA, April 1974.

[14] T.P. Krauss, L. Shure, and J.N. Little. Signal Processing TOOLBOX for use with MATLAB. The Mathworks, Inc., Natick MA, 1994.

[15] M. Abramowitz and I.A. Stegun, editors. Handbook of Mathematical Functions. Dover Publications, New York NY, 1972.

[16] S. Bagchi and S.K. Mitra. Nonuniform Discrete Fourier Transform and Its Signal Processing Applications. Kluwer, Boston MA, 1998.

[17] M.R. Bateman and B. Liu. An approach to programmable CTD filters using coefficients 0, +1, and −1. IEEE Trans. on Circuits and Systems, CAS-27:451- 456, June 1980.

[18] J. Cioffi. A Multicarrier Primer. ANSI T1E1.4 Committee Contribution, Boca Raton FL, November 1991.

[19] A.C. Constantinides. Spectral transformations for digital filters. Proc. IEE (London), 117:1585–1590, August 1970.

[20] R.E. Crochiere and A.V. Oppenheim. Analysis of linear digital networks. Proc. IEEE, 62:581–595, April 1975.

[21] L. Gaszi. Explicit formulas for lattice wave digital filters. IEEE Trans. on Circuits & Systems, CAS-32:68–88, January 1985.

[22] A.H. Gray, Jr. and J. D. Markel. Digital lattice and ladder filter synthesis. IEEE Trans. on Audio and Electroacoustics, AU-21:491–500, December 1973.

[23] O. Herrmann, L.R. Rabiner, and D.S.K. Chan. Practical design rules for optimum finite impulse response lowpass digital filters. Bell System Tech. J., 52:769-799, 1973.

[24] L.B. Jackson. On the interaction of roundoff noise and dynamic range in digital filters. Bell System Technical Journal, 49:159–184, February 1970.

[25] L.B. Jackson. Digital Filters and Signal Processing. Kluwer, Boston MA, third edition, 1996.

[26] P. Jarske, Y. Neuvo, and S.K. Mitra. A simple approach to the design of FIR filters with variable characteristics. Signal Processing, 14:313–326, 1988.

[27] J.F. Kaiser. Nonrecursive digital filter design using the I0-sinh window function. Proc. 1974 IEEE International Symposium on Circuits and Systems, pages 20-23, San Francisco CA, April 1974.

[28] T.P. Krauss, L. Shure, and J.N. Little. Signal Processing TOOLBOX for use with MATLAB. The Mathworks, Inc., Natick MA, 1994.