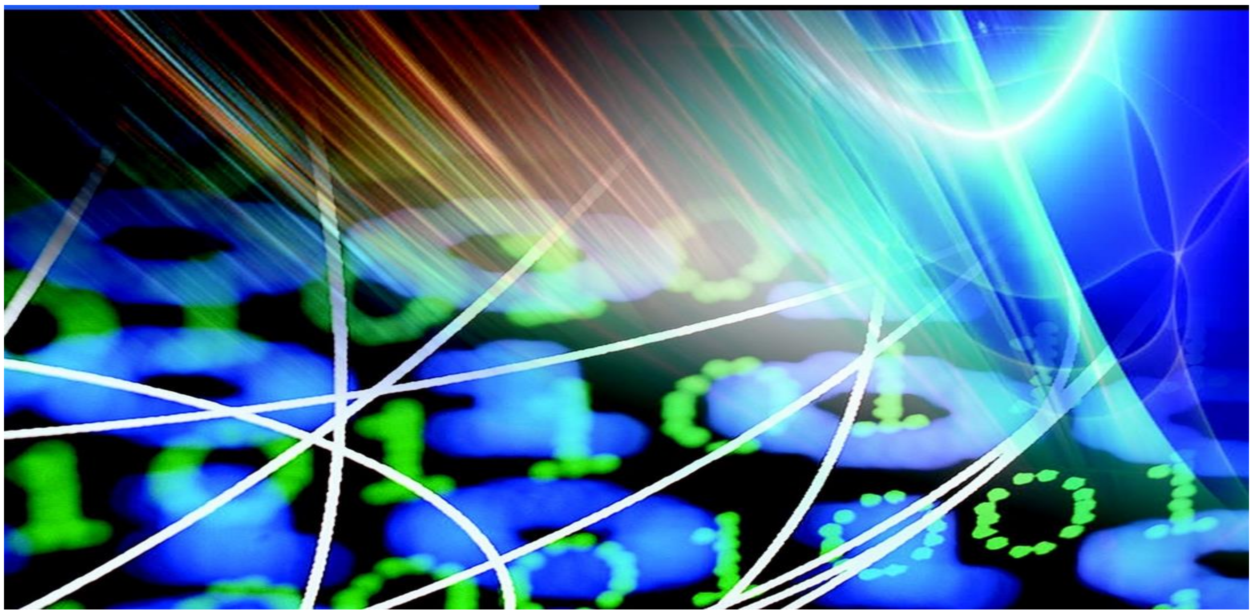
# DIGITAL SIGNAL PROCESSING LAB MANUAL



***Department of Electronics & Communication Engineering***

**VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA**

**NEAR PAKALA, CHITTOOR-517112**

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

# DIGITAL SIGNAL PROCESSING LAB MANUAL

|  |
|---|
| **Name:**________________________________________ |
| **H.T.No:**________________________________________ |
| **Year/Semester:**__________________________________ |

***Department of Electronics & Communication Engineering***

# VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

**NEAR PAKALA, CHITTOOR-517112**

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

**VEMU Institute of Technology**
**Dept. of Electronics and Communication Engineering**

## Vision of the institute

To be one of the premier institutes for professional education producing dynamic and vibrant force of technocrats with competent skills, innovative ideas and leadership qualities to serve the society with ethical and benevolent approach.

## Mission of the institute

**Mission_1:** To create a learning environment with state-of-the art infrastructure, well equipped laboratories, research facilities and qualified senior faculty to impart high quality technical education.
**Mission_2:** To facilitate the learners to inculcate competent research skills and innovative ideas by Industry-Institute Interaction.
**Mission_3:** To develop hard work, honesty, leadership qualities and sense of direction in learners by providing value based education.

## Vision of the department

To develop as a center of excellence in the Electronics and Communication Engineering field and produce graduates with Technical Skills, Competency, Quality, and Professional Ethics to meet the challenges of the Industry and evolving Society.

## Mission of the department

**Mission_1:** To enrich Technical Skills of students through Effective Teaching and Learning practices to exchange ideas and dissemination of knowledge.

**Mission_2:** To enable students to develop skill sets through adequate facilities, training on core and multidisciplinary technologies and Competency Enhancement Programs.

**Mission_3:** To provide training, instill creative thinking and research attitude to the students through Industry-Institute Interaction along with Professional Ethics and values.

## Programme Educational Objectives (PEOs)

**PEO 1:** To prepare the graduates to be able to plan, analyze and provide innovative ideas to investigate complex engineering problems of industry in the field of Electronics and Communication Engineering using contemporary design and simulation tools.

**PEO-2:** To provide students with solid fundamentals in core and multidisciplinary domain for successful implementation of engineering products and also to pursue higher studies.

**PEO-3:** To inculcate learners with professional and ethical attitude, effective communication skills, teamwork skills, and an ability to relate engineering issues to broader social context at work place

## Programme Outcomes(Pos)

| | |
|---|---|
| **PO_1** | **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering |

| | |
|---|---|
| | fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO_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. |
| **PO_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. |
| **PO_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. |
| **PO_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. |
| **PO_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. |
| **PO_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. |
| **PO_8** | **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| **PO_9** | **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO_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. |
| **PO_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. |
| **PO_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. |

## Programme Specific Outcome(PSOs)

| | |
|---|---|
| **PSO_1** | **Higher Education :** Qualify in competitive examination for pursuing higher education by applying the fundamental concepts of Electronics and Communication Engineering domains such as Analog & Digital Electronics, Signal Processing, Communication & Networking, Embeded Systems, VLSI Design and Control systems etc., |
| **PSO_2** | **Employment**: Get employed in allied industries through their proficiency in program specific domain knowledge, Specalized software packages and Computer programming or became an entrepreneur. |

## JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

## III B.Tech. I-Sem (ECE)

# (20A04502P) DIGITAL SIGNAL PROCESSING LAB

**Course Outcomes:**

| CO No. | Description | BL |
|---|---|---|
| C318.1 | Implement various DSP Algorithms using software packages. | 03 |
| C318.2 | Implement DSP algorithms with Digital Signal Processor. | 03 |
| C318.3 | Analyze and observe magnitude and phase characteristics (Frequency response Characteristics) of digital IIR-Butterworth, Chebyshev filters. | 04 |
| C318.4 | Analyze and observe magnitude and phase characteristics (Frequency response Characteristics) of digital FIR filters using window techniques. | 04 |
| C318.5 | Analyze digital filters using Software Tools. | 04 |

**List of Experiments:**

1. Generate the following standard discrete time signals.
   i) Unit Impulse ii) Unit step iii) Ramp iv) Exponential v) Sawtooth
2. Generate sum of two sinusoidal signals and find the frequency response (magnitude and phase).
3. Implement and verify linear and circular convolution between two given signals.
4. Implement and verify autocorrelation for the given sequence and cross correlation between two given signals.
5. Compute and implement the N-point DFT of a given sequence and compute the power density spectrum of the sequence.
6. Implement and verify N-point DIT-FFT of a given sequence and find the frequency response (magnitude and phase).
7. Implement and verify N-point IFFT of a given sequence.
8. Design IIR Butterworth filter and compare their performances with different orders (Low Pass Filter /High Pass Filter)
9. Design IIR Chebyshev filter and compare their performances with different orders (Low Pass Filter /High Pass Filter).
10. Design FIR filter (Low Pass Filter /High Pass Filter) using windowing technique.
    i. Using rectangular window
    ii. Using hamming window
    iii. Using Kaiser window
11. Design and verify Filter (IIR and FIR) frequency response by using Filter design and Analysis Tool.
12. Compute the Decimation and Interpolation for the given signal.
13. Real time implementation of an audio signal using a digital signal processor.
14. Compute the correlation coefficient for the two given audio signals of same length using a digital signal processor.

**Note: Any TWELVE of the experiments are to be conducted.**

**References:**

1. Digital Signal Processing: Alon V. Oppenhelm, PHI
2. Digital Signal processing(II-Edition): S.K. Mitra, TMH

**Online Learning Resources/Virtual Labs:**

**1. http://vlabs.iitkgp.ac.in/dsp/#**

# VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

**NEAR PAKALA, CHITTOOR-517112**

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

**Department of Electronics &Communication Engineering**

## LIST OF EXPERIMENTS TO BE CONDUCTED

### Software Experiments (Using Matlab)

1. Generate the following standard discrete time signals.
   i)Unit Impulse ii) Unit step iii) Ramp iv) Exponential v) Sawtooth
2. Generate sum of two sinusoidal signals and find the frequency response (magnitude and phase).
3. Implement and verify linear and circular convolutions between two given signals.
4. Implement and verify autocorrelation for the given sequence and cross correlation between two given signals.
5. Compute and implement the N-point DFT of a given sequence and compute the power density spectrum of the sequence.
6. Implement and verify N-point DIT-FFT of a given sequence and find the frequency response (magnitude and phase).
7. Implement and verify N-point IFFT of a given sequence.
8. Design IIR Butterworth filter and compare their performances with different orders (Low Pass Filter /High Pass Filter)
9. Design IIR Chebyshev filter and compare their performances with different orders (Low Pass Filter /High Pass Filter).
10. Design FIR filter (Low Pass Filter /High Pass Filter) using windowing technique.
    i.)Using rectangular window
    ii.)Using hamming window
    iii.)Using Kaiser window
11. Design and verify Filter (IIR and FIR) frequency response by using Filter design and Analysis Tool.
12. Compute the Decimation and Interpolation for the given signal.

### Hardware Experiments (Using DSP Processor)

1. Implement and verify linear and circular convolution between two given signals.
2. Implement and verify autocorrelation for the given sequence and cross correlation between two given signals.
3. Find DFT of given discrete time signal.
4. Implement and verify N-point DIT-FFT of a given sequence.
5. Implement and verify N-point IFFT of a given sequence
6. Design and implementation of FIR with low pass / high pass filter using any three windowing techniques. Plot its magnitude and phase responses.
7. Design and implementation of IIR Butterworth (LP/HP) filter..
8. Design and implementation of IIR Chebyshev (LP/HP) filter.
9. Compute the Decimation and Interpolation for the given signal.

### Advanced Experiments (Beyond Curriculum)

1. Convert CD Data to DVD Data.
2. Power Density Spectrum

# CONTENTS

## DOS & DONTS IN LABORATORY

1. While entering the Laboratory, the students should follow the dress code (Wear shoes, White Apron & Female students should tie their hair back).
2. The students should bring their observation note book, Lab manual, record note book, calculator, and necessary stationary items.
3. While sitting in front of the system, check all the cable connections and Switch on the computer.
4. If a student notices any fluctuations in power supply, immediately the same thing is to be brought to the notice of technician/lab in charge.
5. At the end of practical class the system should be switch off safely and arrange the chairs properly.
6. Each program after completion should be written in the observation note book and should be corrected by the lab in charge on the same day of the practical class.
7. Each experiment should be written in the record note book only after getting signature from the lab in charge in the observation note book.
8. Record should be submitted in the successive lab session after completion of the experiment.
9. 100% attendance should be maintained for the practical classes.

## SCHEME OF EVALUVATION

| S No | Date | Name Of The Experiment | Marks Awarded | | | Sign. |
|---|---|---|---|---|---|---|
| | | | Observation (10M) | Viva voce (10M) | Total (20M) | |
| **Software Experiments(Using Matlab)** | | | | | | |
| 1. | | Generation of discrete time Signals | | | | |
| 2. | | Generate sum of two sinusoidal Signals. | | | | |
| 3. | | Linear and Circular Convolutions between two given Signals. | | | | |
| 4 | | Autocorrelation for the given sequence and Cross Correlation between two given signals. | | | | |
| 5 | | N-point DFT of a given Sequence . | | | | |
| 6 | | N-point DIT-FFT of a given Sequence | | | | |
| 7 | | N-point IFFT of a given Sequence | | | | |
| 8 | | Design and implementation of FIR filter with (LP/HP) filter using any three windowing techniques. Plot its magnitude and phase responses | | | | |
| 9 | | Design and implementation of IIR Butterworth (LP/HP) filter | | | | |
| 10 | | Design and implementation of IIR Chebyshev (LP/HP) filter. | | | | |
| 11 | | Filter (IIR and FIR) frequency response by using Filter design and Analysis Tool. | | | | |
| 12 | | Decimation and Interpolation for the given signal. | | | | |
| **Hardware Experiments (Using DSP Processor)** | | | | | | |
| 1 | | Linear and circular convolution between two given signals | | | | |
| 2 | | Autocorrelation for the given Sequence and cross correlation between two given signals. | | | | |
| 3 | | N-point DFT of a given Sequence. | | | | |
| 4 | | N-point DIT-FFT of a given Sequence. | | | | |
| 5 | | N-point IFFT of a given Sequence. | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6 | | Design and implementation of IIR Butterworth (LP/HP) filter | | | | |
| 7 | | Design and implementation of IIR Chebyshev (LP/HP) filter. | | | | |
| 8 | | Design and implementation of FIR filter with (LP/HP) filter using any three windowing techniques. Plot its magnitude and phase responses. | | | | |
| 9 | | Decimation and Interpolation for the given signal. | | | | |
| **Advanced experiments (Beyond Curriculum)** | | | | | | |
| 1 | | Convert CD Data to DVD Data. | | | | |
| 2 | | Power Spectral Density | | | | |

**Signature of Lab In-charge**

# SOFTWARE EXPERIMENTS (USING MATLAB)

**Exp .No:01** **Date:**

# GENERATION OF DISCRETE TIME SIGNALS

**AIM: -**To write a "MATLAB" Program to generate of discrete time signals like unit impulse, unit step, unit ramp, exponential signal and sawtooth signals.

**SOFTWARE REQURIED :-**

- PC and MATLAB Software (2019b 9.7version)

**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

**PROGRAM:-**

```
%unit impulse function%
%Discrete%
n=-10:10;
Xn=(n==0);
subplot(3,2,1);
stem(n,Xn);
axis([-11 11 -0.5 1.5]);
xlabel('Samples');
ylabel('amplitude');
title(' unit impulse function');
%unit step function%
%Discrete%
n=-10:10;
Xn=(n>=0);
subplot(3,2,2);
stem(n,Xn);
axis([-11 11 -0.5 1.5]);
xlabel('Samples');
ylabel('amplitude');
title(' discrete unit step function');

%unit ramp function%
%Discrete%
n=-10:10;
Xn=(n>=0).*n;
subplot(3,2,3);
stem(n,Xn);
axis([-11 11 -1 11]);
xlabel(' Samples');
ylabel('amplitude');
title(' discrete unit ramp function');
```

```
% exponential signal:
%Discrete%
n2=input('enter the  length of the exponential sequence');
t=0:n2;
a=input('enter the a value');
y2=exp(a*t);
subplot(3,2,4);
stem(t,y2);
ylabel('amplitude');
xlabel('time period');
title('exponential sequence')

%sawtooth signal
%Discrete%
n=0:10;
Xn=sawtooth(pi*n/4);
subplot(3,2,5);
stem(n,Xn);
axis([-0.5 11 -1.5 1.5])
xlabel('time period');
ylabel('amplitude');
title('sawtooth sequence');
```

**OUTPUT:-**

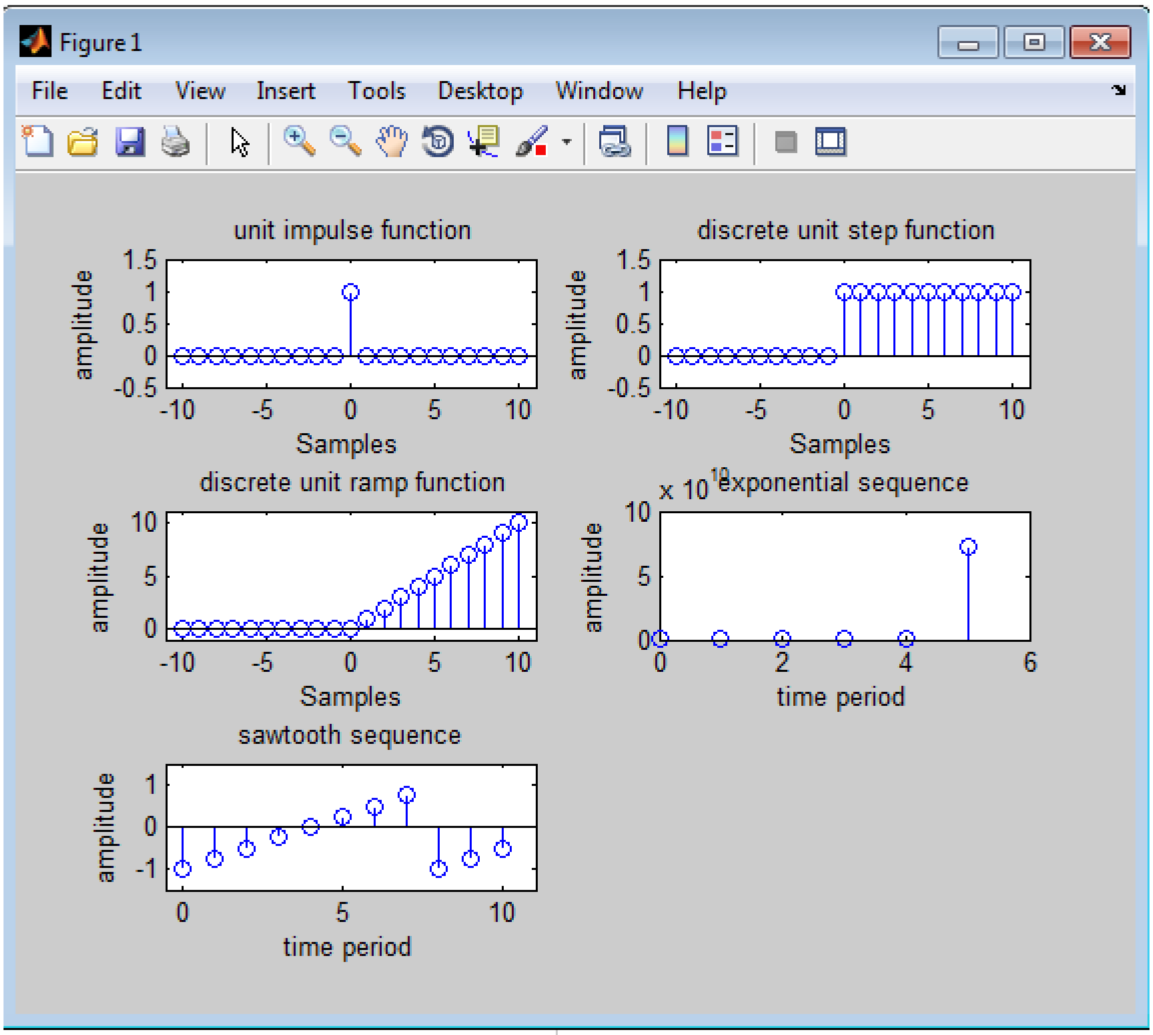


Figure 1
File Edit View Insert Tools Desktop Window Help
unit impulse function
amplitude
Samples
discrete unit step function
discrete unit ramp function
x 10^10 exponential sequence
time period
sawtooth sequence

**RESULT:-**

**CONCLUSIONS:**

**VIVA QUESTIONS:**

1. Define impulse, unit step, ramp signals and write their expressions?
2. Define exponential and sinusoidal signals and write their expressions?
3. Express unit step signal in terms of unit impulse?
4. Express ramp signal in terms of unit step signal?
5. Represent the signal x[n]={1,2,-1,3,2}using impulse signal?

**Exp .No:02** **Date:**

## SUM OF TWO SINUSOIDAL SIGNALS

**AIM: -** To write a MATLAB program to find the sum of two sinusoidal signals and to find frequency response (magnitude and phase).

**SOFTWARE REQURIED :-**

- PC and MATLAB Software (2019b 9.7version)

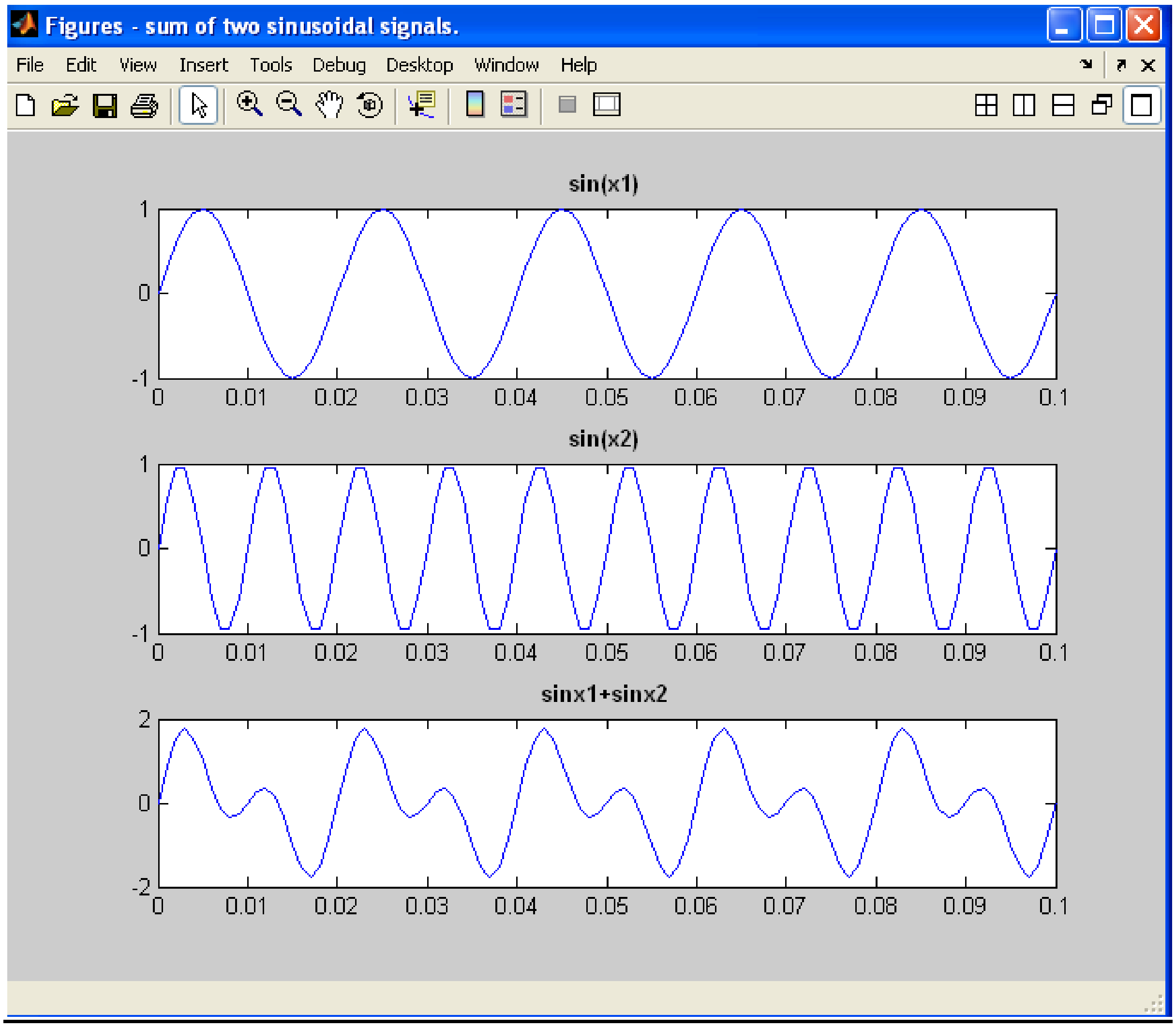
**PROCEDURE:-**

- 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

**PROGRAM:-**

```
clc;
Clear all;
Close all;
t=0:0.001:0.1;
f1=50;
x1=2*pi*f1*t;
y1=sin(x1);
figure;
subplot (3,1,1);
plot (t,y1);
title('sin(x1');
f2=100;
x2=2*pi*f2*t;
y2=sin(x2);
subplot(3,1,2);
plot(t,y2);
title('sin(x2)');
y=y1+y2;
subplot(3,1,3);
plot(t,y);
title('sinx1=sinx2')
```

## OUTPUT:



## RESULT:

## CONCLUSIONS:

**VIVA QUESTIONS**

1. How do you find sum of sinusoid?

2. How do you add two sinusoidal currents?

3. What is amplitude of sinusoidal function?

4. What is meant by sinusoidal signal?

5. What is phase of sinusoidal function?

**Exp .No:03** **Date:**

## LINEAR AND CIRCULAR CONVOLUTIONS

**AIM: -** To write MATLAB programs to find out the linear convolution and Circular convolution of two sequences.

**SOFTWARE REQURIED :-**

- PC and MATLAB Software (2019b 9.7version)

**PROCEDURE:-**

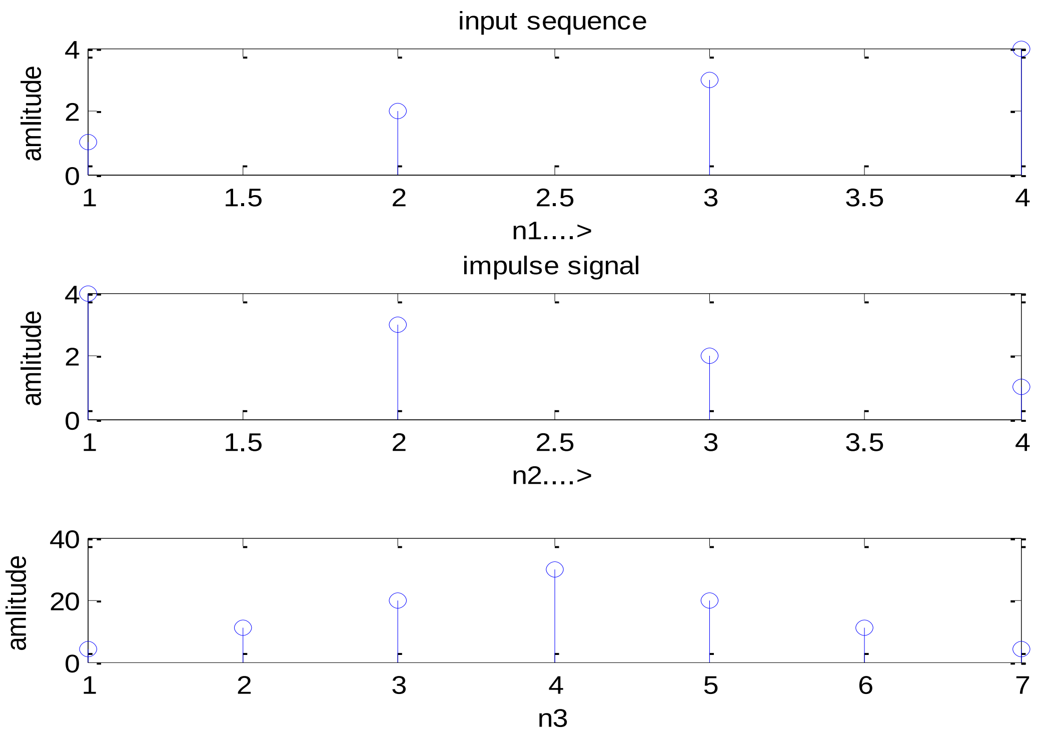
- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

**PROGRAM:-**

```
%Program for linear convolution
%to get the input sequence
n1=input('enter the length of input sequence');
n2=input('enter the length of impulse sequence');
x=input('enter the input sequence');
h=input('enter the impulse sequence');

%convolution operation
y=conv(x,h);
%to plot the signal
subplot(3,1,1);
stem(x);
ylabel('amplitude');
xlabel('n1....>');
title('input sequence')
subplot(3,1,2);
stem(h);
ylabel('amplitude');
xlabel('n2....>');
title('impulse signal')
subplot(3,1,3);
stem(y);
ylabel('amplitude');
xlabel('n3');
disp('the resultant signal is');y
```

OUTPUT : LINEAR CONVOLUTION

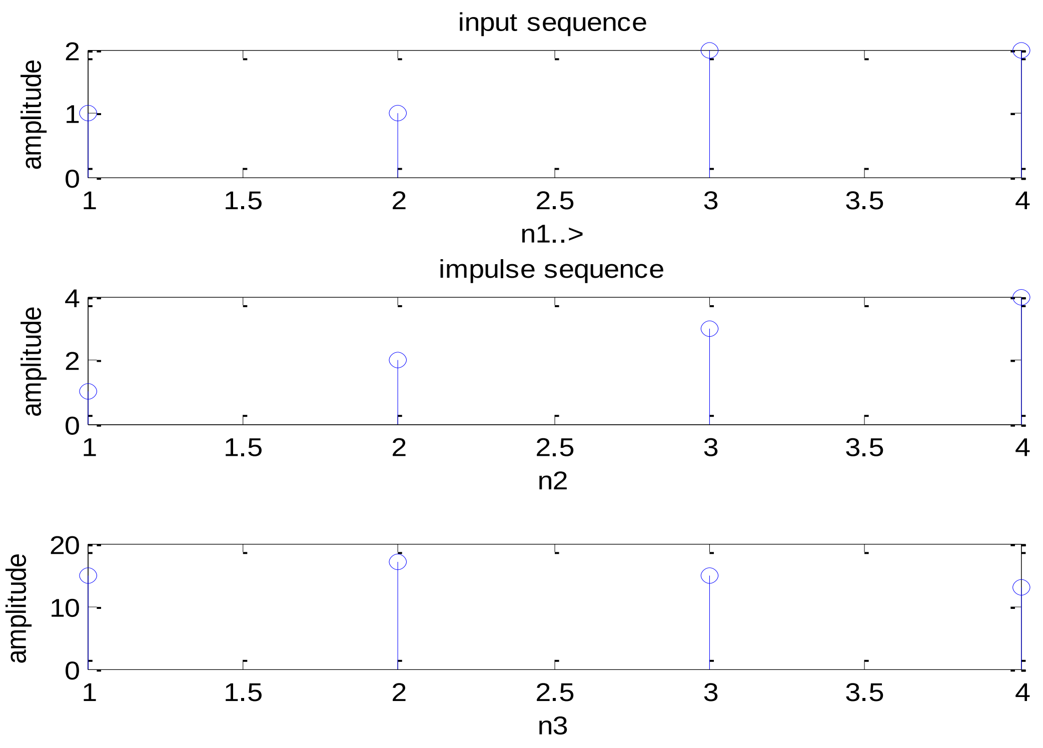
```
Enter the length of input sequence 4
Enter the length of impulse sequence 4
Enter the input sequence   [1 2 3 4]
Enter the impulse sequence   [4   3 2 1]

The resultant signal is
y=  4    11    20     30     20     11     4
```



```
%%Program for circular convolution
clc;
clear all;
close all;
%to get the input sequence
g=input('enter the input sequence');
h=input('enter the impulse sequence');
N1=length(g);
N2=length(h);
N=max(N1,N2);
N3=N1-N2
%loop for getting equal length sequence
if(N3>=0)
h=[h,zeros(1,N3)];
else
g=[g,zeros(1,-N3)];
end
%computation of circular convoluted sequence
for n=1:N;
y(n)=0;
for i=1:N;
j=n-i+1;
if(j<=0)
  j=N+j;
end
y(n)=y(n)+g(i)*h(j);
end
end
figure;
subplot(3,1,1);
stem(g);
ylabel('amplitude');
xlabel('n1..>');
title('input sequence')
subplot(3,1,2);
stem(h);
ylabel('amplitude');
xlabel('n2');
title('impulse sequence')
subplot(3,1,3);
stem(y);
ylabel('amplitude');
xlabel('n3');
disp('the resultant signal is');
```

OUTPUT : CIRCULAR CONVOLUTION

```
Enter the input sequence   [1   2    2   1]
Enter the impulse sequence   [4    3   2   1]

The resultant signal is
y=   15      17       15       13
```



**RESULT:**

**CONCLUSIONS:**

**VIVA QUESTIONS:**

1.) Define Convolution?

2.) What are the types of Convolution?

3.) What is Linear Convolution & Circular Convolution?

4.) State the methods available to compute Convolution Sum?

5.) What is the significance of convolution?

**Exp .No: 04** **Date:**

# AUTO-CORRELATION AND CROSS-CORRELATION

**AIM: -** To write a Matlab program to compute autocorrelation for the given sequence and cross correlation between two signals.

**SOFTWARE REQURIED :-**

- PC and MATLAB Software (2019b 9.7version)

**.PROCEDURE:**

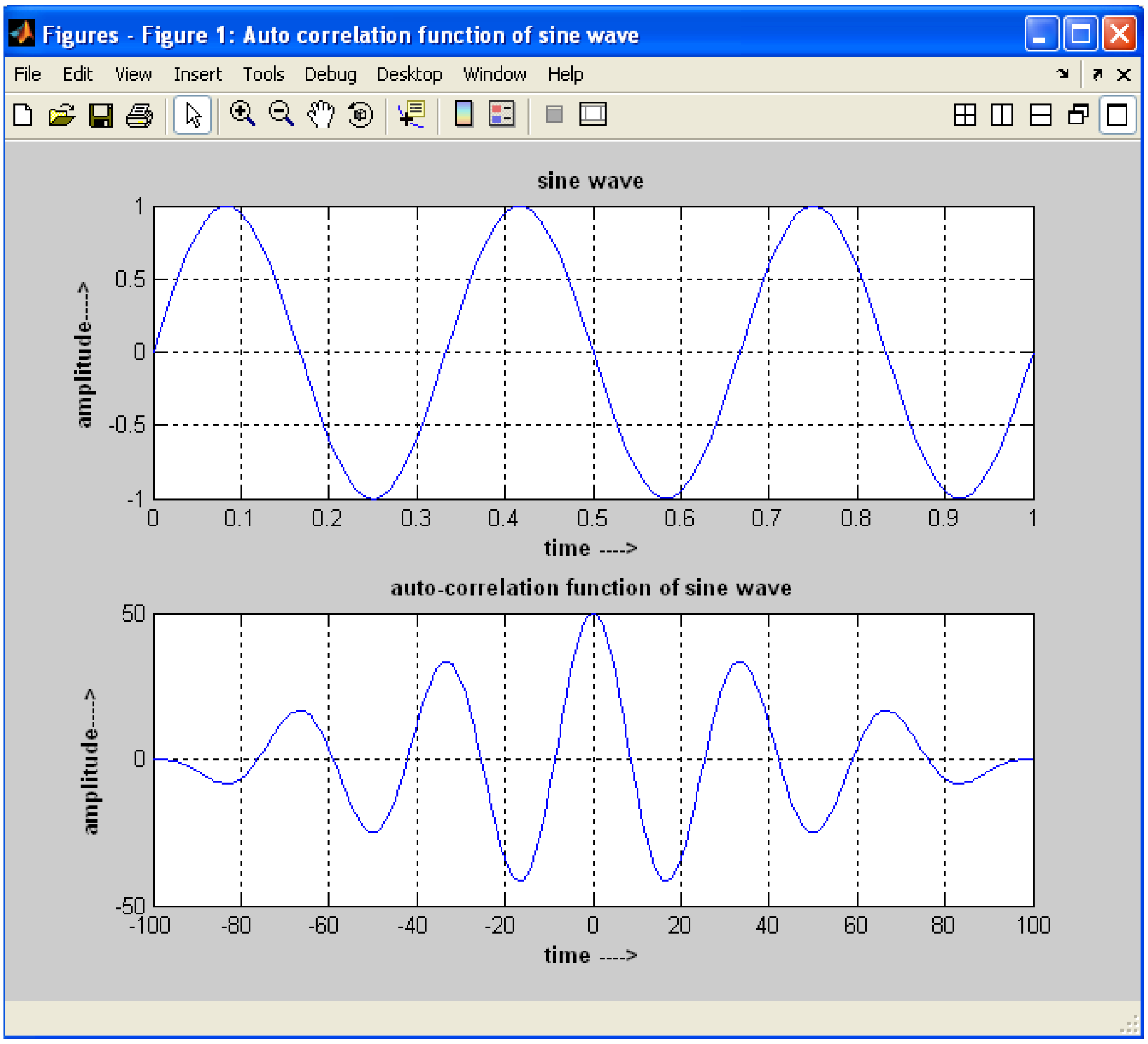
- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

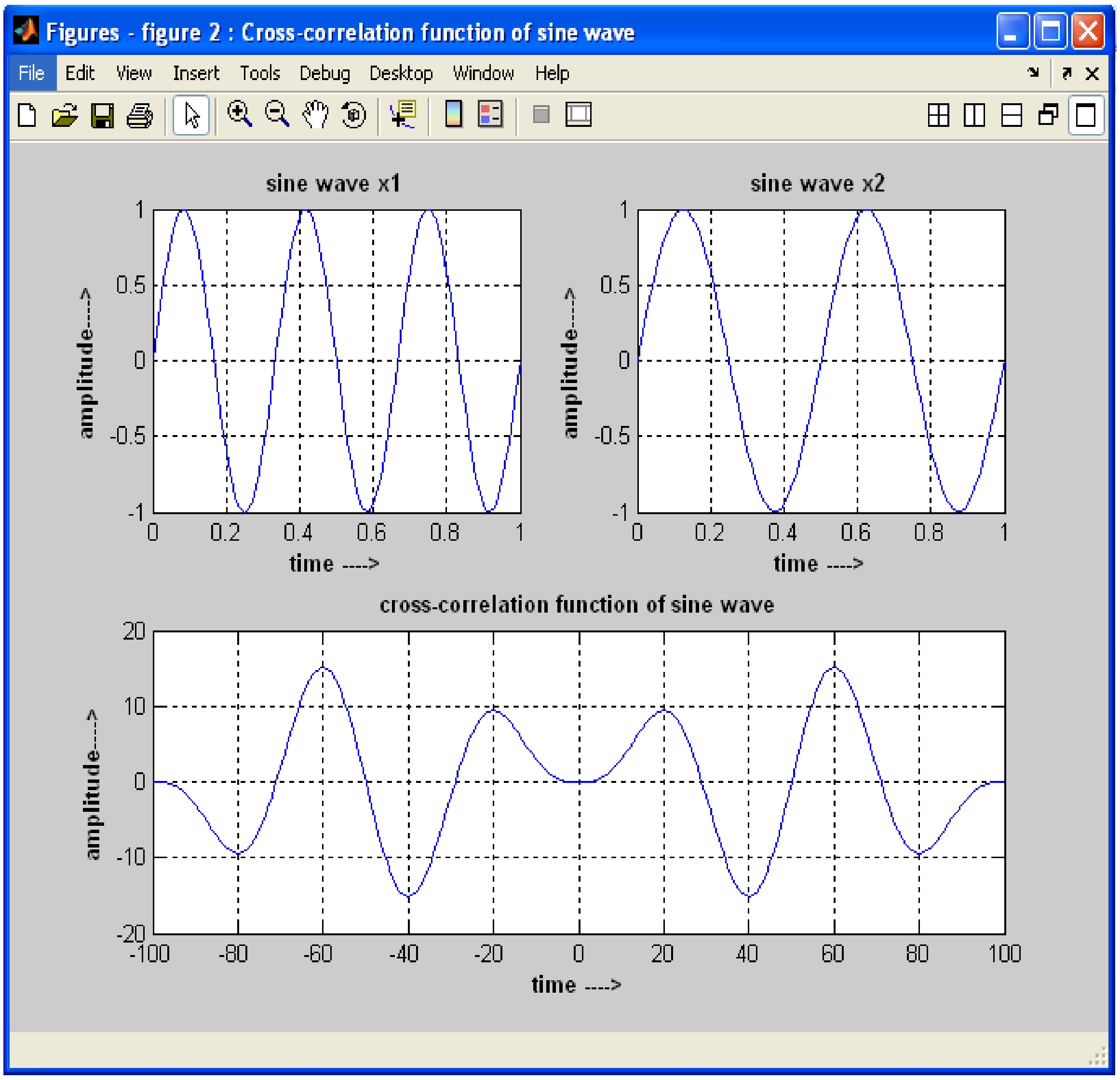
**PROGRAM:**

```
clc; clear all; close all;
t=0:0.01:1;
f1=3;
x1=sin(2*pi*f1*t);
figure;
subplot(2,1,1);
plot(t,x1);
title('sine wave');
xlabel('time ---->');
 ylabel('amplitude---->');
grid;
[rxx lag1]=xcorr(x1);
subplot(2,1,2);
plot(lag1,rxx);
grid;
title('auto-correlation function of sine wave');
figure;
subplot(2,2,1);
plot(t,x1);
title('sine wave x1');
xlabel('time ---->');
ylabel('amplitude---->');
grid;
 f2=2;
x2=sin(2*pi*f2*t);
subplot(2,2,2);
plot(t,x2);
```

```
title('sine wave x2');
xlabel('time ---->');,ylabel('amplitude---->');
grid;
[cxx lag2]=xcorr(x1,x2);
subplot(2,2,[3,4]);
plot(lag2,cxx);
grid;
title('cross-correlation function of sine wave');
```

OUTPUT:




Figures - figure 2 : Cross-correlation function of sine wave
File Edit View Insert Tools Debug Desktop Window Help
sine wave x1
sine wave x2
amplitude----->
time ---->
cross-correlation function of sine wave
1
0.5
0
-0.5
-1
0 0.2 0.4 0.6 0.8 1
20
10
0
-10
-20
-100 -80 -60 -40 -20 0 20 40 60 80 100

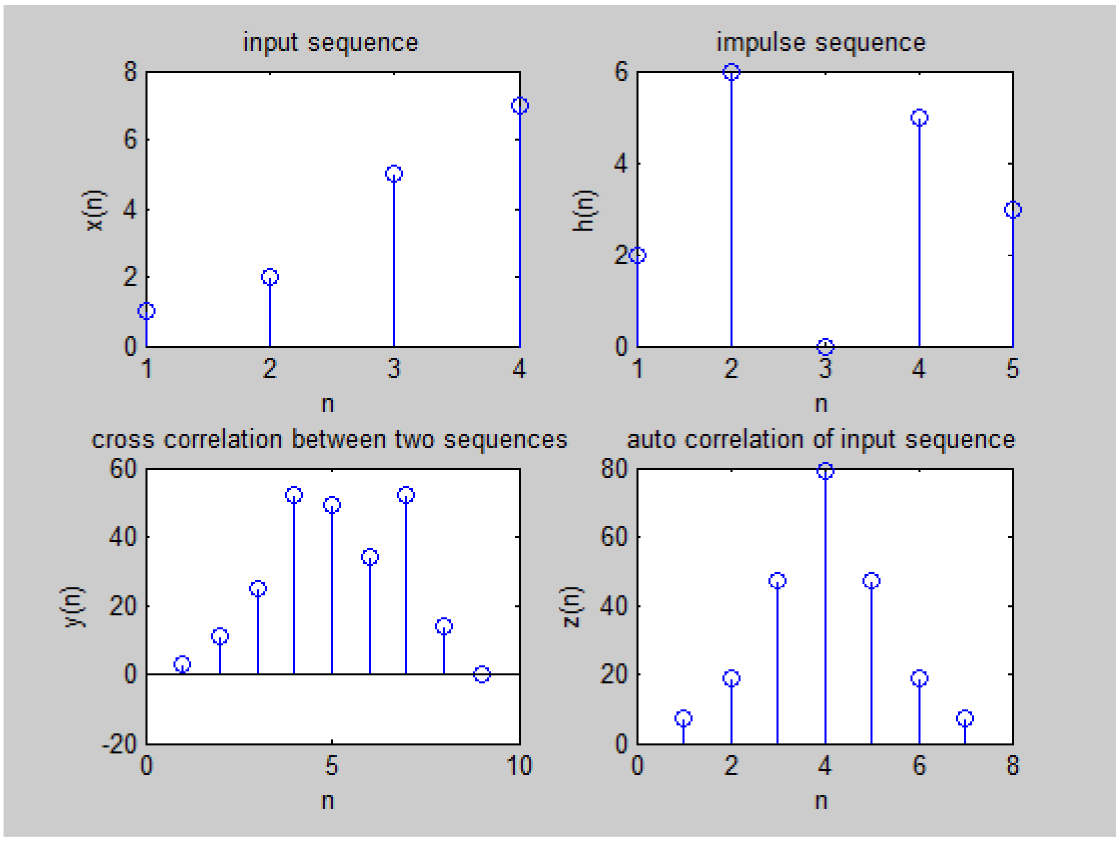
**PROGRAM:**

```
clc;
close all;
clear all;
% two input sequences
x=input('enter input sequence');
h=input('enter the impulse suquence');
subplot(2,2,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input sequence');
subplot(2,2,2);
stem(h);
xlabel('n');
ylabel('h(n)');
title('impulse sequence');
% cross correlation between two sequence
y=xcorr(x,h);
subplot(2,2,3);
stem(y);
xlabel('n');
ylabel('y(n)');
title(' cross correlation between two sequences ');
% auto correlation of input sequence
z=xcorr(x,x);
subplot(2,2,4);
stem(z);
xlabel('n');
ylabel('z(n)');
title('auto correlation of input sequence');
```

**INPUT:**

```
enter input sequence [1 2 5 7]
enter the impulse sequence [2 6 0 5 3]
```

**OUTPUT:**



**RESULT:**

**CONCLUSIONS:**

## VIVA QUESTIONS:

1.) Define Cross Correlation?

2.) What are the applications of Cross Correlation in signal de noising?

3.) Define Cross Power Spectral Density?

4.) How Cross Correlation is different Auto Correlation?

5.) Describe the formula to compute Cross Correlation?

**Exp .No: 05** **Date:**

# DISCRETE FOURIER TRANSFORM

**AIM:** To find DFT of a given sequence and compute the power density spectrum of the sequence.

**SOFTWARE REQUIRED:**

- PC and MATLAB Software (2019b 9.7version)

**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

## THEORY:

Basic equation to find the DFT of a sequence is given below.

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

$$where \ W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \text{ [TWIDDLE FACTOR]}$$

Basic equation to find the IDFT of a sequence is given below.

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn} \qquad n = 0, \dots, N-1.$$

**4. PROGRAM:**

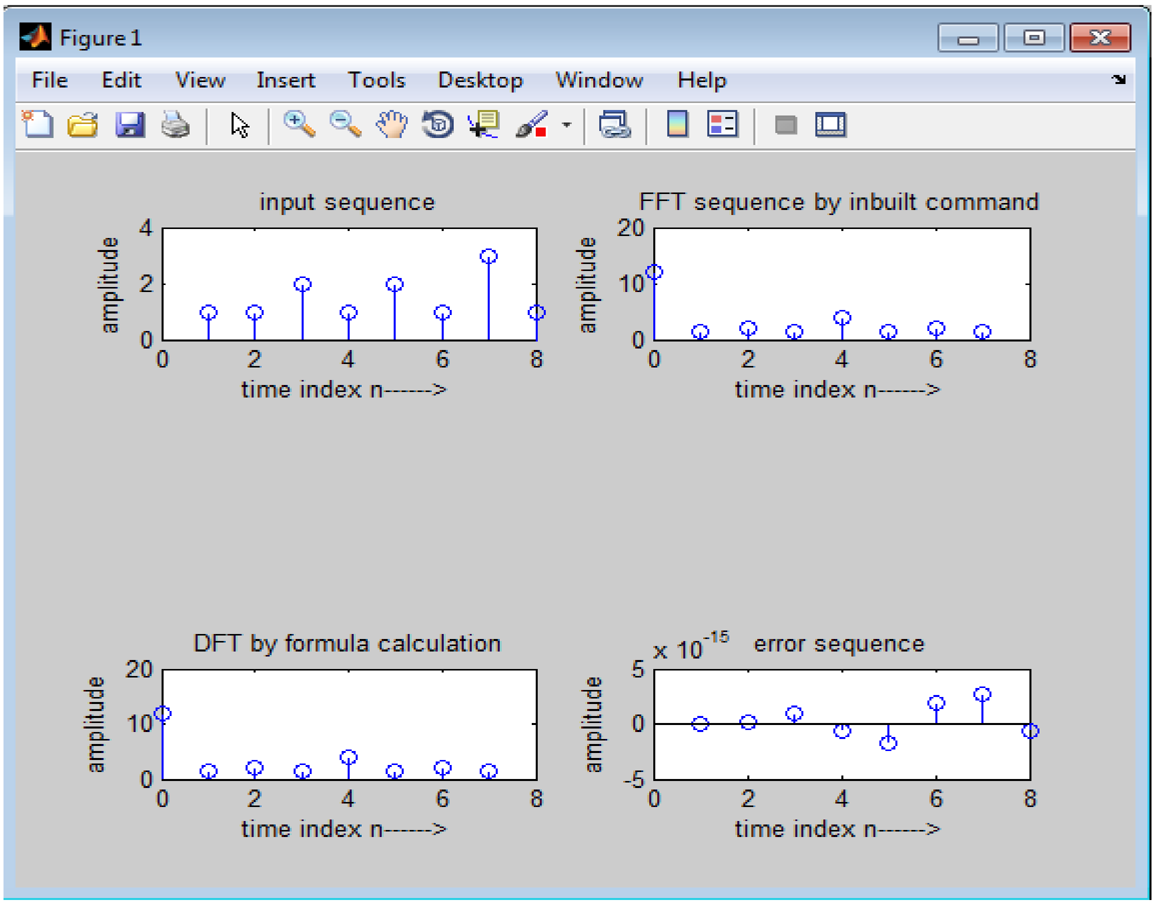
```
clc;
close all;
clear all;
tic;
fprintf('Date & Time:');
Date= datestr(now);
disp(Date);
%DFT
disp('D.F.T');
a=input('Enter the input sequence:');
n=length(a);
x=fft(a,n);
for k=1:n;
y(k)=0;
for i=1:n
y(k)=y(k)+a(i)*exp((-j)*2*pi*(i-1)*(k-1)*(1/n));
end
end
error=x-y;
disp(x);
disp(y);
disp(error);
subplot(3,2,1);
stem(a);
xlabel('time index n------>');
ylabel('amplitude');
title('input sequence');
subplot(3,2,2);
stem(0:n-1,abs(x));
xlabel('time index n------>');
ylabel('amplitude');
title('FFT sequence by inbuilt command');
subplot(3,2,5);
stem(0:n-1,abs(y));
xlabel('time index n------>');
ylabel('amplitude');
title('DFT by formula calculation');
subplot(3,2,6);
stem(error);
xlabel('time index n------>');
ylabel('amplitude');
title('error sequence');
% Power Spectral Density
P = x.* conj(x) / 512;
f = 1000*(0:256)/512;
figure,plot(f,P(1:257))
title('Frequency content of y');
xlabel('frequency (Hz)');
```

**INPUT:**

```
D.F.T
Enter the input sequence:[1 1 2 1 2 1 3 1]
```

**OUTPUT**

```
Columns 1 through 4
12.0000 -1.0000 + 1.0000i -2.0000 -1.0000 - 1.0000i
Columns 5 through 8
4.0000 -1.0000 + 1.0000i -2.0000 -1.0000 - 1.0000i
Columns 1 through 4
12.0000 -1.0000 + 1.0000i -2.0000 - 0.0000i -1.0000 - 1.0000i
Columns 5 through 8
4.0000 + 0.0000i -1.0000 + 1.0000i -2.0000 - 0.0000i -1.0000 - 1.0000i
1.0e-014 *
Columns 1 through 4
0 0.0444 + 0.0222i 0.0444 + 0.0888i -0.0666 - 0.0666i
Columns 5 through 8
0 - 0.1715i 0.7105 + 0.1887i 0.2665 + 0.2665i -0.1887 - 0.0666i
Elapsed time is 27.683359 seconds
```



**RESULT:**

**CONCLUSIONS:**

**VIVA QUESTIONS:**

1. What is the difference between DTFT and DFT?

2. Write any Two Properties of DFT?

3. What is zero Padding and Explain the effect of it on magnitude Spectrum?

4. Write the Two properties of Twiddle Factor?

5. How many no. of Complex Multiplications and Additions are required to compute N-Point DFT?

**Exp .No: 06** **Date:**

## FFT USING Decimation-In-Time (DIT) ALGORITHM

**1. AIM:** Write a MATLAB program to perform N-point DIT- FFT of a given signal and to plot its magnitude and phase spectrum.

**2. SOFTWARE REQUIRED:**

- PC and MATLAB Software (2019b 9.7version)

**3. 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\

**PROGRAM:**

```
clc;
close all;
clear all;
tic;
fprintf('Date & Time:');
Date= Datestr(now);
disp(Date);
fprintf('DIT-FFT');
fprintf('\n\n');
N=input('Enter the length of the input sequence:');
for i=1:N
re(i)=input('Enter the real part of the time domain sequence:');
im(i)=input('Enter the imaginary part of the time domain sequence:');
end
[re1,im1]=DITFFT(re,im,N);
subplot(3,2,1);
stem(0:N-1,re);
xlabel('time index n------>');
ylabel('amplitude');
title('input real sequence');
subplot(3,2,2);
stem(0:N-1,im);
xlabel('time index n------>');
ylabel('amplitude');
title('input imaginary sequence');
subplot(3,2,5);
stem(0:N-1,re1);
xlabel('frequency------>');
```

```
ylabel('amplitude');
title('output real sequence');
subplot(3,2,6);
stem(0:N-1,im1);
xlabel('frequency------>');
ylabel('amplitude');
title('output imaginary sequence');
disp(re1);
disp(im1);
```

**DIT-FFT**

```
Enter the length of the input sequence:8
Enter the real part of the time domain sequence:1
Enter the imaginary part of the time domain sequence:2
Enter the real part of the time domain sequence:1
Enter the imaginary part of the time domain sequence:3
Enter the real part of the time domain sequence:1
Enter the imaginary part of the time domain sequence:2
Enter the real part of the time domain sequence:2
Enter the imaginary part of the time domain sequence:1
Enter the real part of the time domain sequence:2
Enter the imaginary part of the time domain sequence:1
Enter the real part of the time domain sequence:3
Enter the imaginary part of the time domain sequence:1
Enter the real part of the time domain sequence:2
Enter the imaginary part of the time domain sequence:1
Enter the real part of the time domain sequence:3
Enter the imaginary part of the time domain sequence:1
15.0000 0.7071 2.0000 0.1213 -3.0000 -0.7071 -2.0000 -4.1213
12.0000 5.5355 1.0000 0.7071 0 -1.5355 -1.0000 -0.7071
Elapsed time is 23.565254 seconds.
```

**6. RESULT:**

**7. CONCLUSION:**

**VIVA QUESTIONS:**

1.Write the Block diagram of 8-Point DIT FFT & DIF DFFT radix 2 Algorithm?

2.Explain using convolution the effects of taking an FFT of a sample with no windowing

3.What’s the difference between FFT and DFT?

**4.** Why do we need Fourier transform in DSP?

**5.** What is "decimation-in-time" versus "decimation-in-frequency"?

**Exp .No: 07** **Date:**

# INVERSE FAST FOURIER TRANSFORM (IFFT)

**AIM:** Write a MATLAB program to perform N-point IFFT of a given sequence.

**SOFTWARE REQUIRED:**

- PC and MATLAB Software (2019b 9.7version)

**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\

**4. PROGRAM:**

```
clc;
clear all;
close all;
n=input('enter value of n=');
x=input('enter input sequence=');
a=1:1:n;
y=fft(x,n);
disp('fft of input sequence');
disp(y);
z=ifft(y);
disp('ifft of input sequence');
disp(z);
```

**5. OUTPUT:**

```
enter value of n=8
enter input sequence=[1 2 3 4 5 6 7 8]
fft of input sequence
 Columns 1 through 6
 36.0000 + 0.0000i  -4.0000 + 9.6569i  -4.0000 + 4.0000i  -4.0000 + 1.6569i  -4.0000 + 0.0000i  -
4.0000 - 1.6569i
 Columns 7 through 8
 -4.0000 - 4.0000i  -4.0000 - 9.6569i
ifft of input sequence
     1     2     3     4     5     6     7     8
```

**RESULT:**

**CONCLUSION:**

**VIVA QUESTIONS:**

1.) What is the need of FFT?

2.) Write the Applications of FFT?

3.) FFT is in complex domain how to use it in real life signals optimally?

4.) What is the difference between FFT and IFFT?

5.) How many no. of Complex Multiplications and Additions are required to compute N-Point DFT using FFT?

**Exp .No: 08** **Date:**

# DESIGN OF IIR BUTTERWORTH (LP/HP) FILTERS

**AIM:** To write a MATLAB program to design Butterworth IIR LP\HP filters.

**SOFTWARE REQUIRED:**

- PC and MATLAB Software (2019b 9.7version)

**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

**PROGRAM:**

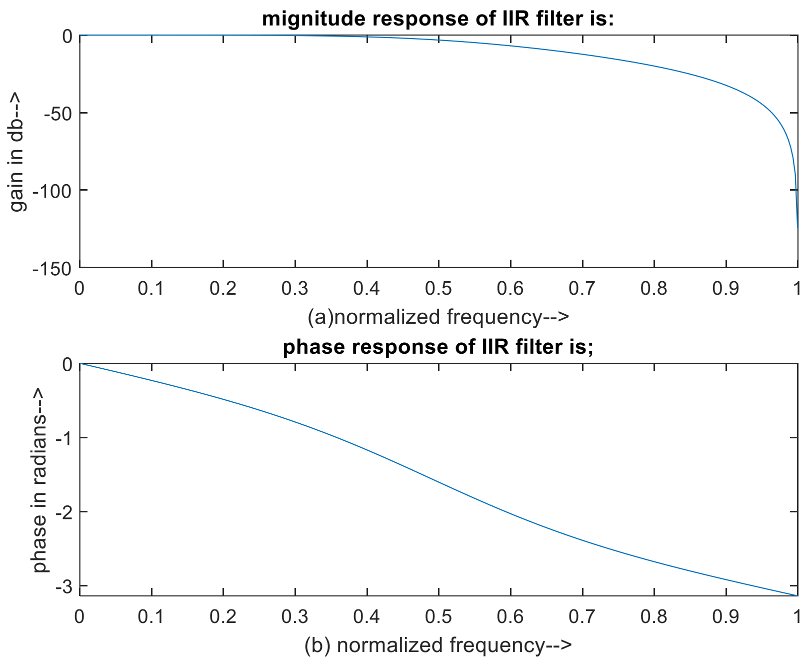
```
clc;
clear all;
close all;
display('enter the iir filter design specifications');
        rp=input('enter the pass band ripple:');
        rs=input('enter the stop band ripple:');
        wp=input('enter the pass band freq:');
ws=input('enter the stop band freq:');
fs=input('enter the sampling freq:');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs);
c=input('enter choice filter 1.lpf 2.hpf /n');
if(c==1)
display('frequency response of IIR lpf is:');
[b,a]=butter(n,wn,'low');
end
if(c==2)
display('freq response of IIR hpf IS:');
[b,a]=butter(n,wn,'high');
end
w=0:0.01:pi;
h=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure;
subplot(2,1,1);
plot(w/pi,m);
title('mignitude response of IIR filter is:');
xlabel('(a)normalized frequency-->');
ylabel('gain in db-->');
subplot(2,1,2);
plot(w/pi,an);
titel('phase response of IIR filter is;');
```

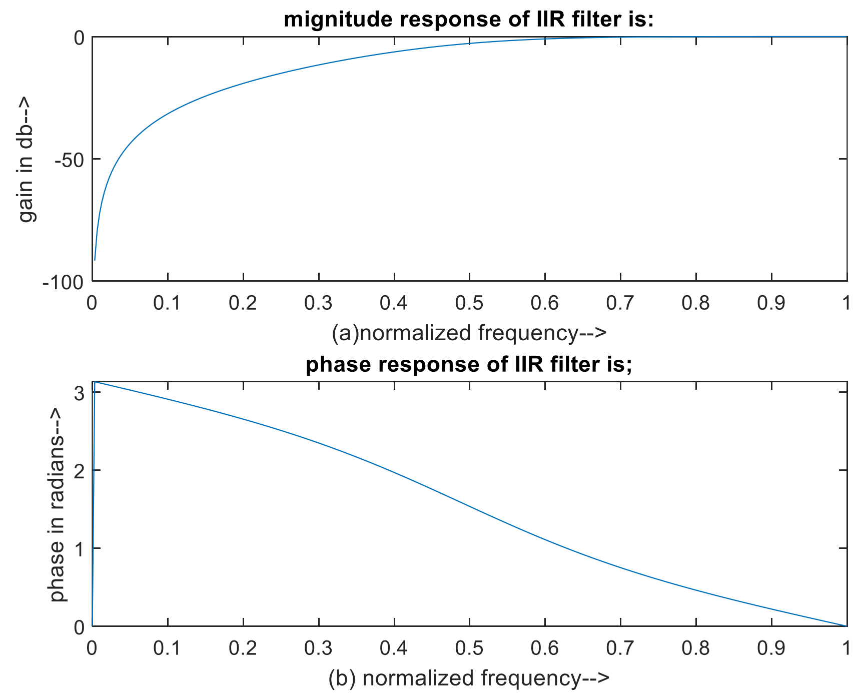
```
xlabel('(b) normalized frequency-->');
ylabel('phase in radians-->');
```

**INPUT:**

```
enter the iir filter design specifications
enter the pass band ripple:2
enter the stop band ripple:20
enter the pass band freq:1000
enter the stop band freq:2000
enter the sampling freq:5000
enter choice filter 1.lpf 2.hpf /n
```

**5. OUTPUT:**





**RESULT:**

**CONCLUSION:**

**VIVA QUESTIONS:**

1.) What do you mean by cut-off frequency?

2.) Give the differences between analog and digital filters?

3.) What is the difference between type 1 and type 2 filter structures?

4.) What is the role of delay element in filter design?

5.) List the Differences between Butterworth and chebyshev filters?

**Exp .No: 09** **Date:**

# DESIGN OF IIR CHEBYSHEV (LP/HP) FILTERS

**AIM:** To write a MATLAB program to design Chebyshev IIR LP\HP filters.

**SOFTWARE REQUIRED:**

- PC and MATLAB Software (2019b 9.7version)

**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

**4. PROGRAM:**

***% Program for the design of Chebyshev Type-1 low-pass filter***

```
clc;
close all;clear all;
format long
rp=input(„enter the passband ripple...‟);
rs=input(„enter the stopband ripple...‟);
wp=input(„enter the passband freq...‟);
ws=input(„enter the stopband freq...‟);
fs=input(„enter the sampling freq...‟);
w1=2*wp/fs;w2=2*ws/fs;
[n,wn]=cheb1ord(w1,w2,rp,rs,‟s‟);
[b,a]=cheby1(n,rp,wn,‟s‟);
W=0:.01:pi;
[h,om]=freqs(b,a,w);
M=20*log10(abs(h));
An=angle(h);
subplot(2,1,1);
plot(om/pi,m);
ylabel(„Gain in dB --.‟);
xlabel(„(a) Normalised frequency --.‟);
subplot(2,1,2);
plot(om/pi,an);
xlabel(„(b) Normalised frequency --.‟);
ylabel(„Phase in radians --.‟);
```

***% Program for the design of Chebyshev Type-2 High pass analog filter***

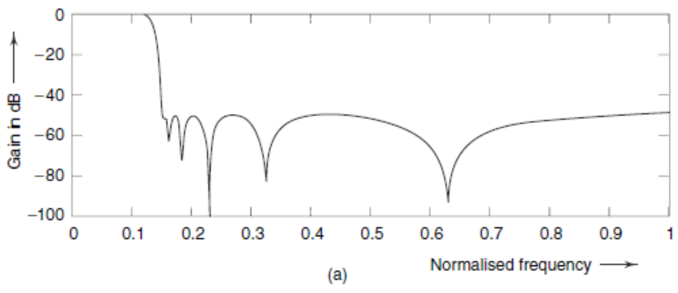
```
clc;
close all;clear all;
format long
rp=input('enter the passband ripple...');
rs=input('enter the stopband ripple...');
wp=input('enter the passband freq...');
ws=input('enter the stopband freq...');
```

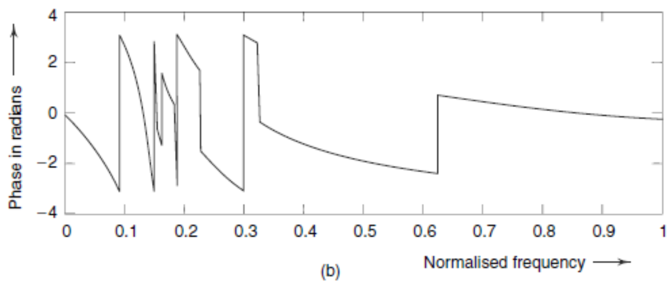
```
fs=input('enter the sampling freq...');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=cheb2ord(w1,w2,rp,rs,'s');
[b,a]=cheby2(n,rs,wn,'high','s');
w=0:.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(2,1,1);
plot(om/pi,m);
ylabel('Gain in dB --.');
xlabel('(a) Normalised frequency --.');
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b) Normalised frequency --.');
ylabel('Phase in radians --.');
```

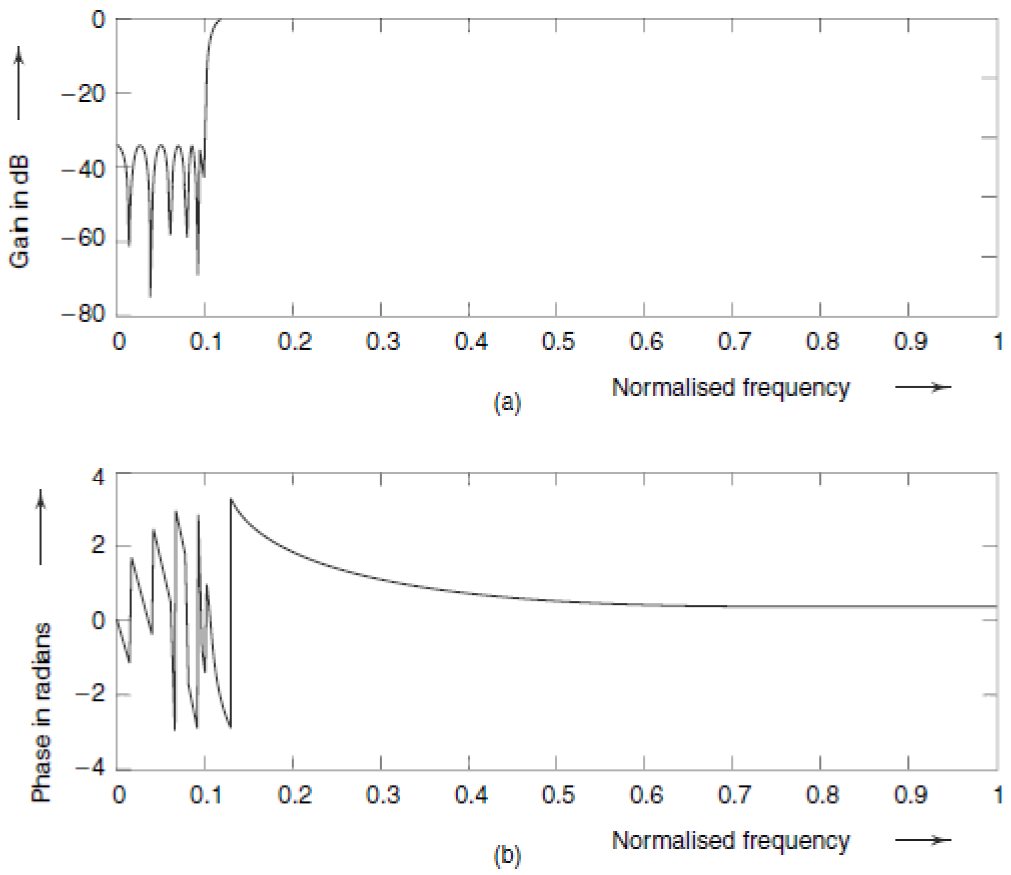
**INPUT:**

```
LPF
enter the passband ripple... 0.23
enter the stopband ripple... 47
enter the passband freq... 1300
enter the stopband freq... 1550
enter the sampling freq... 7800
HPF
enter the passband ripple... 0.34
enter the stopband ripple... 34
enter the passband freq... 1400
enter the stopband freq... 1600
enter the sampling freq... 10000
```

**OUTPUT:**







*Chebyshev Type - 2 High-pass Analog Filter*
*(a) Amplitude Response and (b) Phase Response*

**RESULT:**

**CONCLUSION:**

**VIVA QUESTIONS:**

1.) What is the difference between type 1 and type 2 filter structures?

2.)IIR Filters are non linear phase filters. Why?

3) IIR filters are not always stable. Why?

4.) Write the realization Techniques for IIR Filter?

5.) Compare all realization techniques vailable for IIR Filter?

**Exp .No: 10** **Date:**

# DESIGN OF FIR (LP/HP) USING WINDOWING TECHNIQUE

**AIM:** To write a MATLAB program to design FIR with Low pass filter and High Pass filter using any three Windowing techniques.

**SOFTWARE REQUIRED:**

- PC and MATLAB Software (2019b 9.7version) **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

**PROGRAM:**

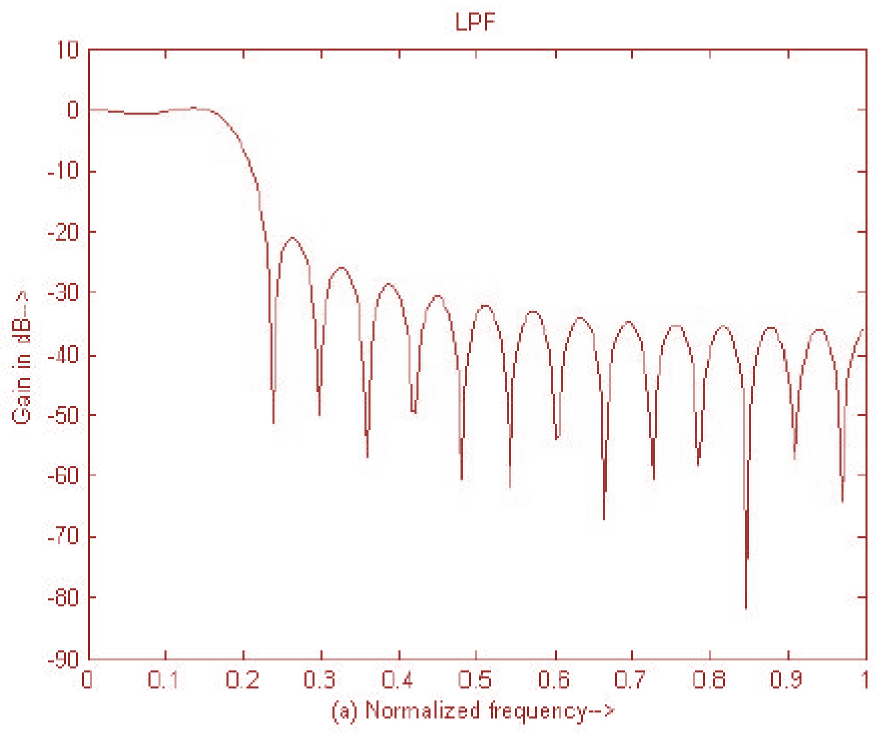
```
%FIR Low Pass/High pass filter design using Rectangular/Hamming/Kaiser window
clc; clear all; close all;
rp=input('enter passband ripple');
rs=input('enter the stopband ripple');
wp=input('enter passband freq');
ws=input('enter stopband freq');
fs=input('enter sampling freq ');
beta=input('enter beta value');
w1=2*wp/fs;
w2=2*ws/fs;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(ws-wp)/fs;
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. Hamming 3.kaiser: \n ');
if(c==1)
y=rectwin(n1);
disp('Rectangular window filter response');
end
if (c==2)
y=hamming(n1);
disp('Hamming window filter response');
end
if(c==3)
y=kaiser(n1,beta);
disp('kaiser window filter response');
```

```
end
ch=input('give type of filter 1:LPF,2:HPF');
switch ch
case 1
b=fir1(n,w1,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
plot(o/pi,m);
title('LPF');
xlabel('(a) Normalized frequency-->');
ylabel('Gain in dB-->');
case 2
b=fir1(n,w1,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
plot(o/pi,m);
title('HPF');
xlabel('(b) Normalized frequency-->');
ylabel('Gain in dB-->');
end
```

**OUTPUT:**

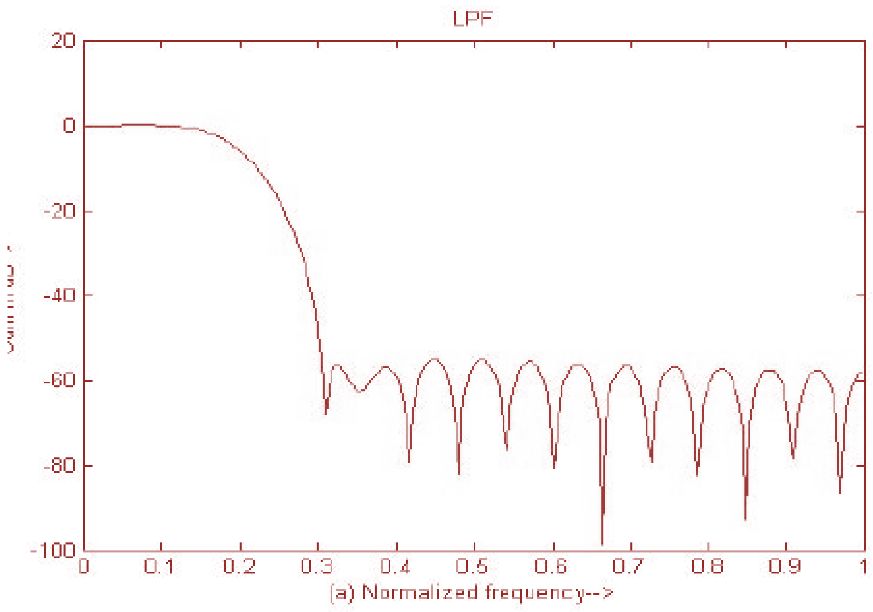
```
enter passband ripple 0.02
enter the stopband ripple
enter passband freq 1000
enter stopband freq 1500
enter sampling freq 10000
enter beta value 5
enter your choice of window function 1. rectangular 2.Hamming 3.kaiser:1

Rectangular window filter response
give type of filter 1:LPF,2:HPF
1:LPF
```

**Low passfilter using Rectangular Window**



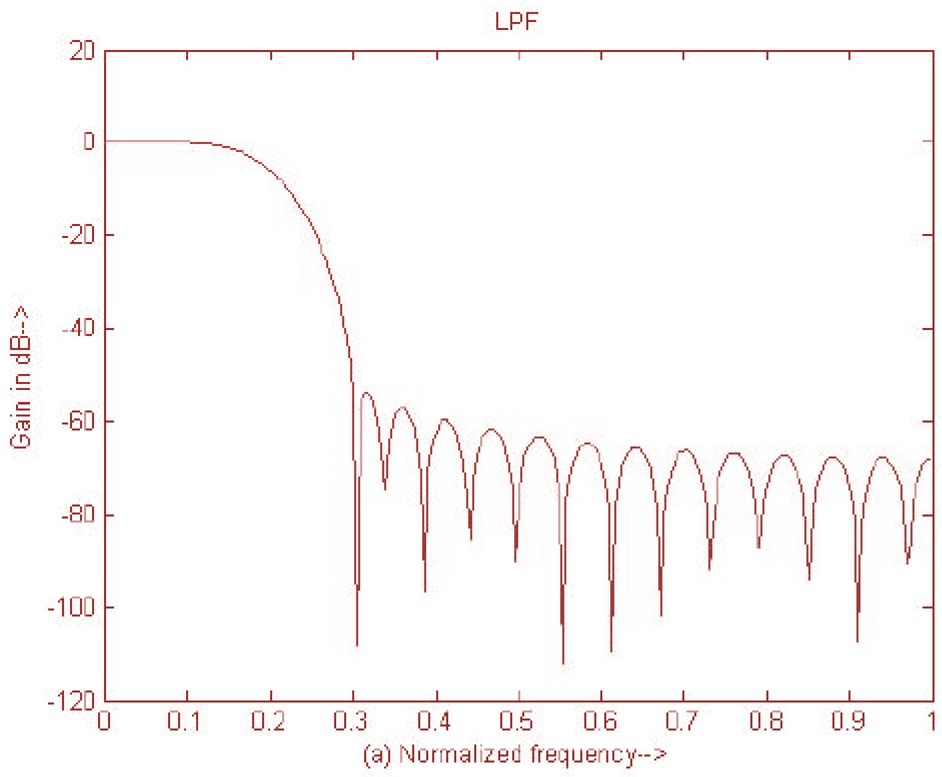
```
enter your choice of window function 1. rectangular 2.
Hamming 3.kaiser:
2
Hamming window filter response
give type of filter 1:LPF,2:HPF
1:LPF
```

**Low pass FIR filter using Hamming window**



```
enter your choice of window function 1. rectangular 2.
Hamming 3.kaiser:
3
kaiser window filter response
give type of filter 1:LPF,2:HPF
1:LPF
```

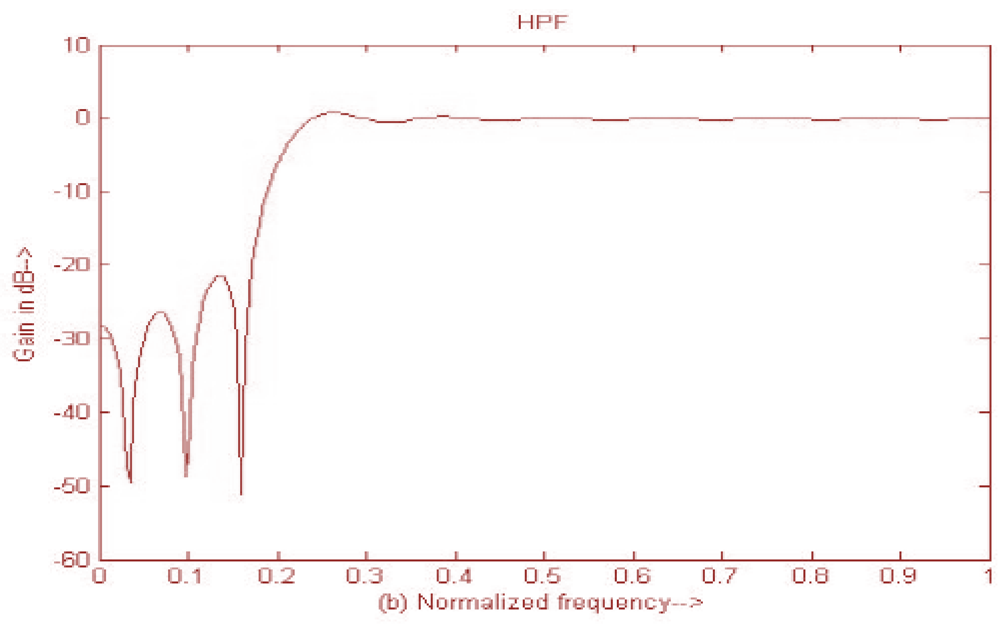
**Low pass FIR filter using Kaiser Window**



## FIR High pass Filter design

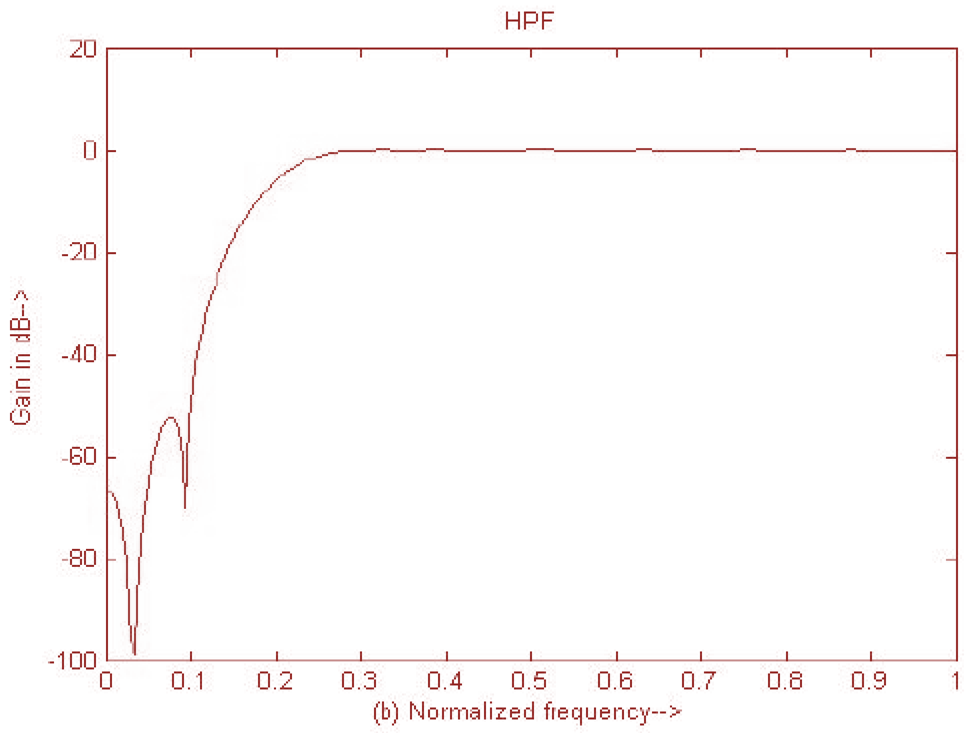
```
enter passband ripple
enter the stopband ripple
enter passband freq 1000
enter stopband freq 1500
enter sampling freq 10000
enter beta value 5
enter your choice of window function 1. rectangular 2.
Hamming 3.kaiser:
1
Rectangular window filter response
give type of filter 1:LPF,2:HPF
2:HPF
```

## High pass FIR filter using Rectangular Window



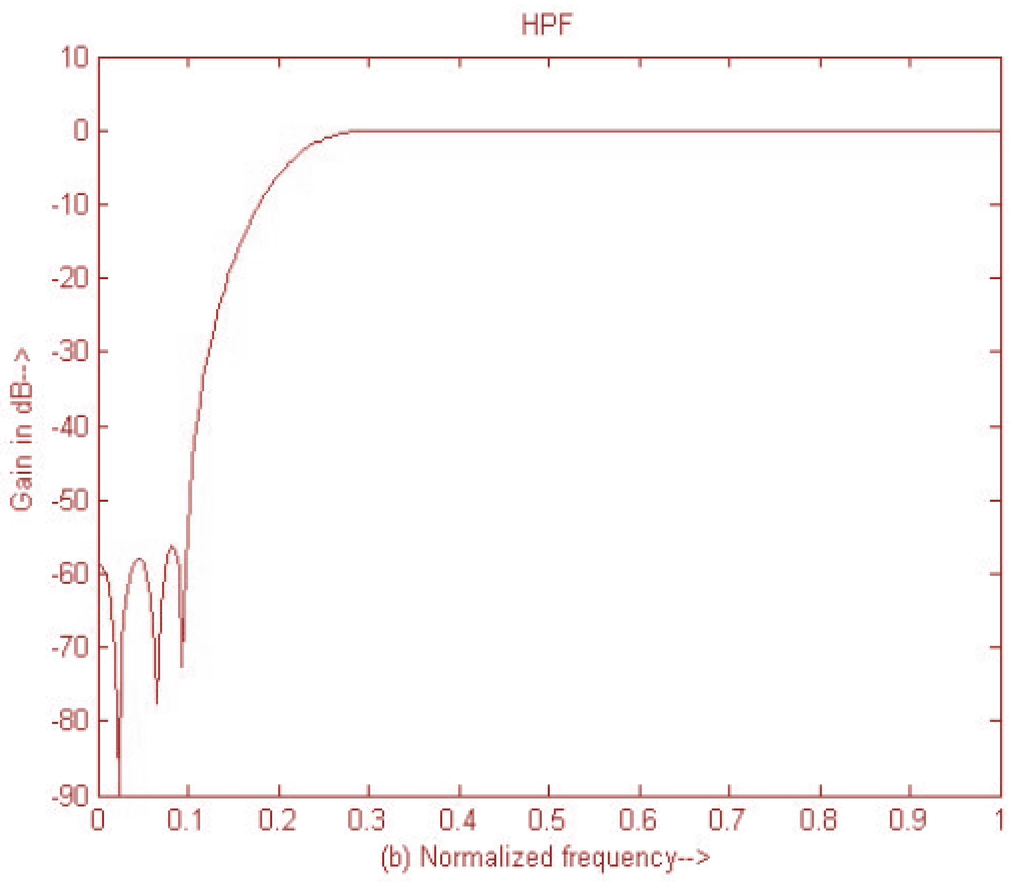
**enter your choice of window function 1. rectangular 2. Hamming 3.kaiser:**
**2**
**Hamming window filter response**
**give type of filter 1:LPF,2:HPF**
**2:HPF**

## High pass FIR filter using Hamming Window



enter your choice of window function 1. rectangular 2. Hamming 3.kaiser:
3
kaiser window filter response
give type of filter 1:LPF,2:HPF
2: HPF

## High pass FIR filter using Kaiser Window



**RESULT:**

**CONCLUSIONS:**

**VIVA QUESTIONS:**

1. What is filter?

2. What is FIR and IIR filter define, and distinguish between these two?

3. What is window method? How you will design an FIR filter using window method?

4. What are low-pass and band-pass filter and what is the difference between these two?

5. What is the matlab command for Hamming window? Explain.

**Exp .No: 11** **Date:**

## A.GENERATION OF FIR LOW PASS/HIGH PASS FILTER COEFFICIENTS

**AIM: -**

To generate FIR filter (LP/HP) coefficients using windowing techniques

- Using Rectangular Window
- Using Triangular Window
- Using Keiser Window

**APPARATUS REQUIRED**: -

- PC and MATLAB Software (2019b 9.7version)

**THEORY**: -

**Steps to design linear phase FIR filter using windowing methods:**

1. Clearly specify the filter specifications
   a. Order of filter (**N**)
   b. Sampling Rate (**$F_s$**)
   c. Cutoff frequency (**$f_c$**)
2. Compute the Cutoff frequency ($\omega_c$)
   1. **$\omega_c=2\pi*f_c/F_s$**
3. Compute the desired impulse response h(n) using particular window.
   - Lowpass Filter with Cutoff frequency($\omega_c$): -
   - Coefficients of linear phase filter

$$h_d(n)= \omega_c/\pi \qquad \text{for } n=\alpha$$
$$= \sin(\omega_c(n-\alpha))/(\pi(n-\alpha)) \qquad \text{for } n\neq\alpha$$

**Where $\alpha=(N-1)/2$.**

- Window Sequence
  **a.** For Rectagular Window

$$W(n)=1 \qquad \text{for } 0\leq n\leq(N-1)$$
$$=0 \qquad \text{otherwise}$$

  **b.** For Triangular Window

$$W(n)=2n/(N-1) \qquad \text{for } 0\leq n\leq(N-1)/2$$
$$=2-\{2n/(N-1) \qquad \text{for } (N-1)/2\leq n\leq(N-1)$$

  **c.** For Keiser Window

$$W(n)=I_0\,[\beta((N-1)^2/4-[n-(N-1)/2]^2)^{1/2}]/I_0[\beta(N-1)/2] \text{ for } 0\leq n\leq(N-1)$$
$$=0 \qquad \text{otherwise}$$

where $\beta$ is the variable parameter whose choice control the tradeoff between side lobe amplitude and side lobe width.

$4<\beta[(N-1)/2]<9$

- The Impulse Response

  $h(n)=h_d(n)*w(n)$

➢ Highpass Filter with Cutoff frequency($\omega_c$): -

- Coefficients of linear phase filter

  $$\mathbf{h_d(n)= 1-\omega_c/\pi \qquad for\ n=\alpha}$$
  $$\mathbf{= (1/(\pi(n-\alpha))) [\sin(\pi(n-\alpha))-\sin(\omega_c(n-\alpha))] \qquad for\ n\neq\alpha}$$

  **Where $\alpha=(N-1)/2$.**

- Window Sequence

  **d.** For Rectagular Window

  $$\mathbf{W(n)=1 \qquad for\ 0\leq n\leq(N-1)/2}$$
  $$\mathbf{=0 \qquad otherwise}$$

  **e.** For Triangular Window

  $$\mathbf{W(n)=2n/(N-1) \qquad for\ 0\leq n\leq(N-1)/2}$$
  $$\mathbf{=2-\{2n/(N-1) \qquad for\ (N-1)/2\leq n\leq(N-1)}$$

  **f.** For Keiser Window

  $$\mathbf{W(n)=I_0\ [\beta((N-1)^2/4-[n-(N-1)/2]^2)^{1/2}]/I_0[\beta(N-1)/2]} \quad \text{for } 0\leq n\leq(N-1)$$
  $$\mathbf{=0 \qquad otherwise}$$

  where $\beta$ is the variable parameter whose choice control the tradeoff between side lobe amplitude and side lobe width.

  $4<\beta[(N-1)/2]<9$

- The Impulse Response

  $h(n)=h_d(n)*w(n)$

4. Convolve input sequence with truncated impulse response

   $x(n)*h(n)$.

**PROGRAM: -**

## MATLAB Program to generate 'FIR Filter-Low Pass' Coefficients using FIR1

```
% FIR Low  pass filters using rectangular, triangular and kaiser windows

% sampling rate - 8000

order = 30;

cf=[500/4000,1000/4000,1500/4000];     cf--> contains set of cut-off  frequencies[Wc ]

% cutoff frequency - 500

b_rect1=fir1(order,cf(1),boxcar(31));          Rectangular

b_tri1=fir1(order,cf(1),bartlett(31)); Triangular

b_kai1=fir1(order,cf(1),kaiser(31,8));         Kaisar [Where 8-->Beta Co-efficient]

% cutoff frequency - 1000

b_rect2=fir1(order,cf(2),boxcar(31));

b_tri2=fir1(order,cf(2),bartlett(31));

b_kai2=fir1(order,cf(2),kaiser(31,8));

% cutoff frequency - 1500

b_rect3=fir1(order,cf(3),boxcar(31));

b_tri3=fir1(order,cf(3),bartlett(31));

b_kai3=fir1(order,cf(3),kaiser(31,8));

fid=fopen('FIR_lowpass_rectangular.txt','wt');

fprintf(fid,'\t\t\t\t\t\t%s\n','Cutoff -400Hz');
fprintf(fid,'\nfloat b_rect1[31]={');

fprintf(fid,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,\n',b_rect1);

fseek(fid,-1,0);

fprintf(fid,'};');

fprintf(fid,'\n\n\n\n');

fprintf(fid,'\t\t\t\t\t\t%s\n','Cutoff -800Hz');

fprintf(fid,'\nfloat b_rect2[31]={');

fprintf(fid,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,\n',b_rect2);

fseek(fid,-1,0);

fprintf(fid,'};');

fprintf(fid,'\n\n\n\n');

fprintf(fid,'\t\t\t\t\t\t%s\n','Cutoff -1200Hz');

fprintf(fid,'\nfloat b_rect3[31]={');

fprintf(fid,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,\n',b_rect3);

fseek(fid,-1,0);

fprintf(fid,'};');

fclose(fid);

winopen('FIR_highpass_rectangular.txt');
```

## MATLAB Program to generate 'FIR Filter-High Pass' Coefficients using FIR1

```
% FIR High pass filters using rectangular, triangular and kaiser windows

% sampling rate - 8000
order = 30;

cf=[400/4000,800/4000,1200/4000];                    ;cf--> contains set of cut-off
frequencies[Wc]

% cutoff frequency - 400
b_rect1=fir1(order,cf(1),'high',boxcar(31));
b_tri1=fir1(order,cf(1),'high',bartlett(31));
b_kai1=fir1(order,cf(1),'high',kaiser(31,8)); Where Kaiser(31,8)--> '8'defines the value of
'beta'.

% cutoff frequency - 800
b_rect2=fir1(order,cf(2),'high',boxcar(31));
b_tri2=fir1(order,cf(2),'high',bartlett(31));
b_kai2=fir1(order,cf(2),'high',kaiser(31,8));

% cutoff frequency - 1200
b_rect3=fir1(order,cf(3),'high',boxcar(31));
b_tri3=fir1(order,cf(3),'high',bartlett(31));
b_kai3=fir1(order,cf(3),'high',kaiser(31,8));

fid=fopen('FIR_highpass_rectangular.txt','wt');
fprintf(fid,'\t\t\t\t\t\t%s\n','Cutoff -400Hz');
fprintf(fid,'\nfloat b_rect1[31]={');
fprintf(fid,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,\n',b_rect1);
fseek(fid,-1,0);
fprintf(fid,'};');

fprintf(fid,'\n\n\n\n');
fprintf(fid,'\t\t\t\t\t\t%s\n','Cutoff -800Hz');
fprintf(fid,'\nfloat b_rect2[31]={');
fprintf(fid,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,\n',b_rect2);
fseek(fid,-1,0);
fprintf(fid,'};');

fprintf(fid,'\n\n\n\n');
fprintf(fid,'\t\t\t\t\t\t%s\n','Cutoff -1200Hz');
fprintf(fid,'\nfloat b_rect3[31]={');
fprintf(fid,'%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,\n',b_rect3);
fseek(fid,-1,0);
fprintf(fid,'};');

fclose(fid);
winopen('FIR_highpass_rectangular.txt');
```

***T.1 : Matlab generated Coefficients for FIR Low Pass Kaiser filter:***

**Cutoff -500Hz**

```
float b_kai1[31]={-0.000019,-0.000170,-0.000609,-0.001451,-0.002593,-0.003511,-
0.003150,0.000000,0.007551,0.020655,0.039383,0.062306,0.086494,0.108031,0.122944,
0.128279,0.122944,0.108031,0.086494,0.062306,0.039383,0.020655,0.007551,0.000000,
-0.003150,-0.003511,-0.002593,-0.001451,-0.000609,-0.000170,-0.000019};
```

**Cutoff -1000Hz**

```
float b_kai2[31]={-0.000035,-0.000234,-0.000454,0.000000,0.001933,0.004838,0.005671,
-0.000000,-0.013596,-0.028462,-0.029370,0.000000,0.064504,0.148863,0.221349,0.249983,
0.221349,0.148863,0.064504,0.000000,-0.029370,-0.028462,-0.013596,-0.000000,0.005671,
0.004838,0.001933,0.000000,-0.000454,-0.000234, -0.000035};
```

**Cutoff -1500Hz**

```
float b_kai3[31]={-0.000046,-0.000166,0.000246,0.001414,0.001046,-0.003421,-0.007410,
0.000000,0.017764,0.020126,-0.015895,-0.060710,-0.034909,0.105263,0.289209,0.374978,
0.289209,0.105263,-0.034909,-0.060710,-0.015895,0.020126,0.017764,0.000000,-0.007410,
-0.003421,0.001046,0.001414,0.000246,-0.000166, -0.000046};
```

***T.2 :Matlab generated Coefficients for FIR Low Pass Rectangular filter***

**Cutoff -500Hz**

```
float b_rect1[31]={-0.008982,-0.017782,-0.025020,-0.029339,-0.029569,-0.024895,
-0.014970,0.000000,0.019247,0.041491,0.065053,0.088016,0.108421,0.124473,0.134729,
0.138255,0.134729,0.124473,0.108421,0.088016,0.065053,0.041491,0.019247,0.000000,
-0.014970,-0.024895,-0.029569,-0.029339,-0.025020,-0.017782,-0.008982};
```

**Cutoff -1000Hz**

```
float b_rect2[31]={-0.015752,-0.023869,-0.018176,0.000000,0.021481,0.033416,0.026254,-
0.000000,-0.033755,-0.055693,-0.047257,0.000000,0.078762,0.167080,0.236286,0.262448,
0.236286,0.167080,0.078762,0.000000,-0.047257,-0.055693,-0.033755,-0.000000,0.026254,
0.033416,0.021481,0.000000,-0.018176,-0.023869,-0.015752};
```

**Cutoff -1500Hz**

```
float b_rect2[31]={-0.020203,-0.016567,0.009656,0.027335,0.011411,-0.023194,-0.033672,
0.000000,0.043293,0.038657,-0.025105,-0.082004,-0.041842,0.115971,0.303048,0.386435,
0.303048,0.115971,-0.041842,-0.082004,-0.025105,0.038657,0.043293,0.000000,-0.033672,
-0.023194,0.011411,0.027335,0.009656,-0.016567,-0.020203};
```

***T.3 : Matlab generated Coefficients for FIR Low Pass Triangular filter***

**Cutoff -500Hz**
```
float b_tri1[31]={0.000000,-0.001185,-0.003336,-0.005868,-0.007885,-0.008298,-0.005988,
0.000000,0.010265,0.024895,0.043368,0.064545,0.086737,0.107877,0.125747,0.138255,
0.125747,0.107877,0.086737,0.064545,0.043368,0.024895,0.010265,0.000000,-0.005988,
-0.008298,-0.007885,-0.005868,-0.003336,-0.001185,0.000000};
```

**Cutoff -1000Hz**
```
float b_tri2[31]={0.000000,-0.001591,-0.002423,0.000000,0.005728,0.011139,0.010502,
-0.000000,-0.018003,-0.033416,-0.031505,0.000000,0.063010,0.144802,0.220534,0.262448,
0.220534,0.144802,0.063010,0.000000,-0.031505,-0.033416,-0.018003,-0.000000,0.010502,
0.011139,0.005728,0.000000,-0.002423,-0.001591,0.000000};
```

**Cutoff -1500Hz**
```
float b_tri3[31]={0.000000,-0.001104,0.001287,0.005467,0.003043,-0.007731,-0.013469,
0.000000,0.023089,0.023194,-0.016737,-0.060136,-0.033474,0.100508,0.282844,0.386435,
0.282844,0.100508,-0.033474,-0.060136,-0.016737,0.023194,0.023089,0.000000,-0.013469,
-0.007731,0.003043,0.005467,0.001287,-0.001104,0.000000};
```

***T.1 : MATLAB generated Coefficients for FIR High Pass Kaiser filter:***

**Cutoff -400Hz**
```
float b_kai1[31]={0.000050,0.000223,0.000520,0.000831,0.000845,-0.000000,-0.002478,
-0.007437,-0.015556,-0.027071,-0.041538,-0.057742,-0.073805,-0.087505,-0.096739,
0.899998,-0.096739,-0.087505,-0.073805,-0.057742,-0.041538,-0.027071,-0.015556,
-0.007437,-0.002478,-0.000000,0.000845,0.000831,0.000520,0.000223,0.000050};
```

**Cutoff -800Hz**
```
float b_kai2[31]={0.000000,-0.000138,-0.000611,-0.001345,-0.001607,-0.000000,0.004714,
0.012033,0.018287,0.016731,0.000000,-0.035687,-0.086763,-0.141588,-0.184011,0.800005,
-0.184011,-0.141588,-0.086763,-0.035687,0.000000,0.016731,0.018287,0.012033,0.004714,
-0.000000,-0.001607,-0.001345,-0.000611,-0.000138,0.000000};
```

**Cutoff -1200Hz**
```
float b_kai3[31]={-0.000050,-0.000138,0.000198,0.001345,0.002212,-0.000000,-0.006489,
-0.012033,-0.005942,0.016731,0.041539,0.035687,-0.028191,-0.141589,-0.253270,0.700008,
-0.253270,-0.141589,-0.028191,0.035687,0.041539,0.016731,-0.005942,-0.012033,-0.006489,
-0.000000,0.002212,0.001345,0.000198,-0.000138,-0.000050};
```

***T.2 :MATLAB generated Coefficients for FIR High pass Rectangular filter***

**Cutoff -400Hz**

```
float b_rect1[31]={0.021665,0.022076,0.020224,0.015918,0.009129,-0.000000,-0.011158,
-0.023877,-0.037558,-0.051511,-0.064994,-0.077266,-0.087636,-0.095507,-.100422,0.918834,
-0.100422,-0.095507,-0.087636,-0.077266,-0.064994,-0.051511,-0.037558,-0.023877,
-0.011158,-0.000000,0.009129,0.015918,0.020224,0.022076,0.021665};
```

**Cutoff -800Hz**

```
float b_rect2[31]={0.000000,-0.013457,-0.023448,-0.025402,-0.017127,-0.000000,0.020933,
0.038103,0.043547,0.031399,0.000000,-0.047098,-0.101609,-0.152414,-0.188394,0.805541,
-0.188394,-0.152414,-0.101609,-0.047098,0.000000,0.031399,0.043547,0.038103,0.020933,
-0.000000,-0.017127,-0.025402,-0.023448,-0.013457,0.000000};
```

**Cutoff -1200Hz**

```
float b_rect3[31]={-0.020798,-0.013098,0.007416,0.024725,0.022944,-0.000000,-0.028043,
-0.037087,-0.013772,0.030562,0.062393,0.045842,-0.032134,-0.148349,-0.252386,0.686050,
-0.252386,-0.148349,-0.032134,0.045842,0.062393,0.030562,-0.013772,-0.037087,-0.028043,
-0.000000,0.022944,0.024725,0.007416,-0.013098,-0.020798};
```

***T.3 : MATLAB generated Coefficients for FIR High Pass Triangular filter***

**Cutoff -400Hz**

```
float b_tri1[31]={0.000000,0.001445,0.002648,0.003127,0.002391,-0.000000,-0.004383,
-0.010943,-0.019672,-0.030353,-0.042554,-0.055647,-0.068853,-0.081290,-0.092048,
0.902380,-0.092048,-0.081290,-0.068853,-0.055647,-0.042554,-0.030353,-0.019672,
-0.010943,-0.004383,-0.000000,0.002391,0.003127,0.002648,0.001445,0.000000};
```

**Cutoff -800Hz**

```
float b_tri2[31]={0.000000,-0.000897,-0.003126,-0.005080,-0.004567,-0.000000,0.008373,
0.017782,0.023225,0.018839,0.000000,-0.034539,-0.081287,-0.132092,-0.175834,0.805541,
-0.175834,-0.132092,-0.081287,-0.034539,0.000000,0.018839,0.023225,0.017782,0.008373,
-0.000000,-0.004567,-0.005080,-0.003126,-0.000897,0.000000};
```

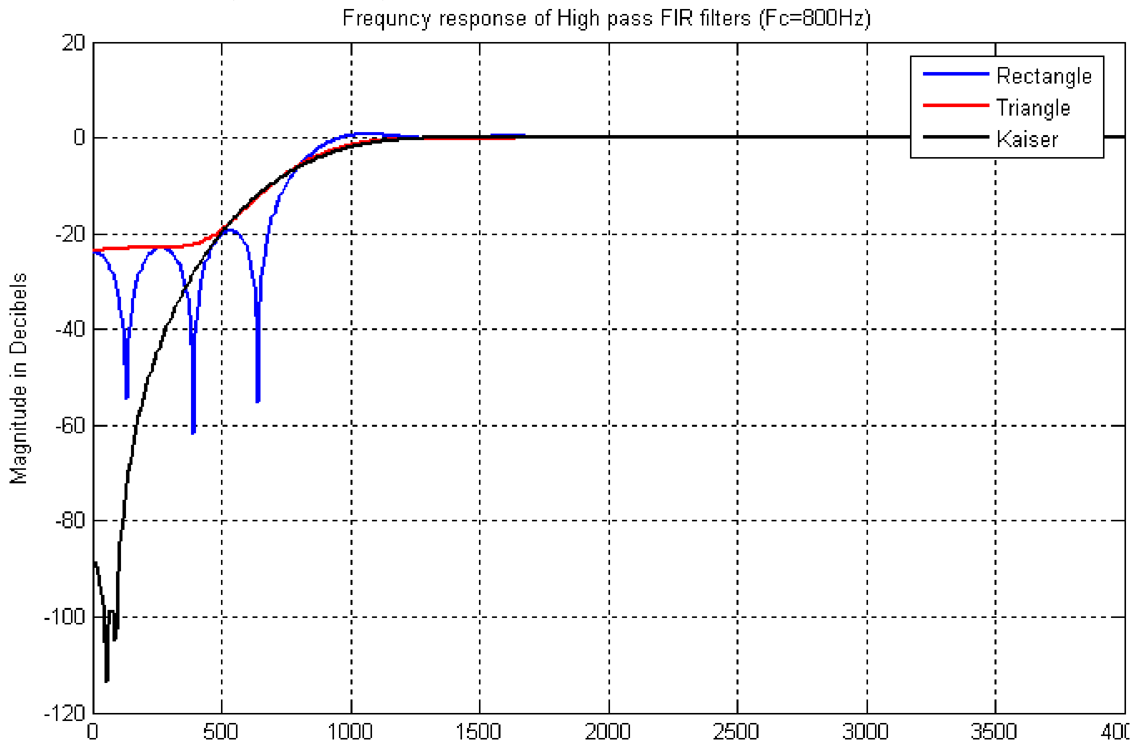
**Cutoff -1200Hz**

```
float b_tri3[31]={0.000000,-0.000901,0.001021,0.005105,0.006317,-0.000000,-0.011581,
-0.017868,-0.007583,0.018931,0.042944,0.034707,-0.026541,-0.132736,-0.243196,0.708287,
-0.243196,-0.132736,-0.026541,0.034707,0.042944,0.018931,-0.007583,-0.017868,-0.011581,
-0.000000,0.006317,0.005105,0.001021,-0.000901,0.000000};
```

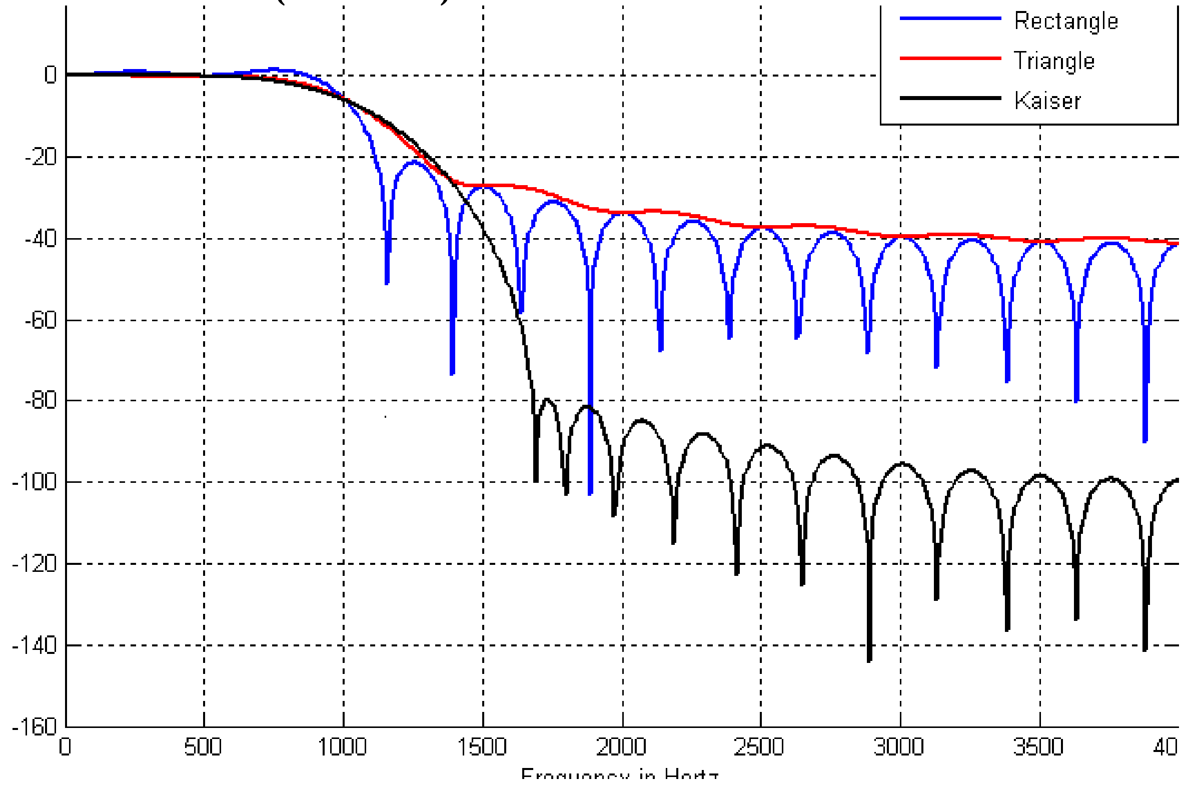
## 5.MODEL GRAPHS:-

## MATLAB GENERATED FREQUENCY RESPONSE

**High Pass FIR filter(Fc= 800Hz).**



**Low Pass FIR filter (Fc=1000Hz)**



**RESULT**: -

**CONCLUSION**

**VIVA QUESTIONS:**

1. What are windowing techniques in FIR filters?

2. Which window is best for FIR filter?

3.Which technique is not used in design os FIR FILETR?

4. Why FIR filters are always stable?

5.Why windows are necessary in FIR filters

**Exp .No: 11** **Date:**

## B.GENERATION OF IIR BUTTERWORTH/CHEBYSHEV FILTER COEFFICIENTS

**AIM: -**To generate IIR filter Butterworth/Chebyshev coefficients

**APPARATUS REQUIRED**: -

MATLAB and PC

**PROGRAM: -**

```
% IIR Low pass Butterworth and Chebyshev filters
% sampling rate - 24000

order = 2;
cf=[2500/12000,8000/12000,1600/12000];

% cutoff frequency - 2500
[num_bw1,den_bw1]=butter(order,cf(1));
[num_cb1,den_cb1]=cheby1(order,3,cf(1));

% cutoff frequency - 8000
[num_bw2,den_bw2]=butter(order,cf(2));
[num_cb2,den_cb2]=cheby1(order,3,cf(2));

fid=fopen('IIR_LP_BW.txt','wt');
fprintf(fid,'\t\t-----------Pass band range: 0-2500Hz----------\n');
fprintf(fid,'\t\t-----------Magnitude response: Monotonic-----\n\n\');
fprintf(fid,'\n float num_bw1[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',num_bw1);
fprintf(fid,'\nfloat den_bw1[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',den_bw1);

fprintf(fid,'\n\n\n\t\t-----------Pass band range: 0-8000Hz----------\n');
fprintf(fid,'\t\t-----------Magnitude response: Monotonic-----\n\n');
fprintf(fid,'\nfloat num_bw2[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',num_bw2);
fprintf(fid,'\nfloat den_bw2[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',den_bw2);

fclose(fid);
winopen('IIR_LP_BW.txt');

fid=fopen('IIR_LP_CHEB Type1.txt','wt');
fprintf(fid,'\t\t-----------Pass band range: 2500Hz----------\n');
fprintf(fid,'\t\t-----------Magnitude response: Rippled (3dB) -----\n\n\');
fprintf(fid,'\nfloat num_cb1[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',num_cb1);
fprintf(fid,'\nfloat den_cb1[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',den_cb1);
fprintf(fid,'\n\n\n\t\t-----------Pass band range: 8000Hz----------\n');
fprintf(fid,'\t\t-----------Magnitude response: Rippled (3dB)-----\n\n');
fprintf(fid,'\nfloat num_cb2[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',num_cb2);
fprintf(fid,'\nfloat den_cb2[9]={');
fprintf(fid,'%f,%f,%f,%f,%f,\n%f,%f,%f,%f};\n',den_cb2);
fclose(fid);
winopen('IIR_LP_CHEB Type1.txt');

%%%%%%%%%%%%%%%%%%
figure(1);
[h,w]=freqz(num_bw1,den_bw1);
w=(w/max(w))*12000;
plot(w,20*log10(abs(h)),'linewidth',2)
hold on
[h,w]=freqz(num_cb1,den_cb1);
w=(w/max(w))*12000;
plot(w,20*log10(abs(h)),'linewidth',2,'color','r')
grid on
legend('Butterworth','Chebyshev Type-1');
xlabel('Frequency in Hertz');
ylabel('Magnitude in Decibels');
title('Magnitude response of Low pass IIR filters (Fc=2500Hz)');
figure(2);
[h,w]=freqz(num_bw2,den_bw2);
w=(w/max(w))*12000;
plot(w,20*log10(abs(h)),'linewidth',2)
hold on
[h,w]=freqz(num_cb2,den_cb2);
w=(w/max(w))*12000;
plot(w,20*log10(abs(h)),'linewidth',2,'color','r')
grid on
legend('Butterworth','Chebyshev Type-1 (Ripple: 3dB)');
xlabel('Frequency in Hertz');
ylabel('Magnitude in Decibels');
title('Magnitude response in the passband');
axis([0 12000 -20 20]);
```

## IIR_CHEB_LP FILTER CO-EFFICIENTS:

| Co-Efficients | Fc=2500Hz | | Fc=800Hz | | Fc=8000Hz | |
|---|---|---|---|---|---|---|
| | **Floating Point Values** | **Fixed Point Values (Q15)** | **Floating Point Values** | **Fixed Point Values (Q15)** | **Floating Point Values** | **Fixed Point Values (Q15)** |
| **B0** | 0.044408 | 1455 | 0.005147 | 168 | 0.354544 | 11617 |
| **B1** | 0.088815 | 1455[B1/2] | 0.010295 | 168[B1/2] | 0.709088 | 11617 [B1/2] |
| **B2** | 0.044408 | 1455 | 0.005147 | 168 | 0.354544 | 11617 |
| **A0** | 1.000000 | 32767 | 1.000000 | 32767 | 1.000000 | 32767 |
| **A1** | - 1.412427 | -23140 [A1/2] | -1.844881 | -30225 [A1/2] | 0.530009 | 8683[A1/2] |
| **A2** | 0.663336 | 21735 | 0.873965 | 28637 | 0.473218 | 15506 |

**Note: We have Multiplied Floating Point Values with 32767($2^{15}$) to get Fixed Point Values.**

## IIR_BUTTERWORTH_LP FILTER CO-EFFICIENTS:

| Co-Efficients | Fc=2500Hz | | Fc=800Hz | | Fc=8000Hz | |
|---|---|---|---|---|---|---|
| | **Floating Point Values** | **Fixed Point Values (Q15)** | **Floating Point Values** | **Fixed Point Values (Q15)** | **Floating Point Values** | **Fixed Point Values (Q15)** |
| **B0** | 0.072231 | 2366 | 0.009526 | 312 | 0.465153 | 15241 |
| **B1** | 0.144462 | 2366[B1/2] | 0.019052 | 312[B1/2] | 0.930306 | 15241 [B1/2] |
| **B2** | 0.072231 | 2366 | 0.009526 | 312 | 0.465153 | 15241 |
| **A0** | 1.000000 | 32767 | 1.000000 | 32767 | 1.000000 | 32767 |
| **A1** | - 1.109229 | -18179 [A1/2] | -1.705552, | -27943 [A1/2] | 0.620204 | 10161 [A1/2] |
| **A2** | 0.398152 | 13046 | 0.743655 | 24367 | 0.240408 | 7877 |

**Note: We have Multiplied Floating Point Values with 32767($2^{15}$) to get Fixed Point Values.**

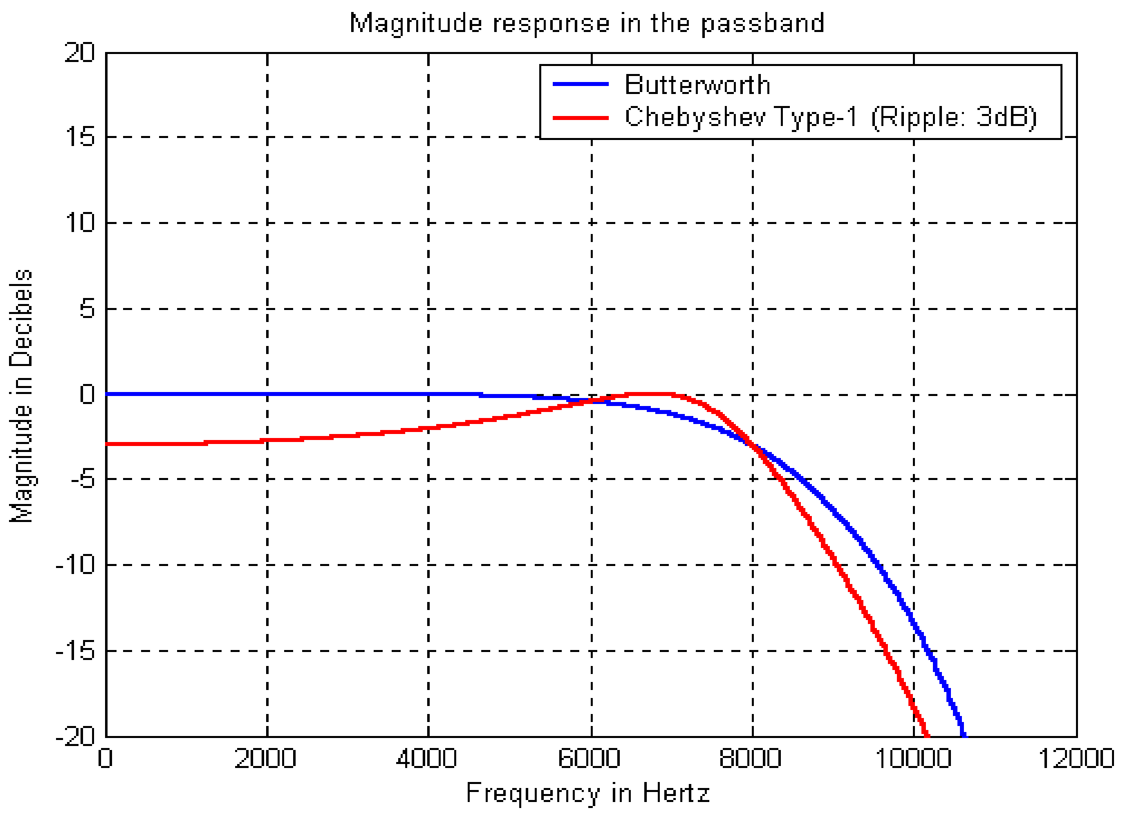
**IIR_CHEB_HP FILTER CO-EFFICIENTS:**

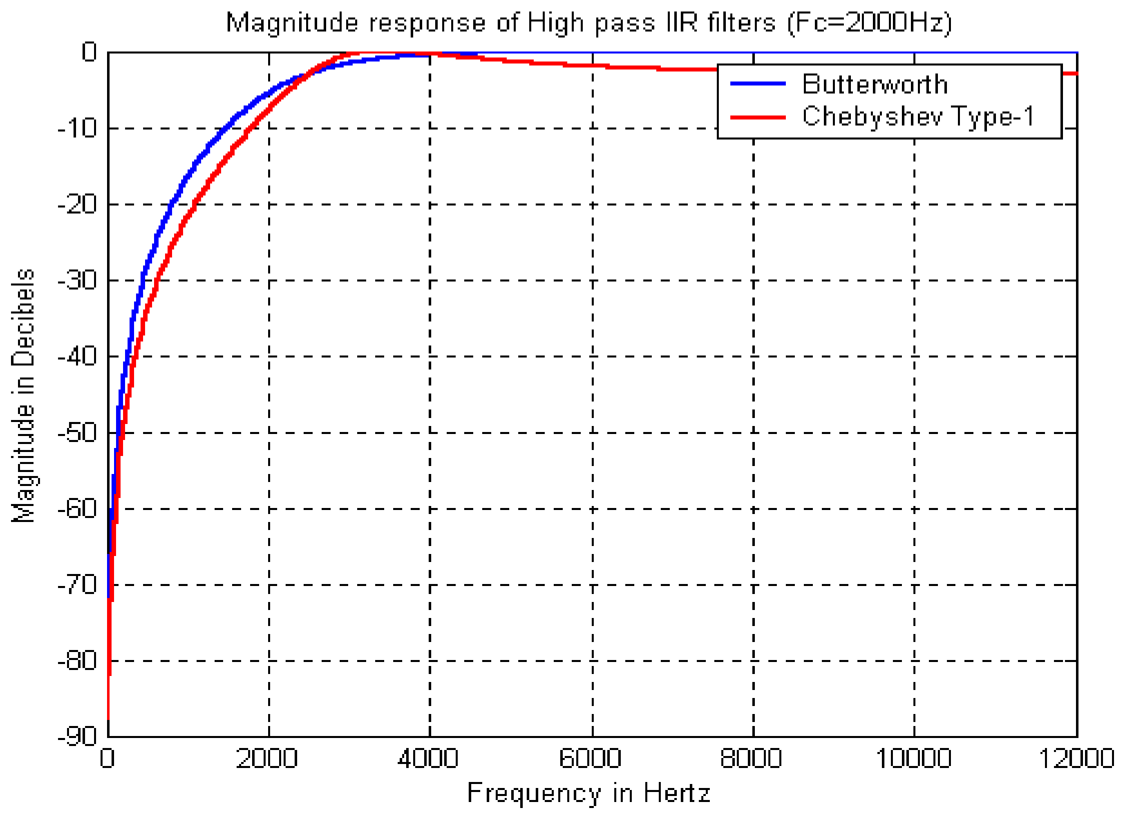
| Co-Efficients | Fc=2500Hz | | Fc=4000Hz | | Fc=7000Hz | |
|---|---|---|---|---|---|---|
| | Floating Point Values | Fixed Point Values (Q15) | Floating Point Values | Fixed Point Values (Q15) | Floating Point Values | Fixed Point Values (Q15) |
| **B0** | 0.388513 | 12730 | 0.282850 | 9268 | 0.117279 | 3842 |
| **B1** | -0.777027 | -12730[B1/2] | -0.565700 | -9268[B1/2] | -0.234557 | -3842[B1/2] |
| **B2** | 0.388513 | 12730 | 0.282850 | 9268 | 0.117279 | 3842 |
| **A0** | 1.000000 | 32767 | 1.000000 | 32767 | 1.000000 | 32767 |
| **A1** | -1.118450 | -18324[A1/2] | -0.451410 | -7395[A1/2] | 0.754476 | 12360[A1/2] |
| **A2** | 0.645091 | 21137 | 0.560534 | 18367 | 0.588691 | 19289 |

**Note: We have Multiplied Floating Point Values with 32767($2^{15}$) to get Fixed Point Values.**

**IIR_BUTTERWORTH_HP FILTER CO-EFFICIENTS:**

| Co-Efficients | Fc=2500Hz | | Fc=4000Hz | | Fc=7000Hz | |
|---|---|---|---|---|---|---|
| | Floating Point Values | Fixed Point Values (Q15) | Floating Point Values | Fixed Point Values (Q15) | Floating Point Values | Fixed Point Values (Q15) |
| **B0** | 0.626845 | 20539 | 0.465153 | 15241 | 0.220195 | 7215 |
| **B1** | -1.253691 | -20539 [B1/2] | -0.930306 | -15241 [B1/2] | -0.440389 | -7215 [B1/2] |
| **B2** | 0.626845 | 20539 | 0.465153 | 15241 | 0.220195 | 7215 |
| **A0** | 1.000000 | 32767 | 1.000000 | 32767 | 1.000000 | 32767 |
| **A1** | -1.109229 | -18173 [A1/2] | -0.620204 | -10161 [A1/2] | 0.307566 | 5039 [A1/2] |
| **A2** | 0.398152 | 13046 | 0.240408 | 7877 | 0.188345 | 6171 |


Magnitude response in the passband
Butterworth
Chebyshev Type-1 (Ripple: 3dB)
Magnitude in Decibels
20
15
10
5
0
-5
-10
-15
-20
0
2000
4000
6000
8000
10000
12000
Frequency in Hertz


Magnitude response of High pass IIR filters (Fc=2000Hz)
Butterworth
Chebyshev Type-1
Magnitude in Decibels
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
0
2000
4000
6000
8000
10000
12000
Frequency in Hertz

**RESULT**: -

**CONCLUSION**

**VIVA QUESTIONS:**

1. What is bilinear transformation in IIIR filter design?

2. Which  method is not suitable for designing IIR filters?

3. What are the steps involved to design digital filter using bilinear transformations?

4. What are the limitations of impulse invariance method

5. What are the differences between IIR and FIR filters

**Exp .No: 12** **Date:**

# DECIMATION AND INTERPOLATION

**AIM:** To write a MATLAB program to decimation and interpolation of a give sequence

**SOFTWARE REQUIRED:**

- PC and MATLAB software

**PROCEDURE:**

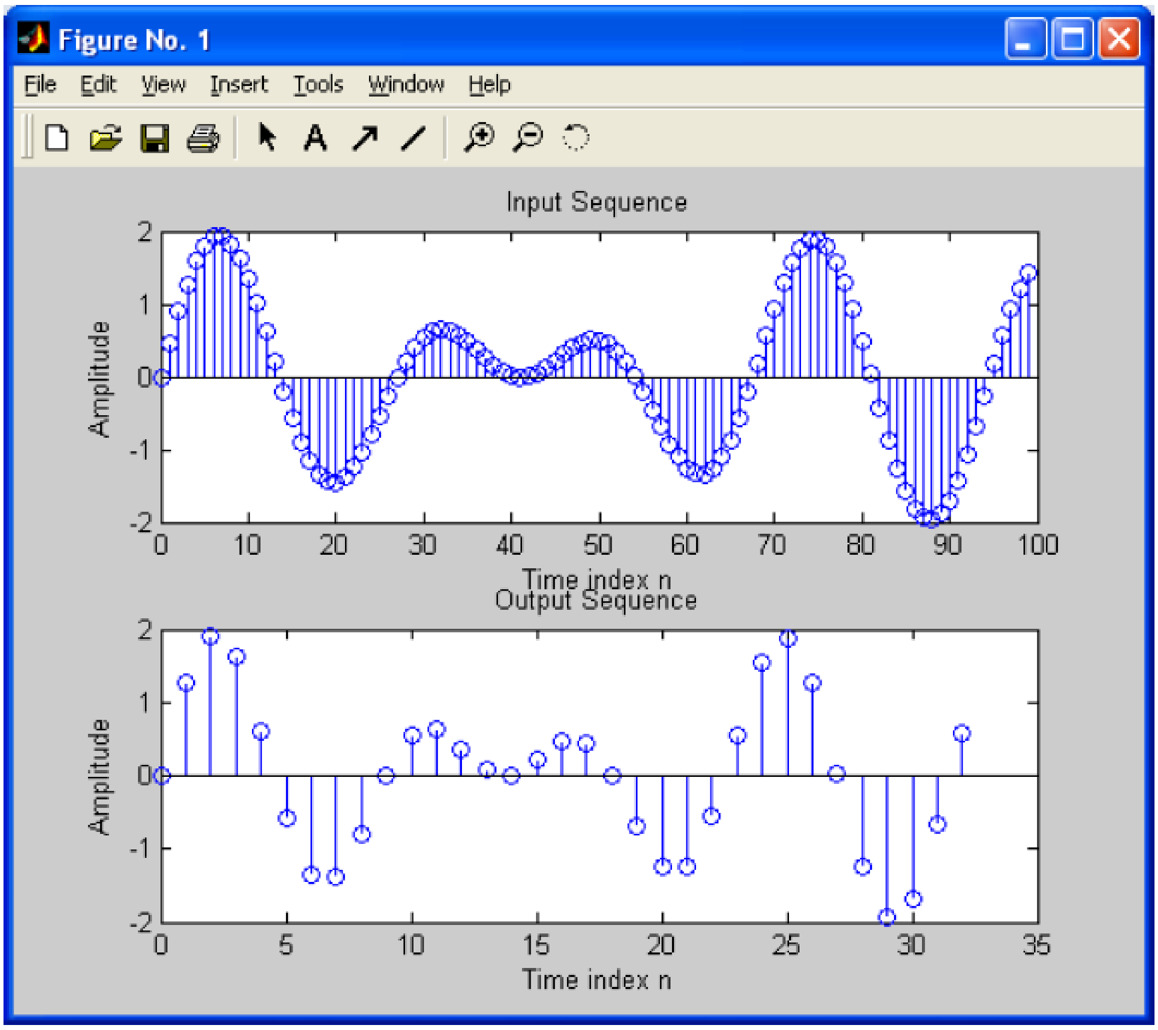
- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window
-

**PROGRAM DECIMATION:**

```
clc;
close all;
clear all;
M = input('enter Down-sampling factor : ');
N = input('enter number of samples :');
n = 0:N-1;
x = sin(2*pi*0.043*n) + sin(2*pi*0.031*n);
y = decimate(x,M,'fir');
subplot(2,1,1);
stem(n,x(1:N));
title('Input Sequence');
xlabel('Time index n');
ylabel('Amplitude');
subplot(2,1,2);
m = 0:(N/M)-1
title('Output Sequence');
xlabel('Time index n');ylabel('Amplitude');
```

**INPUT:**
enter Down-sampling factor : 3
enter number of samples :100

**OUTPUT**



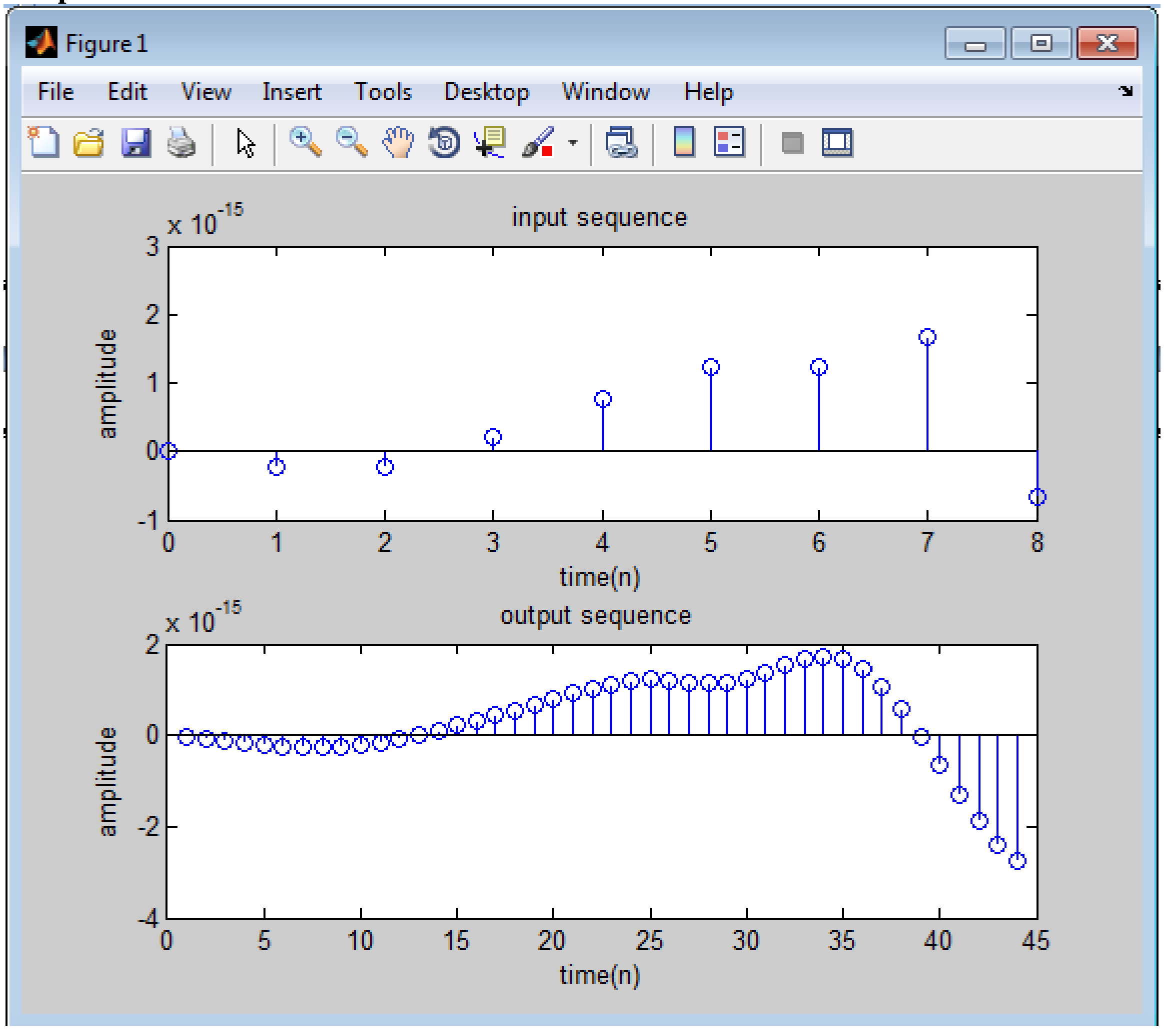
## PROGRAM INTERPOLATION

```
clc;
clear all;
close all;
L=input('enter the upsampling factor');
N=input('enter the length of the input signal'); % Length should be greater
than 8
f1=input('enter the frequency of first sinusodal');
f2=input('enter the frequency of second sinusodal');
 n=0:N-1;
 x=sin(2*pi*f1*n)+sin(2*pi*f2*n);
 y=interp(x,L);
 subplot(2,1,1)
 stem(n,x(1:N))
title('input sequence');
xlabel('time(n)');
ylabel('amplitude');
subplot(2,1,2)
 m=0:N*L-1;
 stem(m,y(1:N*L))
 title('output sequence ');
 xlabel('time(n)');
ylabel('amplitude');
```

**INPUT:**

```
 enter the upsampling factor = 5
enter the length of the input signal = 9
enter the frequency of first sinusoidal = 0.1
enter the frequency of second sinusoidal = 0.3
```

**Output Waveforms:**



**RESULT**: -

**CONCLUSION**

**VIVA QUESTIONS:**

1. Explain about multi rate digital signal processing.

2. List the Applications of multi rate digital signal processing.

3. Define interpolation.

4. Define decimation.

5. Define aliasing.

# HARDWARE EXPERIMENTS (USING DSP PROCESSOR)

**Exp .No:01** **Date:**

## LINEAR AND CIRCULAR CONVOLUTIONS

**AIM:** To write a C- program to find linear convolution and circular convolution of given sequences.

**Procedure to Work on Code Composer Studio**

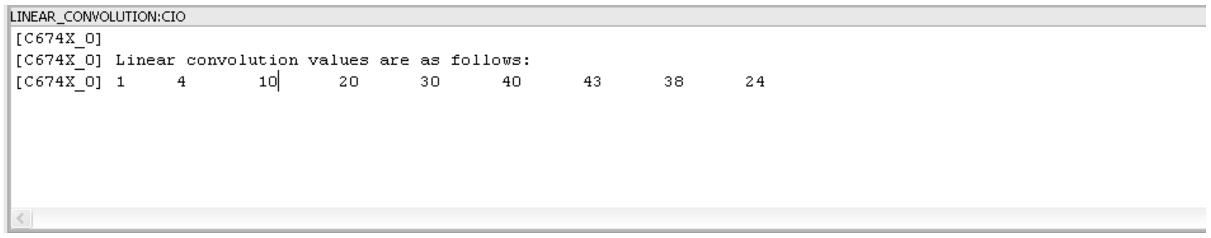
To create the New Project
Project→ New CCS Project (Give name to project with location to save or use default location)
Select project type→ Executable
Device Family→C6000
Variant→C674xFloating-point DSP
Connection →Texas Instruments XDS100V2USB Emulator
Click on Empty Project then Finish
To create a Source file
File →New→ Source file (Save & give file name, Eg: sum.c).Click on Finish
Write the C-Program To build the program project →Build All
After Build Finished without errors, Now DSP kit is turned ON
Debug the Program after loading is done
Run the Program and verify the output

**PROGRAM LINEAR CONVOLUTION:**

```
#include<stdio.h>
#define LENGHT1 6 /*Lenght of i/p samples sequence*/
#define LENGHT2 4 /*Lenght of impulse response Co-efficients */
int x[2*LENGHT1-1]={1,2,3,4,5,6,0,0,0,0,0}; /*Input Signal Samples*/
int h[2*LENGHT1-1]={1,2,3,4,0,0,0,0,0,0,0}; /*Impulse Response Co-efficients*/
int y[LENGHT1+LENGHT2-1];
void main()
{
int i=0,j;
for(i=0;i<(LENGHT1+LENGHT2-1);i++)
{
y[i]=0;
for(j=0;j<=i;j++)
y[i]+=x[j]*h[i-j];
}
printf("\nLinear convolution values are as follows:\n");
for(i=0;i<(LENGHT1+LENGHT2-1);i++)
printf("%d\t",y[i]);
}
```

**OUTPUT:**

- In this program x[n]={1,2,3,4,5,6}, & h[n]={1,2,3,4}.
- So output of linear convolution is y[k]={1,4,10,20,30,40,43,38,24};

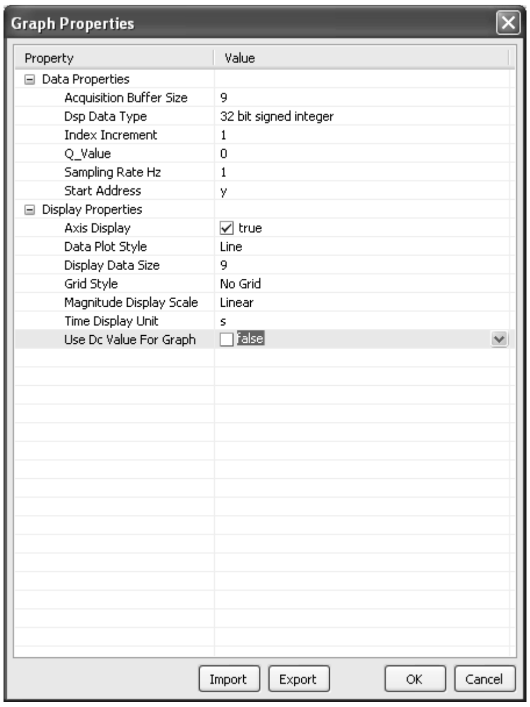


```
LINEAR_CONVOLUTION:CIO
[C674X_0]
[C674X_0] Linear convolution values are as follows:
[C674X_0] 1     4       10      20      30      40      43      38      24
```

- To view output graphically,

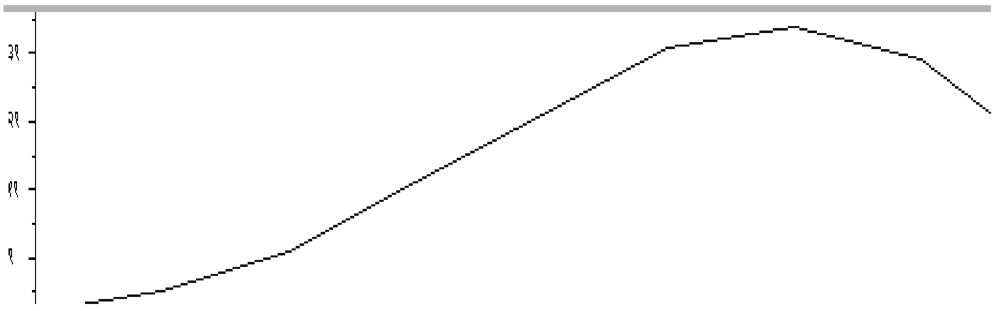
**Tools→ graph →single time**.

Graph setting & graph as follows



Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 9 |
| Dsp Data Type | 32 bit signed integer |
| Index Increment | 1 |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address | y |
| ⊟ Display Properties | |
| Axis Display | ✓ true |
| Data Plot Style | Line |
| Display Data Size | 9 |
| Grid Style | No Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | s |
| Use Dc Value For Graph | ☐ false |

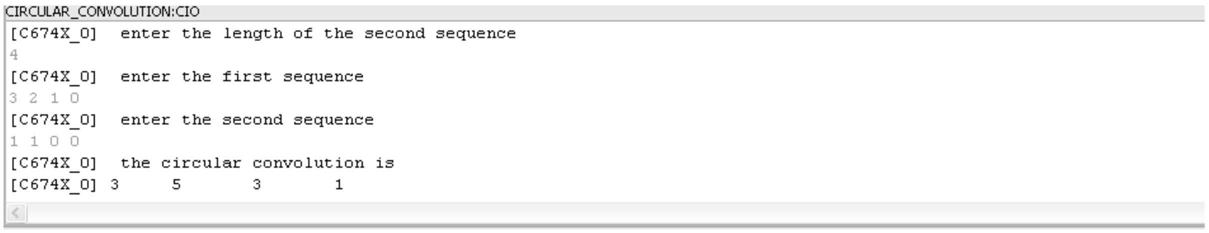


## CIRCULAR CONVOLUTION

```
#include<stdio.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]);
}
```

## OUTPUT

In this program x[4]={3,2,1,0}, & h[4]={1,1.0.0} as you enter.
So output of circular convolution is y[k]={3,5,3,1};



```
CIRCULAR_CONVOLUTION:CIO
[C674X_0]  enter the length of the second sequence
4
[C674X_0]  enter the first sequence
3 2 1 0
[C674X_0]  enter the second sequence
1 1 0 0
[C674X_0]  the circular convolution is
[C674X_0] 3     5      3      1
```

**RESULT:**

**CONCLUSION**

## VIVA QUESTION:

1. Explain the significance of convolution.

2. Define linear convolution.

3. Why linear convolution is called as a periodic convolution?

4. Why zero padding is used in linear convolution?

5. What are the four steps to find linear convolution?

**Exp .No: 02** **Date:**

## AUTO-CORRELATION & CROSS-CORRELATION

**AIM:** Auto correlation and Cross-correlation of a given sequence and verification of its properties

**APPARATUS REQUIRED**

- CODE COMPOSER STUDIO(CCS)V5
- TMS320C6745 DSP TRAINER KIT

**PROCEDURE**

1. To create the New Project
2. Project→ New CCS Project (Give name to project with location to save or use default
3. location)
4. Select project type→ Executable
5. Device Family→C6000
6. Variant→C674xFloating-point DSP
7. Connection →Texas Instruments XDS100V2USB Emulator
8. Click on Empty Project then Finish
9. To create a Source file
10. File →New→ Source file (Save &amp; give file name, Eg: sum.c).Click on Finish
11. Write the C-Program To build the program project →Build All
12. After Build Finished without errors, Now DSP kit is turned ON
13. Debug the Program after loading is done
14. Run the Program and verify the output
15. Program asks for values of x[N], you have to enter values as 1st real & 2nd imaginary for all N no of values.

## PROGRAM AUTO-CORRELATION:

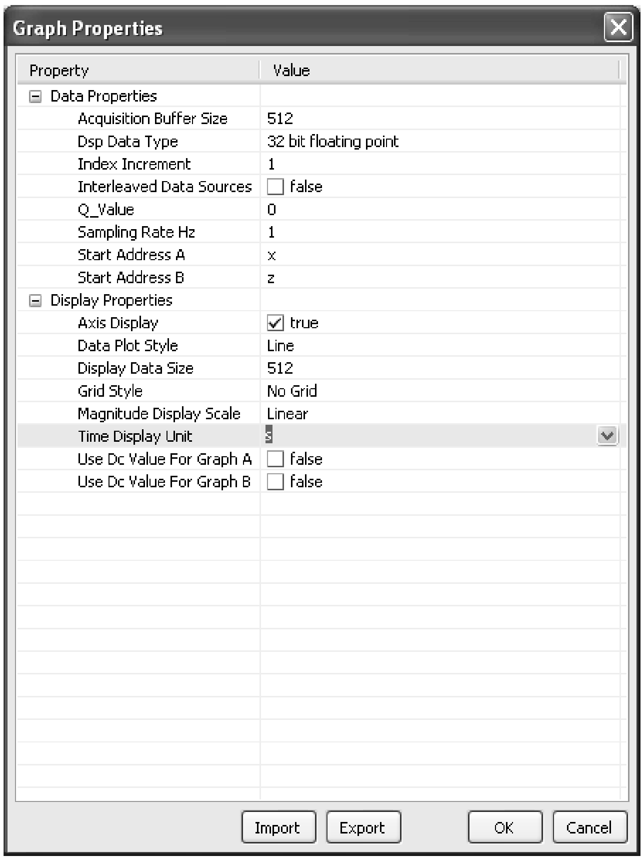
```
#include<stdio.h>
#include<math.h>
#define PI 3.14
#define PTS 512

void delay1();
void sinewave(float fm1, float fc1, int no, int result);

float x[PTS];   //INPUT ARRAY
float y[PTS];
float z[PTS];//OUTPUT ARRAY

void main()
{

	int i,j;
	for (i = 0 ; i < PTS ; i++)
	{
		sinewave(100, 8000, i, 0);
		//x[i] = sin(2 * PI * 100 * i / 8000.0);
		y[i]=0.0;
	}

		for(i=0 ; i< PTS ; i++)
		{
			for(j = 1 ; j < (PTS-i+1) ;j++)
			{
				y[i]=y[i]+x[j]*x[j+i-1];
			}
			z[i]=y[i]/(PTS);
			//printf("\n%f",z[i]);

		}

}


void sinewave(float fm1, float fc1, int no, int result)
{

	float m;
	float n;

			delay1();
			delay1();
			m = 6.28 * fm1 * no;
			delay1();
			delay1();
			n = (m / fc1);
			delay1();
			delay1();
			if(result == 0)
			{
				delay1();
```

```
                                        x[no]= sin(n);
                                        delay1();
                              }
                                        delay1();
                                        delay1();

}

void delay1()
{
          int y=150;
          while(y!=0)
                    y--;

}
```

## OUTPUT

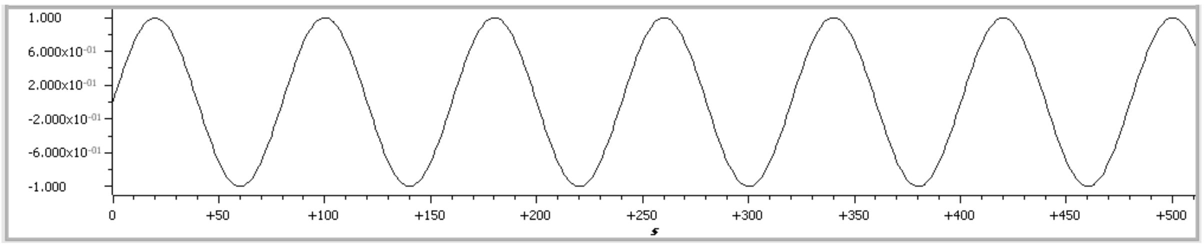
### 1. For auto correlation

- We generated samplesine wave as a input named as x.
- Final output of auto correlation named as z.
- To view input / output graphically,
  Select **Tool→ graph →Dual time**

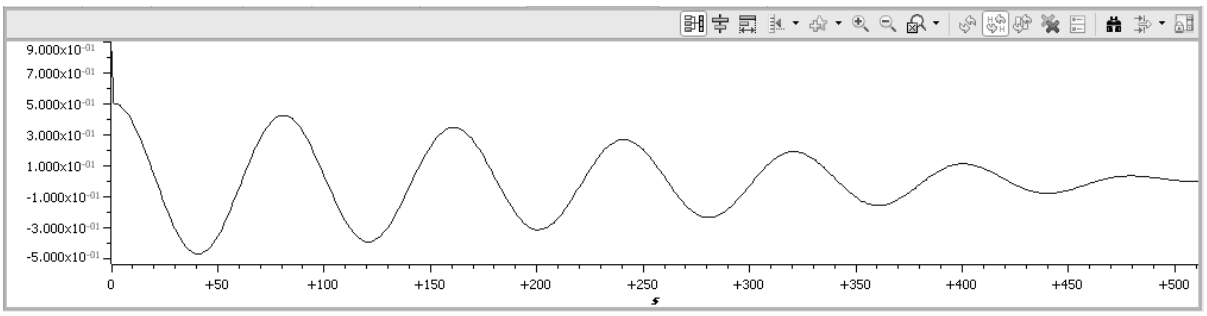

Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 512 |
| Dsp Data Type | 32 bit floating point |
| Index Increment | 1 |
| Interleaved Data Sources | ☐ false |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address A | x |
| Start Address B | z |
| ⊟ Display Properties | |
| Axis Display | ✓ true |
| Data Plot Style | Line |
| Display Data Size | 512 |
| Grid Style | No Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | s |
| Use Dc Value For Graph A | ☐ false |
| Use Dc Value For Graph B | ☐ false |

Graph of Input sine wave named as X



Graph of Auto correlation named as Z



## PROGRAM CROSS-CORRELATION

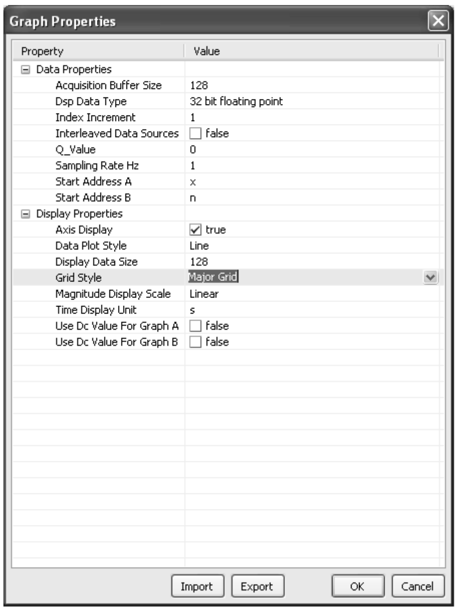
```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<time.h>
#define PI 3.14
#define PTS 128

void delay1();
void sinewave(float fm1, float fc1, int no, int result);

float x[PTS];
float y[PTS];
float z[PTS];
float n[PTS];

void main()
{

        int i,j;
        float xx;
        for (i = 0 ; i < PTS ; i++)
        {
                sinewave(20, 128, i, 0);
                //x[i] = sin(2*PI*i*20/128.0);
                y[i]=0.0;
                n[i]=x[i] + rand() * 10 ;
```

```
      }

            for(i=0;i< PTS ;i++)
            {
                  for(j = 1 ; j < (PTS-i+1) ;j++)
                  {
                        xx = (x[j]*n[j+i-1]);
                        y[i]=(y[i]+xx);
                  }
                  z[i]=y[i]/(PTS);
                  //delay1();
                  //delay1();
                  //printf("\n%f",z[i]);
                  //delay1();
                  //delay1();

            }

}


void sinewave(float fm1, float fc1, int no, int result)
{

      float m;
      float n;

                  delay1();
                  delay1();
                  m = 6.28 * fm1 * no;
                  delay1();
                  delay1();
                  n = (m / fc1);
                  delay1();
                  delay1();
                  if(result == 0)
                  {
                        delay1();
                        x[no]= sin(n);
                        delay1();
                  }
                      delay1();
                        delay1();

}

void delay1()
{
      int y=150000;
      while(y!=0)
            y--;

}
```

## OUTPUT

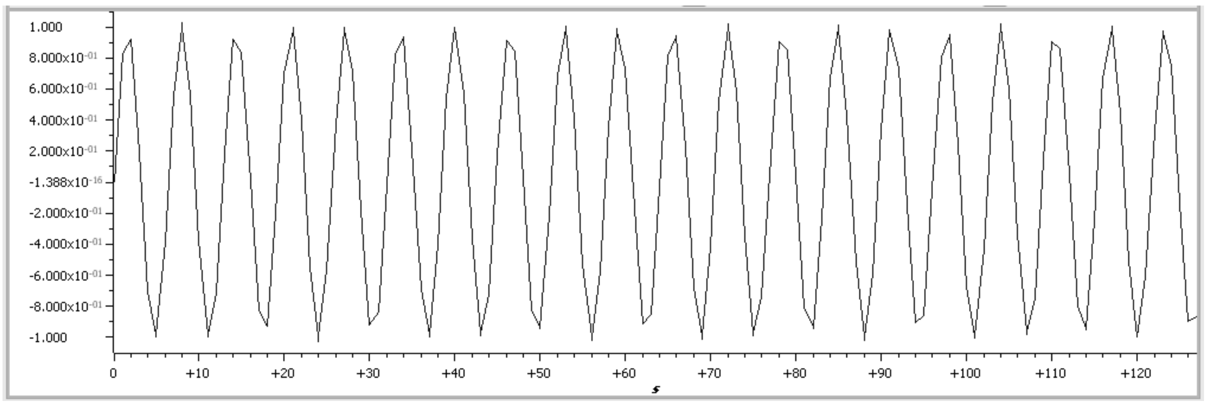
- We generated sample sine wave as a input named as x.
- Then generate noise signal using random function and added with original signal, that
is denoted as n
- Cross-correlation output saved in z variable.
- To view input graphically,

Select **Tool→ graph →Dual time**
Graph setting &Graph

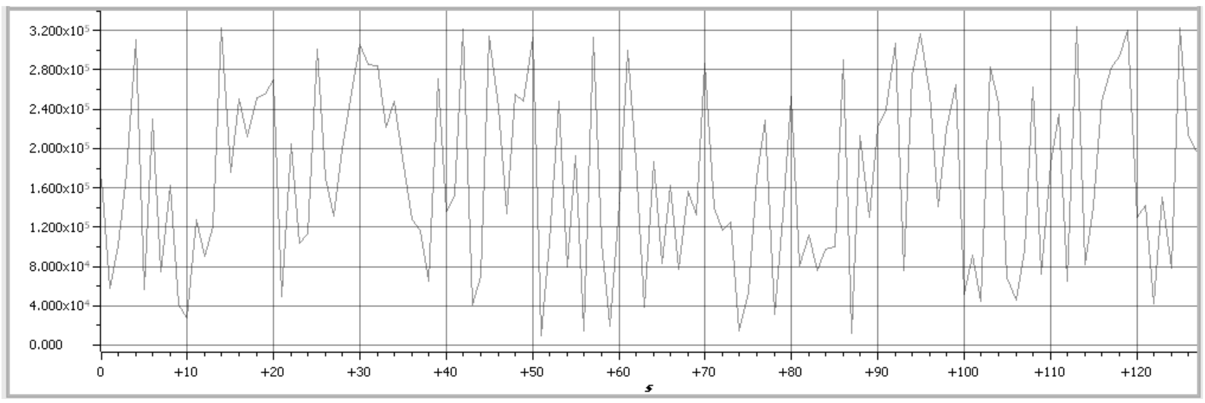

Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 128 |
| Dsp Data Type | 32 bit floating point |
| Index Increment | 1 |
| Interleaved Data Sources | ☐ false |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address A | x |
| Start Address B | n |
| ⊟ Display Properties | |
| Axis Display | ☑ true |
| Data Plot Style | Line |
| Display Data Size | 128 |
| Grid Style | Major Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | s |
| Use Dc Value For Graph A | ☐ false |
| Use Dc Value For Graph B | ☐ false |

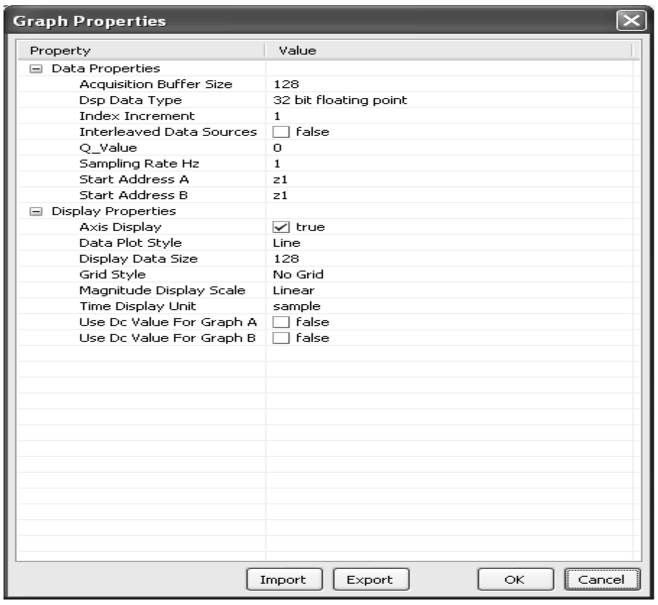
Sample sine wave named as X



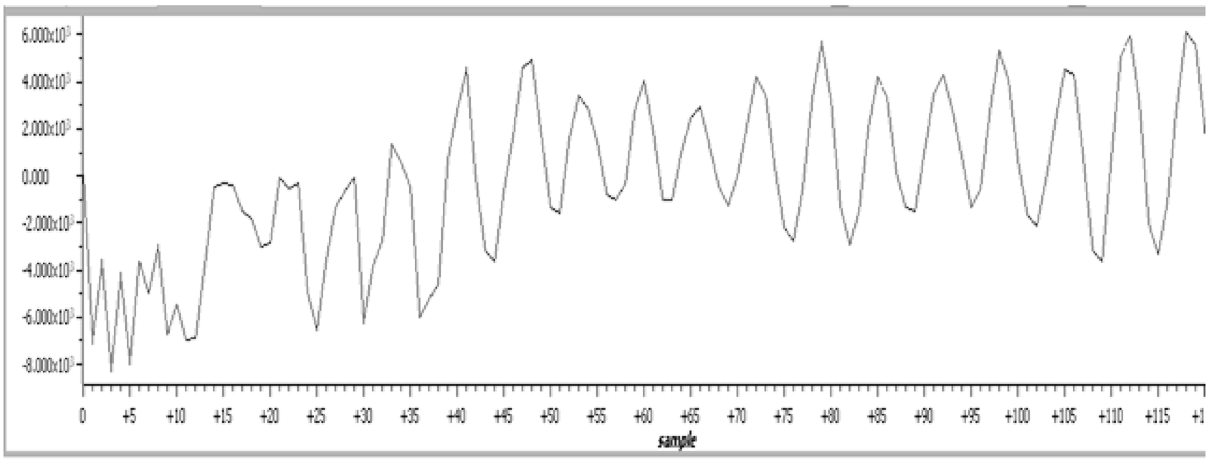
Sample noise signal as n.



To view cross correlation graphically,
Select **Tool→ graph → single time**


Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 128 |
| Dsp Data Type | 32 bit floating point |
| Index Increment | 1 |
| Interleaved Data Sources | ☐ false |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address A | z1 |
| Start Address B | z1 |
| ⊟ Display Properties | |
| Axis Display | ✓ true |
| Data Plot Style | Line |
| Display Data Size | 128 |
| Grid Style | No Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | sample |
| Use Dc Value For Graph A | ☐ false |
| Use Dc Value For Graph B | ☐ false |



**RESULT:**

**CONCLUSION**

## VIVA QUESTION

1. Write mathematical formula to find auto correlation?

2. Define auto correlation?

3. Define cross correlation?

4. Difference between Auto correlation and convolution?

5. Difference between Auto correlation and cross correlation?

**Exp .No: 03** **Date:**

## DFT OF A SIGNAL

**AIM: -** To compute the N (=4/8/16) point DFT of the given sequence.

**APPARATUS REQUIRED**

- CODE COMPOSER STUDIO(CCS)V5
- TMS320C6745 DSP TRAINER KIT

**PROCEDURE**

16. To create the New Project
17. Project→ New CCS Project (Give name to project with location to save or use default
18. location)
19. Select project type→ Executable
20. Device Family→C6000
21. Variant→C674xFloating-point DSP
22. Connection →Texas Instruments XDS100V2USB Emulator
23. Click on Empty Project then Finish
24. To create a Source file
25. File →New→ Source file (Save &amp; give file name, Eg: sum.c).Click on Finish
26. Write the C-Program To build the program project →Build All
27. After Build Finished without errors, Now DSP kit is turned ON
28. Debug the Program after loading is done
29. Run the Program and verify the output
30. Program asks for values of x[N], you have to enter values as 1st real & 2nd imaginary for all N no of values.

    Eg. x[N]={1+0j,1+0j,1+0j,1+0j,1+0j,1+0j,1+0j,1+0j}.
    It enter in program as → 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

**PROGRAM DFT:**

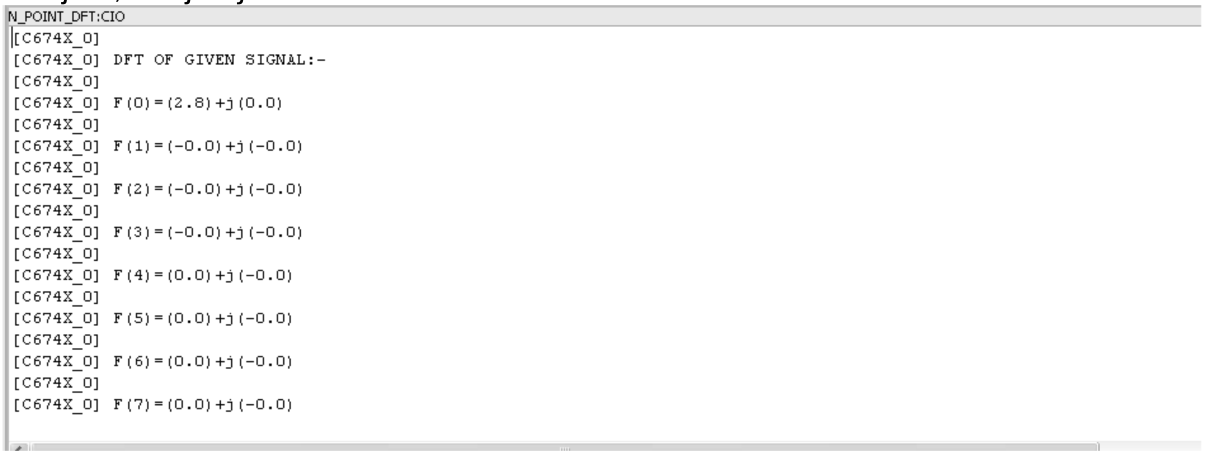
```
#include<stdio.h>
#include<math.h>
#define pi 3.14

typedef struct {
                        float real;
                        float img;
                }com;

void main()
{
      com x[50],y[50],temp[50];
      int pofd=0,i,k,n;
      float WN,c,s,WK;
      for(i=0;i<50;i++)                 /*make output array value to 0.0*/
      {
            y[i].real=0.0;
            y[i].img=0.0;
      }

      printf("\nEnter the no of points of dft:");
      scanf("%d",&pofd);
      printf("\nEnter the function variables of f(n) to calculate DFT in the
complex form (e.g. 3+j4 enter as a 3 4)");
      for(i=0;i<pofd;i++)
      {
            //scanf("%f%f",&x[i].real,&x[i].img);
            scanf("%f",&x[i].real);
            scanf("%f",&x[i].img);
      }
      WN=(2*pi)/pofd;
      for(k=0;k<pofd;k++)
      {
            temp[k].real=0.0;
            temp[k].img=0.0;

                  WK=k*WN;
                  for(n=0;n<pofd;n++)       /*DFT calculation*/
                  {
                        c=cos(WK*n);
                        s=sin(WK*n);
                        temp[k].real=temp[k].real+(x[n].real*c)+(x[n].img*s);
                        temp[k].img=temp[k].img+(x[n].img*c)-(x[n].real*s);
                  }


                   y[k].real=1/sqrt(pofd)*temp[k].real;
                   y[k].img=1/sqrt(pofd)*temp[k].img;


              }
              printf("\nDFT OF GIVEN SIGNAL:-\n");   /*Display values of F[k]*/
              for(i=0;i<pofd;i++)
              {
                    printf("\nF(%d)=(%0.1f)+j(%0.1f)\n",i,y[i].real,y[i].img);
              }
}
```

**OUTPUT:**

**<u>DFT:</u>**

• Input x[n]= {1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j}.

• Output X[K]={2.8+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0,0.0+j0.0}.



```
N_POINT_DFT:CIO
[C674X_0]
[C674X_0] DFT OF GIVEN SIGNAL:-
[C674X_0]
[C674X_0] F(0)=(2.8)+j(0.0)
[C674X_0]
[C674X_0] F(1)=(-0.0)+j(-0.0)
[C674X_0]
[C674X_0] F(2)=(-0.0)+j(-0.0)
[C674X_0]
[C674X_0] F(3)=(-0.0)+j(-0.0)
[C674X_0]
[C674X_0] F(4)=(0.0)+j(-0.0)
[C674X_0]
[C674X_0] F(5)=(0.0)+j(-0.0)
[C674X_0]
[C674X_0] F(6)=(0.0)+j(-0.0)
[C674X_0]
[C674X_0] F(7)=(0.0)+j(-0.0)
```

**RESULT:**

**CONCLUSION**

## VIVA QUESTIONS

1. Define DFT of a discrete time sequence?

2. Define inverse DFT.

3. What is the relation between DTFT and DFT?

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

5. List any four properties of DFT.

**Exp .No: 04** **Date:**

## DFT (N- POINTS)USING DIT-FFT ALGORITHM

**AIM: :** To write a C-program to find DFT N-points using DIT-FFT of a given sequence.

**APPARATUS REQUIRED**

- CODE COMPOSER STUDIO
- TMS320C6745 DSP TRAINER KIT

**PROCEDURE**

1. To create the New Project
2. Project→ New CCS Project (Give name to project with location to save or use default
3. location)
4. Select project type→ Executable
5. Device Family→C6000
6. Variant→C674xFloating-point DSP
7. Connection →Texas Instruments XDS100V2USB Emulator
8. Click on Empty Project then Finish
9. To create a Source file
10. File →New→ Source file (Save &amp; give file name, Eg: sum.c).Click on Finish
11. Write the C-Program To build the program project →Build All
12. After Build Finished without errors, Now DSP kit is turned ON
13. Debug the Program after loading is done
14. Run the Program and verify the output
15. Program ask for values of x[N], you have to enter values as 1st real & 2nd imaginary for all N no of values.
16. Eg. x[N]={1+0j,1+0j,1+0j,1+0j,1+0j,1+0j,1+0j,1+0j}.
17. It enter in program as → 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

## PROGRAM

```
#include<stdio.h>
//#include<conio.h>
#include<math.h>
#define PI 3.14

typedef struct
        {
                float real,imag;
        }com;
void main()
{
        com xx[8],x[8],temp[8],temp1[8],y[8],a[8],b[8],w[4];
        int i,j=0;//loop counter variables
       //      printf("Enter the no of pionts of FFT==");
       //      scanf("%d", &PTS);
        printf("\nEnter values==");
        for(i=0;i<8;i++)
        {
                scanf("%f",&xx[i].real);
                scanf("%f",&xx[i].imag);
        }
        j=0;
        for(i=0;i<8;i=i+2)
        {
                x[j].real=xx[i].real;
                x[j].imag=xx[i].imag;
                x[j+1].real=xx[i+4].real;
                x[j+1].imag=xx[i+4].imag;
                if(i==2)
                i=-1;
                j=j+2;
        }
        for(i=0;i<4;i++)
        {
                w[i].real=cos(2*PI*i/8);
                w[i].imag=-sin(2*PI*i/8);
        }


        for(i=0;i<8;i=i+2)
        {
                temp[i].real=x[i].real+x[i+1].real;
                temp[i].imag=x[i].imag+x[i+1].imag;
                temp[i+1].real=x[i].real-x[i+1].real;
                temp[i+1].imag=x[i].imag-x[i+1].imag;
        }
        for(i=2;i<8;i=3*i)
        {
                a[i].real=temp[i].real*w[0].real-temp[i].imag*w[0].imag;
                a[i].imag=temp[i].real*w[0].imag+temp[i].imag*w[0].real;
                a[i+1].real=temp[i+1].real*w[2].real-temp[i+1].imag*w[2].imag;
                a[i+1].imag=temp[i+1].real*w[2].imag+temp[i+1].imag*w[2].real;
                temp[i].real=a[i].real;
                temp[i].imag=a[i].imag;
                temp[i+1].real=a[i+1].real;
```

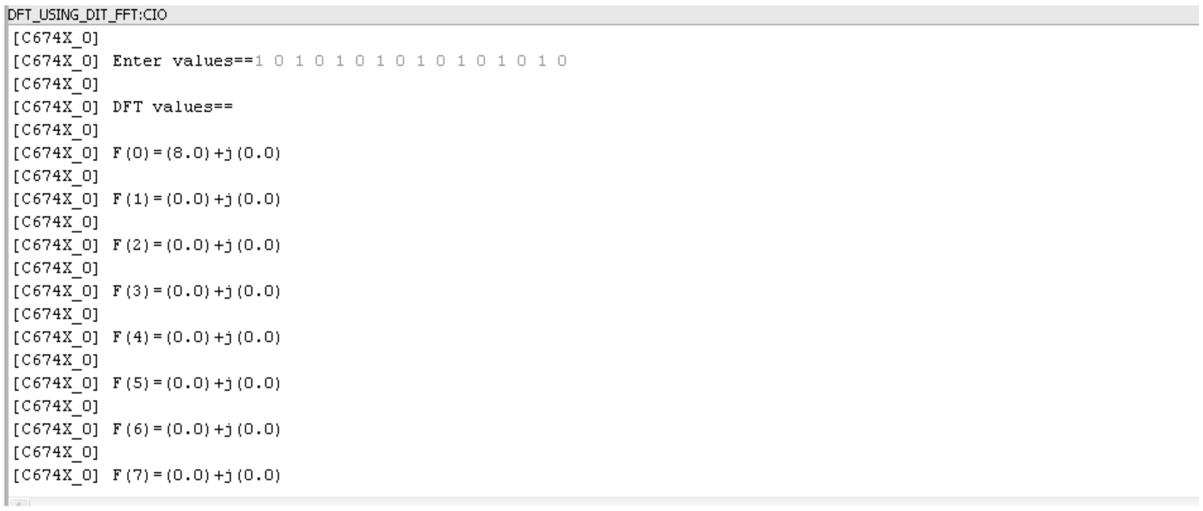
```
		temp[i+1].imag=a[i+1].imag;

	}
	for(i=0;i<6;i++)
	{
		temp1[i].real=temp[i].real+temp[i+2].real;
		temp1[i].imag=temp[i].imag+temp[i+2].imag;
		temp1[i+2].real=temp[i].real-temp[i+2].real;
		temp1[i+2].imag=temp[i].imag-temp[i+2].imag;
		if(i==1)
		i=3;
	}
	for(i=4;i<8;i++)
	{
		b[i].real=temp1[i].real*w[i-4].real-temp1[i].imag*w[i-4].imag;
		b[i].imag=temp1[i].real*w[i-4].imag+temp1[i].imag*w[i-4].real;
		temp1[i].real=b[i].real;
		temp1[i].imag=b[i].imag;
	}
	for(i=0;i<4;i++)
	{
		y[i].real=temp1[i].real+temp1[i+4].real;
		y[i].imag=temp1[i].imag+temp1[i+4].imag;
		y[i+4].real=temp1[i].real-temp1[i+4].real;
		y[i+4].imag=temp1[i].imag-temp1[i+4].imag;
	}
	printf("\nDFT values==\n");
	 for(i=0;i<8;i++)
		  {
			printf("\nF(%d)=(%0.1f)+j(%0.1f)\n",i,y[i].real,y[i].imag);
		  }
}
```

## OUTPUT

• Input x[n]= {1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j}; for both DIT & DIF –FFT algorithm.

• Output X[K]={8+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0}; for both DIT & DIF-FFT algorithm.

Result of DFT using DIT-FFT method



DFT_USING_DIT_FFT:CIO

```
[C674X_0]
[C674X_0] Enter values==1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
[C674X_0]
[C674X_0] DFT values==
[C674X_0]
[C674X_0] F(0)=(8.0)+j(0.0)
[C674X_0]
[C674X_0] F(1)=(0.0)+j(0.0)
[C674X_0]
[C674X_0] F(2)=(0.0)+j(0.0)
[C674X_0]
[C674X_0] F(3)=(0.0)+j(0.0)
[C674X_0]
[C674X_0] F(4)=(0.0)+j(0.0)
[C674X_0]
[C674X_0] F(5)=(0.0)+j(0.0)
[C674X_0]
[C674X_0] F(6)=(0.0)+j(0.0)
[C674X_0]
[C674X_0] F(7)=(0.0)+j(0.0)
```

**RESULT:**

**CONCLUSION**

**VIVA QUESTIONS**

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

2. Compare FFT and DFT?

3. What are the various algorithms to calculate FFT?

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

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

**Exp .No: 05** **Date:**

# N POINT IFFT ALGORITHM

**AIM:** To write a C-program to compute IFFT N-point of a given signal using DIT- FFT

**APPARATUS REQUIRED**

- CODE COMPOSER STUDIO
- TMS320C6745 DSP TRAINER KIT

**PROCEDURE**

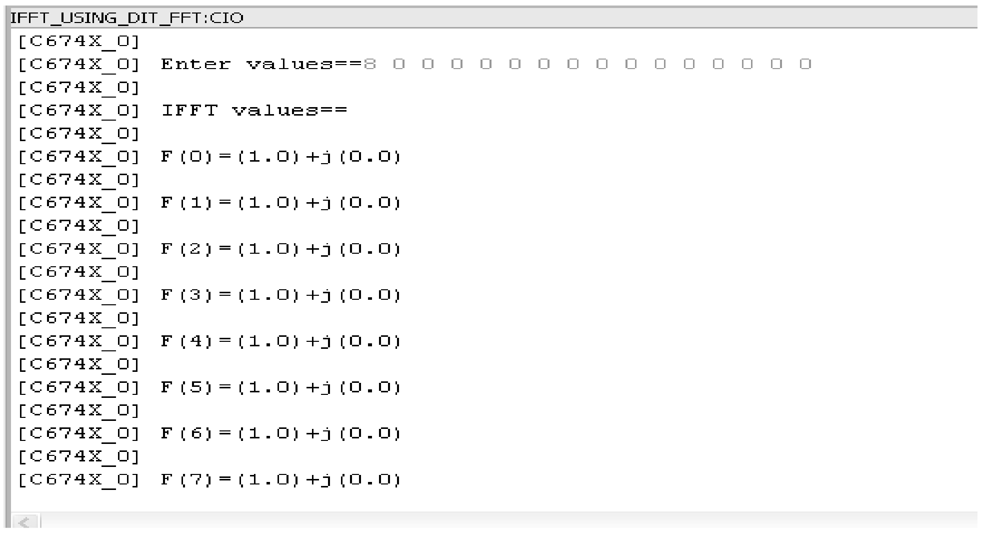
18. To create the New Project
19. Project→ New CCS Project (Give name to project with location to save or use default
20. location)
21. Select project type→ Executable
22. Device Family→C6000
23. Variant→C674xFloating-point DSP
24. Connection →Texas Instruments XDS100V2USB Emulator
25. Click on Empty Project then Finish
26. To create a Source file
27. File →New→ Source file (Save &amp; give file name, Eg: sum.c).Click on Finish
28. Write the C-Program To build the program project →Build All
29. After Build Finished without errors, Now DSP kit is turned ON
30. Debug the Program after loading is done
31. Run the Program and verify the output
32. Program ask for values of x[N], you have to enter values as 1st real & 2nd imaginary for all N no of values.
33. Eg. x[N]= { 8.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0,
34. 0.0+j0.0, 0.0+j0.0}.
35. It enter in program as → 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**PROGRAM:**

```
#include<stdio.h>
//#include<conio.h>
#include<math.h>
#define PI 3.14

typedef struct
        {
                float real,imag;
        }com;
void main()
{
        com xx[8],x[8],temp[8],temp1[8],y[8],a[8],b[8],w[4];
        int i,j=0;//loop counter variables
       //     printf("Enter the no of pionts of FFT==");
       //     scanf("%d", &PTS);
        printf("\nEnter values==");
        for(i=0;i<8;i++)
        {
                scanf("%f",&xx[i].real);
                scanf("%f",&xx[i].imag);
        }
        j=0;
        for(i=0;i<8;i=i+2)
        {
                x[j].real=xx[i].real;
                x[j].imag=xx[i].imag;
                x[j+1].real=xx[i+4].real;
                x[j+1].imag=xx[i+4].imag;
                if(i==2)
                i=-1;
                j=j+2;
        }
        for(i=0;i<4;i++)
        {
                w[i].real=cos(2*PI*i/8);
                w[i].imag=sin(2*PI*i/8);
        }


        for(i=0;i<8;i=i+2)
        {
                temp[i].real=x[i].real+x[i+1].real;
                temp[i].imag=x[i].imag+x[i+1].imag;
                temp[i+1].real=x[i].real-x[i+1].real;
                temp[i+1].imag=x[i].imag-x[i+1].imag;
        }
        for(i=2;i<8;i=3*i)
        {
                a[i].real=temp[i].real*w[0].real-temp[i].imag*w[0].imag;
                a[i].imag=temp[i].real*w[0].imag+temp[i].imag*w[0].real;
                a[i+1].real=temp[i+1].real*w[2].real-temp[i+1].imag*w[2].imag;
                a[i+1].imag=temp[i+1].real*w[2].imag+temp[i+1].imag*w[2].real;
                temp[i].real=a[i].real;
                temp[i].imag=a[i].imag;
                temp[i+1].real=a[i+1].real;
                temp[i+1].imag=a[i+1].imag;
        }
        for(i=0;i<6;i++)
        {
                temp1[i].real=temp[i].real+temp[i+2].real;
                temp1[i].imag=temp[i].imag+temp[i+2].imag;
                temp1[i+2].real=temp[i].real-temp[i+2].real;
                temp1[i+2].imag=temp[i].imag-temp[i+2].imag;
                if(i==1)
                i=3;
        }
        for(i=4;i<8;i++)
        {
                b[i].real=temp1[i].real*w[i-4].real-temp1[i].imag*w[i-4].imag;
                b[i].imag=temp1[i].real*w[i-4].imag+temp1[i].imag*w[i-4].real;
                temp1[i].real=b[i].real;
                temp1[i].imag=b[i].imag;
        }
        for(i=0;i<4;i++)
        {
                y[i].real=temp1[i].real+temp1[i+4].real;
                y[i].imag=temp1[i].imag+temp1[i+4].imag;
                y[i+4].real=temp1[i].real-temp1[i+4].real;
                y[i+4].imag=temp1[i].imag-temp1[i+4].imag;
        }
        printf("\nIFFT values==\n");
         for(i=0;i<8;i++)
                {
                        printf("\nF(%d)=(%0.1f)+j(%0.1f)\n",i,y[i].real/8,y[i].imag/8);
                }
}
```

## OUTPUT

• Input x[n]= {8.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0, 0.0+j0.0};
for both DIT & DIF–FFT algorithm.
• Output X[K]={1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j, 1+0j }; for both DIT & DIF-FFT Algorithm



```
IFFT_USING_DIT_FFT:CIO
[C674X_0]
[C674X_0]  Enter values==8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[C674X_0]
[C674X_0]  IFFT values==
[C674X_0]
[C674X_0]  F(0)=(1.0)+j(0.0)
[C674X_0]
[C674X_0]  F(1)=(1.0)+j(0.0)
[C674X_0]
[C674X_0]  F(2)=(1.0)+j(0.0)
[C674X_0]
[C674X_0]  F(3)=(1.0)+j(0.0)
[C674X_0]
[C674X_0]  F(4)=(1.0)+j(0.0)
[C674X_0]
[C674X_0]  F(5)=(1.0)+j(0.0)
[C674X_0]
[C674X_0]  F(6)=(1.0)+j(0.0)
[C674X_0]
[C674X_0]  F(7)=(1.0)+j(0.0)
```

**RESULT:**

**CONCLUSION**

**VIVA QUESTIONS**

1. Define transform. What is the need for transform?

2. Differentiate Fourier transform and discrete Fourier transform.

3. Differentiate DFT and DTFT.

4. What are the advantages of FFT over DFT?

5. Differentiate DITFFT and DIFFFT algorithms.

**Exp .No: 06** **Date:**

## DESIGN OF FIR FILTER USING WINDOWING TECHNIQUES

**AIM:** To design and implement a FIR LP/HP filters using any three windowing techniques.

a. Rectangular window
b. Triangular window
c. Kaiser window

**APPARATUS REQUIRED**

- CODE COMPOSER STUDIO
- TMS320C6745 DSP TRAINER KIT

**PROCEDURE:**

1. To create the New Project
2. Project→ New CCS Project (Give name to project with location to save or use default
3. location)
4. Select project type→ Executable
5. Device Family→C6000
6. Variant→C674xFloating-point DSP
7. Connection →Texas Instruments XDS100V2USB Emulator
8. Click on Empty Project then Finish
9. To create a Source file
10. File →New→ Source file (Save &amp; give file name, Eg: sum.c).Click on Finish
11. Write the C-Program To build the program project →Build All
12. After Build Finished without errors, Now DSP kit is turned ON
13. Debug the Program after loading is done
14. Run the Program and verify the output
15. Connect CRO to the **LINE OUT.**
16. Connect a Signal Generator to the **LINE IN.**
17. Switch on the Signal Generator with a sine wave of frequency 100 Hz. and Vp-p=1.0v&vary the frequency.

**1. FOR LOW PASS FILTER:**
a. For fir low pass rectangular window (cutoff 500Hz)
b. For fir low pass triangular window (cutoff 1000Hz)
C. For fir low pass Kaiser Window (cutoff 1500Hz)
**2. FOR HIGH PASS FILTER**:
a. For fir high pass rectangular window (cutoff 400Hz)
b. For fir high pass triangular window (cutoff 800Hz)
C. For fir high pass triangular window (cutoff 1200Hz)

## 1(a) RECTANGULAR WINDOW FOR LPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;

void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {
    }
}
void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );
    /* Release SPI */
    SPI_SPIGCR0 = 1;

    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS
      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //
    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );

    SPI_SPIDAT1 = spidat1;
    SPI_SPIDELAY = 0
```

```
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY
   SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high
    SPI_SPIINT = 0;
    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}

float filter_Coeff[] ={-0.008982,-0.017782,-0.025020,-0.029339,-0.029569,-0.024895,
-0.014970,0.000000,0.019247,0.041491,0.065053,0.088016,0.108421,0.124473,0.134729,
0.138255,0.134729,0.124473,0.108421,0.088016,0.065053,0.041491,0.019247,0.000000,
-0.014970,-0.024895,-0.029569,-0.029339,-0.025020,-0.017782,-0.008982};
           static short in_buffer[100];
           signed int FIR_FILTER(float  h[], signed int x);
           void main( )
              {
          Uint32 GBLCTL_XHCLKRST_ON    =      0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =      0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =      0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =      0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =      0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =      0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =      0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =      0X00000008;
          Uint32 GBLCTL_XFRST_ON              =      0X00001000;
          Uint32 GBLCTL_RFRST_ON              =      0X00000010;
       int temp,i;
      signed int input_data1, input_data2;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8,
       0X0811, 0X0A00, 0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

             for(i=0; i<10; i++)
             {
                    SPI_SPIDAT1= temp | codec_reg_val[i];
                    while( !(SPI_SPIFLG&0x200));
                    DSP6745_wait( 1000 );
             }
             SPI_SPIGCR0 = 0;
                    DSP6745_wait( 1000 );

       MCASP0_GBLCTL     = 0;
       MCASP0_RGBLCTL    = 0;
       MCASP0_XGBLCTL    = 0;
       MCASP0_PWRDEMU    = 1;
       MCASP0_RMASK      = 0xFFFFFFFF;
       MCASP0_RFMT       = 0x00018078; // MSB 16bit, 0-delay, no pad, CFGBus
       MCASP0_AFSRCTL    = 0x00000000; // 2TDM,1bit Rising edge INTERNAL FS, word
      MCASP0_ACLKRCTL   =  0x00000081;//0x000000AF;//Rising INTERNAL CLK(from tx
side)
      MCASP0_AHCLKRCTL  = 0x00008000;//INT CLK (from tx side)
      MCASP0_RTDM       = 0x00000003; // Slots 0,1
      MCASP0_RINTCTL    = 0x00000000; // Not used
                        MCASP0_RCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_XMASK      = 0xFFFFFFFF; // No padding used
                        MCASP0_XFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSXCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKXCTL   = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                        MCASP0_AHCLKXCTL  = 0x00008000;// INT CLK, div-by-4
                        MCASP0_XTDM       = 0x00000003; // Slots 0,1
                        MCASP0_XINTCTL    = 0x00000000; // Not used
                        MCASP0_XCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_SRCTL1     = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                        MCASP0_SRCTL0     = 0x0001;     // MCASP0.AXR0[0] --> DIN

                          MCASP0_PFUNC      = 0x00;
                        MCASP0_PDIR            = 0x00000001;
                        MCASP0_DITCTL     = 0x00000000; // Not used
                        MCASP0_DLBCTL     = 0x00000000; // Not used
                        MCASP0_AMUTE      = 0x00000000; // Not used


             /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                 MCASP0_XSTAT = 0x0000ffff;        // Clear all
                 MCASP0_RSTAT = 0x0000ffff;        // Clear all

                 MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);
```

```
                  /* Write a 0, so that no underrun occurs after releasing the state
machine */

                  MCASP0_XBUF0 = 0;

                  MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);



                  MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                  MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );


                  /* Start by sending a dummy write */
                  while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                  MCASP0_XBUF0 = 0;

                 while(1)
                  {

                           while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                           input_data1 = MCASP0_RBUF1_32BIT;
                           input_data2 = FIR_FILTER(filter_Coeff , input_data1);
                           while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                           MCASP0_XBUF0_32BIT = (input_data2 << 16);

                  }


}



signed int FIR_FILTER(float  h[], signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* new input at buffer[0] */
for(i=29;i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
for(i=0;i<31;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

## 1(b) TRIANGULAR WINDOW FOR LPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;
void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}
void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );
              /* Release SPI */
    SPI_SPIGCR0 = 1;

    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );
```

```
 SPI_SPIDAT1 = spidat1;
         SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;
    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}

float filter_Coeff[] ={0.000000,-0.001591,-
0.002423,0.000000,0.005728,0.011139,0.010502,
-0.000000,-0.018003,-0.033416,-
0.031505,0.000000,0.063010,0.144802,0.220534,0.262448,
0.220534,0.144802,0.063010,0.000000,-0.031505,-0.033416,-0.018003,-
0.000000,0.010502,
0.011139,0.005728,0.000000,-0.002423,-0.001591,0.000000};

static short in_buffer[100];

signed int FIR_FILTER(float  h[], signed int x);
void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =     0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =     0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =     0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =     0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =     0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =     0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =     0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =     0X00000008;
          Uint32 GBLCTL_XFRST_ON       =     0X00001000;
          Uint32 GBLCTL_RFRST_ON       =     0X00000010;
      int temp,i;
      signed int input_data1, input_data2;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

             for(i=0; i<10; i++)
             {
                    SPI_SPIDAT1= temp | codec_reg_val[i];
                    while( !(SPI_SPIFLG&0x200));
                    DSP6745_wait( 1000 );
             }
             SPI_SPIGCR0 = 0;
                    DSP6745_wait( 1000 );
                        MCASP0_GBLCTL     = 0;
                        MCASP0_RGBLCTL    = 0;
                        MCASP0_XGBLCTL    = 0;
                        MCASP0_PWRDEMU    = 1;

                        MCASP0_RMASK      = 0xFFFFFFFF;
                        MCASP0_RFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSRCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKRCTL   = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
                        MCASP0_AHCLKRCTL  = 0x00008000;//INT CLK (from tx side)
                        MCASP0_RTDM       = 0x00000003; // Slots 0,1
                        MCASP0_RINTCTL    = 0x00000000; // Not used
                        MCASP0_RCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256
                         MCASP0_XMASK      = 0xFFFFFFFF; // No padding used
                        MCASP0_XFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSXCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKXCTL   = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                        MCASP0_AHCLKXCTL  = 0x00008000;// INT CLK, div-by-4
                        MCASP0_XTDM       = 0x00000003; // Slots 0,1
                        MCASP0_XINTCTL    = 0x00000000; // Not used
                        MCASP0_XCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_SRCTL1     = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                        MCASP0_SRCTL0     = 0x0001;     // MCASP0.AXR0[0] --> DIN

                          MCASP0_PFUNC      = 0x00;
                        MCASP0_PDIR            = 0x00000001;
                        MCASP0_DITCTL     = 0x00000000; // Not used
                        MCASP0_DLBCTL     = 0x00000000; // Not used
                        MCASP0_AMUTE      = 0x00000000; // Not used

             /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                 MCASP0_XSTAT = 0x0000ffff;        // Clear all
                 MCASP0_RSTAT = 0x0000ffff;        // Clear all
                 MCASP0_XGBLCTL  |= GBLCTL_XSRCLR_ON;
// Serialize
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                 MCASP0_RGBLCTL  |= GBLCTL_RSRCLR_ON;
// Serialize
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                 /* Write a 0, so that no underrun occurs after releasing the state
machine */

                 MCASP0_XBUF0 = 0;

                 MCASP0_XGBLCTL  |= GBLCTL_XSMRST_ON;
// State Machine
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                 MCASP0_RGBLCTL  |= GBLCTL_RSMRST_ON;
// State Machine
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);



                 MCASP0_XGBLCTL  |= GBLCTL_XFRST_ON;
// Frame Sync
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                 MCASP0_RGBLCTL  |= GBLCTL_RFRST_ON;
// Frame Sync
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );


                 /* Start by sending a dummy write */
                 while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                 MCASP0_XBUF0 = 0;

                while(1)
                 {
                          while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                          input_data1 = MCASP0_RBUF1_32BIT;
                          input_data2 = FIR_FILTER(filter_Coeff , input_data1);
                          while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                          MCASP0_XBUF0_32BIT = (input_data2 << 16);
                                     }
                    }
signed int FIR_FILTER(float  h[], signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* new input at buffer[0] */
for(i=29;i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
for(i=0;i<31;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

## 1(c) KAISER WINDOW FOR LPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;
void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {
    }
}
void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );
    /* Release SPI */
    SPI_SPIGCR0 = 1;
              /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS
      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //
    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );
    SPI_SPIDAT1 = spidat1;
    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
```

```
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high
    SPI_SPIINT = 0;
    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}
float filter_Coeff[] ={0.000046,-0.000166,0.000246,0.001414,0.001046,-0.003421,-
0.007410,
0.000000,0.017764,0.020126,-0.015895,-0.060710,-
0.034909,0.105263,0.289209,0.374978,
0.289209,0.105263,-0.034909,-0.060710,-0.015895,0.020126,0.017764,0.000000,-
0.007410,
-0.003421,0.001046,0.001414,0.000246,-0.000166, -0.000046};
static short in_buffer[100];
signed int FIR_FILTER(float  h[], signed int x);
void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =     0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =     0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =     0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =     0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =     0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =     0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =     0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =     0X00000008;
          Uint32 GBLCTL_XFRST_ON       =     0X00001000;
          Uint32 GBLCTL_RFRST_ON       =     0X00000010;
      int temp,i;
      signed int input_data1, input_data2;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

            for(i=0; i<10; i++)
            {
                  SPI_SPIDAT1= temp | codec_reg_val[i];
                  while( !(SPI_SPIFLG&0x200));
                  DSP6745_wait( 1000 );
            }
            SPI_SPIGCR0 = 0;
                  DSP6745_wait( 1000 );

                          MCASP0_GBLCTL     = 0;
                       MCASP0_RGBLCTL    = 0;
                       MCASP0_XGBLCTL    = 0;
                       MCASP0_PWRDEMU    = 1;

                       MCASP0_RMASK      = 0xFFFFFFFF;
                       MCASP0_RFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                       MCASP0_AFSRCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                       MCASP0_ACLKRCTL   = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
```

```
                        MCASP0_AHCLKRCTL  = 0x00008000;//INT CLK (from tx side)
                        MCASP0_RTDM       = 0x00000003; // Slots 0,1
                        MCASP0_RINTCTL    = 0x00000000; // Not used
                        MCASP0_RCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_XMASK      = 0xFFFFFFFF; // No padding used
                        MCASP0_XFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSXCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKXCTL   = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                        MCASP0_AHCLKXCTL  = 0x00008000;// INT CLK, div-by-4
                        MCASP0_XTDM       = 0x00000003; // Slots 0,1
                        MCASP0_XINTCTL    = 0x00000000; // Not used
                        MCASP0_XCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_SRCTL1     = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                        MCASP0_SRCTL0     = 0x0001;     // MCASP0.AXR0[0] --> DIN

                        MCASP0_PFUNC    = 0x00;
                        MCASP0_PDIR        = 0x00000001;
                        MCASP0_DITCTL     = 0x00000000; // Not used
                        MCASP0_DLBCTL     = 0x00000000; // Not used
                        MCASP0_AMUTE      = 0x00000000; // Not used


             /* Starting sections of the McASP*/
                  MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                  MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                  MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                  MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                  MCASP0_XSTAT = 0x0000ffff;        // Clear all
                  MCASP0_RSTAT = 0x0000ffff;        // Clear all

                  MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
```

```
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                 /* Write a 0, so that no underrun occurs after releasing the state
machine */

                 MCASP0_XBUF0 = 0;

                 MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);



                 MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );


                 /* Start by sending a dummy write */
                 while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                 MCASP0_XBUF0 = 0;

                while(1)
                 {

                         while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                         input_data1 = MCASP0_RBUF1_32BIT;
                         input_data2 = FIR_FILTER(filter_Coeff , input_data1);
                         while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                         MCASP0_XBUF0_32BIT = (input_data2 << 16);

                 }


}



signed int FIR_FILTER(float  h[], signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* new input at buffer[0] */
for(i=29;i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
for(i=0;i<31;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

## 2(a) RECTANGULAR WINDOW FOR HPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;
void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}
void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );

    /* Release SPI */
    SPI_SPIGCR0 = 1;
    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers

    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );
    SPI_SPIDAT1 = spidat1;
    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;

    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}

float filter_Coeff[] ={0.021665,0.022076,0.020224,0.015918,0.009129,-0.000000,-
0.011158,
-0.023877,-0.037558,-0.051511,-0.064994,-0.077266,-0.087636,-0.095507,-
.100422,0.918834,
-0.100422,-0.095507,-0.087636,-0.077266,-0.064994,-0.051511,-0.037558,-0.023877,
-0.011158,-0.000000,0.009129,0.015918,0.020224,0.022076,0.021665};
static short in_buffer[100];
signed int FIR_FILTER(float  h[], signed int x);
void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =     0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =     0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =     0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =     0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =     0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =     0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =     0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =     0X00000008;
          Uint32 GBLCTL_XFRST_ON       =     0X00001000;
          Uint32 GBLCTL_RFRST_ON       =     0X00000010;

      int temp,i;
      signed int input_data1, input_data2;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

             for(i=0; i<10; i++)
             {
                    SPI_SPIDAT1= temp | codec_reg_val[i];
                    while( !(SPI_SPIFLG&0x200));
                    DSP6745_wait( 1000 );
             }
             SPI_SPIGCR0 = 0;
                    DSP6745_wait( 1000 );

                        MCASP0_GBLCTL     = 0;
                        MCASP0_RGBLCTL    = 0;
                        MCASP0_XGBLCTL    = 0;
                        MCASP0_PWRDEMU     = 1;

                        MCASP0_RMASK       = 0xFFFFFFFF;
                        MCASP0_RFMT        = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSRCTL     = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKRCTL    = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
                        MCASP0_AHCLKRCTL   = 0x00008000;//INT CLK (from tx side)
                        MCASP0_RTDM        = 0x00000003; // Slots 0,1
                        MCASP0_RINTCTL     = 0x00000000; // Not used
                        MCASP0_RCLKCHK     = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_XMASK       = 0xFFFFFFFF; // No padding used
                        MCASP0_XFMT        = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSXCTL     = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKXCTL    = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                        MCASP0_AHCLKXCTL   = 0x00008000;// INT CLK, div-by-4
                        MCASP0_XTDM        = 0x00000003; // Slots 0,1
                        MCASP0_XINTCTL     = 0x00000000; // Not used
                        MCASP0_XCLKCHK     = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_SRCTL1      = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                        MCASP0_SRCTL0      = 0x0001;     // MCASP0.AXR0[0] --> DIN

                        MCASP0_PFUNC       = 0x00;
                        MCASP0_PDIR            = 0x00000001;
                        MCASP0_DITCTL      = 0x00000000; // Not used
                        MCASP0_DLBCTL      = 0x00000000; // Not used
                        MCASP0_AMUTE       = 0x00000000; // Not used


             /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                 MCASP0_XSTAT = 0x0000ffff;        // Clear all
```

```
                  MCASP0_RSTAT = 0x0000ffff;        // Clear all

                  MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                  /* Write a 0, so that no underrun occurs after releasing the state
machine */

                  MCASP0_XBUF0 = 0;

                  MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);


                  MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                  MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );

                  /* Start by sending a dummy write */
                  while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                  MCASP0_XBUF0 = 0;

                 while(1)
                  {         while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                           input_data1 = MCASP0_RBUF1_32BIT;
                           input_data2 = FIR_FILTER(filter_Coeff , input_data1);
                           while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                           MCASP0_XBUF0_32BIT = (input_data2 << 16);

                  }
}
signed int FIR_FILTER(float  h[], signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* new input at buffer[0] */
for(i=29;i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
for(i=0;i<31;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

## 2(b) TRIANGULAR WINDOW FOR HPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char

static Uint32 spidat1;

void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}
void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );

    /* Release SPI */
    SPI_SPIGCR0 = 1;

    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers

    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );    // Char Len | ( 1F << 0 );

    SPI_SPIDAT1 = spidat1;

    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;

    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}

float filter_Coeff[] ={0.000000,-0.000897,-0.003126,-0.005080,-0.004567,-
0.000000,0.008373,
0.017782,0.023225,0.018839,0.000000,-0.034539,-0.081287,-0.132092,-
0.175834,0.805541,
-0.175834,-0.132092,-0.081287,-
0.034539,0.000000,0.018839,0.023225,0.017782,0.008373,
-0.000000,-0.004567,-0.005080,-0.003126,-0.000897,0.000000};

static short in_buffer[100];

signed int FIR_FILTER(float  h[], signed int x);

void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =     0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =     0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =     0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =     0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =     0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =     0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =     0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =     0X00000008;
          Uint32 GBLCTL_XFRST_ON       =     0X00001000;
          Uint32 GBLCTL_RFRST_ON       =     0X00000010;

      int temp,i;
      signed int input_data1, input_data2;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

            for(i=0; i<10; i++)
            {
                  SPI_SPIDAT1= temp | codec_reg_val[i];
                  while( !(SPI_SPIFLG&0x200));
                  DSP6745_wait( 1000 );
```

```
             }
             SPI_SPIGCR0 = 0;
                   DSP6745_wait( 1000 );
                       MCASP0_GBLCTL     = 0;
                       MCASP0_RGBLCTL    = 0;
                       MCASP0_XGBLCTL    = 0;
                       MCASP0_PWRDEMU    = 1;

                       MCASP0_RMASK      = 0xFFFFFFFF;
                       MCASP0_RFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                       MCASP0_AFSRCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                       MCASP0_ACLKRCTL   = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
                       MCASP0_AHCLKRCTL  = 0x00008000;//INT CLK (from tx side)
                       MCASP0_RTDM       = 0x00000003; // Slots 0,1
                       MCASP0_RINTCTL    = 0x00000000; // Not used
                       MCASP0_RCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256
                       MCASP0_XMASK      = 0xFFFFFFFF; // No padding used
                       MCASP0_XFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                       MCASP0_AFSXCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                       MCASP0_ACLKXCTL   = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                       MCASP0_AHCLKXCTL  = 0x00008000;// INT CLK, div-by-4
                       MCASP0_XTDM       = 0x00000003; // Slots 0,1
                       MCASP0_XINTCTL    = 0x00000000; // Not used
                       MCASP0_XCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256
                       MCASP0_SRCTL1     = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                       MCASP0_SRCTL0     = 0x0001;     // MCASP0.AXR0[0] --> DIN
                       MCASP0_PFUNC      = 0x00;
                       MCASP0_PDIR          = 0x00000001;
                       MCASP0_DITCTL     = 0x00000000; // Not used
                       MCASP0_DLBCTL     = 0x00000000; // Not used
                       MCASP0_AMUTE      = 0x00000000; // Not used

             /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );
```

```
                  MCASP0_XSTAT = 0x0000ffff;        // Clear all
                  MCASP0_RSTAT = 0x0000ffff;        // Clear all

                  MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                  /* Write a 0, so that no underrun occurs after releasing the state
machine */

                  MCASP0_XBUF0 = 0;

                  MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);
                  MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                  MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );
                  /* Start by sending a dummy write */
                  while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                  MCASP0_XBUF0 = 0;

                 while(1)
                  {
                           while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                           input_data1 = MCASP0_RBUF1_32BIT;
                           input_data2 = FIR_FILTER(filter_Coeff , input_data1);
                           while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                           MCASP0_XBUF0_32BIT = (input_data2 << 16);

                  }
}
signed int FIR_FILTER(float  h[], signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* new input at buffer[0] */
for(i=29;i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
for(i=0;i<31;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

## 2(c) KAISER WINDOW FOR HPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;
void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}
void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );

    /* Release SPI */
    SPI_SPIGCR0 = 1;

    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers

    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );

    SPI_SPIDAT1 = spidat1;
    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;
    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}
float filter_Coeff[] ={-0.000050,-0.000138,0.000198,0.001345,0.002212,-0.000000,-
0.006489,
-0.012033,-0.005942,0.016731,0.041539,0.035687,-0.028191,-0.141589,-
0.253270,0.700008,
-0.253270,-0.141589,-0.028191,0.035687,0.041539,0.016731,-0.005942,-0.012033,-
0.006489,
-0.000000,0.002212,0.001345,0.000198,-0.000138,-0.000050};
static short in_buffer[100];
signed int FIR_FILTER(float  h[], signed int x);
void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =     0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =     0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =     0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =     0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =     0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =     0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =     0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =     0X00000008;
          Uint32 GBLCTL_XFRST_ON       =     0X00001000;
          Uint32 GBLCTL_RFRST_ON       =     0X00000010;
      int temp,i;
      signed int input_data1, input_data2;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

            for(i=0; i<10; i++)
            {
                  SPI_SPIDAT1= temp | codec_reg_val[i];
                  while( !(SPI_SPIFLG&0x200));
                  DSP6745_wait( 1000 );
            }
            SPI_SPIGCR0 = 0;
                  DSP6745_wait( 1000 );
                      MCASP0_GBLCTL     = 0;
                      MCASP0_RGBLCTL    = 0;
                      MCASP0_XGBLCTL    = 0;
```

```
                        MCASP0_PWRDEMU    = 1;

                        MCASP0_RMASK      = 0xFFFFFFFF;
                        MCASP0_RFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSRCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKRCTL   = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
                        MCASP0_AHCLKRCTL  = 0x00008000;//INT CLK (from tx side)
                        MCASP0_RTDM       = 0x00000003; // Slots 0,1
                        MCASP0_RINTCTL    = 0x00000000; // Not used
                        MCASP0_RCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256                     MCASP0_XMASK      = 0xFFFFFFFF; // No padding used
                        MCASP0_XFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSXCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKXCTL   = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                        MCASP0_AHCLKXCTL  = 0x00008000;// INT CLK, div-by-4
                        MCASP0_XTDM       = 0x00000003; // Slots 0,1
                        MCASP0_XINTCTL    = 0x00000000; // Not used
                        MCASP0_XCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_SRCTL1     = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                        MCASP0_SRCTL0     = 0x0001;     // MCASP0.AXR0[0] --> DIN

                        MCASP0_PFUNC      = 0x00;
                        MCASP0_PDIR           = 0x00000001;
                        MCASP0_DITCTL     = 0x00000000; // Not used
                        MCASP0_DLBCTL     = 0x00000000; // Not used
                        MCASP0_AMUTE      = 0x00000000; // Not used
                                 /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                 MCASP0_XSTAT = 0x0000ffff;        // Clear all
                 MCASP0_RSTAT = 0x0000ffff;        // Clear all

                 MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
```

```
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);
                 /* Write a 0, so that no underrun occurs after releasing the state
machine */

                 MCASP0_XBUF0 = 0;
                 MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);
                 MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );
                 /* Start by sending a dummy write */
                 while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                 MCASP0_XBUF0 = 0;

                while(1)
                 {
                         while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                          input_data1 = MCASP0_RBUF1_32BIT;
                          input_data2 = FIR_FILTER(filter_Coeff , input_data1);
                          while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                          MCASP0_XBUF0_32BIT = (input_data2 << 16);

                 }
}
signed int FIR_FILTER(float  h[], signed int x)
{
int i=0;
signed long output=0;
in_buffer[0] = x; /* new input at buffer[0] */
for(i=29;i>0;i--)
in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
for(i=0;i<31;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

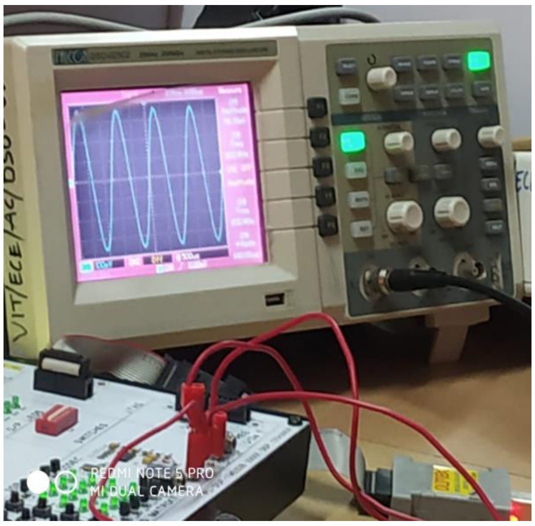
## OUTPUT:

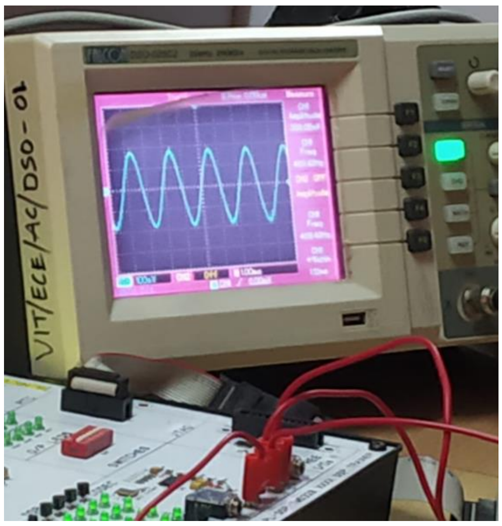
- As we giving applied sine input through line in, the output will appear as per the filter type on DSO.

- Output will decreasing after the cutoff frequency for low pass filter
- Output will appear at the cutoff frequency for high pass filter

LPF

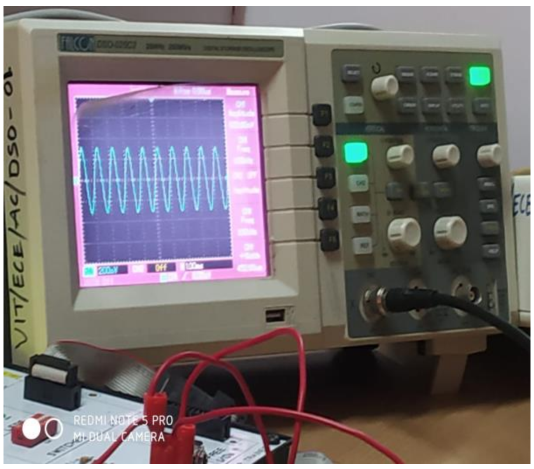
INPUT FREQUENCY (500Hz) OUTPUT

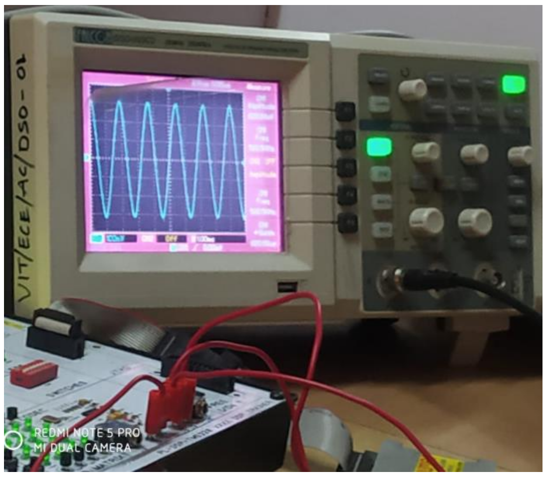




HPF

INPUT FREQUENCY (400Hz) OUTPUT





**5. Result**

**Conclusion**

**Viva Questions**

1. Write the procedure for design of FIR filters by using Kaiser Window?

2. Write the expression for rectangular widow and what is Peak amplitude of side lobe (db, Main lobe width and minimum stop band attenuation?

3. Write the expression for Triangular widow and what is Peak amplitude of side lobe (db, Main lobe width and minimum stop band attenuation?

4. Write the expression for Hanning widow and what is Peak amplitude of side lobe (db, Main lobe width and minimum stop band attenuation?

5. Write the expression for Hamming widow and what is Peak amplitude of side lobe (db, Main lobe width and minimum stop band attenuation?

**Exp .No: 07** **Date:**

# DESIGN OF IIR BUTTERWORTH LP/HP FILTER

**AIM:** To design and implement an IIR Butterworth LP/HP Filter for given specification.
**APPARATUS REQUIRED**

- CODE COMPOSER STUDIO
- TMS320C6745 DSP TRAINER KIT

**Procedure**

1. To create the New Project
2. Project→ New CCS Project (Give name to project with location to save or use default location)Select project type→ Executable
3. Device Family→C6000
4. Variant→C674xFloating-point DSP
5. Connection →Texas Instruments XDS100V2USB Emulator
6. Click on Empty Project then Finish
7. To create a Source file
8. File →New→ Source file (Save & give file name, Eg: sum.c).Click on Finish
9. Write the C-Program To build the program project →Build All
10. After Build Finished without errors, Now DSP kit is turned ON
11. Debug the Program after loading is done
12. Run the Program and verify the output
13. Connect CRO to the LINE OUT.
14. Connect a Signal Generator to the LINE IN.
15. Switch on the Signal Generator with a sine wave of frequency 100 Hz. and Vp-p=1.0v & vary the frequency.

1. IIR low pass Butterworth filter (cutoff 800Hz)
2. IIR high pass Butterworth filter (cutoff 2500Hz)

## 1. IIR Butterworth LPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char

static Uint32 spidat1;

void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}


void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );
              /* Release SPI */
    SPI_SPIGCR0 = 1;
    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );

    SPI_SPIDAT1 = spidat1;
    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;

    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}
const signed int filter_Coeff[] =
{
312,312,312,32767,-27943,24367 /*LP 800 */
} ;
signed int IIR_FILTER(const signed int  h[], signed int x1);

void main( )
{
             Uint32 GBLCTL_XHCLKRST_ON =      0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =      0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =      0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =      0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =      0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =      0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =      0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =      0X00000008;
          Uint32 GBLCTL_XFRST_ON       =      0X00001000;
          Uint32 GBLCTL_RFRST_ON       =      0X00000010;

      int temp,i;
      signed int input_data1, input_data2;
      //Uint16 x, y, z;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

             for(i=0; i<10; i++)
             {
                    SPI_SPIDAT1= temp | codec_reg_val[i];
                    while( !(SPI_SPIFLG&0x200));
                    DSP6745_wait( 1000 );
             }
             SPI_SPIGCR0 = 0;
                    DSP6745_wait( 1000 );

                           MCASP0_GBLCTL     = 0;
                        MCASP0_RGBLCTL     = 0;
                        MCASP0_XGBLCTL     = 0;
                        MCASP0_PWRDEMU     = 1;

                        MCASP0_RMASK       = 0xFFFFFFFF;
                        MCASP0_RFMT        = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSRCTL     = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKRCTL    = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
                        MCASP0_AHCLKRCTL   = 0x00008000;//INT CLK (from tx side)
                        MCASP0_RTDM        = 0x00000003; // Slots 0,1
                        MCASP0_RINTCTL     = 0x00000000; // Not used
                        MCASP0_RCLKCHK     = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_XMASK       = 0xFFFFFFFF; // No padding used
                        MCASP0_XFMT        = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSXCTL     = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKXCTL    = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                        MCASP0_AHCLKXCTL   = 0x00008000;// INT CLK, div-by-4
                        MCASP0_XTDM        = 0x00000003; // Slots 0,1
                        MCASP0_XINTCTL     = 0x00000000; // Not used
                        MCASP0_XCLKCHK     = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_SRCTL1      = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                        MCASP0_SRCTL0      = 0x0001;     // MCASP0.AXR0[0] --> DIN

                        MCASP0_PFUNC       = 0x00;
                        MCASP0_PDIR       = 0x00000001;
                        MCASP0_DITCTL      = 0x00000000; // Not used
                        MCASP0_DLBCTL      = 0x00000000; // Not used
                        MCASP0_AMUTE       = 0x00000000; // Not used

             /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                 MCASP0_XSTAT = 0x0000ffff;        // Clear all
```

```
                  MCASP0_RSTAT = 0x0000ffff;        // Clear all

                  MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                  /* Write a 0, so that no underrun occurs after releasing the state
machine */

                  MCASP0_XBUF0 = 0;

                  MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                  MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);

                  MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                  while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                  MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
                  while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );


                  /* Start by sending a dummy write */
                  while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                  MCASP0_XBUF0 = 0;

                  while(1)
                  {
                           while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                           input_data1 = MCASP0_RBUF1_32BIT;
                           input_data2 = IIR_FILTER(filter_Coeff , input_data1);
                           while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                           MCASP0_XBUF0_32BIT = (input_data2 << 16);

                  }

}
signed int IIR_FILTER(const signed int  h[], signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Must be
static */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Must be
static */
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-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Divide temp by coefficients[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);
}
```

## 2. Butterworth HPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;

void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}

void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );

    /* Release SPI */
    SPI_SPIGCR0 = 1;

    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
```

```
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
       //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers

    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );

    SPI_SPIDAT1 = spidat1;

    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;

    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}

const signed int filter_Coeff[] =
{
             3403, -3403, 3403, 32767, 15565, 17595//1621,-
1621,1621,32767,20964,15649//17926, -17926, 17926, 32767, -14372, 10192//
} ;
signed int IIR_FILTER(const signed int  h[], signed int x1);

void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =      0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =      0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =      0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =      0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =      0X00000400;
```

```
		Uint32 GBLCTL_RSRCLR_ON 	= 	0X00000004;
		Uint32 GBLCTL_XSMRST_ON 	= 	0X00000800;
		Uint32 GBLCTL_RSMRST_ON 	= 	0X00000008;
		Uint32 GBLCTL_XFRST_ON 		= 	0X00001000;
		Uint32 GBLCTL_RFRST_ON 		= 	0X00000010;

	int temp,i;
	signed int input_data1, input_data2;
	//Uint16 x, y, z;
	Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
	PINMUX7 = 0x10110000;
	PINMUX9 = 0x11011000;
	PINMUX10 = 0x00000110;
	spi_init( );
	temp = SPI_SPIDAT1;

		for(i=0; i<10; i++)
		{
			SPI_SPIDAT1= temp | codec_reg_val[i];
			while( !(SPI_SPIFLG&0x200));
			DSP6745_wait( 1000 );
		}
		SPI_SPIGCR0 = 0;
			DSP6745_wait( 1000 );

				 MCASP0_GBLCTL		= 0;
				MCASP0_RGBLCTL		= 0;
				MCASP0_XGBLCTL		= 0;
				MCASP0_PWRDEMU		= 1;

				MCASP0_RMASK		= 0xFFFFFFFF;
				MCASP0_RFMT		= 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
				MCASP0_AFSRCTL		= 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
				MCASP0_ACLKRCTL		= 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
				MCASP0_AHCLKRCTL	= 0x00008000;//INT CLK (from tx side)
				MCASP0_RTDM		= 0x00000003; // Slots 0,1
				MCASP0_RINTCTL		= 0x00000000; // Not used
				MCASP0_RCLKCHK		= 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

				MCASP0_XMASK		= 0xFFFFFFFF; // No padding used
				MCASP0_XFMT		= 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
				MCASP0_AFSXCTL		= 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
				MCASP0_ACLKXCTL		= 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
				MCASP0_AHCLKXCTL	= 0x00008000;// INT CLK, div-by-4
				MCASP0_XTDM		= 0x00000003; // Slots 0,1
				MCASP0_XINTCTL		= 0x00000000; // Not used
				MCASP0_XCLKCHK		= 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

				MCASP0_SRCTL1		= 0x0002;	// MCASP0.AXR0[1] <-- DOUT
```

```
                        MCASP0_SRCTL0     = 0x0001;     // MCASP0.AXR0[0] --> DIN

                          MCASP0_PFUNC      = 0x00;
                        MCASP0_PDIR          = 0x00000001;
                        MCASP0_DITCTL     = 0x00000000; // Not used
                        MCASP0_DLBCTL     = 0x00000000; // Not used
                        MCASP0_AMUTE      = 0x00000000; // Not used

             /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                 MCASP0_XSTAT = 0x0000ffff;        // Clear all
                 MCASP0_RSTAT = 0x0000ffff;        // Clear all

                 MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                 /* Write a 0, so that no underrun occurs after releasing the state
machine */

                 MCASP0_XBUF0 = 0;

                 MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);

                 MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
```

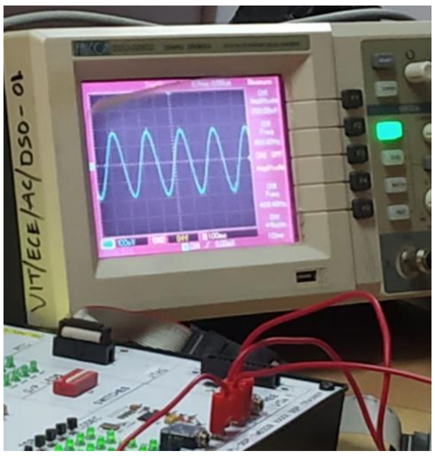
```
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );


                 /* Start by sending a dummy write */
                 while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                 MCASP0_XBUF0 = 0;

                 while(1)
                 {

                          while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                          input_data1 = MCASP0_RBUF1_32BIT;
                          input_data2 = IIR_FILTER(filter_Coeff , input_data1);
                          while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                          MCASP0_XBUF0_32BIT = (input_data2 << 16);

                 }


}

signed int IIR_FILTER(const signed int  h[], signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Must be
static */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Must be
static */
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-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Divide temp by coefficients[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);
```

}

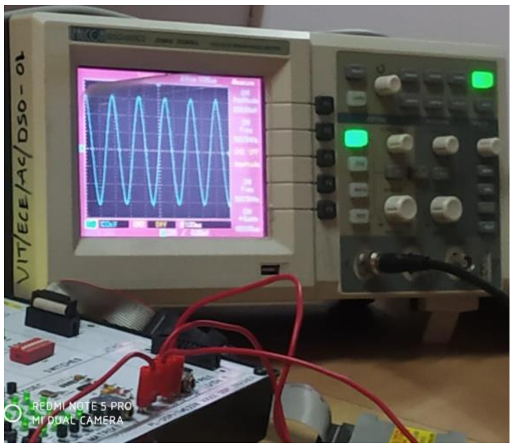
## OUTPUT:

- As we giving applied sine input through line in, the output will appear as per the filter type on DSO.
- Output will decreasing after the cutoff frequency for low pass filter
- Output will appear at the cutoff frequency for high pass filter

INPUT FREQUENCY (800Hz)



OUTPUT



## RESULT

## CONCLUSION

## VIVA QUESTIONS

1. Write the design procedure of butter worth IIR filter using Bilinear Transformation Method?

2. Write the design procedure of butter worth IIR filter using Impulse Invariant Method?

3. Write relation between S & Z in BLT?

4. Write relation between S & Z in IIM?

5. Compare BLT & IIM?

**Exp .No: 08** **Date:**

# DESIGN OF IIR CHEBYSHEV LP/HP FILTER

**AIM:** To design and implement IIR chebyshev LP/HP Filter for given specification.

**APPARATUS REQUIRED**

CODE COMPOSER STUDIO
TMS320C6745 DSP TRAINER KIT

**Procedure**

1. To create the New Project
2. Project→ New CCS Project (Give name to project with location to save or use default location)Select project type→ Executable
3. Device Family→C6000
4. Variant→C674xFloating-point DSP
5. Connection →Texas Instruments XDS100V2USB Emulator
6. Click on Empty Project then Finish
7. To create a Source file
8. File →New→ Source file (Save & give file name, Eg: sum.c).Click on Finish
9. Write the C-Program To build the program project →Build All
10. After Build Finished without errors, Now DSP kit is turned ON
11. Debug the Program after loading is done
12. Run the Program and verify the output
13. Connect CRO to the LINE OUT.
14. Connect a Signal Generator to the LINE IN.
15. Switch on the Signal Generator with a sine wave of frequency 100 Hz. and Vp-p=1.0v & vary the frequency.

1. Low pass Chebyshev filter (cutoff 1000Hz)
2. High pass Chebyshev filter (cutoff 1000Hz)

## 1. Chebyshev LPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;
void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}

void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );

    /* Release SPI */
    SPI_SPIGCR0 = 1;

    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers

    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );
    SPI_SPIDAT1 = spidat1;

    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;

    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}

const signed int filter_Coeff[] =
{
             1455,1455,1455,32767,-23140,21735 /*LP 2500 */
} ;

signed int IIR_FILTER(const signed int  h[], signed int x1);

void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =      0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =      0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =      0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =      0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =      0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =      0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =      0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =      0X00000008;
          Uint32 GBLCTL_XFRST_ON       =      0X00001000;
          Uint32 GBLCTL_RFRST_ON       =      0X00000010;
      int temp,i;
      signed int input_data1, input_data2;
      //Uint16 x, y, z;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

             for(i=0; i<10; i++)
             {
                    SPI_SPIDAT1= temp | codec_reg_val[i];
                    while( !(SPI_SPIFLG&0x200));
                    DSP6745_wait( 1000 );
             }
             SPI_SPIGCR0 = 0;
                    DSP6745_wait( 1000 );

                        MCASP0_GBLCTL     = 0;
                        MCASP0_RGBLCTL    = 0;
                        MCASP0_XGBLCTL     = 0;
                        MCASP0_PWRDEMU     = 1;

                        MCASP0_RMASK       = 0xFFFFFFFF;
                        MCASP0_RFMT        = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSRCTL     = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKRCTL    = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
                        MCASP0_AHCLKRCTL   = 0x00008000;//INT CLK (from tx side)
                        MCASP0_RTDM        = 0x00000003; // Slots 0,1
                        MCASP0_RINTCTL     = 0x00000000; // Not used
                        MCASP0_RCLKCHK     = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_XMASK       = 0xFFFFFFFF; // No padding used
                        MCASP0_XFMT        = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                        MCASP0_AFSXCTL     = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                        MCASP0_ACLKXCTL    = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                        MCASP0_AHCLKXCTL   = 0x00008000;// INT CLK, div-by-4
                        MCASP0_XTDM        = 0x00000003; // Slots 0,1
                        MCASP0_XINTCTL     = 0x00000000; // Not used
                        MCASP0_XCLKCHK     = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                        MCASP0_SRCTL1      = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                        MCASP0_SRCTL0      = 0x0001;     // MCASP0.AXR0[0] --> DIN

                          MCASP0_PFUNC      = 0x00;
                        MCASP0_PDIR          = 0x00000001;
                        MCASP0_DITCTL      = 0x00000000; // Not used
                        MCASP0_DLBCTL      = 0x00000000; // Not used
                        MCASP0_AMUTE       = 0x00000000; // Not used

             /* Starting sections of the McASP*/
                 MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                 MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                 MCASP0_XSTAT = 0x0000ffff;        // Clear all
                 MCASP0_RSTAT = 0x0000ffff;        // Clear all

                 MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                 /* Write a 0, so that no underrun occurs after releasing the state
machine */

                 MCASP0_XBUF0 = 0;

                 MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                 MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);

                 MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                 while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
                 MCASP0_RGBLCTL |= GBLCTL_RFRST_ON;
// Frame Sync
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );


                 /* Start by sending a dummy write */
                 while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                 MCASP0_XBUF0 = 0;

                 while(1)
                 {

                          while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                          input_data1 = MCASP0_RBUF1_32BIT;
                          input_data2 = IIR_FILTER(filter_Coeff , input_data1);
                          while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                          MCASP0_XBUF0_32BIT = (input_data2 << 16);
                 }


}
signed int IIR_FILTER(const signed int  h[], signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Must be
static */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Must be
static */
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-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Divide temp by coefficients[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);
}
```

## 2. Chebyshev HPF

**PROGRAM:**

```
#include<stdio.h>
#include "sysreg.h"
#define Uint32  unsigned int
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char
static Uint32 spidat1;

void DSP6745_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ )
    {

    }
}

void spi_init( )
{
    /* Reset SPI */
    SPI_SPIGCR0 = 0;
      DSP6745_wait( 1000 );

    /* Release SPI */
    SPI_SPIGCR0 = 1;

    /* SPI 4-Pin Mode setup */
    SPI_SPIGCR1 = 0
        | ( 0 << 24 )//Deactivates SPI
        | ( 0 << 16 )//Internal loop-back test mode disabled/enabled=0/1
        //| ( 1 << 1 )//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers
        | ( 3 << 0 );//MASTER MODE. SPIx_CLK is an output and the SPI initiates
transfers

    SPI_SPIPC0 = 0
        | ( 1 << 11 )   // DI
        | ( 1 << 10 )   // DO
        | ( 1 << 9 )    // CLK
        | ( 1 << 8 )    // EN0
        | ( 1 << 0 );   // CS

      spidat1 = 0
        | ( 0 << 28 )   // CSHOLD
        | ( 0 << 24 )   // DFSEL Format [0]
        | ( 1 << 26 )
        | ( 0 << 16 )   // CSNR
        | ( 0 << 0 );   //

    SPI_SPIFMT0 = 0
        | ( 0 << 20 )   // SHIFTDIR
        | ( 0 << 17 )   // Polarity
        | ( 1 << 16 )   // Phase
        | ( 14 << 8 )    // Prescale to 30MHz (150/(value+1))//29
        | ( 16 << 0 );   // Char Len | ( 1F << 0 );

    SPI_SPIDAT1 = spidat1;

    SPI_SPIDELAY = 0
        | ( 8 << 24 )   // C2TDELAY
        | ( 8 << 16 );  // T2CDELAY

    SPI_SPIDEF = 0
        | ( 1 << 1 )    // EN1 inactive high
        | ( 1 << 0 );   // EN0 inactive high

    SPI_SPIINT = 0;

    SPI_SPIDEF = 0x01;
    SPI_SPIGCR1 |= ( 1 << 24 );  //Enable SPI
}

const signed int filter_Coeff[] =
{
             11617, -11617, 11617, 32767, -8683, 15505//963, -963, 963, 32767,
25417, 23512//
} ;
signed int IIR_FILTER(const signed int  h[], signed int x1);

void main( )
{
          Uint32 GBLCTL_XHCLKRST_ON    =      0X00000200;
          Uint32 GBLCTL_RHCLKRST_ON    =      0x00000002;
          Uint32 GBLCTL_XCLKRST_ON     =      0X00000100;
          Uint32 GBLCTL_RCLKRST_ON     =      0X00000001;
          Uint32 GBLCTL_XSRCLR_ON      =      0X00000400;
          Uint32 GBLCTL_RSRCLR_ON      =      0X00000004;
          Uint32 GBLCTL_XSMRST_ON      =      0X00000800;
          Uint32 GBLCTL_RSMRST_ON      =      0X00000008;
          Uint32 GBLCTL_XFRST_ON       =      0X00001000;
          Uint32 GBLCTL_RFRST_ON       =      0X00000010;

      int temp,i;
      signed int input_data1, input_data2;
      //Uint16 x, y, z;
      Uint16 codec_reg_val[] = {0X0017, 0X0217, 0X04D8, 0X06D8, 0X0811, 0X0A00,
0X0C00, 0X0E53, 0X100C, 0x1201};
      PINMUX7 = 0x10110000;
      PINMUX9 = 0x11011000;
      PINMUX10 = 0x00000110;
      spi_init( );
      temp = SPI_SPIDAT1;

            for(i=0; i<10; i++)
            {
                  SPI_SPIDAT1= temp | codec_reg_val[i];
                  while( !(SPI_SPIFLG&0x200));
                  DSP6745_wait( 1000 );
            }
            SPI_SPIGCR0 = 0;
                  DSP6745_wait( 1000 );

                        MCASP0_GBLCTL     = 0;
                       MCASP0_RGBLCTL    = 0;
                       MCASP0_XGBLCTL    = 0;
                       MCASP0_PWRDEMU    = 1;

                       MCASP0_RMASK      = 0xFFFFFFFF;
                       MCASP0_RFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                       MCASP0_AFSRCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                       MCASP0_ACLKRCTL   = 0x00000081;//0x000000AF; // Rising
INTERNAL CLK(from tx side)
                       MCASP0_AHCLKRCTL  = 0x00008000;//INT CLK (from tx side)
                       MCASP0_RTDM       = 0x00000003; // Slots 0,1
                       MCASP0_RINTCTL    = 0x00000000; // Not used
                       MCASP0_RCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256

                       MCASP0_XMASK      = 0xFFFFFFFF; // No padding used
                       MCASP0_XFMT       = 0x00018078; // MSB 16bit, 0-delay, no
pad, CFGBus
                       MCASP0_AFSXCTL    = 0x00000000; // 2TDM, 1bit Rising edge
INTERNAL FS, word
                       MCASP0_ACLKXCTL   = 0x00000081;//0x000000AF; // ASYNC,
Rising INTERNAL CLK, div-by-16
                       MCASP0_AHCLKXCTL  = 0x00008000;// INT CLK, div-by-4
                       MCASP0_XTDM       = 0x00000003; // Slots 0,1
                       MCASP0_XINTCTL    = 0x00000000; // Not used
                       MCASP0_XCLKCHK    = 0x00FF0008; // 255-MAX 0-MIN, div-by-
256
```

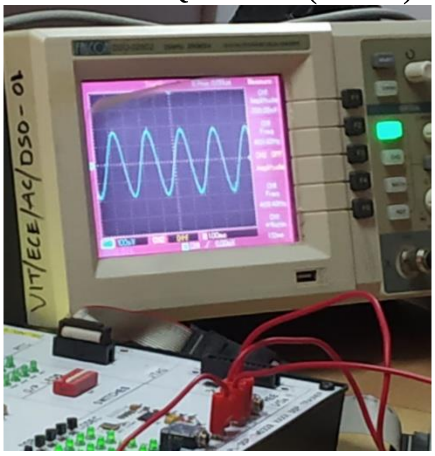
```
                                MCASP0_SRCTL1     = 0x0002;     // MCASP0.AXR0[1] <-- DOUT
                                MCASP0_SRCTL0     = 0x0001;     // MCASP0.AXR0[0] --> DIN

                                  MCASP0_PFUNC      = 0x00;
                                MCASP0_PDIR         = 0x00000001;
                                MCASP0_DITCTL     = 0x00000000; // Not used
                                MCASP0_DLBCTL     = 0x00000000; // Not used
                                MCASP0_AMUTE      = 0x00000000; // Not used

                      /* Starting sections of the McASP*/
                          MCASP0_XGBLCTL |= GBLCTL_XHCLKRST_ON;
// HS Clk
                          while ( ( MCASP0_XGBLCTL & GBLCTL_XHCLKRST_ON ) !=
GBLCTL_XHCLKRST_ON );
                          MCASP0_RGBLCTL |= GBLCTL_RHCLKRST_ON;
// HS Clk
                          while ( ( MCASP0_RGBLCTL & GBLCTL_RHCLKRST_ON ) !=
GBLCTL_RHCLKRST_ON );

                          MCASP0_XGBLCTL |= GBLCTL_XCLKRST_ON;
// Clk
                          while ( ( MCASP0_XGBLCTL & GBLCTL_XCLKRST_ON ) !=
GBLCTL_XCLKRST_ON );
                          MCASP0_RGBLCTL |= GBLCTL_RCLKRST_ON;
// Clk
                          while ( ( MCASP0_RGBLCTL & GBLCTL_RCLKRST_ON ) !=
GBLCTL_RCLKRST_ON );

                          MCASP0_XSTAT = 0x0000ffff;        // Clear all
                          MCASP0_RSTAT = 0x0000ffff;        // Clear all

                          MCASP0_XGBLCTL |= GBLCTL_XSRCLR_ON;
// Serialize
                          while ( ( MCASP0_XGBLCTL & GBLCTL_XSRCLR_ON ) != GBLCTL_XSRCLR_ON
);
                          MCASP0_RGBLCTL |= GBLCTL_RSRCLR_ON;
// Serialize
                          while ( ( MCASP0_RGBLCTL & GBLCTL_RSRCLR_ON ) != GBLCTL_RSRCLR_ON
);

                          /* Write a 0, so that no underrun occurs after releasing the state
machine */

                          MCASP0_XBUF0 = 0;

                          MCASP0_XGBLCTL |= GBLCTL_XSMRST_ON;
// State Machine
                          while ( ( MCASP0_XGBLCTL & GBLCTL_XSMRST_ON ) != GBLCTL_XSMRST_ON
);
                          MCASP0_RGBLCTL |= GBLCTL_RSMRST_ON;
// State Machine
                          while ( ( MCASP0_RGBLCTL & GBLCTL_RSMRST_ON ) != GBLCTL_RSMRST_ON
);

                          MCASP0_XGBLCTL |= GBLCTL_XFRST_ON;
// Frame Sync
                          while ( ( MCASP0_XGBLCTL & GBLCTL_XFRST_ON ) != GBLCTL_XFRST_ON );
```

```
                 MCASP0_RGBLCTL  |= GBLCTL_RFRST_ON;
// Frame Sync
                 while ( ( MCASP0_RGBLCTL & GBLCTL_RFRST_ON ) != GBLCTL_RFRST_ON );


                 /* Start by sending a dummy write */
                 while( ! ( MCASP0_SRCTL0 & 0x10 ) );  // Check for Tx ready
                 MCASP0_XBUF0 = 0;

                 while(1)
                 {

                          while ( ! ( MCASP0_SRCTL1 & 0x20 ) );
                          input_data1 = MCASP0_RBUF1_32BIT;
                          input_data2 = IIR_FILTER(filter_Coeff , input_data1);
                          while ( ! ( MCASP0_SRCTL0 & 0x10 ) );
                          MCASP0_XBUF0_32BIT = (input_data2 << 16);

                 }


}

signed int IIR_FILTER(const signed int  h[], signed int x1)
{
static signed int x[6] = { 0, 0, 0, 0, 0, 0 }; /* x(n), x(n-1), x(n-2). Must be
static */
static signed int y[6] = { 0, 0, 0, 0, 0, 0 }; /* y(n), y(n-1), y(n-2). Must be
static */
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-1) */
temp -= ( (int)h[5] * y[2]); /* A2 * y(n-2) */
/* Divide temp by coefficients[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);
}
```

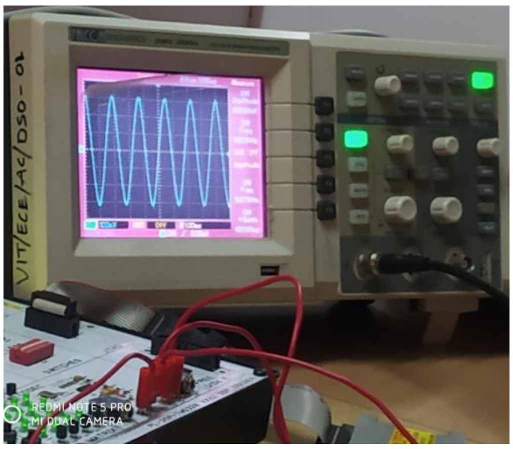
**OUTPUT:**

- As we giving applied sine input through line in, the output will appear as per the filter type on DSO.
- Output will decreasing after the cutoff frequency for low pass filter
- Output will appear at the cutoff frequency for high pass filter

INPUT FREQUENCY (800Hz)



OUTPUT



**RESULT**

**CONCLUSION**

**Viva Questions**

1. Write the expression for the order of chebyshev filter?

2. Write the steps in designing chebyshev filter?

3. State the equation for finding the poles in chebyshev filter.

4. What is warping effect?

5. What are the properties of chebyshev filter?

**Exp .No:09** **Date:**

# DECIMATION AND INTERPOLATION

**AIM:** To write a C- program Implementation of Decimation process and Interpolation process

**Procedure to Work on Code Composer Studio**

To create the New Project
Project→ New CCS Project (Give name to project with location to save or use default location)
Select project type→ Executable
Device Family→C6000
Variant→C674xFloating-point DSP
Connection →Texas Instruments XDS100V2USB Emulator
Click on Empty Project then Finish
To create a Source file
File →New→ Source file (Save & give file name, Eg: sum.c).Click on Finish
Write the C-Program To build the program project →Build All
After Build Finished without errors, Now DSP kit is turned ON
Debug the Program after loading is done
Run the Program and verify the output

**1. PROGRAM:**

Decimation process

```
include<stdio.h>
#include<math.h>
#define TIME 100
#define PI 3.14
#define L 4
#define M 4
float x[TIME],y[TIME*L],z[TIME*L],z1[TIME*L],y1[TIME];
float temp12_L[31], temp12_M[31];
static float in_buffer[100];

//float FIR_FILTER(float * , float );

float FIR_FILTER(float  h[] , float x)
{
	int i=0;
	float output=0;
	in_buffer[0] = x; /* new input at buffer[0] */
	for(i=29;i>0;i--)
	in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
	for(i=0;i<31;i++)
	output = output + h[i] * in_buffer[i];
	return(output);
}

void delay1()
{
	int y=150000;
	while(y!=0)
		y--;

}

void sinewave(float fm, float fc, int no, int result)
{

	float m;
	float n;

				delay1();
				delay1();
				m = 6.28 * fm * no;
				delay1();
				delay1();
				n = (m / fc);
				delay1();
				delay1();
				if(result == 0)
				{
					delay1();
					x[no]= sin(n);
					delay1();
				}
				if(result == 1)
				{
					delay1();
					temp12_L[no]= sin(n);
```

```
                        delay1();
                }
                if(result == 2)
                {
                        delay1();
                        temp12_M[no]= sin(n);
                        delay1();
                }
                delay1();
                delay1();

}



void main()
{
        int i,j,p,q;
    float filter_Coeff_L[31],xx,yy,zz,zz1,zz2,filter_Coeff_M[31];

        for(i=0;i<TIME;i++)
                sinewave(1000, 8000.0, i, 0);
        printf("\ninterpolation\n");

        filter_Coeff_L[0]=1.0;
        for(i=1;i<31;i++)
        {

                sinewave(0.5, L, i, 1);
                xx = temp12_L[i];
                yy = PI * i;
                zz = yy/ L;
                filter_Coeff_L[i]=(xx/zz);
        }

        printf("\ndecimation\n");

        filter_Coeff_M[0]= L / M;
        for(i=1;i<31;i++)
        {

                sinewave(0.5, M, i, 2);
                xx = temp12_M[i];
                yy = PI * i;
                zz = yy/ M;
                zz1 = xx / zz;
                zz2 = (zz1 * L);
                filter_Coeff_M[i]=(zz2/M);
        }

        for(i=0;i<TIME*L;i=i+L)
        {
                p = (i / L);
                y[i]=x[p];
                for(j=i+1;j<(i+L);j++)
                        y[j]=0.000000;
        }
```

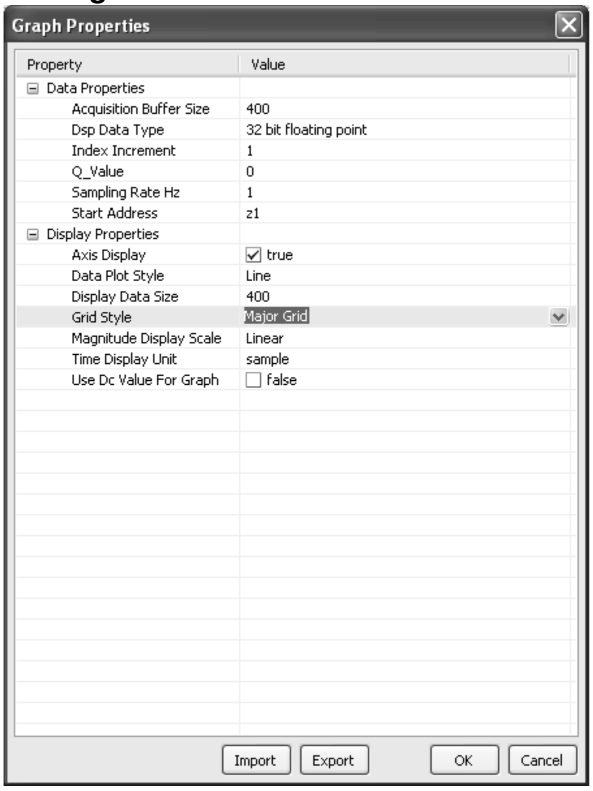
```
	for(i=0; i<100; i++)
	in_buffer[i] = 0;

	for(i=0;i<TIME*L;i++)
	{
		delay1();
		z[i]=FIR_FILTER(filter_Coeff_L ,y[i]);
		delay1();

	}
	for(i=0; i<100; i++)
	in_buffer[i] = 0;

	for(i=0;i<TIME*L;i++)
	{
		delay1();
		z1[i]=FIR_FILTER(filter_Coeff_M ,z[i]);
		delay1();
	}

	for(i=0;i<TIME;i++)
	{
		delay1();
		q = (i * L);
		delay1();
		y1[i]=z1[q];
		delay1();
		printf("\n%f",y1[i]);
	}


	printf("\nCompleted\n");

}
```

## OUTPUT

- We generated sample interpolated input named as z1
- Then it pass through decimation process, that output named as y1
- To view input graphically,

.

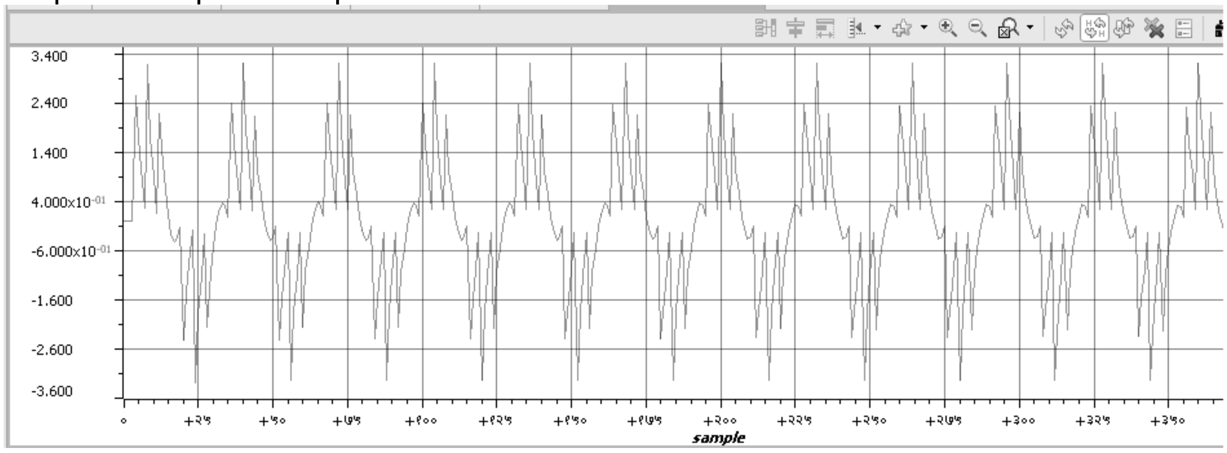
Select **Tool→ graph →single time**



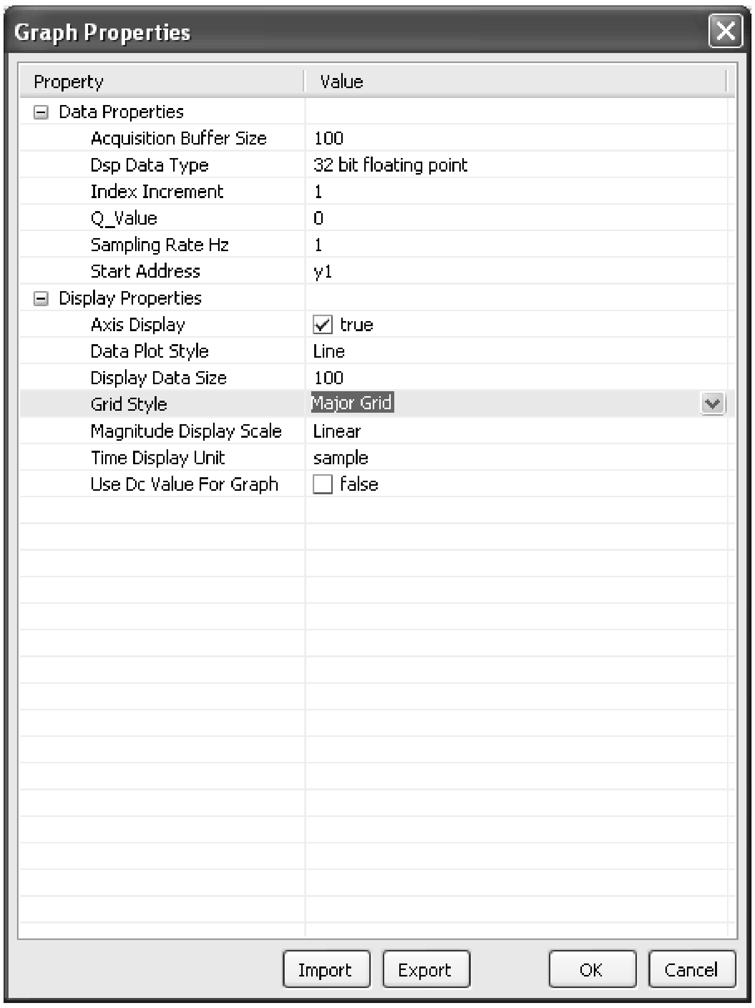
Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 400 |
| Dsp Data Type | 32 bit floating point |
| Index Increment | 1 |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address | z1 |
| ⊟ Display Properties | |
| Axis Display | ✓ true |
| Data Plot Style | Line |
| Display Data Size | 400 |
| Grid Style | Major Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | sample |
| Use Dc Value For Graph | ☐ false |

Graph of interpolated input



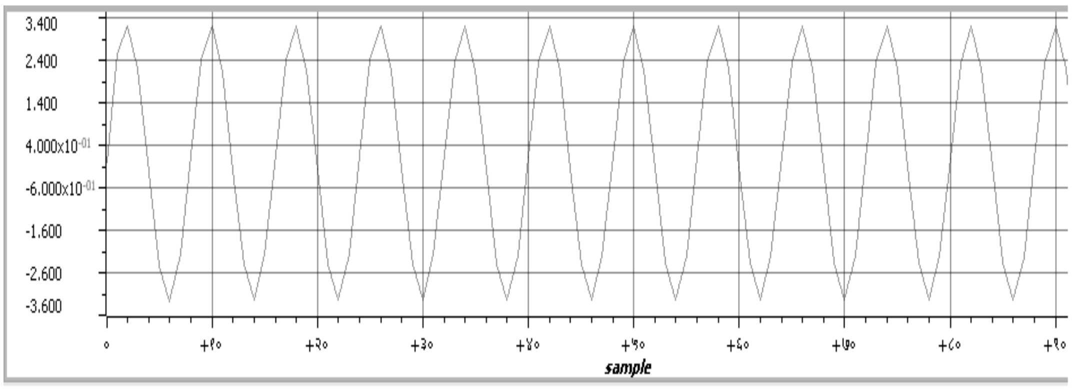
To view output graphically,
Select **Tool→ graph →single time**.



Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 100 |
| Dsp Data Type | 32 bit floating point |
| Index Increment | 1 |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address | y1 |
| ⊟ Display Properties | |
| Axis Display | ✓ true |
| Data Plot Style | Line |
| Display Data Size | 100 |
| Grid Style | Major Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | sample |
| Use Dc Value For Graph | ☐ false |

Graph of output after decimation



**2. PROGRAM:**

**Interpolation process**

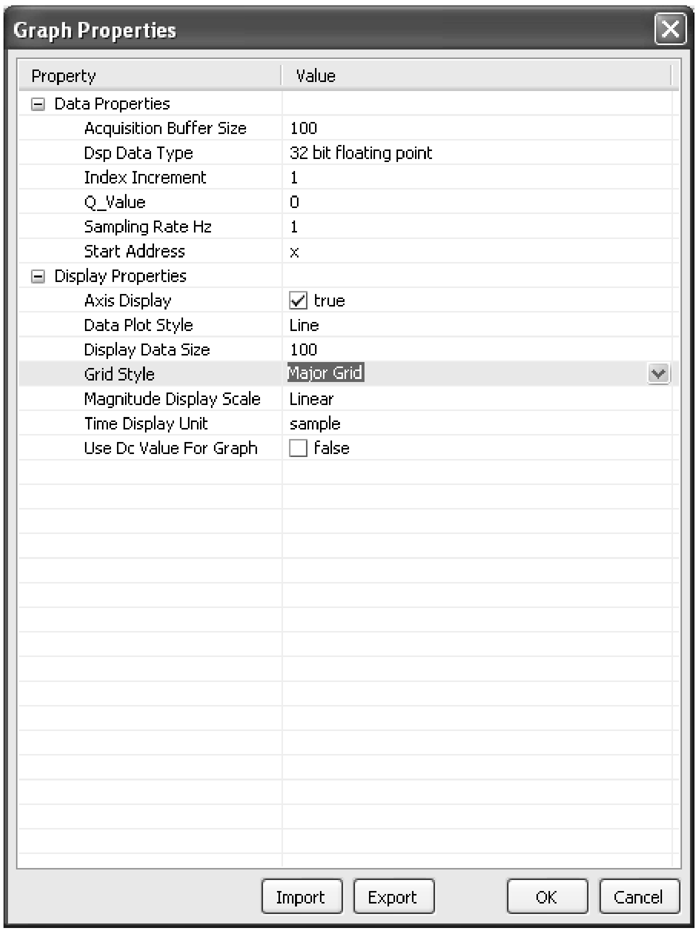
```
#include<stdio.h>
#include<math.h>
#define TIME 100
#define PI 3.14
#define L 4
float x[TIME],y[TIME*L],z[TIME*L];
float temp12[31];
static float in_buffer[100]={0.0};

//float FIR_FILTER(float * , float );

float FIR_FILTER(float  h[] , float x)
{
	int i=0;
	float output=0;
	in_buffer[0] = x; /* new input at buffer[0] */
	for(i=29;i>0;i--)
	in_buffer[i] = in_buffer[i-1]; /* shuffle the buffer */
	for(i=0;i<31;i++)
	output = output + h[i] * in_buffer[i];
	return(output);
}

void delay1()
{
	int y=150000;
	while(y!=0)
		y--;

}

void sinewave(float fm, float fc, int no, int result)
{

	float m;
	float n;

			delay1();
			delay1();
			m = 6.28 * fm * no;
			delay1();
			delay1();
			n = (m / fc);
			delay1();
			delay1();
			if(result == 0)
			{
				delay1();
				x[no]= sin(n);
				delay1();
			}
			else
			{
			if(result == 1)
```

```
                    {
                          delay1();
                          temp12[no]= sin(n);
                          delay1();
                    }
                    }
                    delay1();
                    delay1();

}



void main()
{
      int i,j;
    float filter_Coeff_L[31],xx,yy,zz;

      for(i=0;i<TIME;i++)
            sinewave(1000, 8000.0, i, 0);
            //x[i]=sin(2*3.14*1000*i/8000);
      filter_Coeff_L[0]=1.0;
      for(i=1;i<31;i++)
      {

            sinewave(0.5, 4, i, 1);
            xx = temp12[i];
            yy = PI * i;
            zz = yy/ L;
            filter_Coeff_L[i]=(xx/zz);
            //printf("\n%f",filter_Coeff_L[i]);
      }
      for(i=0;i<TIME*L;i=i+L)
      {
            y[i]=x[i/L];
            for(j=i+1;j<(i+L);j++)
                  y[j]=0.000000;
      }

      for(i=0;i<TIME*L;i++)
      {
            delay1();
            z[i]=FIR_FILTER(filter_Coeff_L ,y[i]);
            delay1();
            printf("\n%f",z[i]);
            //delay1();
      }



}
```

## OUTPUT

- We generated sample sine wave as a input named as x
- Then it pass through interpolation process, that output named as z
- To view input graphically,

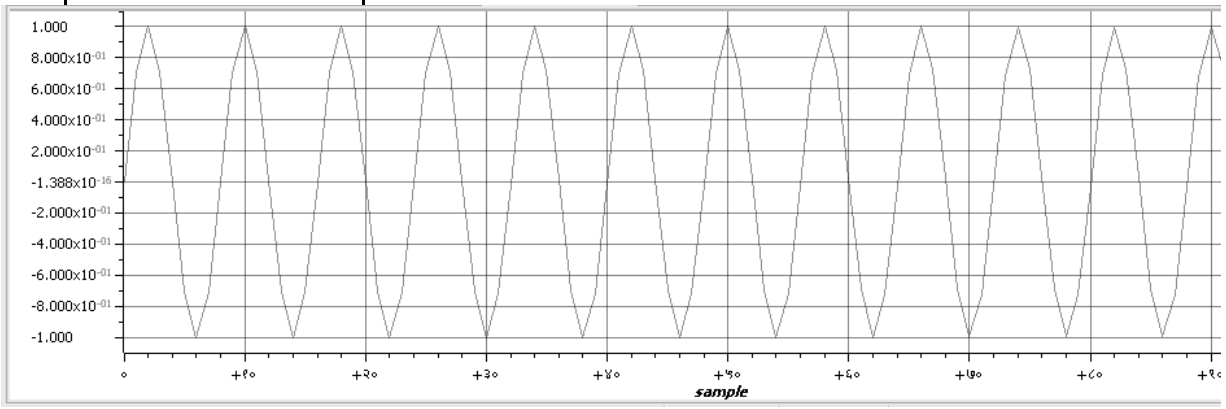
Select Tool→ graph →single time



Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 100 |
| Dsp Data Type | 32 bit floating point |
| Index Increment | 1 |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address | x |
| ⊟ Display Properties | |
| Axis Display | ✓ true |
| Data Plot Style | Line |
| Display Data Size | 100 |
| Grid Style | Major Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | sample |
| Use Dc Value For Graph | ☐ false |

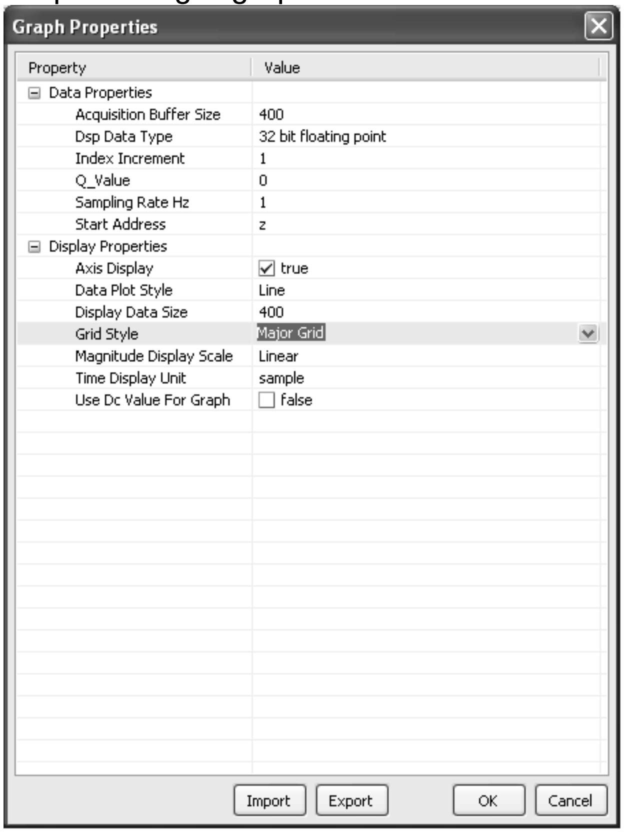
Sample sine wave as a input X



To view interpolation graphically,

Select Tool→ graph →single time

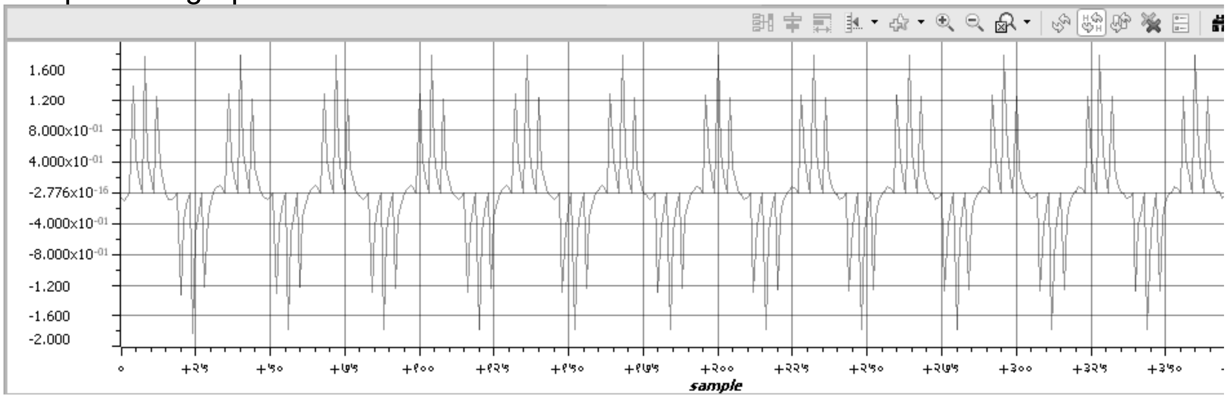
## Graph setting & graph



Graph Properties

| Property | Value |
|---|---|
| ⊟ Data Properties | |
| Acquisition Buffer Size | 400 |
| Dsp Data Type | 32 bit floating point |
| Index Increment | 1 |
| Q_Value | 0 |
| Sampling Rate Hz | 1 |
| Start Address | z |
| ⊟ Display Properties | |
| Axis Display | ✓ true |
| Data Plot Style | Line |
| Display Data Size | 400 |
| Grid Style | Major Grid |
| Magnitude Display Scale | Linear |
| Time Display Unit | sample |
| Use Dc Value For Graph | ☐ false |

## Interpolation graph



**RESULT:**

**CONCLUSION**

**VIVA QUESTIONS**

1. Explain about multi rate digital signal processing.

2. List the Applications of multi rate digital signal processing.

3. Define interpolation.

4. Define decimation.

5. Define aliasing

# ADVANCED EXPERIMENTS

**Exp .No: 01** **Date:**

## CONVERTING CD DATA TO DVD DATA

**1. AIM:** To write a MATLAB program to Converting CD DATA TO DVD DATA.

**2. SOFTWARE REQUIRED:**

- PC and MATLAB software

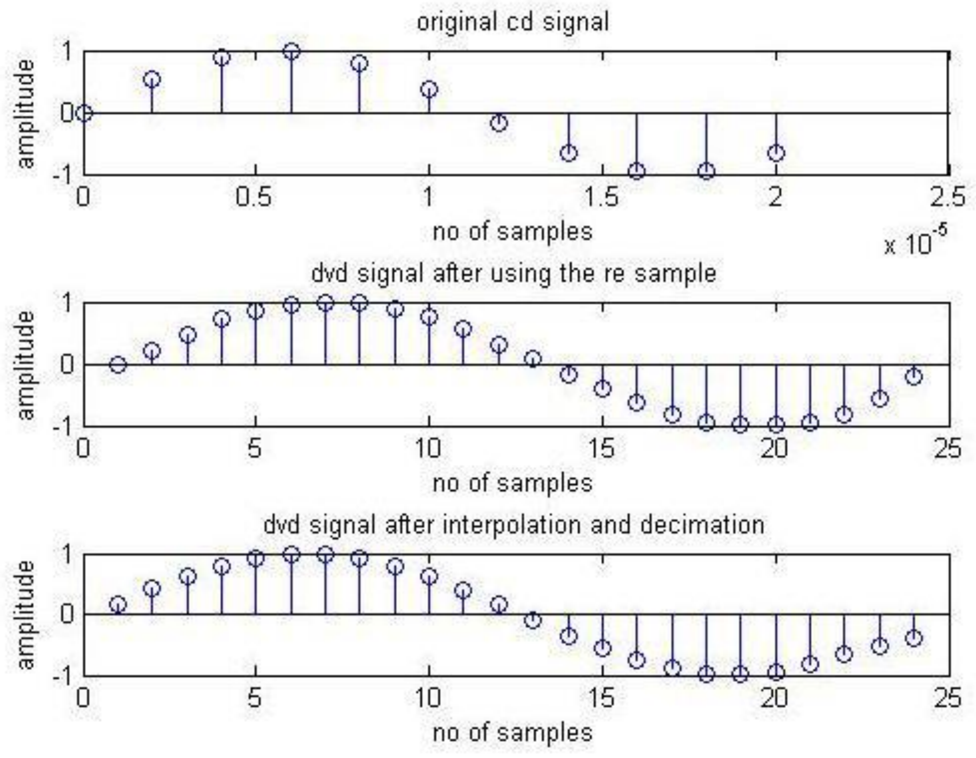
**3. PROCEDURE:**

- 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

**4. PROGRAM:**

```
clc;
clear all;
fc=44100;
t=0:0.000002:0.00002;
x=sin(2*pi*fc*t);
subplot(3,1,1);
stem(t,x);
title(&#39;original cd signal&#39;);
xlabel(&#39;no of samples&#39;);
ylabel(&#39;amplitude&#39;);
i=13;
d=6;
y=resample(x,i,d);
subplot(3,1,2);
stem(y);
title(&#39;dvd signal after using the re sample&#39;);
xlabel(&#39;no of samples&#39;);
ylabel(&#39;amplitude&#39;);
in=interp(x,i);
de=decimate(in,d);
subplot(3,1,3);
stem(de);
title(&#39;dvd signal after interpolation and decimation&#39;);
xlabel(&#39;no of samples&#39;);
ylabel(&#39;amplitude&#39;);
```

**OUTPUT:**



6.

**RESULT:**

**CONCLUSION**

## 7. VIVA QUESTIONS

1. How does poly phase filtering save computations in an interpolation filter?

2. Why do we need I&Q signals?

3. What is Interpolation and decimation filters and why we need it?

4. What are "up sampling" and "interpolation"?

5. What is the "interpolation factor"?

**Exp .No: 02** **Date:**

## POWER DENSITY SPECTRUM

**AIM:** To write a C- program to compute power density spectrum of given one – dimensional signal and plot.

**APPARATUS REQUIRED:**

- Code Composer Studio
- TMS320C6745 DSP Trainer Kit

**PROCEDURE:**

1. To create the New Project
2. Project→ New CCS Project (Give name to project with location to save or use default
3. location)
4. Select project type→ Executable
5. Device Family→C6000
6. Variant→C674xFloating-point DSP
7. Connection →Texas Instruments XDS100V2USB Emulator
8. Click on Empty Project then Finish
9. To create a Source file
10. File →New→ Source file (Save &amp; give file name, Eg: sum.c).Click on Finish
11. Write the C-Program To build the program project →Build All
12. After Build Finished without errors, Now DSP kit is turned ON
13. Debug the Program after loading is done
14. Run the Program and verify the output

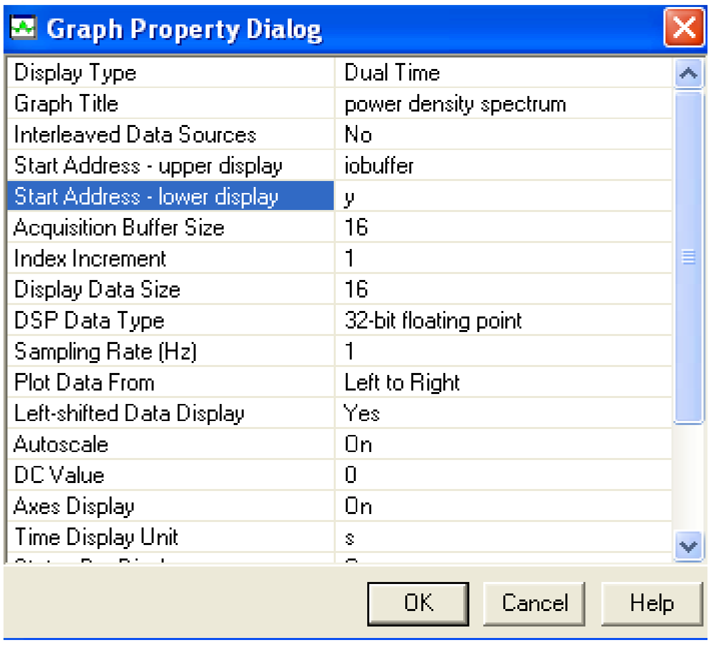
**PROGRAM:**

```
#include<stdio.h>
#include<math.h>
#define N 16
#define PI 3.14159
typedef struct
{
float real,imag;
}
complex;
complex x[N];
float iobuffer[N];
float x1[N],y[N];
int i;
main()
{
complex w[N];
complex temp1,temp2;
float sum=0.0;
int j,k,n,upper_leg,lower_leg,leg_diff,index,step;
```

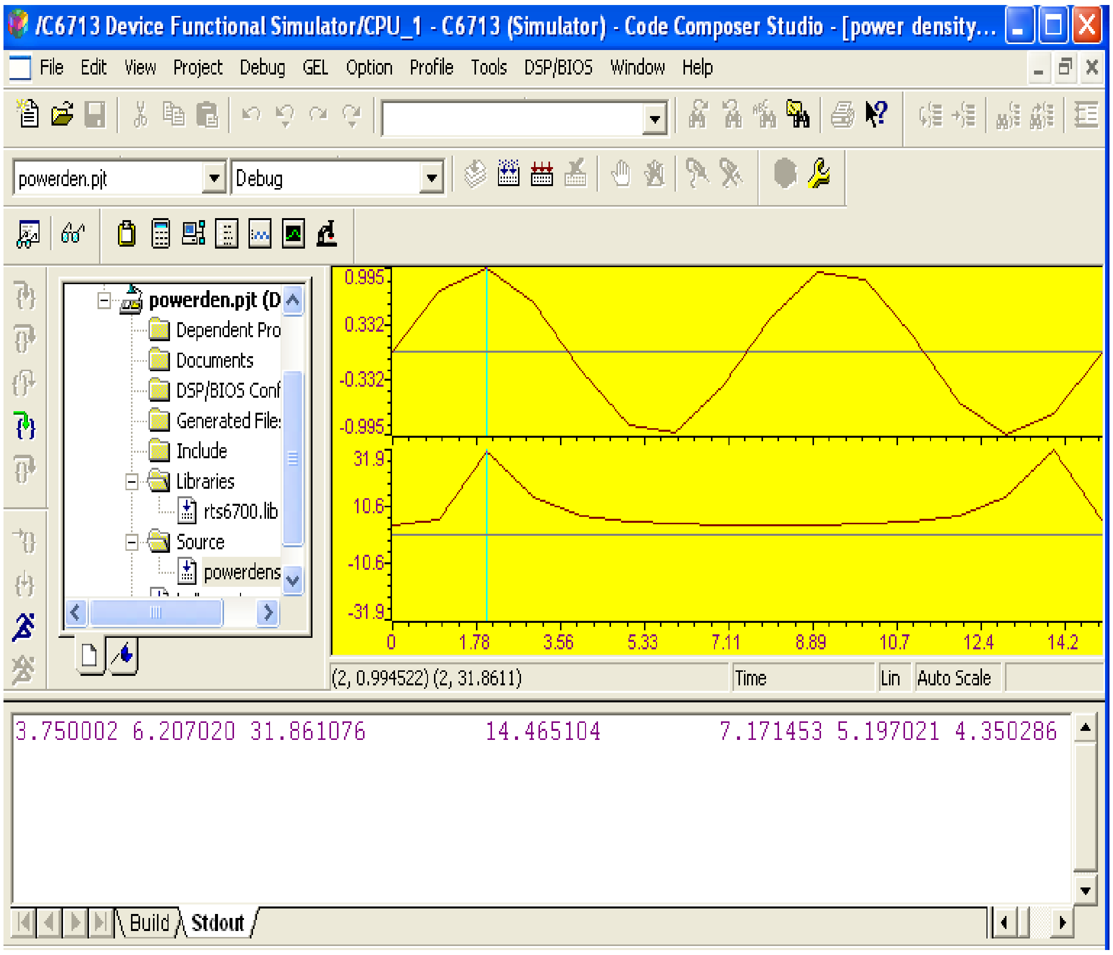
```
for(i=0;i<N;i++)
{

iobuffer[i]=sin((2*PI*2*i)/15.0);
}
for(n=0;n<N;n++)
{
sum=0;
for(k=0;k<N-n;k++)
{
sum=sum+(iobuffer[k]*iobuffer[n+k]);
}
x1[n]=sum;
}
for(i=0;i<N;i++)
{
x[i].real=x1[i];
x[i].imag=0.0;
}
for(i=0;i<N;i++)
{
w[i].real=cos((2*PI*i)/(N*2.0));
w[i].imag=-sin((2*PI*i)/(N*2.0));
}
leg_diff=N/2;
step=2;
for(i=0;i<4;i++)
{
index=0;
for(j=0;j<leg_diff;j++)
{
for(upper_leg=j;upper_leg<N;upper_leg+=(2*leg_diff))
{
lower_leg=upper_leg+leg_diff;
temp1.real=(x[upper_leg]).real+(x[lower_leg]).real;
temp1.imag=(x[upper_leg]).imag+(x[lower_leg]).imag;
temp2.real=(x[upper_leg]).real-(x[lower_leg]).real;
temp2.imag=(x[upper_leg]).imag-(x[lower_leg]).imag;
(x[lower_leg]).real=temp2.real*(w[index]).real-temp2.imag*(w[index]).imag;
(x[lower_leg]).imag=temp2.real*(w[index]).imag+temp2.imag*(w[index]).real;
(x[upper_leg]).real=temp1.real;
(x[upper_leg]).imag=temp1.imag;
}
index+=step;
}

leg_diff=(leg_diff)/2;
step=step*2;
}
j=0;
```

```
for(i=1;i<(N-1);i++)
{
k=N/2;
while(k<=j)
{
j=j-k;
k=k/2;
}
j=j+k;
if(i<j)
{
temp1.real=(x[j]).real;
temp1.imag=(x[j]).imag;
(x[j]).real=(x[i]).real;
(x[j]).imag=(x[i]).imag;
(x[i]).real=temp1.real;
(x[i]).imag=temp1.imag;
}
}
for(i=0;i<N;i++)
{
y[i]=sqrt((x[i].real*x[i].real)+(x[i].imag*x[i].imag));
}
for(i=0;i<N;i++)
{
printf("%f\t",y[i]);
}
return(0);
}
```

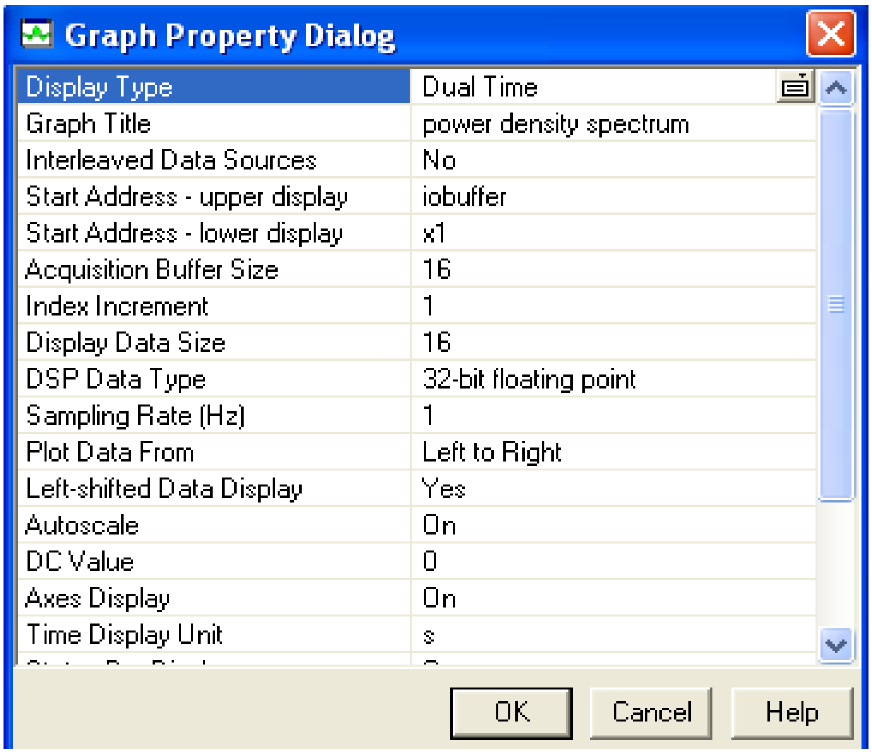
**OUTPUT:**



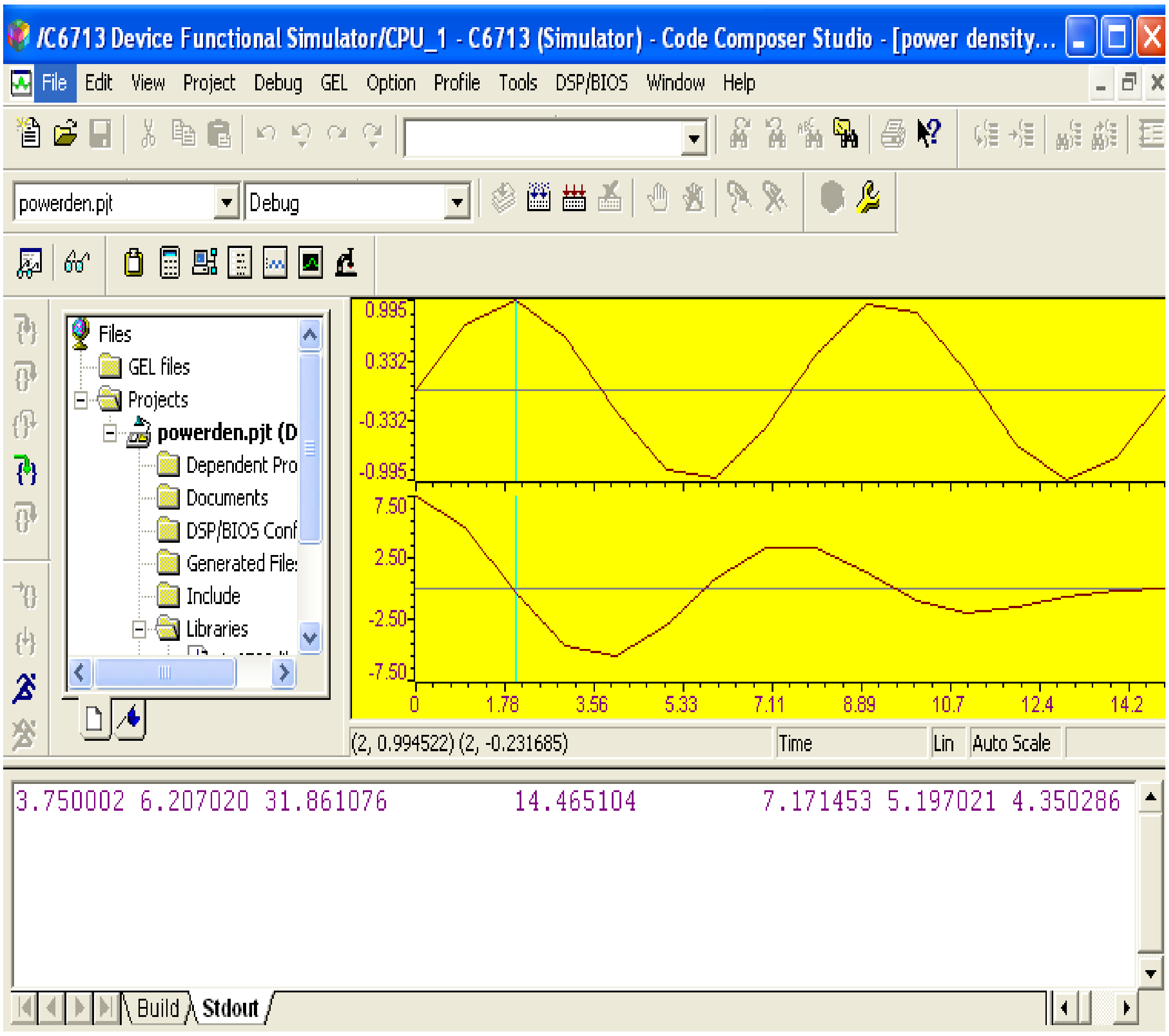
| | |
|---|---|
| Display Type | Dual Time |
| Graph Title | power density spectrum |
| Interleaved Data Sources | No |
| Start Address - upper display | iobuffer |
| Start Address - lower display | y |
| Acquisition Buffer Size | 16 |
| Index Increment | 1 |
| Display Data Size | 16 |
| DSP Data Type | 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 |



```
3.750002 6.207020 31.861076        14.465104        7.171453 5.197021 4.350286
```



| | |
|---|---|
| Display Type | Dual Time |
| Graph Title | power density spectrum |
| Interleaved Data Sources | No |
| Start Address - upper display | iobuffer |
| Start Address - lower display | x1 |
| Acquisition Buffer Size | 16 |
| Index Increment | 1 |
| Display Data Size | 16 |
| 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 |



```
(2, 0.994522) (2, -0.231685)
```

```
3.750002 6.207020 31.861076          14.465104          7.171453 5.197021 4.350286
```

**RESULT:**

**CONCLUSION**

## VIVA QUESTIONS

1**.** What do you mean by phase spectrum and magnitude spectrum give comparison?

2. What is power spectral density used for?

**3.** How Power Spectral Density is used to Classify Noise?

4. What is Energy Spectral Density?

5. Which has the same power spectral density?