Gopalan College of Engineering and Management

ISO 9001 : 2008

APPROVED BY AICTE NEW DELHI, AFFILIATED TO VTU BELGAUM

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

DIGITAL SIGNAL PROCESSING LAB (10ECL57)
Vth Sem- ECE
2016-2017

Prepared by:	Reviewed by:	Approved by:
Srilakshmi B Dept. of ECE GCEM	Kavitha M V Head of the Department Dept. of ECE GCEM	Dr. A.A. Powly Thomas Principal GCEM

81/1, 182/1, Hoodi Village, Sonnenahalli, K.R. Puram, Bengaluru, Karnataka-560048.

INDEX

SYLLABUS

DIGITAL SIGNAL PROCESSING LABORATORY

Subject Code : **10ECL57** IA Marks : 25
No. of Practical Hrs/Week: 03 Exam Hours : 03
Total no. of Practical Hrs. : 42 Exam Marks : 50

A LIST OF EXPERIMENTS USING MATLAB

1. Verification of Sampling theorem.
2. Impulse response of a given system
3. Linear convolution of two given sequences.
4. Circular convolution of two given sequences
5. Autocorrelation of a given sequence and verification of its properties. Cross correlation of given sequences and verification of its properties.
6. Solving a given difference equation.
7. Computation of N point DFT of a given sequence and to plot magnitude and phase spectrum.
8. Linear convolution of two sequences using DFT and IDFT.
9. Circular convolution of two given sequences using DFT and IDFT
10. Design and implementation of FIR filter to meet given specifications.
11. Design and implementation of IIR filter to meet given specifications.

LIST OF EXPERIMENTS USING DSP PROCESSOR

1. Linear convolution of two given sequences.
2. Circular convolution of two given sequences.
3. Computation of N- Point DFT of a given sequence
4. Realization of an FIR filter (any type) to meet given specifications .The input can be a signal from function generator / speech signal.
5. Audio applications such as to plot time and frequency (Spectrum) display of Microphone output plus a cosine using DSP. Read a wav file and match with their respective spectrograms
6. Noise: Add noise above 3kHz and then remove; Interference suppression using 400 Hz tone.
7. Impulse response of first order and second order system

Objectives:

- Analyze and implement digital signal processing systems in time domain.
- Design frequency-selective digital filters.
- Design digital filters using windows.
- Sample and reconstruct analog signals.
- Compute circular convolution,linear convolution and the discrete Fourier transform (DFT) of discrete-time signals.
- Analyze and implement digital systems using the DFT and the Fast Fourier Transform (FFT).
- Use MATLAB for DSP system analysis and design.

Outcomes:

- The student will be able to carry out simulation of DSP systems.
- The student will be able to demonstrate their abilities towards DSP processor based implementation of DSP systems.
- The student will be able to analyze Finite word length effect on DSP systems.
- The student will be able to demonstrate the applications of FFT to DSP.
- The student will be able to implement adaptive filters for various applications of DSP.

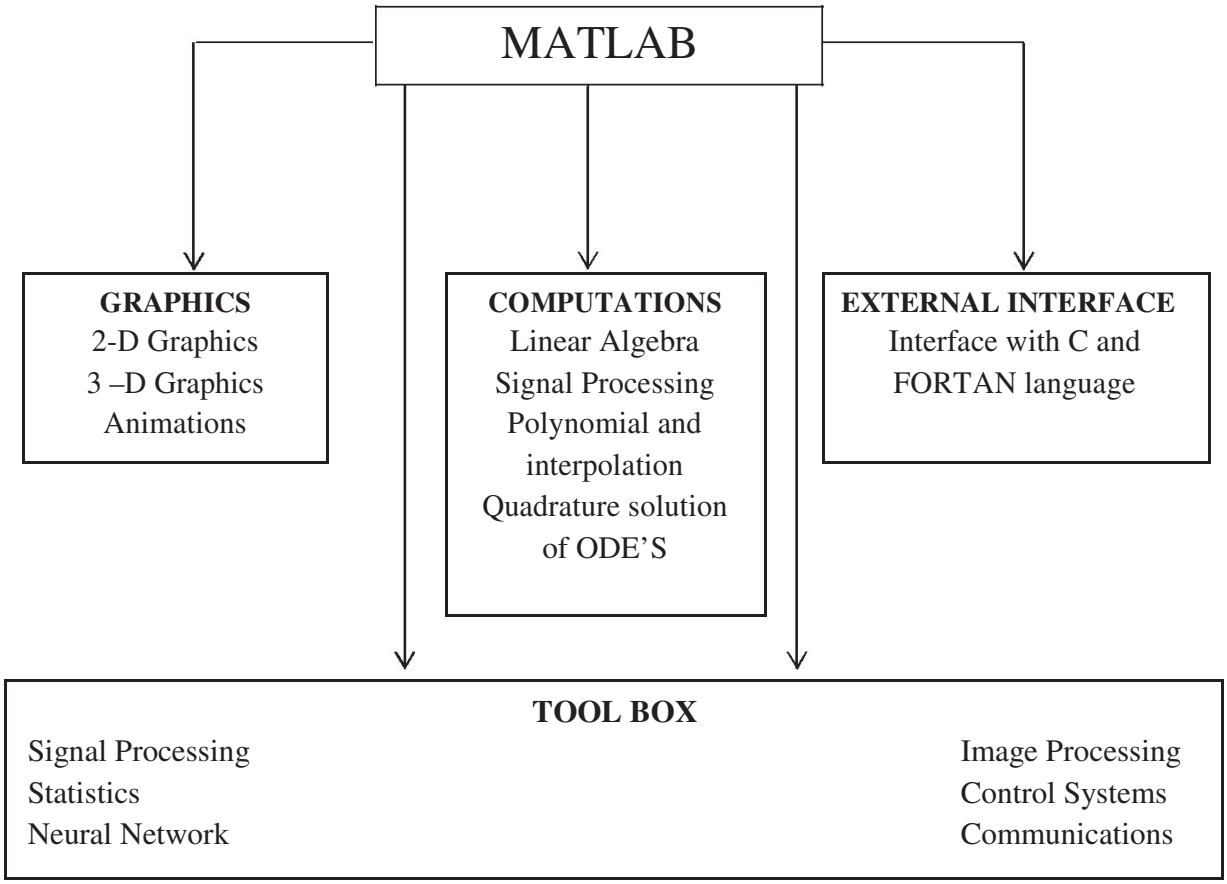
Do's and Dont's:

- Do not handle any equipment without reading the instructions /Instruction manuals.
- Observe type of sockets of equipment power to avoid mechanical damage.
- Do not insert connectors forcefully in the Sockets.
- Strictly observe the instructions given by the Teacher/ Lab Instructor.
- It is mandatory to come to lab in a formal dress (Shirts, Trousers, ID card, and Shoes for boys).
- Strictly no Jeans for both Girls and Boys.
- It is mandatory to come with observation book and lab record in which previous experiment should be written in Record and the present lab's experiment in Observation book.
- Observation book of the present lab experiment should be get corrected on the same day and
- Record should be corrected on the next scheduled lab session.
- Mobile Phones should be Switched OFF in the lab session.
- Students have to come to lab in-time. Late comers are not allowed to enter the lab.
- Prepare for the viva questions. At the end of the experiment, the lab faculty will ask the viva
- Questions and marks are allotted accordingly.
- Bring all the required stationery like graph sheets, pencil & eraser, different color pens etc. for the lab class.

List of Experiments

DSP USING MATLAB

MATLAB:MATLAB is a software package for high performance numerical computation and visualization provides an interactive environment with hundreds of built in functions for technical computation, graphics and animation. The MATLAB name stands for MATrix Laboratory The diagram shows the main features and capabilities of MATLAB.



At its core ,MATLAB is essentially a set (a "toolbox") of routines (called "m files" or "mex files") that sit on your computer and a window that allows you to create new variables with names (e.g. voltage and time) and process those variables with any of those routines (e.g. plot voltage against time, find the largest voltage, etc).

It also allows you to put a list of your processing requests together in a file and save that combined list with a name so that you can run all of those commands in the same order at some later time. Furthermore, it allows you to run such lists of commands such that you

pass in data and/or get data back out (i.e. the list of commands is like a function in most programming languages). Once you save a function, it becomes part of your toolbox (i.e. it now looks to you as if it were part of the basic toolbox that you started with).

For those with computer programming backgrounds: Note that MATLAB runs as aninterpretive language (like the old BASIC). That is, it does not need to be compiled. It simply reads through each line of the function, executes it, and then goes on to the next line. (In practice, a form of compilation occurs when you first run a function, so that it can run faster the next time you run it.

MATLAB Windows:

MATLAB works with through three basic windows

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

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

Edit window: This is where you write edit, create and save your own programs in filescalled M files.

INPUT-OUTPUT:

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

Data Type: the fundamental data –type in MATLAB is the array. It encompasses several distinct data objects- integers, real numbers, matrices, character strings, structures and cells.There is no need to declare variables as real or complex, MATLAB automatically sets the variable to be real.

Dimensioning: Dimensioning is automatic in MATLAB. No dimension statements are required for vectors or arrays .we can find the dimensions of an existing matrix or a vector with the size and length commands.

BASIC INSTRUCTIONS IN MAT LAB

1. **T = 0: 1:10**

This instruction indicates a vector T which as initial value 0 and final value 10 with an increment of 1

Therefore T = [0 1 2 3 4 5 6 7 8 9 10]

2. **F= 20: 1: 100**

Therefore F = [20 21 22 23 24 …… 100]

3. **T= 0:1/pi: 1**

Therefore T= [0, 0.3183, 0.6366,

0.9549] 4. **zeros (1, 3)**

The above instruction creates a vector of one row and three columns whose values are zero
Output= [0 0 0]

5. **ones (5,2)**

The above instruction creates a vector of five rows and two columns

Output = 1 1

1 1

1 1

1 1

1 1

6. **zeros (2, 4)**

Output = 0 0 0 0

0 0 0 0

7. a = [1 2 3] , b = [4 5 6]

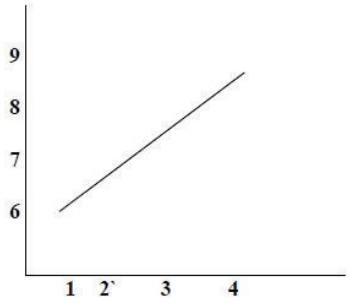
THEN **a.*b** = [4 10 18]

i.e. [4*1 5*2 6*3]

8. if C= [2 2 2] and b = [4 5 6]
   b.*C results in [8 10 12]

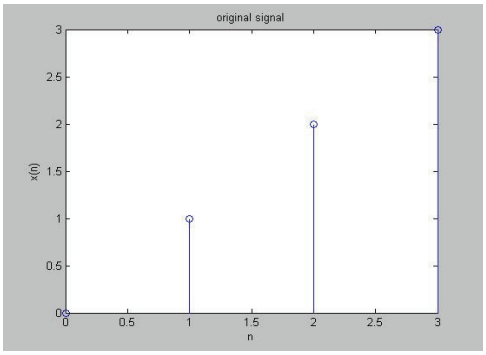
9. **plot (t, x)**

If x = [6 7 8 9] and t = [1 2 3 4]; This instruction will display a figure window which indicates the plot of x versus t

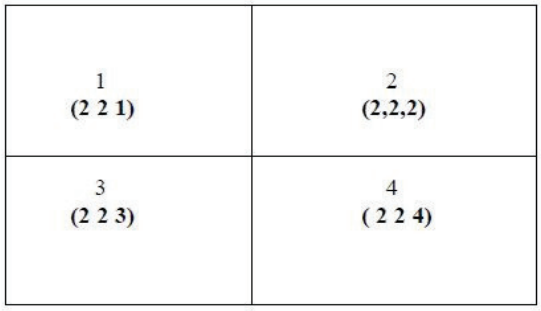


10. **stem (t,x**)

If x(n) = [0 1 2 3];This instruction will display a figure window as shown



11. **Subplot**: This function divides the figure window into rows and columns Subplot (2 2 1) divides the figure window into 2 rows and 2 columns 1 represent number of the figure



Subplot(3, 1, 3)

12. Filter:**Syntax:** y = filter(b,a,X)

Description:y = filter(b,a,X) filters the data in vector X with the filter described by numerator coefficient vector b and denominator coefficient vector a. If a(1) is not equal to 1, filter normalizes the filter coefficients by a(1). If a(1) equals 0, filter returns an error.

13. Fliplr: **Syntax: B =** fliplr(A)

Description:B = fliplr(A) returns A with columns flipped in the left-right direction, that is, about a vertical axis.If A is a row vector, then fliplr(A) returns a vector of the same length with the order of its elements reversed. If A is a column vector, then fliplr(A) simply returns A.

14. Conv: **Syntax:** w = conv(u,v)

Description:w = conv(u,v) convolves vectors u and v. Algebraically, convolution is the same operation as multiplying the polynomials whose coefficients are the elements of u and v.

15. Impz:**Syntax:** [h,t] = impz(b,a,n)

Description:[h,t] = impz(b,a,n) computes the impulse response of the filter with numerator coefficients b and denominator coefficients a and computes n samples of the impulse response when n is an integer (t = [0:n-1]'). If n is a vector of integers, impz

computes the impulse response at those integer locations, starting the response computation from 0 (and t = n or t = [0 n]).If, instead of n, you include the empty vector for the second argument, the number of samples is computed automatically by default.

16. Disp:**Syntax:** disp(X)

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

17. xlabel:**Syntax:** xlabel('string')

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

18. **ylabel: Syntax:**ylabel('string')

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

19. **Title: Syntax:**title('string')

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

20. **grid on: Syntax:**grid on

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

PRACTICE PROGRAM WITH RESULT

GENERATION OF ELEMENTARY SIGNALS

Impulse Signal

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \qquad \delta(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases}$$

Unit Step Signal

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n \leq 0 \end{cases} \qquad u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t \leq 0 \end{cases}$$

Ramp Signal

$$x(n) = \begin{cases} n & n \geq 0 \\ 0 & n \leq 0 \end{cases} \qquad u(t) = \begin{cases} t & t \geq 0 \\ 0 & t \leq 0 \end{cases}$$

Exponential Signal

$$x(n) = e^{an}$$

$$x(t) = e^{at}$$

Sine wave

x(n) = SinΩn

x(n) = Sinωt

Cosine wave

x(n) = CosΩn

x(n) = Cosωt

PROGRAM

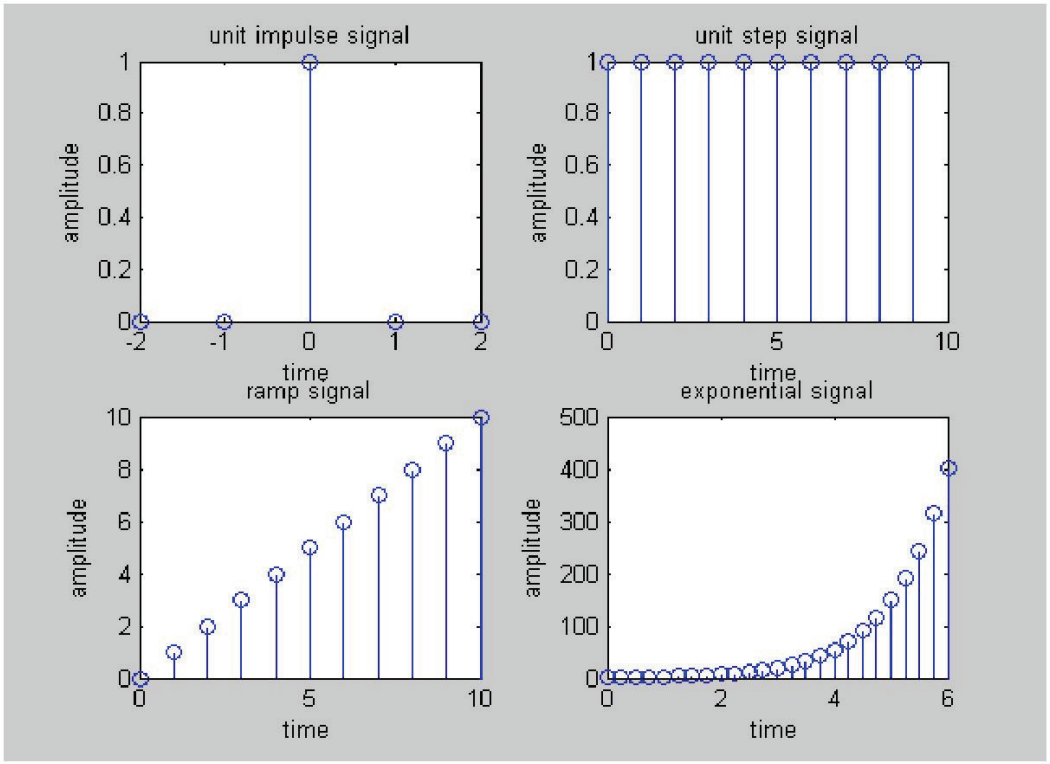
```
clc;                                    % clear screen
clear all;                              % clear workspace
close all;                              % close all figure windows
% to generate unit impulse signal
t = -2:1:2;                             % Define time vector
y = [zeros(1,2), ones(1), zeros(1,2)]; % define amplitude values
subplot(2,2,1);
```

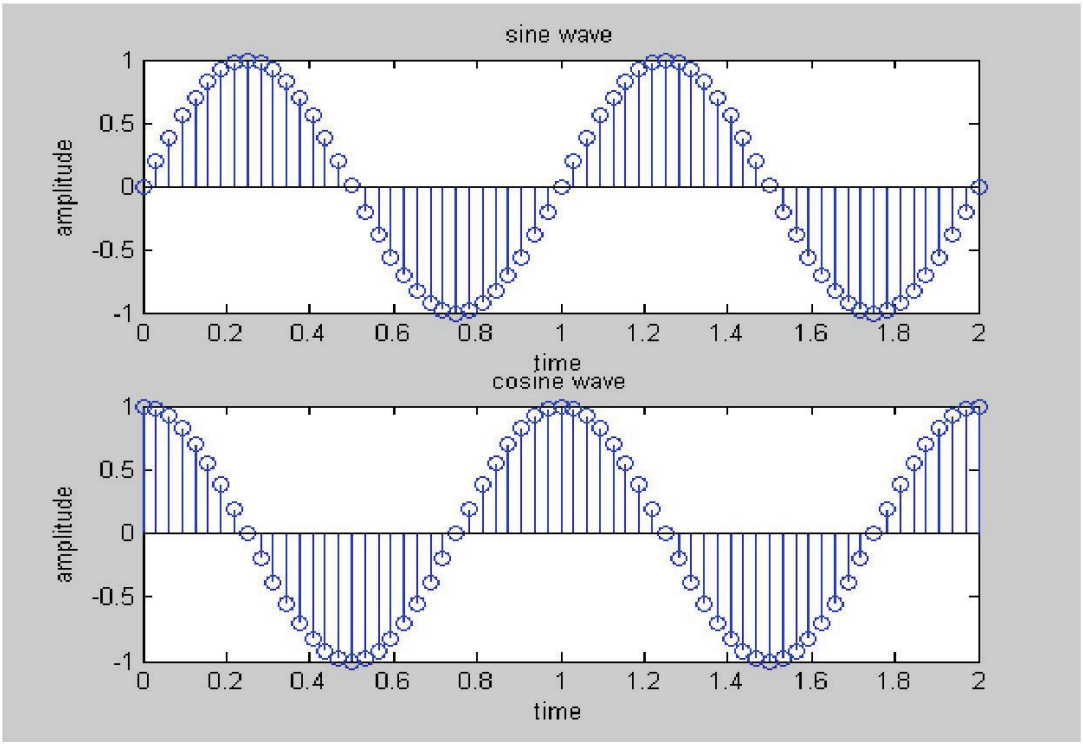
```
stem(t,y);                          % plot the discrete time impulse signal
xlabel('time');                     % label x axis
ylabel('amplitude');                % label y axis
title('Unit Impulse signal');       % graph title
% to generate unit step
N = input('enter the value of N = '); % define the length of unit step signal
t = 0 : N-1;                        % Define time vector
y1=ones(1,N);                       % Define amplitude values
subplot(2,2,2);
stem(t,y1);                         %plot the discrete time unit step signal
xlabel('time');                     % label x axis
ylabel('amplitude');                % label y axis
title('Unit Step signal');          % graph title
% to generate ramp signal
N1 = input('enter the value of N1 = '); % Define the length of unit ramp signal
t = 0 : N1;                         % Define time vector
subplot(2,2,3);
stem(t,t);                          %plot the discrete time impulse signal
xlabel('time');                     % label x axis
ylabel('amplitude');                % label y axis
title('Ramp signal');               % graph title
% to generate exponential
N2 = input('enter the length N2 = '); % define the length of unit step signal
t = 0:1/4:N2;                       % Define time vector
a = input('enter the value of a = '); % if a is +ve –Rising exponential, a is –ve decaying
exponential
y2 = exp(a*t);                      % Define amplitude values
subplot(2,2,4);
stem(t,y2);                         %plot the discrete time impulse signal
xlabel('time');                     % label x axis
ylabel('amplitude');                % label y axis
title('Exponential signal');        % graph title
```

```
% to generate sine wave
t = 0:1/32:2;                    % Define time vector
x = sin(2*pi*t);                 % define amplitude values
figure(2);                       % Open new figure window
subplot(2,1,1);
stem(t,x);                       %plot the discrete time impulse signal
xlabel('time');                  % label x axis
ylabel('amplitude');             % label y axis
title('Sine wave');               % graph title
% to generate Cosine wave
t = 0:1/32:2;                    % Define time vector
x = cos(2*pi*t);                 % Define amplitude values
subplot(2,1,2);
stem(t,x);                       %plot the discrete time impulse signal
xlabel('time');                  % label x axis
ylabel('amplitude');             % label y axis
title('Cosine wave');            % graph title
```

OUTPUT:

```
enter the value of N = 10
enter the value of N1 = 10
enter the length N2 = 6
enter the value of a = 3
```





OUTCOME: Generation of elementary signals are verified using MATLAB

VIVA QUESTIONS WITH ANSWERS

1. Define Signal.

A signal is defined as any physical quantity that varies with time, space or any other independent variable or variables

2. Define system.

A system is a physical device (i.e., hardware) or algorithm (i.e., software) that performs an operation on the system.

3. What is static and dynamic system? Give examples.

A discrete time system is called static (or memory less), if its output at any instant depends on the input sample at the same time (but does not depend on past or future samples). If the response depends on past depends on past or future samples, then the system is called dynamic system.

4. What is multi-dimensional signal? Give example.

A signal which is a function of two or more independent variables is called multi-dimensional signal and can be denoted as I(x, y). The intensity or brightness of black and white photograph at each point is a function of two independent special coordinates x and y. hence it is a two dimensional signal. The intensity of black and white motion picture is a function of x and ycoordinates, and time. Hence it is a three dimensional signal.

5. What is analog signal?

The analog signal is a continuous function of an independent variable such as time, space etc. the analog signal is defines for every instant of the independent variable and so the magnitude of analog signal is continuous in the specified range. Here both the magnitude of the signal and the independent variable are continuous.

6. What is discrete signal?

The discrete signal is a function of a discrete independent variable which is an integer. The independent variable is divided into uniform intervals and each interval is represented by an integer. The discrete signal is defined for every integer value of the independent variable. The magnitude of discrete signal can take any discrete value in the specified range. Here both the value of the signal and the independent variable are discrete.

7. What is digital signal?

The digital signal is same as discrete signal except that the magnitude of the signal is quantized. The magnitude of the signal can take one of the values in a set of quantized values. Here quantization is necessary to represent the signal in binary codes.

8. How will you classify the discrete time signals?

The discrete time signals are classified depending on their characteristics. Some ways of classifying discrete time signals are (a) Energy signals and Power signals, (b) Periodic and Aperiodic signals, (c) Symmetric and Anti symmetric signals.

PART-A

EXPERIMENT NO-1:-SAMPLING THEOREM

AIM: Verification of sampling theorem

OBJECTIVE: To verify sampling theorem forNyquist rate, Under sampling and Over sampling condition in both time and frequency domain using MATLAB.

THEORY:

Sampling: Is the process of converting a continuous time signal into a discrete time signal. It is the first step in conversion from analog signal to digital signal.

Sampling theorem: Sampling theorem states that "Exact reconstruction of a continuous time base-band signal from its samples is possible, if the signal is band-limited and the sampling frequency is greater than twice the signal bandwidth".i.e. fs> 2W, where W is the signal bandwidth.

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

ALGORITHM:

1. Select the frequency of analog signal f Hz
2. To generate sine wave of f Hz define a closely spaced time vector
3. Generate the sinusoid and plot the signal
4. Select the sampling frequency fs= 10f samples/sec. Generate a suitable time scale for this sampling signal
5. Sample the analog signal at the instant specified by n.
6. Modify the time vector n used for discrete simulation
7. Reconstruct the analog signal from its discrete samples
8. Compare the analog and reconstructed signal

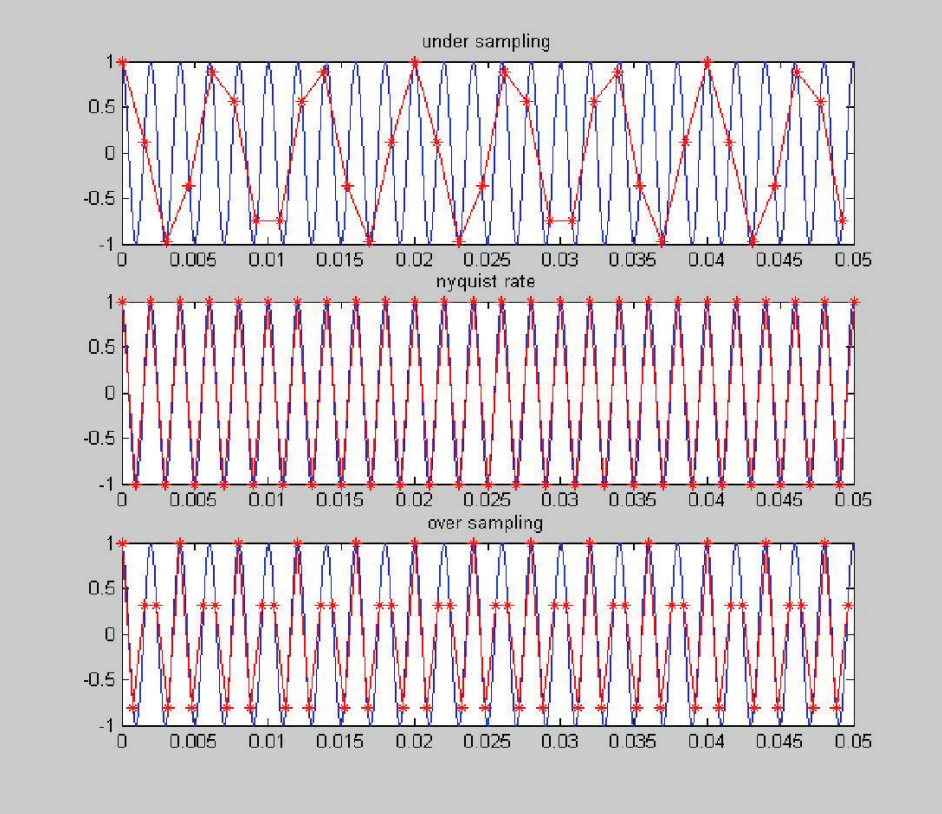
9. Repeat the values experiment for different values of f and verify reconstructed and analog signal

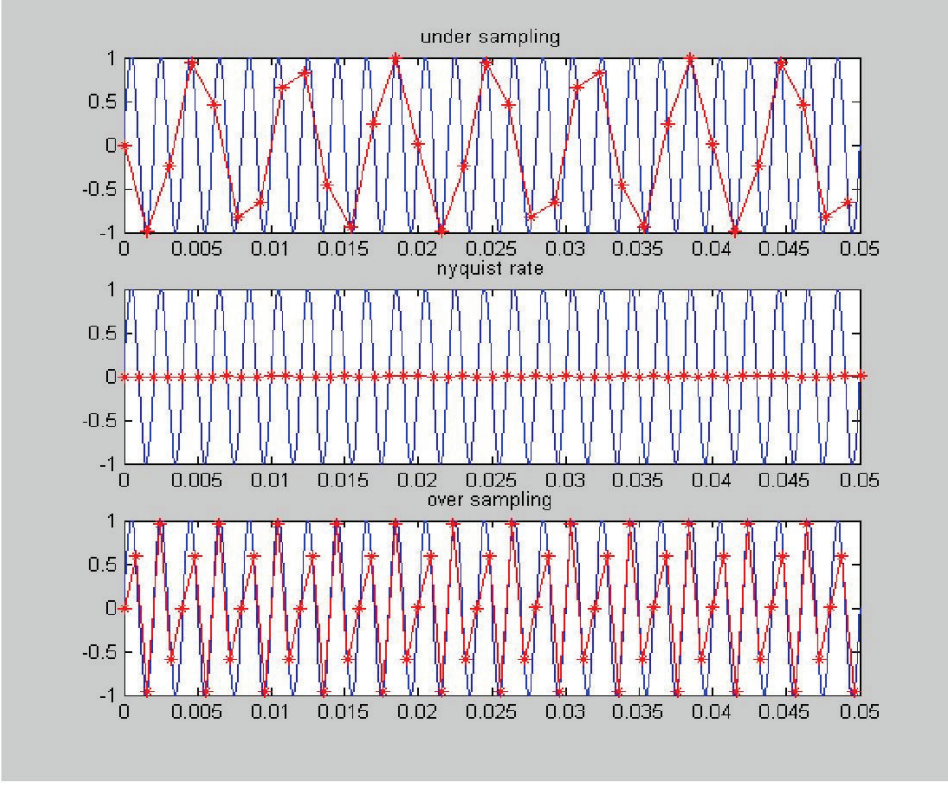
PROGRAM: SAMPLING THEOREM IN TIME DOMAIN

```
clc;                                    % clear screen
clear all;                              % clear work space
close all;                              % close all figure windows
tfinal = 0.05;                          % define final value of time vector
t= 0:0.00005: tfinal;                   % define time vector for analog signal
fd= input('enter the analog frequency'); % enter the analog frequency
xt = sin(2*pi*fd*t);                    % define analog signal
fs1 = 1.3*fd;% simulate condition for under sampling
n1= 0: 1/fs1: tfinal;                   % define time vector for discrete signal
xn = sin(2*pi*n1*fd);% to generate under sampled
signal subplot(3,1,1);
plot(t,xt,'b',n1,xn,'r*-');% plot the analog and sampled signal
title('under sampling');
fs2= 2*fd;% simulate condition for nyquist rate
n2= 0:1/fs2:tfinal; % define time vector for discrete signal
xn = sin(2*pi*n2*fd);% to generate under sampled signal
subplot(3,1,2);
plot(t,xt,'b',n2,xn,'r*-');% plot the analog and sampled signal
title('nyquist rate');
fs3 = 2.5*fd;% simulate condition for over sampling
n3= 0:1/fs3:tfinal;% define time vector for disrete
signal xn = sin(2*pi*n3*fd);%generate over sampling
signal subplot(3,1,3);
plot(t,xt,'b',n3,xn,'r*-');% plot the analog and sampled signal
title('over sampling');
```

OUTPUT:

enter the analog wave frequency = 500


under sampling
nyquist rate
over sampling
1
0.5
0
-0.5
-1
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05


under sampling
nyquist rate
over sampling
1
0.5
0
-0.5
-1
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05

PROGRAM: SAMPLING THEOREM IN FREQUENCY DOMAIN

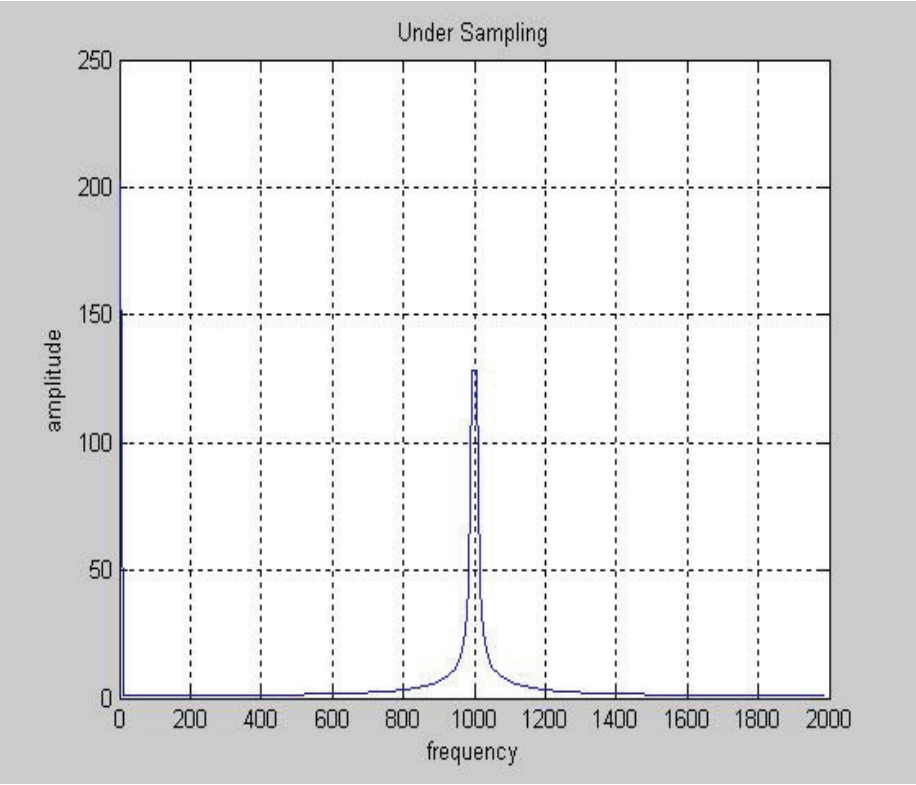
```
clc;                                                    % clear screen
close all;                                              % close all figure windows
clear all;                                              % clear work space
f1 = input('Enter the first sine wave frequency = '); % Enter the first sinusoidal frequency f2
= input('Enter the second sine wave frequency = '); % Enter the second sinusoidal freq
fn = 2*max(f1,f2);                                      % simulate condition for nyquist rate
fs = fn/2;% define sampling frequency for under sampling as half nyquist rate
t = [0:1/fs:0.1];                                       % define time vector
x = cos(2*pi*f1*t)+cos(2*pi*f2*t);                      % Generate the discrète time signal
xk = fft(x);                                % take Fourier transform of discrète time signal
f = [0:length(xk)-1]*fs/length(xk);                     % define the frequency axis
figure(1);                                              % open figure window
plot(f,abs(xk));          % plot the absolute value of fourier transform of discrete signal
xlabel('frequency');                                    % label x axis
ylabel('amplitude');                                    % label y axis
title('Under Sampling');                                % graph title
grid;
fs = fn;                                                % nyquist rate
t = [0:1/fs:0.1];                                       % define time vector
x = cos(2*pi*f1*t)+cos(2*pi*f2*t);                      % Generate the discrète time signal
xk = fft(x);                                % take Fourier transform of discrète time signal
f = [0:length(xk)-1]*fs/length(xk);                     % define the frequency axis
figure(2);                                              % open figure window
plot(f,abs(xk));          % plot the absolute value of Fourier transform of discrete signal
xlabel('frequency');                                    % label x axis
ylabel('amplitude');                                    % label y axis
title('Nyquist Rate Sampling');                         % graph title
grid;

fs = 2*fn;                                              % oversampling
t = [0:1/fs:0.1];                                       % définе time vector
```

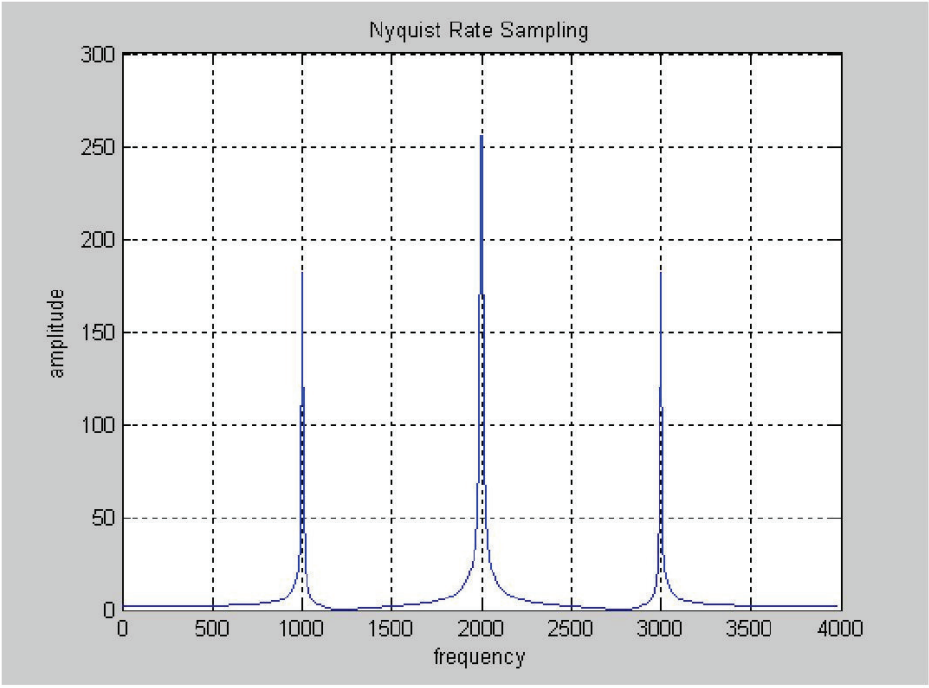
```
x = cos(2*pi*f1*t)+cos(2*pi*f2*t);          %Generate the discrete time signal
xk = fft(x);                                % take Fourier transform of discrete
time                                        %signal
f = [0:length(xk)-1]*fs/length(xk);         % define the frequency axis
figure(3);                                  % open figure window
plot(f,abs(xk));        % plot the absolute value of Fourier transform of discrete signal
xlabel('freq');                             % label x axis
ylabel('amplitude');                        % label y axis
title('Over Sampling');                     % graph title
grid;
```

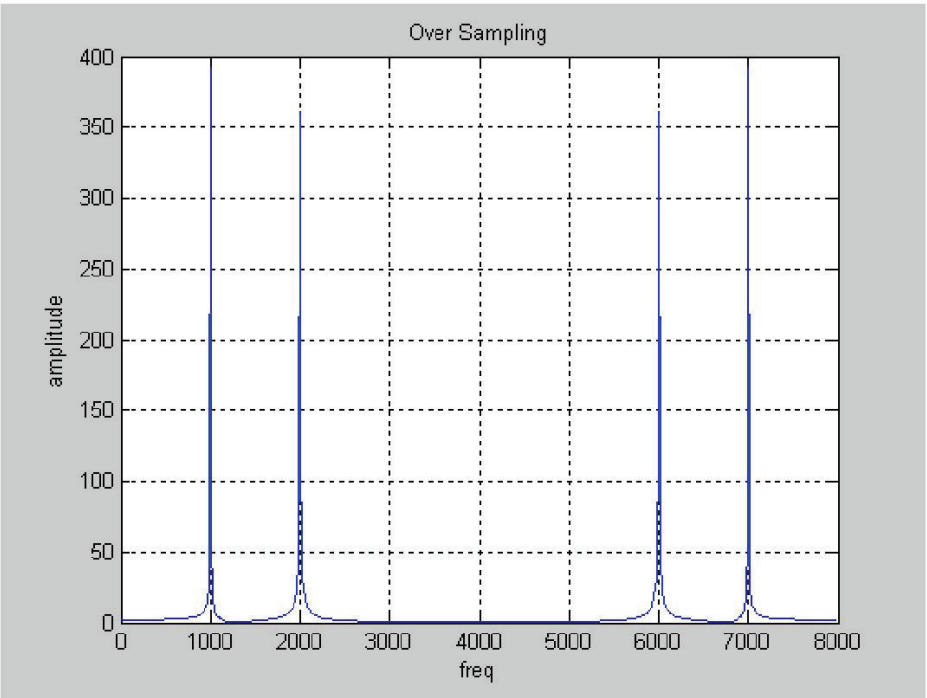
OUTPUT:

Enter the first sine wave frequency = 1000

Enter the second sine wave frequency = 2000







OUTCOME: Sampling theorem is verified for under, Nyquist rate and over sampling using MATLAB

VIVA QUESTIONS WITH ANSWERS

1. What is DSP?

Digital Signal Processing refers to processing of signals by digital systems. DSP is a technique that converts signals from real world sources (usually in analog form) into digital data that can then be analyzed. Analysis is performed in digital form because once a signal has been reduced to numbers; its components can be isolated, analyzed and rearranged more easily than in analog form. Eventually, when the DSP has finished its work, the digital data can be turned back into an analog signal, with improved quality.

2. What are the advantages of DSP?

The advantages of DSP are (i) programs can be modified easily for better performance; (ii) Better accuracy can be achieved by using adaptive algorithms, (iii) digital signals can be easily stored and transported, (iv)digital systems are cheaper than analog equivalent.

3. What are the applications of DSP?

Speech coding & decoding Audio mixing & editing, speech encryption & decryption Image compression & decompression, Speech recognition, Image compression and processing.

4. Why DFT? What is need of Sampling DTFT?

In digital domain for processing, input has to be discrete. For frequency domain analysis, DT signal is converted to frequency domain. Frequency domain representation of DT signal is continuous, NOT discrete. For processing in digital domain we need to take sampled values. The frequency samples thus obtained are called DFT coefficient. That is what DFT is.

5. What is DT system?

A DT system is a device or algorithm that operates on a DT signal according to some well-defined rule, to produce another DT signal. In general a DT system can be thought as a set of operations performed on the input signal x[n] to produce the output signal y[n].

6. How Discrete Time signal is obtained?

DT signal is obtained by sampling CT signal at regular intervals of time. In practical application sampling is implemented using S/H circuit.

7. When a discrete time signal is called periodic?

When a discrete time signal x(n) satisfies the condition x(n) = x(n+N), then it is called periodic, with periodicity of N samples.

8. List the various methods of classifying discrete time systems.

The discrete time systems are classified based in characteristics. Some of the classifications of discrete time systems are

(i) Static and Dynamic Systems,
(ii) Time invariant and time variant systems,
(iii) Linear and Nonlinear systems,
(iv) Causal and Non-causal systems,
(v) Stable and Unstable Systems,
(vi) FIR and IIR systems,
(vii) Recursive and Non-recursive Systems.

9. What is antialiasing filter? Can it be Digital filter? Justify.

When processing the analog signal using DSP System, it is sampled at some rate depending upon the bandwidth. The rate of sampling is decided by the Nyquist criterion. However, signals that are found in physical systems will never be strictly band limited. To eliminate signal content beyond the desired bandwidth, anti-aliasing filter is used. The filter cannot be a digital filter. This is because anti-alias filtering is required to be performed in the analog domain prior to applying the signal to A/D converter where aliasing would take place.

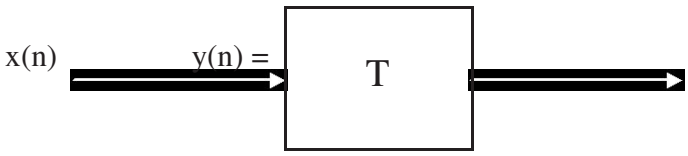
EXPERIMENT NO-2: IMPULSE RESPONSE

AIM: Impulse response of a given system

OBJECTIVE: To learn how to use MATLAB to solve ordinary differential equations (ODEs) and to find the responses of LTI systems for impulse excitation using MATLAB functions DECONV, IMPZ and FILTER.

THEORY:

A discrete time system performs an operation on an input signal based on predefined criteria to produce a modified output signal. The input signal x(n) is the system excitation, and y(n) is the system response. The transform operation is shown as,



If the input to the system is unit impulse i.e. x(n) = δ(n) then the output of the system is known as impulse response denoted by h(n) where,

$$h(n) = T[\delta(n)] \quad (1)$$

We know that any arbitrary sequence x(n) can be represented as a weighted sum of discrete impulses. Now the system response is given by,

$$y(n)=T(x(n)) = \sum_{n=-\infty}^{\infty} x(k)\delta(n-k) \quad (2)$$

For linear system (2) reduces to

$$y(n)= \sum_{n=-\infty}^{\infty} x(k)T[\delta(n-k)] \quad (3)$$

The response to the shifted impulse sequence can be denoted by h(n, k) is denoted by,

$$h(n, k) = T[\delta(n-k)] \quad (4)$$

For a time-invariant system

$$h(n, k) = h(n-k) \quad (5)$$

Using (5) in (4) we obtain,

$$T[\delta(n-k)] = h(n-k) \quad (6)$$

Using (6) in (3) we get,

$$y(n) = \sum_{k=0}^{\infty} x(k)\, h(n-k) \quad (7)$$

For a linear time-invariant system if the input sequence is x(n) and impulse response h(n) is given, we can fine output y(n) by using eqn (7), which is known as convolution sum and can be represented by y(n) = x(n) * h(n)

ALGORITHM:

1. For the given difference equation, rewrite the equation so that y[n] and its delayed samples are on the LHS and x[n] and its delayed samples are on the RHS
2. Create a matrix A for the coefficients of y[n] and its delayed versions
3. Create a matrix B for the coefficients of x[n] and its delayed versions
4. Generate an impulse excitation signal delta[n].
5. Find the response h[n] of the filter defined by A and B coefficients to the input impulse excitation using deconv, impz and filter commands
6. Display and plot the impulse response h[n]

CALCULATION:

$$y(n) - \frac{3}{4}y(n-1) + \frac{1}{8}y(n-2) = x(n) - \frac{3}{4}x(n-1)$$

Taking z-transformers

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

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

Multiply both numerator and denomination by z^2.

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

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

$$H(z) = \frac{z - \frac{3}{4}}{(z - 0.5)(z - 0.25)}$$

$$\frac{z-\frac{3}{4}}{(z-0.5)(z-0.25)} = \frac{A}{z-0.5}+\frac{B}{z-0.25}$$

$$z-\frac{3}{4} = A(z-0.25)+B(z-0.5)$$

When z=0.25, B=2

Z=0.5, A=-1

$$\frac{H(z)}{z} = \frac{-1}{z-0.5}+\frac{2}{z-0.25}$$

$$H(z) = \frac{-1z}{z-0.5}+\frac{2z}{z-0.25}$$

$$h(n) = -0.5^n.u(n)+2.(-0.25)^n.u(n)$$

put n=0,1,2,3,4 [length =5]

$$h(0) = -0.5^0.u(0)+2.(-0.25)^0.u(0) = -1+2 = 1$$

$$h(1) = -0.5^1.u(1)+2.(-0.25)^1.u(1) = 0$$

$$h(2) = -0.5^2.u(2)+2.(-0.25)^2.u(2) = -0.125$$

$$h(3) = -0.0937, \quad h(4) = -0.0546$$

PROGRAM:IMPLULSE RESPONSE USING DECONV FUNCTION

```
clc;                                % clear screen
close all                           % close all figure windows
clear all;                          % clear work space
y = input('Output sequence y(n) of the system = '); % enter the output sequence
x = input('Input sequence x(n) of the system = ');% enter the input sequence
[h,r] = deconv(y,x); % deconvolute output and input to get the impulse response
disp('Impulse response of the system is = ');
disp(h);% display result
N= length(h);% find the length of h n =
0:1:N-1;% define time axis
stem(n,h);% plot the impulse response
```

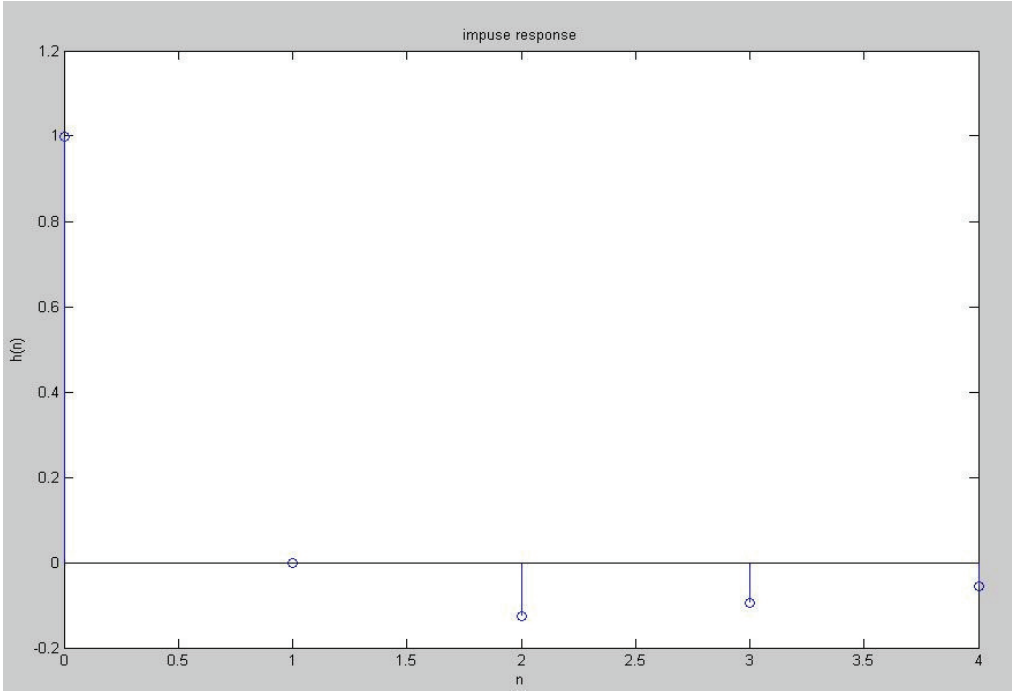
```
xlabel('n');% label x axis
ylabel('h(n)');% label y axis

title('Impulse Response of the system'); % graph title
% verification

yv = conv(x,h)+ r;% verification of result using convolution of input and impulse to get
output

disp('Verified output sequence y(n) is');
disp(yv);% display the value of output
```

OUTPUT:

Output sequence y(n) of the system = [1 -3/4]

Input sequence x (n) of the system = [1 -3/4 1/8]

Impulse response of the system = [1 0 –0.125 -0.0937 -0.0546]



PROGRAM: IMPLULSE RESPONSE USING IMPZ FUNCTION

```
clc;                                    % clear screen
close all;                              % close all figure windows
clear all;                              % clear workspace
```

```
y = input('Output sequence y(n) of the system = ');   % enter the coefficients of y terms
x = input('Input sequence x(n) of the system = ');    % enter the coefficient of x terms
N=input('Enter the length of impulse response =       % define the length of the response
'); h = impz(y,x,N);                                  % calculate the impulse response
disp('Impulse Response of system h(n)');
disp(h);                                              % display the values of impulse response
                                                      %graphical plot
n = 0:1:N-1;                                          % define x axis
stem(n,h);                                            % plot the impulse response
xlabel('n');                                          % label x axis
ylabel('h(n)');                                       % label y axis
title('Impulse Response');                            % graph title
```

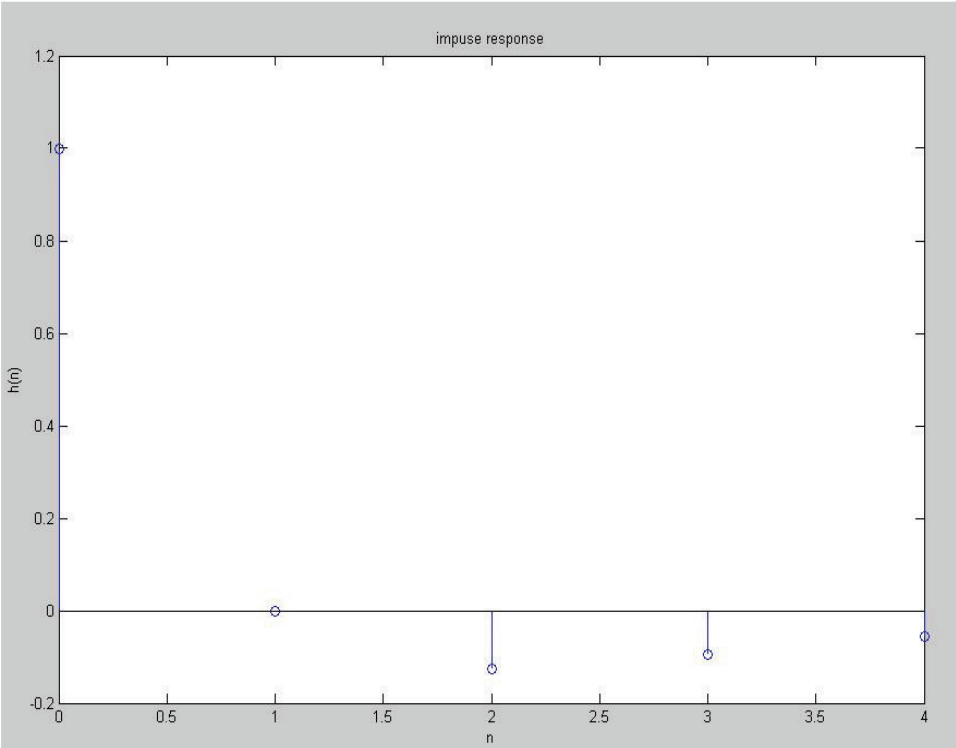
OUTPUT:

Output sequence y(n) of the system = [1 -1/3]

Input sequence x(n) of the system = [1 -3/4 -1/8]

Enter the length of impulse response =5

Impulse Response of system h(n) =1.0000 0 -0.125 -0.0937 -0.0546



PROGRAM: IMPLULSE RESPONSE USING FILTER FUNCTION

```
clc;                                          % clear screen
close all;                                    % close all the figure windows
clear all;                                    % clear the work space
y = input('Enter the co-efficient of y = ');  % define the coefficient of y terms
x = input('Enter the co-efficient of x = ');  % define the coefficient of x terms
N = input('length of impulse sequence = ');   % define the length of the response
xi = [1,zeros(1,N-1)];                        % define the input signal
h = filter(x,y,xi);                           % calculate the impulse response
disp( 'Impulse Response of the system is = ');
disp(h);                                      % display the response
                                              % graphical display
n = 0:1:N-1;                                  % define the time axis
stem(n,h);                                    % plot the impulse response
xlabel('n');                                  % label x axis
ylabel('h(n)');                               % label y axis
```

title('Impulse Response of the system'); % graph title

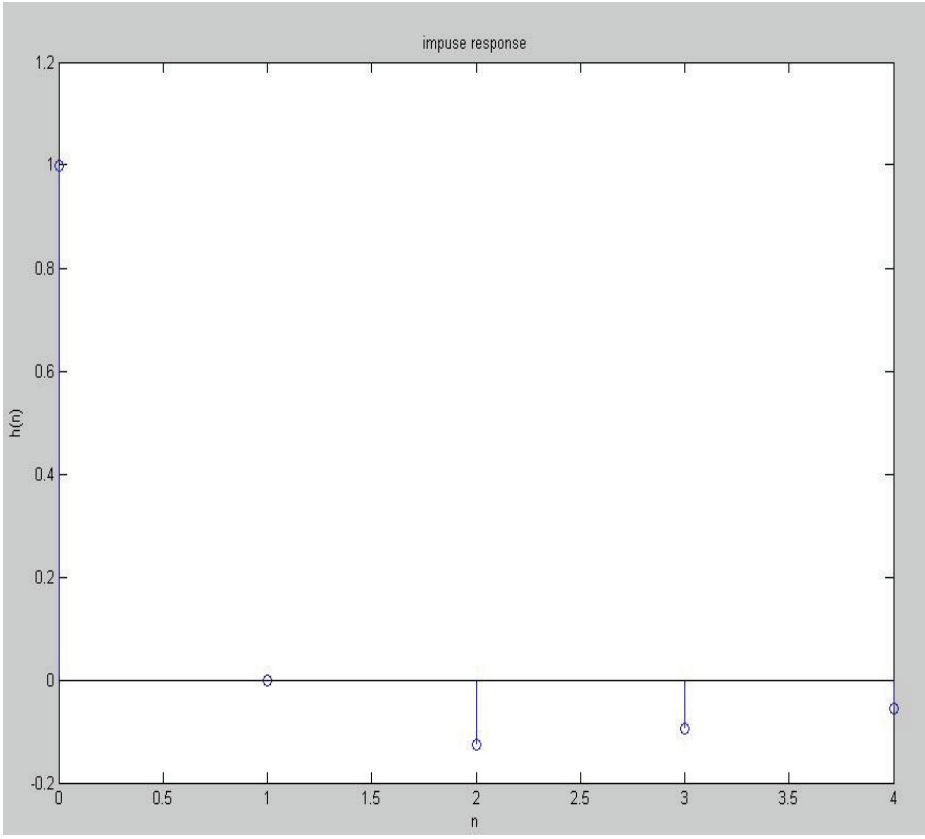
OUTPUT:

Output sequence y(n) of the system = [1 -1/3]

Input sequence x(n) of the system = [1 -3/4 -1/8]

Enter the length of impulse response = 5

Impulse Response of the system is = 1.0000 0 -0.125 -0.0937 -0.0546



OUTCOME: Impulse response and the output response of the system is found and the results are verified in MATLAB.

VIVA QUESTIONS WITH ANSWERS

1. What are the various methods available to determine the response of LTI systems?

The response of LTI system can be determined by the following four methods (i) Direct solution of difference equation governing the LTI system, (ii) Discrete convolution of input and impulse response of the system, (iii)Using Convolution Property of Z transform, (iv) Using convolution Property of FT or DFT

2. What is impulse response and what is its significance?

The response or output of an LTI system for unit impulse input $\delta(n)$ is called impulse response. It is denoted by h(n). The response y(n) of an LTI system for any arbitrary input x(n) is given by convolution of impulse response and input i.e., $y(n) = x(n)* h(n)$

3. Define the transfer function of an LTI system

The transfer function of an LTI system is defined as the ratio of Z transform of output to Z transform of input.

4. What is BIBO stability? What is the condition to be satisfied foe stability?

A system is said to be BIBO stable if and only if every bounded input produces a bounded output. The condition to be satisfied for the stability of an LTI system is that the impulse response of the system should be absolutely summable.

5. What do you mean by real time signal? Give example.

Signal is processed with the same speed it is captured. Signal is captured, sampled and processed with the same speed. Signal is not stored before processing. Entire input signal never available before processing. Processed signal can be stored. For example, in digital telephone system, Signal is captured, Sampled, Processed, and Transmitted and made it available to the end user. Real Time Processing is Online Processing.

EXPERIMENT NO-3:LINEAR CONVOLUTION

AIM: To implement linear convolution of two given sequences

OBJECTIVE:

1. To find linear convolution of right sided sequence using inbuilt MATLAB function "CONV" and its theoretical method to verify the result
2. To find linear convolution of both sided sequence using inbuilt MATLAB function "CONV" and its theoretical method to verify the result
3. To find linear convolution of a given sequences using circular convolution.

THEORY: Convolution is an integral concatenation of two signals. It has many applications in numerous areas of signal processing. The most popular application is the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system. Note that convolving two signals is equivalent to multiplying the Fourier transform of the two signals. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal. In linear convolution length of output sequence is,length(y(n)) = length(x(n)) + length(h(n)) – 1

Mathematical Formula:

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

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

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

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

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

ALGORITHM:

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

CALCULATION:

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

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

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

$$z = x_1 * x_2$$

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

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

On simplification we get,

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

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

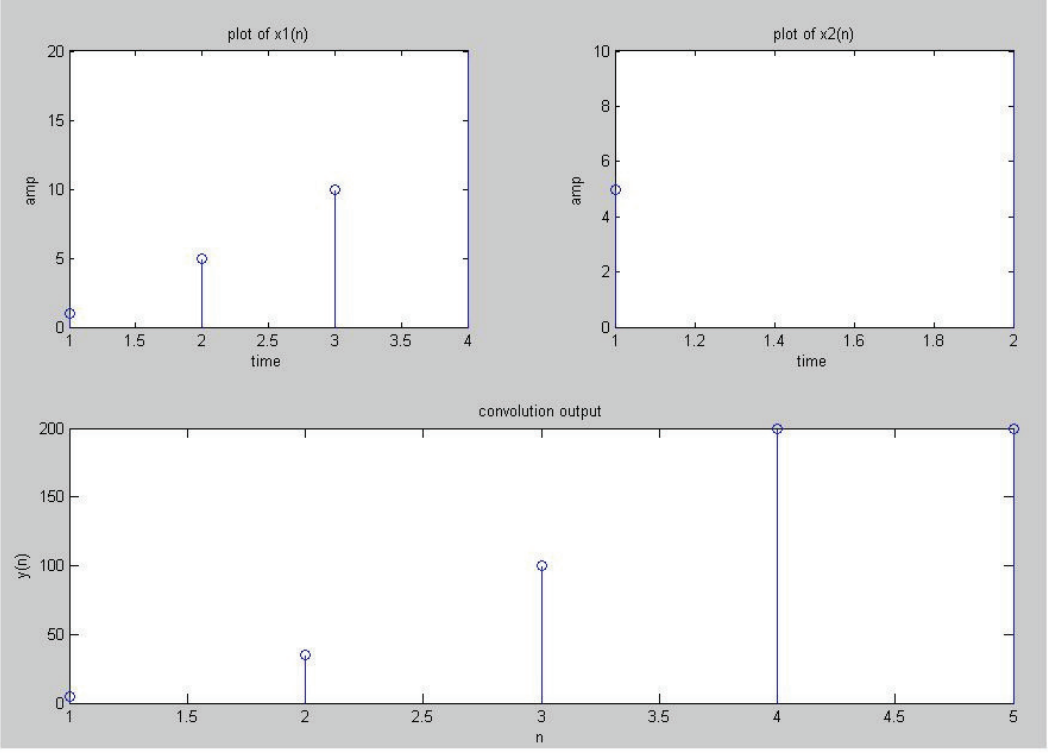
PROGRAM:LINEAR CONVOLUTION OF RIGHT SIDED SEQUENCES

```
clc;                                                    % clear screen
clear all;                                              % clear workspace
close all;                                              % close all figure windows
x1 = input('enter the first sequence x1(n) = ');        % define first sequence
x2 = input('enter the second sequence x2(n) = ');       % define second sequence
y = conv(x1,x2);                                        % convolute first and second sequences
disp('Linear convolution of x1 and x2 is = ');
disp(y);                                                % display the output
%graphical display
subplot(2,2,1); % divide the display screen in four sections and choose the first one to display
stem(x1);                                               % plot the first sequence
xlabel('n');                                            % label x axis
ylabel('x1(n)');                                        % label y axis
title('plot of x1(n)');                                 % graph title
```

```
subplot(2,2,2);% divide the display screen in four sections and choose the first second to
display
stem(x2);                                   % plot the second sequence
xlabel('n');                                % label x axis
ylabel('x2(n)');                            % label y axis
title('plot of x2');                        % graph title
subplot(2,1,2);% divide the display screen in four sections and choose the first second to
display
stem(y);                                    % plot the output sequence
xlabel('n');                                % label x axis
ylabel('y(n)');                             % label y axis
title('convolution output');                % graph title
```

OUTPUT:

enter the first sequence x1(n) = [1 5 10 20]
enter the second sequence x2(n) = [5 10]
Linear convolution of x1 and x2 is = 5 35 100 200 200



PROGRAM: LINEAR CONVOLUTION OF BOTH SIDED SEQUENCES

Calculation:

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

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

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

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

Z= X1 * X2

On Simplification, we get

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

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

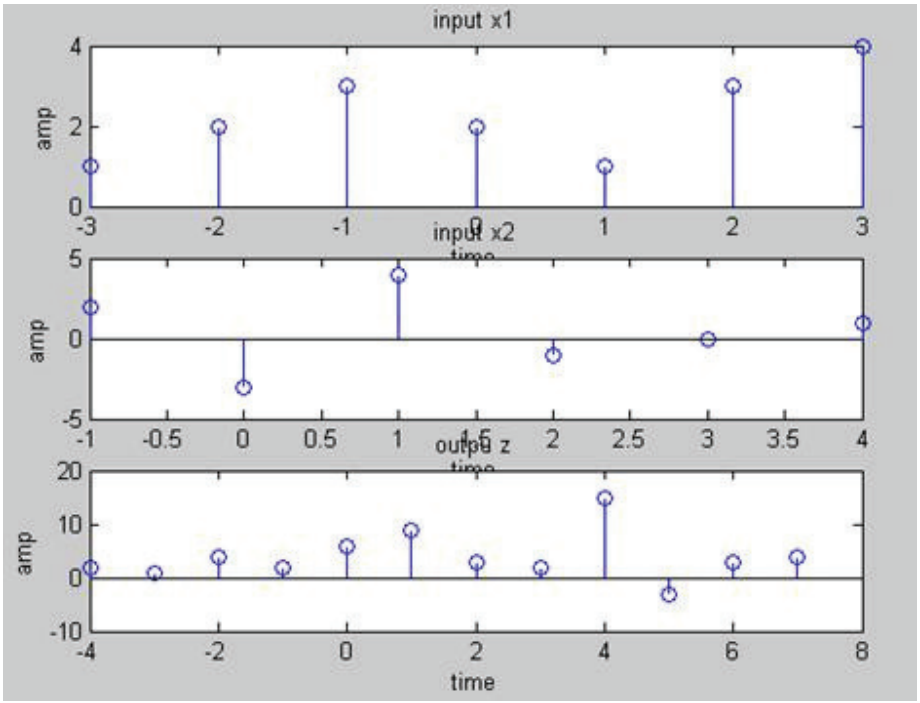
PROGRAM

```
clc;                                                % clear screen
clear all;                                          % clear work space
close all;                                          % close all figure windows
x1 = input('enter the first sequence x1(n) = ');    % define first sequence
x2 = input('enter the second sequence x2(n) = ');   % define second sequence
n1 = -2:2;                                          % time axis for first sequence
n2 = -1:3;                                          % time axis for second sequence
ybegin = n1(1)+n2(1);                               % calculate the first point of x axis of output
yend = n1(length(x1)) + n2(length(x2));             % calculate the end point of x axis of output
ny = [ybegin : yend];                               % define x axis for output
y = conv(x1,x2);                                    % convolute the first and second sequence
disp('Linear convolution of x1 and x2 is = ');
disp(y);                                            % display the output
                                                    %graphical display
subplot(2,2,1);% divide the display screen in four sections and choose the first second to
display
stem(n1,x1);                                        % plot the first sequence
xlabel('n');                                        % label x axis
```

```
ylabel('x1(n)');                                    % label y axis
title('plot of x1');                                % graph title
subplot(2,2,2);% divide the display screen in four sections and choose the second to display
stem(n2,x2);                                        % plot the second sequence
xlabel('n');                                        % label x axis
ylabel('x2(n)');                                    % label y axis
title('plot of x2');                                % graph title
subplot(2,1,2);% divide the display screen in four sections and choose the second to display
stem(ny,y);                                         % plot the second sequence
xlabel('n');                                        % label x axis
ylabel('y(n)');                                     % label y axis
title('convolution output');                        % graph title
```

OUTPUT:

```
enter the first sequence x1(n) = [1 2 3 2 1 3 4]
enter the second sequence x2(n) = [2 -3 4 -1 0 1]
Linear convolution of x1 and x2 is =
    2    1  4  2  6  9  3  2  15  -3  3  4
```



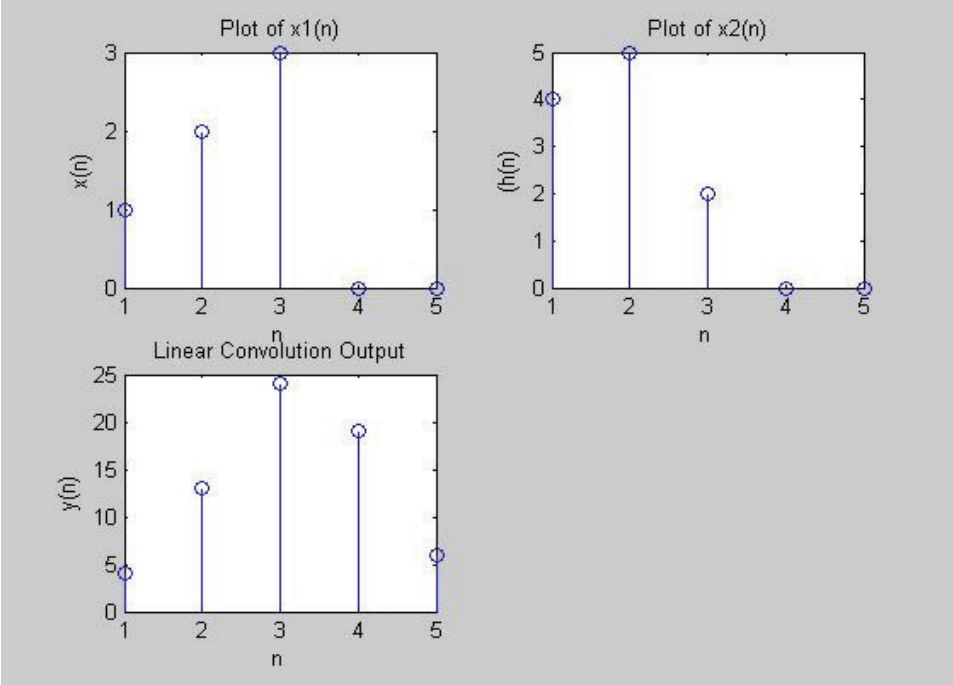
PROGRAM: LINEAR CONVOLUTION USING CIRCULAR CONVOLUTION

```
clc;                                        % clear screen
clear all;                                  % clear work space
close all;                                  % close all figure windows
xn = input('enter the first sequence x(n) = ');
hn = input('enter the second sequence h(n) = ');
l1 = length(xn);                            % length of first sequence
l2 = length(hn);                            % length of second sequence
N = l1+l2-1;                                % length of output
xn = [xn,zeros(1,l2-1)];                    % add zeros to make length of xn and hn equal
hn = [hn,zeros(1,l1-1)];                    % add zeros to make length of xn and hn equal

for n=0:N-1;                                %function for linear convolution in time domain
y(n+1) = 0;
for k=0:N-1;
      i = mod((n-k),N);
if i<0
 i = i+N;
end
y(n+1) = y(n+1)+hn(k+1)*xn(i+1);
end
end
disp('Linear Convolution using Circular Convolution = ');
disp(y);                                    % display output
subplot(2,2,1);                             %Graphical display of first input sequence
stem(xn);
xlabel('n');
ylabel('x(n)');
title('Plot of x1(n)');
subplot(2,2,2);                             %Graphical display of second input sequence
stem(hn);
xlabel('n');
```

```
ylabel('(h(n)');
title('Plot of x2(n)');
subplot(2,2,3); %Graphical display of output sequence stem(y);

xlabel('n');
ylabel('y(n)');
title('Linear Convolution Output');
```

OUTPUT:

```
enter the first sequence x(n) = [1 2 3]
enter the second sequence h(n) = [4 5 2]
Linear Convolution using Circular Convolution =
    4 13 24 19 6
```



OUTCOME: Linear convolution of the sequences is found and the results are verified in MATLAB.

VIVA QUESTIONS WITH ANSWERS

1. What is the length of linearly convolved signals?

Length of linearly convolved signal is always equal to N = L + M - 1 where L is length of first signal and M is length of second signal.

2. What are linear and non-linear systems?

A system is linear, if the response of the system to a weighted sum of the signals is equal to the corresponding weighted sum of the response of the system to each of the individual input signals i.e., if $H[a_1x_1(n) + a_2x_2(n)] = a_1H[x_1(n)] + a_2H[x_2(n)]$. If the system does not satisfy the above condition then it is non-linear.

3. What is meant by discrete convolution?

The convolution of two discrete time signals is called discrete convolution. The discrete convolution of two discrete time signals $x_1(n)$ and $x_2(n)$ is defined as

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

4. Why linear convolution is important in DSP?

The response or output of LTI discrete time system for any input x(n) is given by linear convolution of the input x(n) and the impulse response h(n) of the system. This means that if the impulse response of a system is known, then the response of the system for any input can be determined by convolution operation.

5. Write properties of linear convolution.

The linear convolution satisfies the following properties;

- Commutative Property: x(n) * h(n) = h(n) * x(n)
- Associative Property: [x(n) * h(n)] * g(n) = x(n) * [h(n) * g(n)]
- Distributive Property: [x(n) + h(n)] * g(n) = [x(n) * h(n)] + [x(n) * g(n)]

EXPERIMENT NO-4:- CIRCULAR CONVOLUTION

AIM: Circular convolution of two given sequences

OBJECTIVE: To implement circular convolution of given sequences in time domain using MATLAB and to verify the result theoretically.

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

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

$$x3(n) = \sum_{m=0}^{N-1} x1(m)x2((n-m))\,N$$

CALCULATION:

Let's take $x_1(n) = \{1, 1, 2, 1\}$ and $x_2(n) = \{1, 2, 3, 4\}$

$x_3(0) = x_1(m)\, x_2(-m)$

$= x_1(0)\, x_2(0) + x_1(1)\, x_2(3) + x_1(2)\, x_2(2) + x_1(3)\, x_2(1)$

$= 1 + 4 + 6 + 2 = 13$

$x_3(1) = x_1(m)\, x_2(1-m)$

$= x_1(0)\, x_2(1) + x_1(1)\, x_2(0) + x_1(2)\, x_2(3) + x_1(3)\, x_2(2)$

$= 2 + 1 + 8 + 3 = 14$

$x_3(2) = x_1(m)\, x_2(2-m)$

$= x_1(0)\, x_2(2) + x_1(1)\, x_2(1) + x_1(2)\, x_2(0) + x_1(3)\, x_2(3)$

$= 3 + 2 + 2 + 4 = 11$

$x_3(3) = x_1(m)\, x_2(3-m)$

$= x_1(0)\, x_2(3) + x_1(1)\, x_2(2) + x_1(2)\, x_2(1) + x_1(3)\, x_2(0)$

$= 4 + 3 + 4 + 1 = 12$

The convoluted signal is,

$x_3(n) = \{13, 14, 11, 12\}$

ALGORITHM:

1. Read the first input sequence, x[n] and plot.
2. Read the second input sequence, h[n] and plot
3. Find the length of x[n] and y[n] , l1 and l2 respectively
4. Check if l1=l2. Proceed only if equal.

5. If l1 not equal to l2, zero padding is done to make l1=l2.
6. Initialize a loop variable for the number of output points.
7. For each output sample access the samples of y[n] in cyclic order.
8. Find the sum of products of x[n] and cyclically folded and shifted h[n] to get circular convoluted output.
9. Display and plot the output.

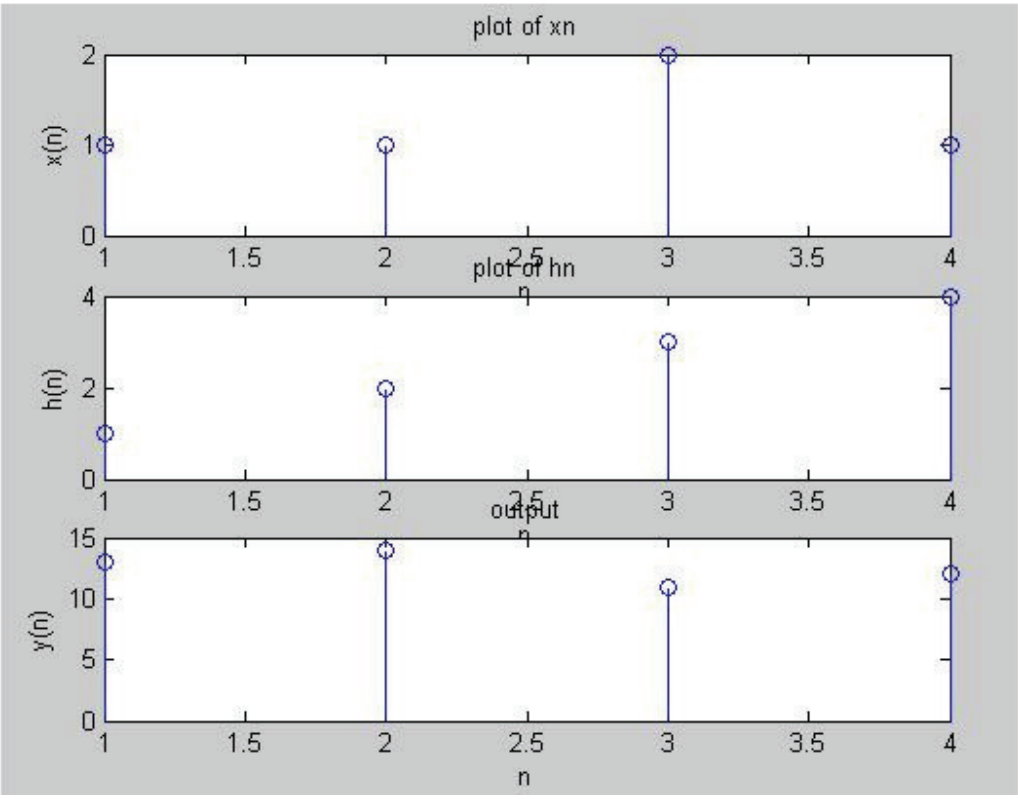
PROGRAM: CIRCULAR CONVOLUTION IN TIME DOMAIN

```
clc;                                                % clear screen
clear all;                                          % clear workspace
close all;                                          % close all figure windows
xn= input('enter the first sequence x(n) = ');      % define first sequence
hn=input('enter the second sequence h(n) = ');      % Define second sequence
l1 = length(xn);                                    % length of first sequence
l2 = length(hn);                                    % length of second sequence
N = max(l1,l2);                                     % Define the length of the output
xn = [xn, zeros(1,N-l1)];                           % zero padding is done to make l1=l2.
hn = [hn, zeros(1,N-l2)];                           % zero padding is done to make l1=l2.
for n=0:N-1;                                        % loop to calculate circular convolution
y(n+1) = 0;
for k=0:N-1
      i = mod((n-k),N);
y(n+1) =y(n+1)+hn(k+1)*xn(i+1);
end ;
end;
disp('Circular convolution in Time Domain = ');
disp(y);                                            % display the output
subplot(2,2,1);                                     % graphical plot the first input sequence
stem(xn);
xlabel('n');
ylabel('x(n)');
title('Plot of x(n)');
```

```
subplot(2,2,2);                          % graphical plot the second input sequence
stem(hn);
xlabel('n');
ylabel('h(n)');
title('Plot of h(n)');
subplot(2,2,3);                          % graphical plot the output sequence
stem(y);
xlabel('n');
ylabel('y(n)');
title('Circular Convolution Output');
```

OUTPUT:

```
enter the first sequence x(n) = [1 1 2 1]
enter the second sequence h(n) = [1 2 3 4]
Circular convolution in Time Domain =
   13    14    11    12
```



OUTCOME: Circular convolution of the sequences are found and the results are verified in MATLAB

VIVA QUESTIONS WITH ANSWERS

1. What is Circular Convolution?

The convolution of two periodic sequences with periodicity N is called circular convolution. If $x_1(n)$ and $x_2(n)$ are two periodic sequences with N samples in a period, then the circular convolution of $x_1(n)$ and $x_2(n)$ is defined as,

$$x3(n) = \sum_{m=0}^{N-1} x1(m)x2((n-m))\,N$$

2. What do you mean by aliasing in circular convolution?

In circular convolution if value of N < L+M-1 then last M-1 values of y[n] wraps around gets added with first M-1 values of y[n]. This is called aliasing.

3. What is the difference between circular convolution and periodic convolution

In periodic convolution input signals are originally periodic with common value of period. In circular convolution, if input signals are not periodic then they are assumed to be periodic with period = N where N = max (L, M) where L is the length of first signal and M is length of second signal.

4. Why circular convolution is important in DSP?

The Discrete Fourier Transform (DFT) is used for the analysis and design of discrete time systems using digital computers. The DFT supports only circular convolution. Hence when DFT techniques are employed, the results of linear convolution are obtained only via circular convolution.

5. How to perform linear convolution using circular convolution?

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

EXPERIMENT NO-5:- AUTOCORRELATION

AIM: Autocorrelation of a given sequence and verification of its properties

OBJECTIVE:

1. To find Autocorrelation of the given sequence using an inbuilt MATLAB function "XCORR" and to verify its properties.
2. To verify the results theoretically.

THEORY: Correlation: Correlation determines the degree of similarity between two signals. If the signals are identical, then the correlation coefficient is 1; if they are totally different, the correlation coefficient is 0, and if they are identical except that the phase is shifted by exactly 180^0 (i.e. mirrored), then the correlation coefficient is -1.

Autocorrelation: The Autocorrelation of a sequence is correlation of a sequence with itself. The autocorrelation of a sequence x(n) is defined by,

$$Rxx(k) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-k)_{\text{k = 0, ±1, ±2, ±3…}}$$

Where k is the shift parameter

Or equivalently

$$Rxx(k) = \sum_{n=-\infty}^{\infty} x(n+k)\, x(n)\text{k = 0, ±1, ±2, ±3 …}$$

Properties of Autocorrelation:

1. Rxx(0) = Energy(x)
2. Autocorrelation function is **symmetric.** i.e. $R_{xx}(m) = R_{xx}(-m)$

ALGORITHM

1. Read the input sequence x[n]
2. Auto correlate the signal using xcorr(x,x)
3. Display the auto correlation result on a suitable axis.
4. Verify the correlation property Rxx(0)= energy(x)
5. Verify the symmetric property

CALCULATION:

X(n) ={ 3, 4, 5, 6}

$$Rxx(K) = \sum_{n=-\infty}^{\infty} x(n)x(n-K)$$

Put K=0 in the above equation, we get

$$Rxx(0) = \sum_{n=-\infty}^{\infty} x(n)x(n)$$

$$Rxx(0) = 9 + 16 + 25 + 36 = 86$$

Put K=1 in the above equation, we get

$$Rxx(1) = \sum_{n=-\infty}^{\infty} x(n)x(n-1)$$

$$Rxx(1) = 0 + 12 + 20 + 30 = 62$$

Put K=2 in the above equation, we get

$$Rxx(2) = \sum_{n=-\infty}^{\infty} x(n)x(n-2)$$

$$Rxx(2) = 0 + 0 + 15 + 24 + 0 = 39$$

Put K=3 in the above equation, we get

$$Rxx(3) = \sum_{n=-\infty}^{\infty} x(n)x(n-3)$$

$$Rxx(3) = 0 + 0 + 0 + 18 + 0 + 0 + 0 = 18$$

Put K=-1 in the above equation, we get

$$Rxx(-1) = \sum_{n=-\infty}^{\infty} x(n)x(n+1)$$

$$Rxx(-1) = 0 + 12 + 20 + 30 = 62$$

Put K=-2 in the above equation, we get

$$Rxx(-2) = \sum_{n=-\infty}^{\infty} x(n)x(n+2)$$

$$Rxx(-2) = 0 + 0 + 15 + 24 = 39$$

Put K=-3 in the above equation, we get

$$Rxx(-3) = \sum_{n=-\infty}^{\infty} x(n)x(n+3)$$

$$Rxx(-3) = 0 + 0 + 0 + 18 + 0 + 0 + 0 = 18$$

Rxx = [18 39 62 86 62 39 18]

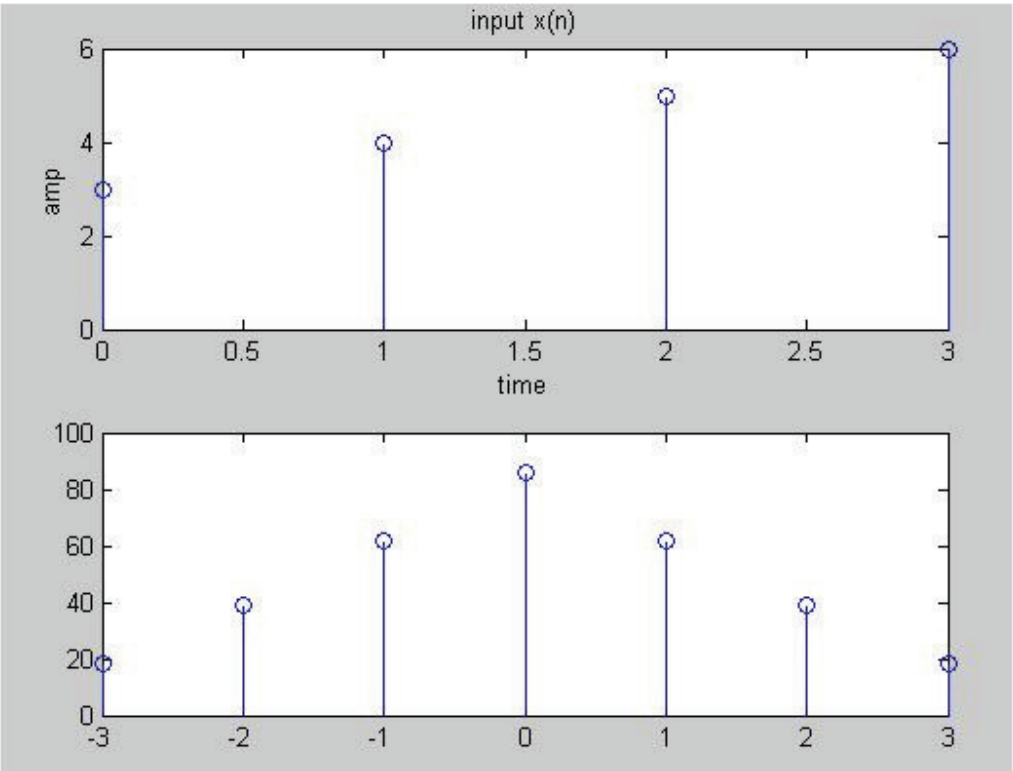
PROGRAM: AUTO CORRELATION

```
clc;                                          % clear screen
clear all;                                    % clear work space
close all;                                    % close all figure windows
                    %computation of autocorrelation of rectangular sequences
n = -5:5;                                     % define the x axis for input sequence
x = ones(1,11);                               % define the amplitude for the input
[Rxx,lag] = xcorr(x, x);                      % calculate the autocorrelation
disp ('Auto correlation sequence r(n) is ');
disp(r);                                      % display the output
subplot(2,1,1);                               % plot the input and output sequences
stem (n, x);
xlabel('n');
ylabel('x(n)');
title('Plot of x(n)');
subplot(2,1,2);
stem(lag,r);
title('Autocorrelation output');
xlabel('n');
ylabel('r(n)');
%Verificaion of the auto correlation properties
% property 1: Rxx(0)gives the energy of the signal
```

```
Energy = sum(x.^2);                                  % calculate the energy of input signal
center_index= ceil(length(Rxx)/2);                   % find the center index
Rxx_0=Rxx(center_index)                               % take the center value of output
if Rxx_0==Energy
disp('Rxx(0) gives energy -- proved');                % display the result
else
disp('Rxx(0) gives energy -- not proved');            % display the result
end
% property 2: Rxx is even
Rxx_Right = Rxx(center_index:1:length(Rxx));         % take the right side values
Rxx_left = Rxx(center_index:-1:1);                    % take the left side values
if Rxx_Right == Rxx_left
disp('Rxx is even');                                  % display the result
else
disp('Rxx is not even');                              % display the result
end
```

OUTPUT:

```
Enter the input signal x[n] [3 4 5 6]
Rxx = [ 18   39   62   86   62   39   18]

Rxx_0 = 30
Rxx(0) gives energy -- proved
Rxx is even
```



OUTCOME: Autocorrelation of the sequence is found and properties of autocorrelation are verified.

VIVA QUESTIONS WITH ANSWER

1. What is correlation?

Correlation gives a measure of similarity between two data sequences. In this process, two signals are compared and the degree to which the two signals are similar is computed.

2. What are the applications of Correlation?

Typical applications of correlation include speech processing, image processing and radar systems. In a radar system, the transmitted signal is correlated with the echo signal to locate the position of the target. Similarly, in speech processing systems, different waveforms are compared for voice recognition.

3. What is correlation property of DFT?

If x[n] X[k] and h[n] H[k] Then DFT {x[n] o h[n]} = X[k] H*[k]

4. What are the applications of FFT?

(i) Linear filtering i.e. to find output of digital filter for any given input sequence (ii) Spectral Analysis i.e. to find magnitude spectrum and phase spectrum (iii) Circular Correlation i.e., to find degree of similarity between two signals.

5. How to find output of the filter using DFT?

Output of the filter is linear convolution of impulse response with the input of the signal. To find output means to find linear convolution by DFT.

6. What is auto-correlation?

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

EXPERIMENT NO-6:- CROSS-CORRELATION

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

OBJECTIVE:

1. To implement Cross correlation of the given sequence using an inbuilt MATLAB function "XCORR" and to verify its properties.
2. To find Cross correlation of the given sequences using convolution.
3. To verify the results theoretically.

THEORY: When two independent signals are compared, the procedure is known as cross correlation. It is given by,

$$Rxy(k) = \sum_{n=-\infty}^{\infty} x(n)\, y(n-k) \quad k = 0, \pm1, \pm2, \pm3 \ldots$$

Where k is the shift parameter

Or equivalently

$$Ryx(k) = \sum_{n=-\infty}^{\infty} y(n+k)\, x(n)$$

Comparing above two equations, we find that, Rxy (k) = Ryx (-k),WhereRyx (-k) is the folded version of Rxy (k) about k = 0.So, we can write Cross correlation of the two sequences is given by,

$$Rxx(k) = \sum_{n=-\infty}^{\infty} x(n)\, y[-(k-n)] \quad k = 0, \pm1, \pm2, \pm3 \ldots$$

$$Rxy(k) = x(k) * y(-k) \qquad (1)$$

Equation (1) shows that cross correlation is the essentially the convolution of two sequences in which one of the sequences has been reversed.

Properties of cross correlation:

1. $R_{xy}(k)$ is always a real valued function which may be a positive or negative.
2. $R_{xy}(-k) = R_{yx}(k)$
3. $|R_{xy}(k)|^2 \leq R_{xx}(0)\, R_{yy}(0)$
4. $|R_{xy}(-k)| \leq [1/2]\, [R_{xy}(-k) + R_{xy}(-k)]$
5. When $R_{xy}(k) = 0$, x(n) and y(n) are said to be uncorrelated or they said to be statistically independent.
6. $R_{xy}(k)$ may not be necessarily have a maximum at k=0 nor $R_{xy}(k)$ an even function.

CALCULATION:

x(n) ={ 1,2,3,4} y(n)={1,2,1,2}

$$R_{xy}(K) = \sum_{n=-\infty}^{\infty} x(n)y(n-K)$$

Put K=0 in the above equation, we get

$$Rxy(0) = \sum_{n=-\infty}^{\infty} x(n)y(n)$$

$$Rxy(0) = 1 + 4 + 3 + 8 = 16$$

Put K=1 in the above equation, we get

$$Rxy(1) = \sum_{n=-\infty}^{\infty} x(n)y(n-1)$$

$$Rxy(1) = 2 + 6 + 4 = 12$$

Put K=2 in the above equation, we get

$$Rxy(2) = \sum_{n=-\infty}^{\infty} x(n)y(n-2)$$

$$Rxy(2) = 3 + 8 = 11$$

Put K=3 in the above equation, we get

$$Rxy(3) = \sum_{n=-\infty}^{\infty} x(n)y(n-3)$$

$$Rxy(3) = 4 = 4$$

Put K=-1 in the above equation, we get

$$Rxy(-1) = \sum_{n=-\infty}^{\infty} x(n)y(n+1)$$

$$Rxy(-1) = 2 + 6 + 2 = 13$$

Put K=-2 in the above equation, we get

$$Rxy(-2) = \sum_{n=-\infty}^{\infty} x(n)y(n+2)$$

$$Rxy(-2) = 1 + 4 = 5$$

Put K=-3 in the above equation, we get

$$Rxy(-3) = \sum_{n=-\infty}^{\infty} x(n)y(n+3)$$

$$Rxy(-3) = 2 = 2$$

Rxy = [2 5 10 16 12 11 4]

PROGRAM: CROSS CORRELATION USING CONV

```
clc;                                                    % clear screen
clear all;                                              % clear work space
close all;                                              % close all figure windows
x = input('Enter the first sequence x(n)=');            % first sequence
y = input('Enter the second sequence y(n)=');           % second sequence
r = conv(x,fliplr(y));                  %convolute first sequence with folded second signal
disp('Cross Correlation Output= ');
disp(r);                                                % display result
n1= length(x)-1;
t1 = 0: n1;                                             %Graphical display of first input
sequence
subplot(2,2,1);
stem(t1,x);
xlabel('n')
ylabel('x(n)');
title('Plot of x(n)');
n2 = length(y)-1;
t2 = 0:n2;                                      %Graphical display of second input sequence
subplot(2,2,2);
stem(t2,y);
```

```
xlabel('n')
ylabel('y(n)');
title('Plot of y(n)');
k = -n2:n1; %Graphical display of output sequence subplot(2,1,2);

stem(k,r);
xlabel('n');
ylabel('r(n)');
title('Cross Correlation Output');
```

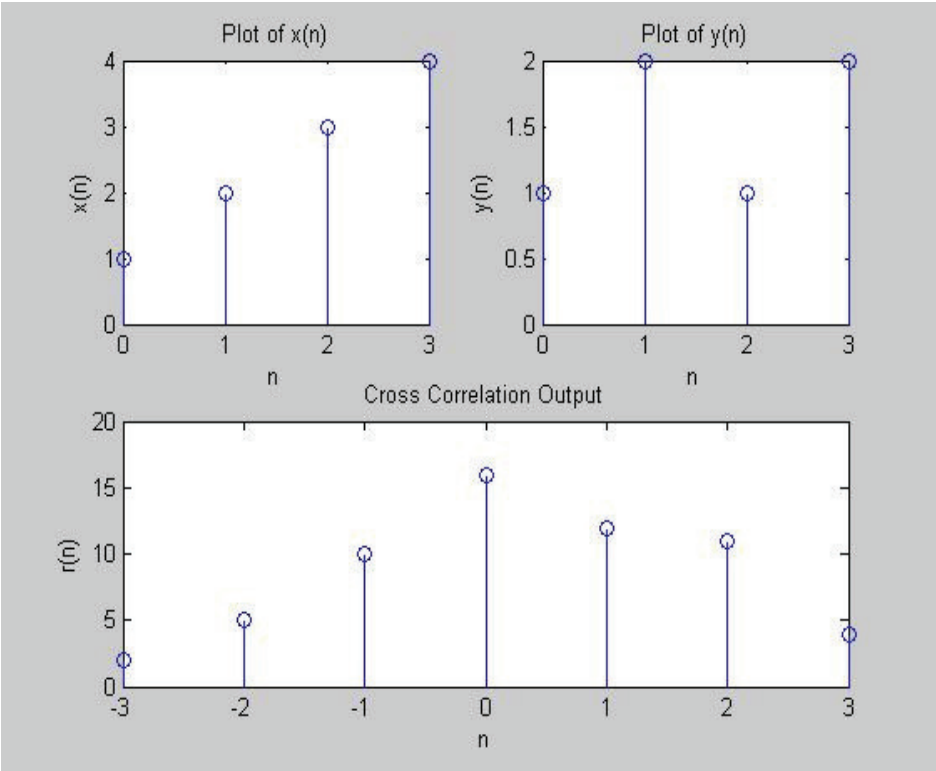
OUTPUT:

Enter the first sequence x(n) = [1 2 3 4]

Enter the second sequence y(n) = [1 2 1 2]

Cross Correlation Output=

2 5 10 16 12 11 4



PROGRAM: CROSS CORRELATION USING XCORR

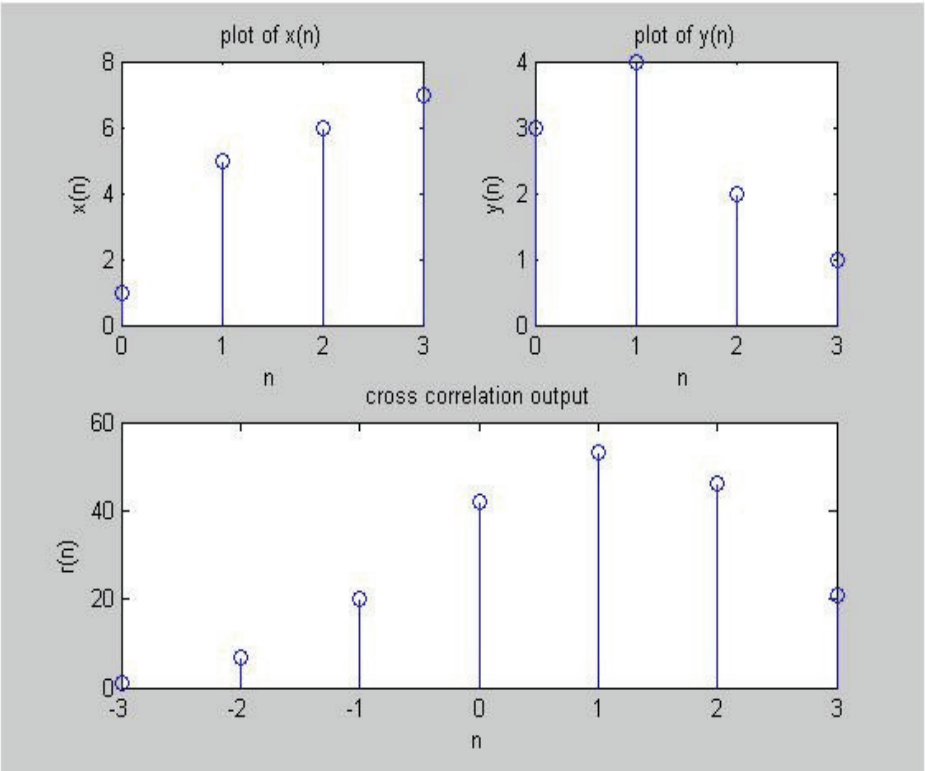
```
clc;                                                    % clear screen
clear all;                                              % clear workspace
close all;                                              % close all figure windows
x = input('Enter the first sequence x(n) =');           % first sequence
y = input('Enter the second sequence y(n) =');          % second sequence
r = xcorr(x,y);                                         % calculate cross correlation
disp('Cross Correlation Output    = ');
disp(r);                                                % display the output
n1 = length(x)-1;                                       % graphical plot of first input sequence
t1 = 0:n1;
subplot(2,2,1);
stem(t1,x);
xlabel('n');
ylabel('x(n)');
title('plot of x(n)');
n2 = length(y)-1;                                       % graphical plot of second input sequence
t2 = 0:n2;
subplot(2,2,2);
stem(t2,y);
xlabel('n')
ylabel('y(n)');
title('plot of y(n)');
N = max(n1,n2);                                         % graphical plot of output sequence
k = -N:N;
subplot(2,1,2);
stem(k,r);
xlabel('n');
ylabel('r(n)');
title('cross correlation output');

x= input("seq1');                                       % Property Rxy(-k) = Ryx(k)
```

```
y = input('seq2');
Rxy = xcorr(x,y);
Ryx=xcorr(y,x);
Rxy1 = fliplr(Rxy);
if Rxy1 = Ryx
disp('Rxy(-k) = Ryx(k) -> proved'); else
disp('Not proved');
end
```

```
Rxx=xcorr(x,x);                                   %|Rxy(k)|^2 < Rxx(-0).Ryy(0)
Ryy = xcorr(y,y);
R =abs(Rxy).^2
CIx = ceil(length(Rxx)/2);
CIy= ceil(length(Ryy)/2);
Rxx.0=Rxx(CIx);
Ryy.0=Ryy(CIy);
R1 = Rxx.0+ Ryy.0;
If R <=R1
disp('Property Proved');
else
disp('Property not proved');
end
```

OUTPUT:

```
Enter the first sequence x(n) = [1 5 6 7]
Enter the second sequence y(n) = [3 4 2 1]
Cross Correlation Output  =
    1    7   20   42   53   46   21
```



OUTCOME: Cross-correlation of the sequence is found and properties of cross correlation are verified.

VIVA QUESTIONS WITH ANSWERS

1. What are even and odd signals?

A discrete time signal x(n) is called even or symmetric signal if it satisfies the condition x(-n) = x(n). A discrete time signal x(n) is called odd or anti-symmetric signal if it satisfies the condition x-(n) = -x(n).

2. What is DTFT?

DTFT is Fourier Transform of DT signal that converts the sampled DT signal from time domain to frequency domain. Frequency domain representation parameters are magnitude and phase. DTFT gives frequency response that includes magnitude response and phase response.

3. If DTFT is Fourier Transform of DT signal then what is DFT?

DFT is frequency sampling of DTFT. When DTFT is sampled in frequency domain we get DFT.

4. Find DTFT and Energy Density Spectrum of x[n] = u[n].

Energy of u[n] is infinite. Therefore u[n] is not energy signal. Fourier Transform is defined only for energy signal.

5. DTFT gives continuous spectra or discrete spectra?

When signal is periodic spectrum is Discrete. If the signal is not- periodic then spectrum is always continuous. DTFT is Fourier transform of Non-periodic signals. Therefore DTFT gives continuous spectra.

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

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

7. What is Periodic Convolution?

Periodic convolution is convolution of two periodic signals of the same period. When two periodic signals are periodic with common period, periodic convolution is similar to circular convolution.

EXPERIMENT NO-7:- DIFFERENCE EQUATION

AIM: To solve a given difference equation

OBJECTIVE:

1. To solve the given difference equation(response of the filter) by varying the input sequences as "Impulse input, Exponential input and Sinusoidal input" with and without initial conditions using an inbuilt MATLAB functions "FILTER and FILTIC"
2. To verify the results theoretically.

THEORY:

A General N^{th} order Difference equations looks like,

$y[n] + a_1y[n-1] + a_2y[n-2] + \ldots + a_Ny[n-N) = b_0x[n] + b_1x[n-1] + \ldots. + b_Mx[n-M]$

Here y[n] is "Most advanced" output sample and y[n-m] is "Least advanced" output sample. The difference between these two index values is the order of the difference equation.

Here we have: n-(n-N) = N

We can rewrite the above equation as, $y[n] + \sum a_i y[n-i] = \sum b_i\, x[n-i]$

y[n] becomes,

$$y[n] = -\sum a_i y[n-i] + \sum b_i\, x[n-i]$$

Example:

$$y[n+2] - 1.5y[n+1] + y[n] = 2x[n]$$

In general we start with the "Most advanced" output sample. Here it is y [n+2].

So, here we need to subtract 2 from each sample argument. Weget

$$y[n] - 1.5y[n-1] + y[n-2] = 2x[n-2]$$

$$\Rightarrow y[n] = 1.5y[n-1] - y[n-2] + 2x[n-2]$$

Lets assume our input $x[n] = u[n] = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$

In our example we have taken $x[n] = u[n] = \begin{cases} 0 & x < 0 \\ 1 & 0 \leq x < 10 \end{cases}$

We need N past values to solve N^{th} order difference equation.

y [-2] = 1

y[-1] = 2

Compute y[n] for various values of n

y[0] = 1.5y[-1] - y[-2] + 2x[-2]

= 1.5*2 – 1 +

2*0 y[0] = 2

y[1] = 1.5y[0] - y[-1] + 2x[-1]

= 1.5*2 - 2 + 2*0

y[1] = 1

y[2] = 1.5y[1] - y[0] + 2x[0]

= 1.5*1 – 2 + 2*1

y[2] = 1.5

And so on…

ALGORITHM:

1. For the given difference equation, rewrite the equation so that y[n] and its delayed samples are on the LHS and x[n] and its delayed samples are on the RHS
2. Create a matrix A for the coefficients of y[n] and its delayed versions
3. Create a matrix B for the coefficients of x[n] and its delayed versions
4. Define the input signal unit impulse, unit step, exponential and sinusoidal
5. Find the response y[n] of the system defined by A and B coefficients to the input excitation using filter command
6. Display and plot the impulse, step, exponential and steady state response y[n]

PROGRAM: SOLUTION OF DIFFERENCE EQUATION WITHOUT INITIAL CONDITIONS

```
clc;                                          % clear screen
close all;                                    % close all figure windows
clear all;                                    % clear work space
                                              %to find impulse response
N= input('Enter the length of response =  '); % define the length of output
b = [-2 5/4];                                 % coefficients of x(n)
a = [1 1/4 -1/8];                             % coefficients of y(n)
```

```
x = [1,zeros(1,N-1)];                          % define the impulse signal
n = 0:N-1;                                     % define x axis
h = filter(b,a,x);                             % calculate the response of the system
disp('Response of filter =');
disp(h);                                       % display the output
subplot(2,1,1);                                % graphical plot of input and output
stem(n,x);
title('Impulse input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title('Impulse response');
xlabel('n');
ylabel('h(n)');
                                               %to find step response
N= input('Enter the length of response =  ');  % define the length of output
b = [-1 2];                                    % coefficients of x(n)
a = [1 -1/4 -3/8];                             % coefficients of y(n)
x = [ones(1,N)];                               % define the unit step signal
n = 0:1:N-1;                                   % define the x axis
h = filter(b,a,x);                             % calculate the step response
disp('Response of filter =');
disp(h);                                       % display the output
subplot(2,1,1);                                % graphical plot of input and output
stem(n,x);
title('Step input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title('Step response');
```

```
xlabel('n');
ylabel('h(n)');
                                        %to find exponential response
N= input('Enter the length of response =    ');   % define the length of response
b = [1];                                % coefficients of x(n)
a = [1 -5/6 1/6];                       % coefficients of y(n)
n = 0:1:N-1;                            % define x axis
x = 2.^n;                               % define exponential input
h = filter(b,a,x);                      % calculate the exponential response
disp('Response of filter =');
disp(h);                                % display the output
subplot(2,1,1);                         % graphical plot of the input and output
stem(n,x);
title('Exponential input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title('Exponential response');
xlabel('n');
ylabel('h(n)');
                                        %to find steady response
N= input('Enter the length of response =    ');   % define the length of response
b = [5];                                % coefficients of x(n)
a = [1 -0.5];                           % coefficients of y(n)
n = 0:1:N-1;                            % define x axis
x = cos(0.5*pi*n);                      % define sinusoidal input
h = filter(b,a,x);                      % calculate the sinusoidal response
disp('Response of filter =');
disp(h);                                % display the output
subplot(2,1,1);                         % graphical plot of the input and output
stem(n,x);
```

```
title('Steady input');
xlabel('n');
ylabel('x(n)');
subplot(2,1,2);
stem(n,h);
title('Steady response');
xlabel('n');
ylabel('h(n)');
```

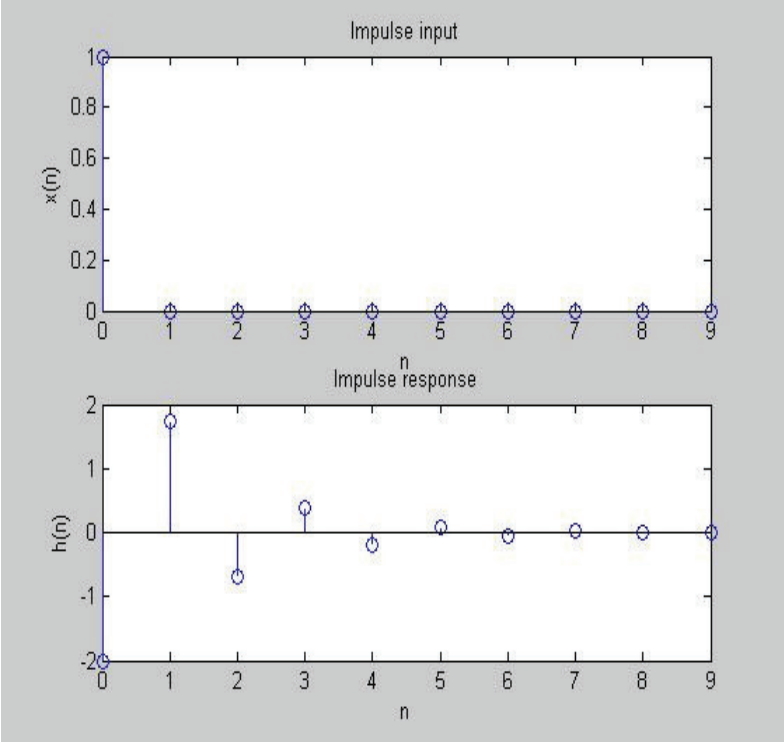
OUTPUT:

Impulse Response:

Enter the length of response = 10

Response of filter =

-2.0000 1.7500 -0.6875 0.3906 -0.1836 0.0947 -0.0466 0.0235 -0.0117 0.0059

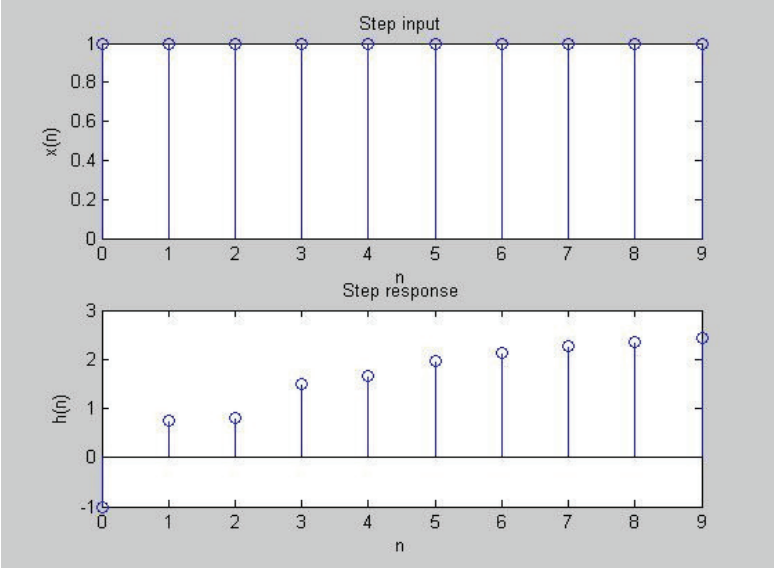


Step Response:

Enter the length of response = 10

Response of filter =

-1.0000 0.7500 0.8125 1.4844 1.6758 1.9756 2.1223 2.2714 2.3637 2.4427

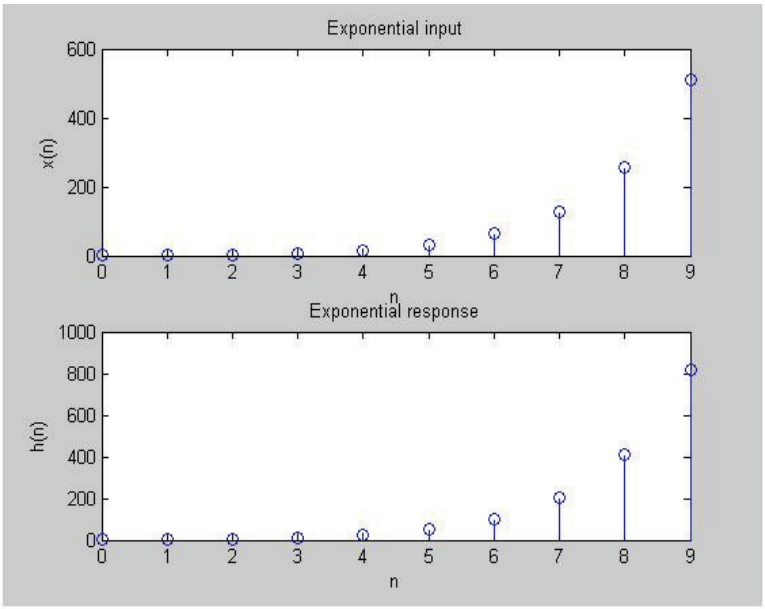


Exponential Response:

Enter the length of response =

10 Response of filter =

1.0000 2.8333 6.1944 12.6898 25.5424 51.1704 102.3849 204.7924 409.5962 819.1981



Steady State Response:

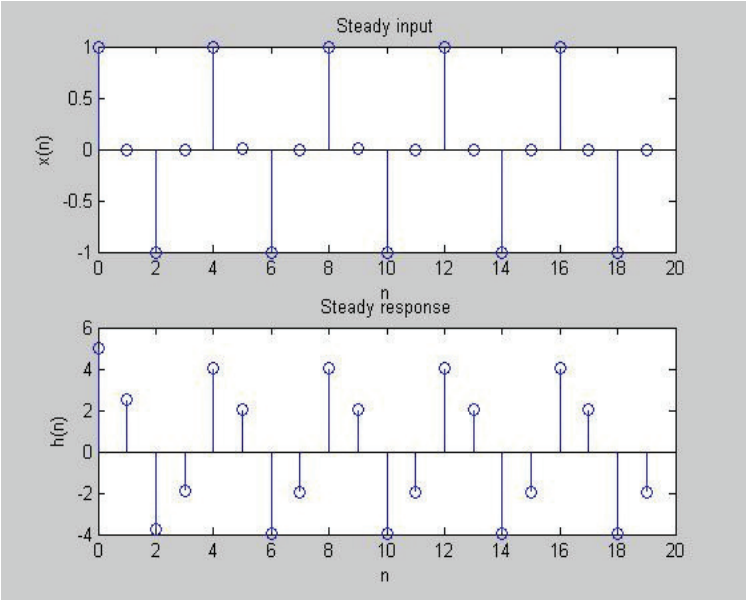
Enter the length of response = 20

Response of filter =

5.0000 2.5000 -3.7500 -1.8750 4.0625 2.0313 -3.9844 -1.9922 4.0039

2.0020 -3.9990 -1.9995 4.0002 2.0001 -3.9999 -2.0000 4.0000 2.0000

-4.0000 -2.0000



PROGRAM: SOLUTION OF DIFFERENCE EQUATION WITH INITIAL CONDITIONS

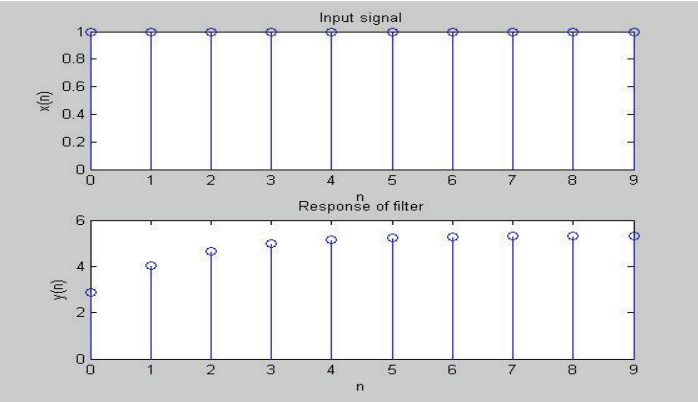
```
clc;                                                    % clear screen
close all;                                              % close all figure windows
clear all;                                              % clear work space
N = input('Enter the length of response = ');           % define the length of response
a= [1 -3/4 1/8];                                        %filter co-efficient
b=[2];
y = [1 -1];                                             %initial conditions
x =[0 0];
xic = filtic(b,a,y,x);
x = [ones(1,N)];                                        %input signal
y = filter(b,a,x,xic);                                  % calculate the response
```

```
disp('Response of filter = ');
disp(y);
n = 0:1:N-1; %graphical plot of input and output sequence subplot(2,1,1);

stem(n,x);
xlabel('n');
ylabel('x(n)');
title('Input signal');
subplot(2,1,2);
stem(n,y);
xlabel('n');
ylabel('y(n)');
title('Response of filter');
```

OUTPUT:

Enter the length of response =
10 Response of filter =

2.8750 4.0313 4.6641 4.9941 5.1626 5.2477 5.2904 5.3119 5.3226 5.3280



OUTCOME: Solution of the difference equation is found and the output response is calculated.

VIVA QUESTIONS WITH ANSWERS

1. What is Transient response?

Transient response of the system is the response of the system that decays to zero.

2. What is Steady State Response?

Everlasting response of the system that depends on magnitude response and phase response of the system is steady state response of the system.

3. What is Infinite Impulse Response?

When length of h[n] is infinite it is called infinite impulse response. E.g. h[n] = (½)n u[n]

4. What is Finite Impulse Response?

When length of h[n] is finite it is called finite impulse response.

5. What is Finite Impulse Response?

When length of h[n] is finite it is called finite impulse response. 6. What is zero input response?

If the initial state of the system is NOT zero and the input x[n] = 0 to all n, then the output of the system with zero input is called the zero input response or natural response or free response of the system. When length of h[n] is finite it is called finite impulse response.

7. What is zero state step response?

If the initial state of the system is zero and the input x[n]=u[n] then the output of the system is called zero step response of the system.

EXPERIMENT NO-8:- N-POINT DFT

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

OBJECTIVE:

1. To find the N point DFT of a given sequence using the MATLAB inbuilt function "FFT" and to find Magnitude and Phase of DFT sequence using functions "ABS and ANGLE"
2. To verify the result theoretically.

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

The sequence of Ncomplex numbers$x_0,...,x_{N-1}$ is transformed into the sequence of N complex numbers $X_0, ..., X_{N-1}$ by the DFT according to the formula:

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

EXAMPLE:

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

We have,

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

For k = 0,

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

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

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

For k = 1,

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

$$X(1) = x(0) + x(1)\, e^{-j\pi/2} + x(2)\, e^{-j\pi} + x(3)\, e^{-j3\pi/2}$$

$$X(1) = 1 + \cos(\pi/2) - j\sin(\pi/2)$$

$$X(1) = 1 - j$$

For k = 2

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

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

$$X(2) = 1 + \cos\pi - j\sin\pi$$

$$X(2) = 1\text{-}1 = 0$$

For k = 3,

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

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

$$X(3) = 1 + \cos(3\pi/2) - j\sin(3\pi/2)$$

$$X(3) = 1 + j$$

The DFT of the given sequence is,

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

To find Magnitude of X(k):

Magnitude= $(a^2+b^2)^{1/2}$

Where a and b are real and imaginary parts respectively

To fine Phase of X (k):

Phase=$\tan^{-1}(b/a)$

ALGORITHM:

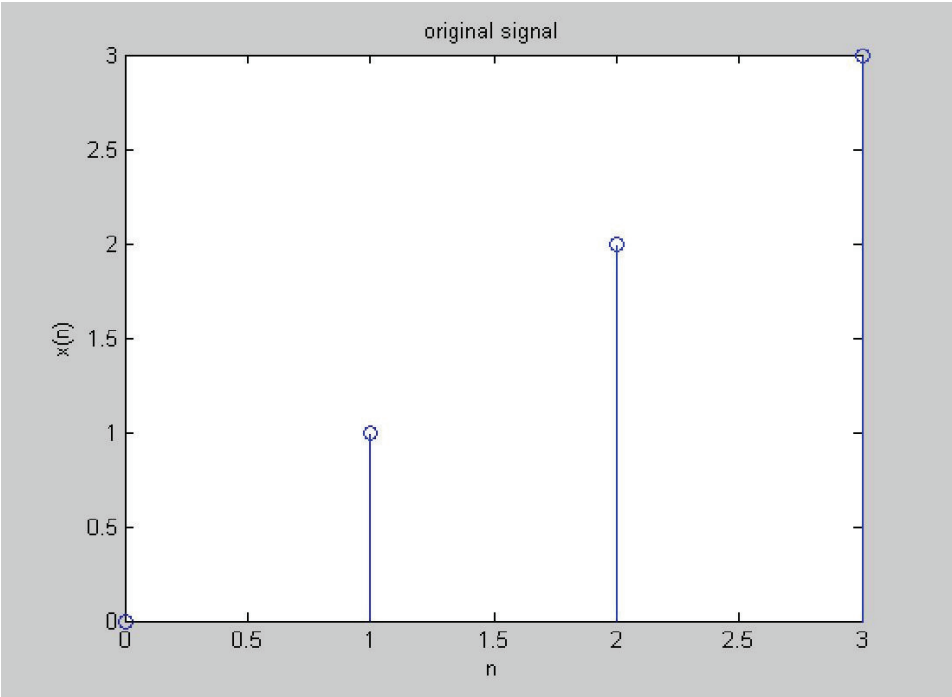
1. Enter the number of points N
2. Enter the input sequence elements x[n]
3. Create a vector for sample index n
4. Calculate DFT using built in function FFT
5. Plot the magnitude and phase spectrum

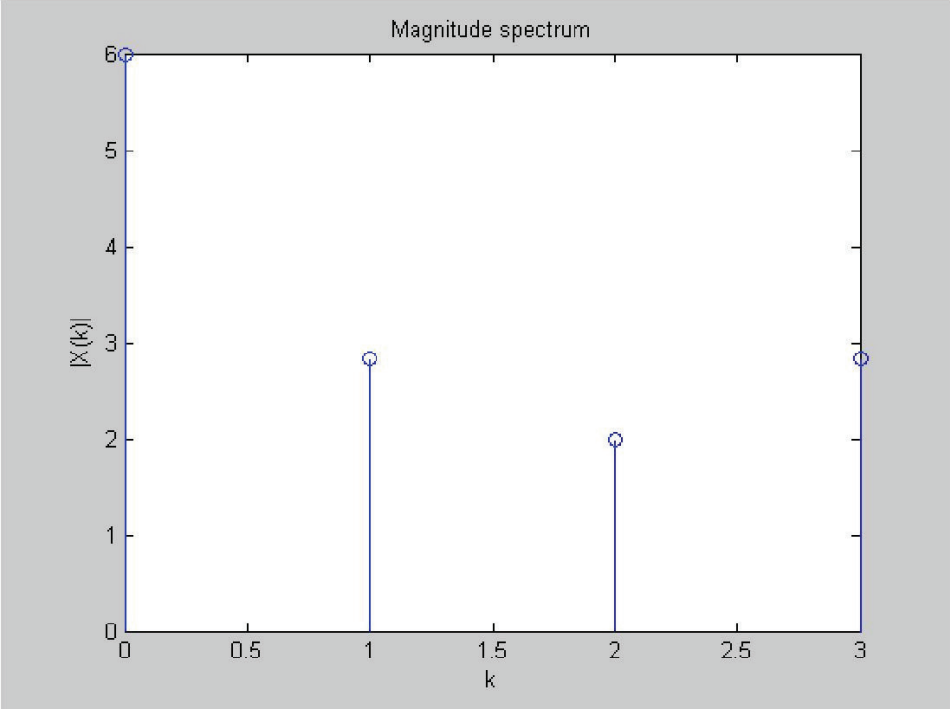
PROGRAM: N POINT DFT

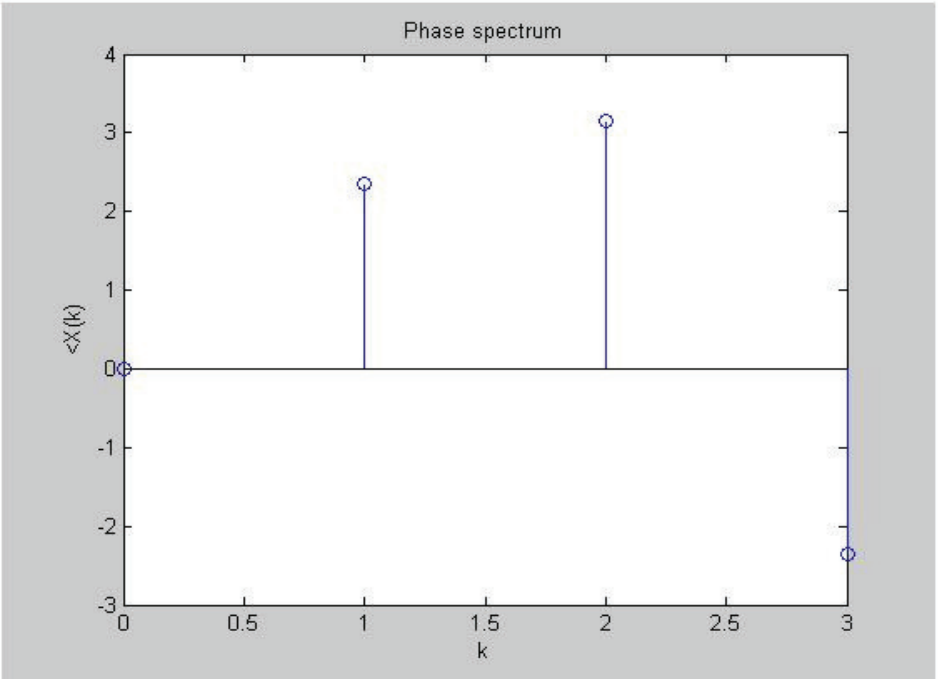
```
clc;                                                    % clear screen
close all;                                              % close all figure windows
clear all;                                              % clear work space
N = input('enter the N point = ');      % define the number of points to be taken for DFT
xn = input('enter the input sequence x(n) = ');         % input sequence
Xk = fft(xn,N);                                         % find the N point DFT
disp('N point DFT of x(n) is = ' );
disp(Xk);                                       % display the DFT of the input sequence
figure(1);
n = 0:1:length(xn)-1;                                   % define x axis for input
stem(n,xn);                                             % plot the input
xlabel('n');
ylabel('x(n)');
title('original signal');
figure(2);
k = 0:N-1;                                              % define the x axis for output sequence
stem(k,abs(Xk));                                        % plot the absolute value of output
xlabel('k');
ylabel('|X(k)|');
title('Magnitude spectrum');
figure(3);
stem(k,angle(Xk));                      % stem(k. (angle(Xk)*180/pi)), plot the phase of DFT
xlabel('k');
ylabel('<X(k)');
title('Phase spectrum');
```

OUTPUT:

```
enter the N point = 4
enter the input sequence x(n) = [0 1 2 3]
N point DFT of x(n) is =
   6.0000          -2.0000 + 2.0000i          -2.0000          -2.0000 - 2.0000i
```


original signal
x(n)
n
0
0.5
1
1.5
2
2.5
3


Magnitude spectrum
|X(k)|
k
0
1
2
3
4
5
6
0.5
1.5
2.5



PROGRAM: N POINT DFT USING FUNCTION

```
clc;                                                    % clear screen
close all;                                              % close all figure windows
clear all;                                              % clear work space
N = input('enter N point =');          % define the number of points to be taken for DFT
xn = input('enter the input sequence x(n) = ');         % input sequence
Xk = dft(xn,N);                                         % find the N point DFT
n = 0:1:N-1;                                            % x axis values for plotting input
k = 0:1:N-1;                                        % x axis values for plotting output
disp('N point DFT of x(n) = ');
disp(Xk);                                               % display DFT
subplot(2,2,1)                                          % graphical plot of input signal
stem(n,xn);
xlabel('n');
ylabel('x(n)');
title(' input signal');
subplot(2,2,2);                                % graphical plot of magnitude of output signal
stem(k,abs(Xk));
```

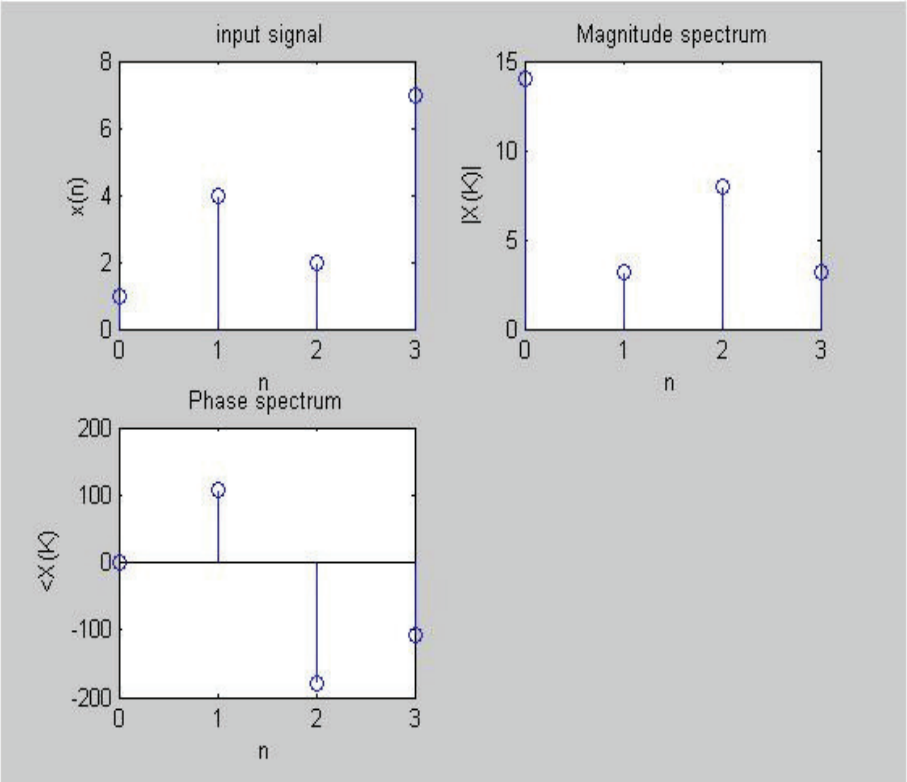
```
xlabel('n');
ylabel('|X(K)|');
title('Magnitude spectrum');
subplot(2,2,3);                          % graphical plot of angle of output signal
stem(k,angle(Xk)*180/pi);
xlabel('n');
ylabel('<X(K)');
title('Phase spectrum');
```

%function file

```
function [Xk] =
dft(xn,N) n = 0:1:N-1;
k = 0:1:N-1;
WN = exp(-
j*2*pi/N); nk = n'*k;
WNnk = WN.^nk;
Xk = xn*WNnk;
```

OUTPUT:

```
enter N point = 4
enter the input sequence x(n) = [1 4 2 7]
N point DFT of x(n) =
  14.0000          -1.0000 + 3.0000i          -8.0000 - 0.0000i          -1.0000 - 3.0000i
```



OUTCOME: DFT of the given sequence is found and the results are verified using MATLAB.

VIVA QUESTIONS WITH ANSWER

1. DFT gives discrete spectrum or continuous spectrum? Justify?

DFT gives discrete spectrum. If the signal is periodic then spectrum is discrete and if the signal is non-periodic then spectrum is continuous. DFT assumes that input signal is periodic and therefore DFT gives discrete spectrum.

2. How to find energy of signal from its DFT?

According to Parseval's energy theorem, Energy of the signal is given by.

3. What is the need of FFT?

It may be noted that the number of complex multiply and add operations required by the simple forms both the DFT and IDFT is of order N^2. This is because there are N data points to calculate, each of which requires N complex arithmetic operations. In computer science jargon, we say they have algorithmic complexity O(N^2). This is not good news. If we

can't do any better than this then the DFT will not be very useful for the majority of practical DSP applications. Fortunately, there are a number of different 'Fast Fourier Transform' (FFT) algorithms that enable us to do very much better than this.

4. What is FFT? What is difference between DIT and DIF FFT?

A fast Fourier transform (FFT) is an efficient algorithmto compute the discrete Fourier transform (DFT) and it's inverse.The most significant difference between simple DIF and DIT algorithms is that DIF starts with normal order input and generates bit reversed order output. In contrast, DIT starts with bit reversed order input and generates normal order output. So if both forward and inverse transforms are required and bit reversed addressing isn't available, then the choice would seem clear. Use DIF for the forward transform and DIT for the inverse transform. (For the inverse transform you will need to conjugate the twiddle factors.) Unfortunately, the issue isn't quite so simple. If your performing FFT's on pure real data it may be simpler to use a modified DIT for the forward transform and a modified DIF for the inverse transform

5. How efficient is the FFT?

The DFT takes N2 operations for N points. Since at any stage the computation required to combine smaller DFTs into larger DFTs is proportional to N, and there are $\log_2(N)$ stages (for radix 2), the total computation is proportional to N $\log_2(N)$. Therefore, the ratio between a DFT computation and an FFT computation for the same N is proportional to N / $\log_2(N)$. In cases where N is small this ratio is not very significant, but when N becomes large, this ratio gets very large. (Every time you double N, the numerator doubles, but the denominator only increases by 1.

6. FFT is faster than DFT. Justify.

FFT produces fast results because calculations are reduced by decomposition technique. In FFT, N point DFT is decomposed into two N/2 point DFT's, N/2 point DFT is decomposed into N/$ point DFT's and so on.. Decomposition reduces calculations. FFT algorithms are implemented using parallel processing techniques. Because calculations are done in parallel, FFT produces fast computations.

7. What do you mean by Decimation?

Decimation means Sampling

8. Which algorithm is more powerful: DIT-FFT or DIF-FFT?

   Computationally, both the algorithms are exactly same

9. Why Radix-2 algorithms are fast compared to radix-3 algorithms. ?

In FFT, N point DFT is decomposed into two N/2 pt DFT's, N/2 pt DFT is decomposed into N/4 pt DFT's and so on... Decomposition reduces calculations this process continues till further decomposition is not possible. In radix-2 last level of decomposition is when the length of signal becomes 2 pt. For minimum calculations there must be maximum levels of decompositions. In Radix-2 algorithms, we get maximum levels of decompositions and therefore a radix-2 algorithm requires less calculation. Radix-2 algorithms are fast algorithms.

EXPERIMENT NO-9:- LINEAR CONVOLUTION USING DFT AND IDFT

AIM: Linear convolution of two given sequences using DFT and IDFT.

OBJECTIVE:

1. To find the Linear convolution of a given sequence using the inbuilt MATLAB functions "FFT and IFFT" for DFT and IDFT and verify the result using the function "CONV"
2. To verify the result theoretically.

THEORY: Convolution is an integral concatenation of two signals. It has many applications in numerous areas of signal processing. The most popular application is the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system. Note that convolving two signals is equivalent to multiplying the Fourier Transform of the two signals.

Mathematic Formula:

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

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

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

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

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

EXAMPLE:

$x_1(n) = \{1, 1, 2\}$

$x_2(n) = \{1, 2\}$

For linear convolution,

Length N = Length(x_1) + Length(x_2) - 1

N = 3 + 2 – 1 = 4

Convolution of two sequences $x_1(n)$ and $x_2(n)$

is, $x_3(n) = IDFT[X_3(k)]$

$x_3(n) = IDFT[X_1(k)\ X_2(k)]$

Where, $X_1(k) = DFT\ [x_1(n)]$ and $X_2(k) = DFT\ [x_2(n)]$

$$X1(k) = \sum_{n=0}^{N-1} x1(n) e^{-j2\pi kn/N} \quad k= 0, 1, 2, \dots, N-1$$

Given $x_1(n) = \{1, 1, 2\}$ and N=4

$X1(0) = \sum_{n=0}^{3} x1(n) = 1 + 1 + 2 = 4$

$X1(1) = \sum_{n=0}^{3} x1(n) e^{-j2\pi n/4} = 1 - j - 2 = -1 - j$

$X1(2) = \sum_{n=0}^{3} x1(n) e^{-j\pi n/2} = 1 - 1 + 2 = 2$

$X1(3) = \sum_{n=0}^{3} x1(n) e^{-j3\pi n/4} = 1 + j - 2 = -1 + j$

$X_1(k) = \{4, -1 - j, 2, -1 + j\}$

Now,

$$X2(k) = \sum_{n=0}^{N-1} x2(n) e^{-j2\pi kn/N} \quad k= 0, 1, 2, \dots, N-1$$

Given $x_2(n) = \{1, 2\}$ and N=4

$X2(0) = \sum_{n=0}^{3} x2(n) = 1 + 2 = 3$

$X2(1) = \sum_{n=0}^{3} x2(n) e^{-j2\pi n/4} = 1 + 2(-j) = 1 - j2$

$X2(2) = \sum_{n=0}^{3} x2(n) e^{-j\pi n/2} = 1 + 2(-1) = -1$

$X2(3) = \sum_{n=0}^{3} x2(n) e^{-j3\pi n/4} = 1 + 2(j) = 1 + j2$

$X_2(k) = \{3, 1-j2, -1, 1+j2\}$

We know that,

$X_3(k) = X_1(k)\ X_2(k)$

$X_3(k) = \{12, -3+j, -2, -3-j\}$

Convolution of two given sequences

is, $x_3(n) = IDFT[X_3(k)]$

$$x3(n) = (1/N) \sum_{k=0}^{N-1} X3(K) e^{j2\pi kn/N} \quad n = 0, 1, 2, \dots, N-1$$

$x3(0) = (1/4) \sum_{k=0}^{3} X3(k)$ = (1/4) [12 – 3 +j -2 -3 –j] = 1

$x3(1) = (1/4) \sum_{k=0}^{3} X3(k) e^{j\pi k/2}$ = (1/4) [12 + (-3 + j) j + (-2) (-1) + (-3 - j) (-j)] = 3

$x3(2) = (1/4) \sum_{k=0}^{3} X3(k) e^{j\pi k}$ = (1/4) [12 + (-3 + j) (-1) + (-2) (1) + (-3 - j) (-1)] = 4

$x3(3) = (1/4) \sum_{k=0}^{3} X3(k) e^{j3\pi k/4}$ = (1/4) [12 + (-3 + j) (-j) + (-2) (-1) + (-3 - j) (j)] = 4

Convoluted sequence of two given sequences
is, $x_3(n) = \{1, 3, 4, 4\}$

ALGORITHM:

1. Find the length of the first sequence x[n]
2. Find the length of second sequence h[n]
3. Define the number of points of DFT N to be taken as length(x[n])+ length(h[n])-1
4. Find the DFT of x[n] and h[n] using fft command as X(k) and H(k) respectively.
5. Find Y(k)= X(k) * H(k).
6. Find y[n] using IDFT of Y(k)
7. Verify y[n]= convolution of x(n) and h(n) using conv command

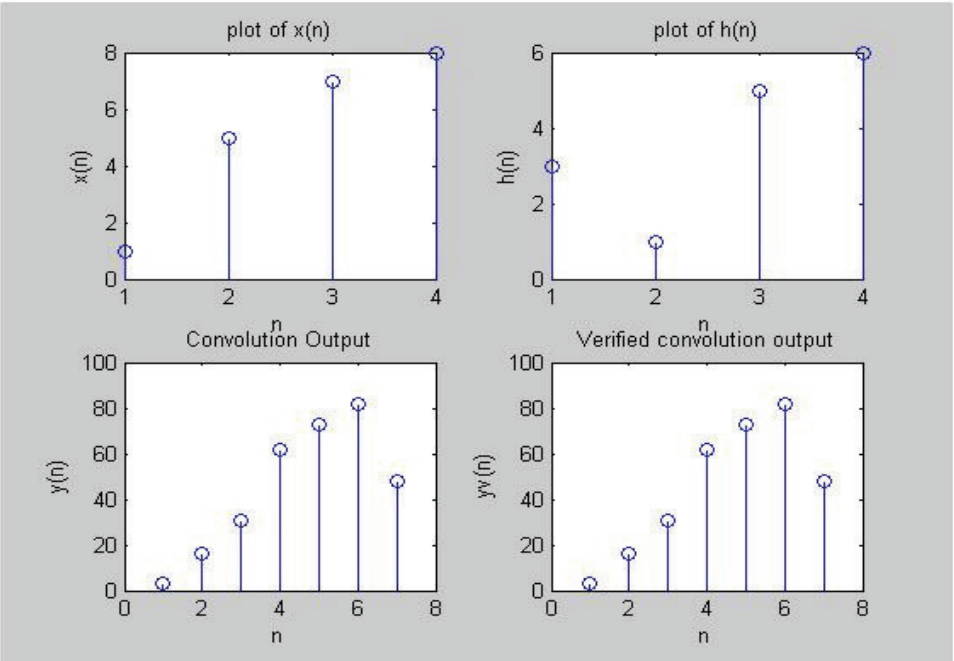
PROGRAM: LINEAR CONVOLUTION USING DFT AND IDFT

```
clc;                                        % clear screen
clear all;                                  % clear work space
close all;                                  % close all figure windows
xn = input('enter the first sequence x(n) = '); % first sequence
hn = input('enter the second sequence h(n) = '); % second sequence
N = length(xn)+length(hn)-1;                % length of output
Xk = fft(xn,N);                             % N point DFT of first sequence
Hk = fft(hn,N);                             % N point DFT of second sequence
Yk = Xk.*Hk;                    % multiplication of DFTs of first and second sequence
yn = ifft(Yk,N);                            % take inverse DFT
disp('Linear convolution of x(n) and h(n) =');
disp(yn);                                    % display the output
subplot(2,2,1);                             % graphical display of first sequence
stem(xn);
```

```
xlabel('n');
ylabel('x(n)');
title('plot of x(n)');
subplot(2,2,2);                          % graphical display of second sequence
stem(hn);
xlabel('n');
ylabel('h(n)');
title('plot of h(n)');
subplot(2,2,3);                          % graphical display of output sequence
stem(yn);
xlabel('n');
ylabel('y(n)');
title('Convolution Output');
yv =conv(xn,hn);                         % verification of linear convolution
disp('Convolution in time domain = ');
disp(yv);
subplot(2,2,4);                          % graphical display of output sequence
stem(yv);
xlabel('n');
ylabel('yv(n)');
title('Verified convolution output');
```

OUTPUT:

```
Enter the first sequence x(n) = [1 5 7 8]
Enter the second sequence h(n) = [3 1 5 6]

Linear convolution of x(n) and h(n) =
    3.0000    16.0000   31.0000   62.0000   73.0000   82.0000    48.0000

Convolution in time domain =
     3    16    31    62    73    82    48
```



OUTCOME: Linear convolution of two given sequences found using DFT and IDFT and the results are verified.

VIVA QUESTIONS WITH ANSWER

1. What is the drawback in Fourier Transform and how it is overcome?

The draw back in Fourier Transform is that it is a continuous function of ω and so it cannot be processed by digital system. This drawback is overcome by using Discrete Fourier Transform. The DFT converts the continuous function of ω to a discrete function of ω

2. Mention the importance of DFT

DFT is used for spectral analysis of signals using digital computer and to perform filtering operations on signals using digital computer.

3. What is the application of Convolution?
4. Application of Convolution is to find output of Digital Filter for any given input signal. Output of Digital filter y[n] is linear Convolution of input signal x[n] and impulse response of the filter h[n].
5. What is convolution property of DFT?

Convolution in time domain corresponds to multiplication in frequency domain. If DFT{x[n]} = X[k] and DFT {h[n]} = H[k] Then DFT{x[n] * h[n]} = X[k] H[k]

6. List the differences between linear convolution and circular convolution.

Linear convolution	Circular convolution
1. The length of the input sequences can be different	1. The length of the input sequences should be same
2. Zero padding is not required	2. If the length of the sequences are different, then zero padding is required
3. The input sequences need not be periodic	3. Atleast one of the input sequence should be periodic or should be periodically extended
4. The output sequence is non-periodic	4. The output sequence is periodic. The periodicity is same as that of input sequences
5. The length of output sequence will be greater than the length of input sequence	5. The length of the input and output sequences are same

EXPERIMENT NO-10:- CIRCULAR CONVOLUTION USING DFT AND IDFT

AIM: To implement circular convolution of two given sequences using DFT and IDFT.

OBJECTIVE:

1. To find the Circular convolution of a given sequence using the inbuilt MATLAB function "FFT and IFFT" for DFT and IDFT.
2. To verify the result theoretically.

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

by, $x_3(n) = IDFT[X_3(k)]$

$x_3(n) = IDFT[X_1(k)\ X_2(k)]$

Where, $X_1(k) = DFT\ [x_1(n)]$, $X_2(k) = DFT\ [x_2(n)]$

EXAMPLE:

Let $x_1(n) = \{1, 1, 2, 1\}$ and $x_2(n) = \{1, 2, 3, 4\}$

$$X1(k) = \sum_{n=0}^{N-1} x1(n) e^{-j2\pi kn/N} \quad k = 0, 1, 2, \ldots, N-1$$

Given $x_1(n) = \{1, 1, 2, 1\}$ and N=4

$X1(0) = \sum_{n=0}^{3} x1(n) = 1 + 1 + 2 + 1 = 5$

$X1(1) = \sum_{n=0}^{3} x1(n) e^{-j2\pi n/4} = 1 - j - 2 + j = -1$

$X1(2) = \sum_{n=0}^{3} x1(n) e^{-j\pi n/2} = 1 - 1 + 2 - 1 = 1$

$X1(3) = \sum_{n=0}^{3} x1(n) e^{-j3\pi n/4} = 1 + j - 2 - j = -1$

$X_1(k) = \{5, -1, 1, -1\}$

Now,

$$X2(k) = \sum_{n=0}^{N-1} x2(n) e^{-j2\pi kn/N} \quad k = 0, 1, 2, \ldots, N-1$$

Given $x_2(n) = \{1, 2, 3, 4\}$ and N=4

$X2(0) = \sum_{n=0}^{3} x2(n) = 1 + 2 + 3 + 4 = 10$

$X2(1) = \sum_{n=0}^{3} x2(n) e^{-j2\pi n/4} = 1 + 2(-j) + 3(-1) + 4(j) = -2 + j2$

$X2(2) = \sum_{n=0}^{3} x2(n) e^{-j\pi n/2} = 1 + 2(-1) + 3(1) + 4(-1) = -2$

$X2(3) = \sum_{n=0}^{3} x2(n)e^{-j3\pi n/4}$ = 1 + 2(j) + 3(-1) + 4(-j) = -2 – j2

$X_2(k)$ = {10, -2+j2, -2, -2-j2}

We know that,

$X_3(k) = X_1(k)\, X_2(k)$

$X_3(k)$ = {50, 2 – j2, -2, 2 + j2}

Convolution of two given sequences
is, $x_3(n) = IDFT[X_3(k)]$

$$x3(n) = (1/N) \sum_{k=0}^{N-1} X3(K)e^{j2\pi kn/N} \quad n = 0, 1, 2, \ldots, N-1$$

$x3(0) = (1/4) \sum_{k=0}^{3} X3(k)$ = (1/4) [50 + 2 - j2 – 2 + 2 + j2] = 13

$x3(1) = (1/4) \sum_{k=0}^{3} X3(k)e^{j\pi k/2}$ = (1/4) [50 + (2 - j2) j + (-2) (-1) + (2 + j2) (-j)] = 14

$x3(2) = (1/4) \sum_{k=0}^{3} X3(k)e^{j\pi k}$ = (1/4) [50 + (2 - j2) (-1) + (-2) (1) + (2 + j2) (-1)] = 11

$x3(3) = (1/4) \sum_{k=0}^{3} X3(k)\, e^{j3\pi k/4}$ = (1/4) [50 + (2 - j2) (-j) + (-2) (-1) + (2 + j2) (j)] = 12

Convoluted sequence of two given sequences
is, $x_3(n)$ = {13, 14, 11, 12}

ALGORITHM:

1. Find the length of the first sequence x[n]
2. Find the length of second sequence h[n]
3. Define the number of points of DFT N to be taken as maximum of length(x[n]) and length(h[n]).
4. Find the DFT of x[n] and h[n] using fft command as X(k) and H(k) respectively.
5. Find Y(k)= X(k) * H(k).
6. Find y[n] using IDFT of Y(k)
7. Verify y[n]= convolution of x(n) and h(n) using conv command

PROGRAM: CIRCULAR CONVOLUTION USING DFT AND IDFT

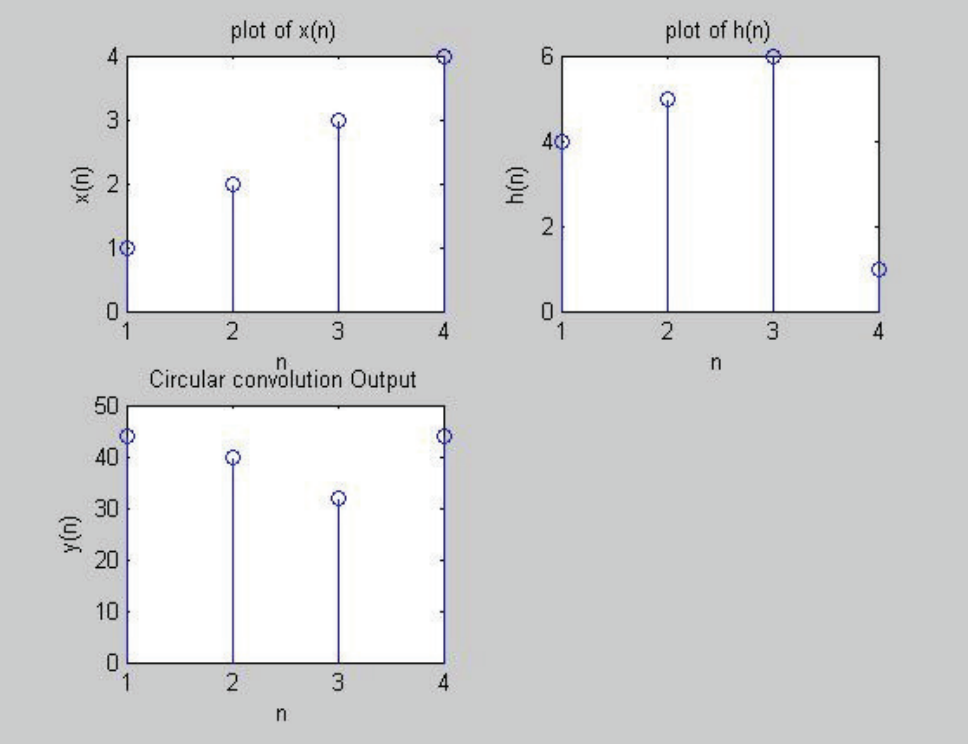
```
clc;                                                    % clear screen
clear all;                                              % clear work space
close all;                                              % close all figure windows
xn = input('enter the first sequence x(n) = ');         % first sequence
hn = input('enter the second sequence h(n) = ');        % second sequence
N = max(length(xn),length(hn));                         % length of convolution
Xk = fft(xn,N);                                         % N point DFT of first sequence
Hk = fft(hn,N);                                         % N point DFT of second sequence
Yk = Xk.*Hk;                                            % multiplication of two DFTs
yn = ifft(Yk,N);                                        % Inverse DFT of the product
disp('Circular convolution of x(n) and h(n) =');
disp(yn);                                               % Display the output
subplot(2,2,1);                                         % graphical display of the first sequence
stem(xn);
xlabel('n');
ylabel('x(n)');
title('plot of x(n)');
subplot(2,2,2);                                 % graphical display of the second sequence
stem(hn);
xlabel('n');
ylabel('h(n)');
title('plot of h(n)');
subplot(2,2,3);                                 % graphical display of the output sequence
stem(yn);
xlabel('n');
ylabel('y(n)');
title('Circular convolution Output');
```

OUTPUT:

```
Enter the first sequence x(n) = [1 2 3 4]
Enter the second sequence h(n) = [4 5 6 1]
```

Circular convolution of x(n) and h(n) =

44 40 32 44



OUTCOME: Circular convolution of two given sequences found using DFT and IDFT and the results are verified.

VIVA QUESTIONS WITH ANSWER

1. Why sectioned convolution is performed?

In linear convolution of two sequences, if one of the sequences is very much larger than the other, then it is difficult to compute the linear convolution using DFT for the following reasons (i) the entire sequence should be available before convolution can be carried out. This makes long delay in getting the output. (ii) large amount of memory is required to store the sequences.

2. What is Zero padding? Why it is needed?

Appending zeros to a sequence in order to increase the size or length of the sequence is called zero padding. In circular convolution, when the two input sequences are of different size, then they are converted to equal size by zero padding.

3. What are the two methods of sectioned convolution?

The two methods of sectioned convolution are overlap add method and overlap save method.

4. In what way zero padding is implemented in overlap save method?

In overlap save method, the zero padding is employed to convert the smaller input sequences to the size of the output sequence of each sectioned convolution. The zero padding is also employed to convert either the last section or the first section of the longer input sequence to the size of the output sequence of each sectioned convolution. This depends on the method of overlapping input sequences.

5. Compare the overlap add and overlap save method of sectioned convolution.

Overlap add method	Overlap save method
1. Linear convolution of each section of longersequencewith smaller sequence is performed	1. Circular convolution of each section of longer sequence with smaller sequence is performed.
2. Zero padding is not required	2. Zero-padding is required to convert the input sequences to the size of output sequence.
3. Overlapping of samples of input sections are not required	3. The N2-1 sample of an input section of longer sequence is overlapped with next input section.
4. The overlapped samples in the output of sectioned convolutions are added to get the overall output.	4. Depending on the method of overlapping the input samples, either the last N2-1 samples or the first N2-1 samples of the output sequence of each sectioned convolution are discarded.

EXPERIMENT NO-11:- DESIGN AND IMPLEMENTATION OF FIR FILTER

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

OBJECTIVE:

1. To design the FIR filter by Hamming window using the inbuilt MATLAB function "FIR1 and HAMMING".
2. To verify the result by theoretical calculations.

THEORY: The FIR filters are of non-recursive type, whereby the present output sample is depending on the present input sample and previous input samples. The transfer function of a FIR causal filter is given by,

$$H(z) = \sum_{n=0}^{N-1} h(n) Z^{-n}$$

Where h(n) is the impulse response of the filter.

The Fourier transform of h(n) is

$$H(e^{jw}) = \sum_{n=0}^{N-1} h(n) e^{-jwn}$$

In the design of FIR filters most commonly used approach is using windows.The desired frequency response $H_d(e^{jw})$ of a filter is periodic in frequency and can be expanded in Fourier series. The resultant series is given by,

$$h_d(n) = (1/2\pi) \int_{-\pi}^{\pi} H(e^{jw}) e^{jwn} dw$$

This is known as Fourier series coefficients having infinite length. One possible way of obtaining FIR filter is to truncate the infinite Fourier series at n = ± [(N-1)/2]Where N is the length of the desired sequence.

The Fourier coefficients of the filter are modified by multiplying the infinite impulse response with a finite weighing sequence w(n) called a window.

Where $w(n) = w(-n) \neq 0$ for $|n| \leq [(N-1)/2]$

$= 0$ for $|n| > [(N-1)/2]$

After multiplying w(n)with $h_d(n)$, we get a finite duration sequence h(n) that satisfies the desired magnitude response,

$h(n) = h_d(n)\, w(n)$ for $|n| \leq [(N-1)/2]$

$= 0$ for $|n| > [(N-1)/2]$

The frequency response $H(e^{jw})$ of the filter can be obtained by convolution of $H_d(e^{jw})$ and $W(e^{jw})$ is given by,

$H(e^{jw}) = (1/2\pi) \int_{-\pi}^{\pi} Hd(ej\theta)\, W(ej(w-\theta))\, d\theta$

$H(e^{jw}) = H_d(e^{jw}) * W(e^{jw})$

ALGORITHM:

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

EXAMPLE:

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

$w_H(n) = 0.54 + 0.46 \cos((2\pi n)/(N-1))$;$-(N-1)/2 \leq n \leq (N-1)/2$

$= 0$; otherwise

The frequency response of Hamming window is,

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

PROGRAM: DESIGN AND IMPLEMENTATION OF FIR FILTER

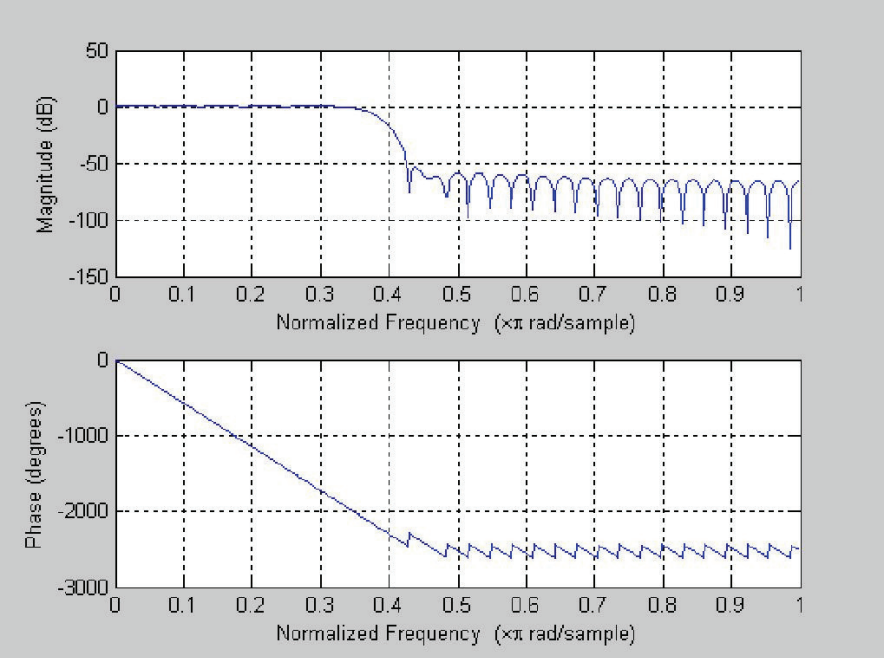
```
clc;                                                    % clear screen

close all;                                              % close all figure windows
clear all; % clear work space
wp = input( 'Enter the 'Pass band edge in radians = ');  % input specifications
```

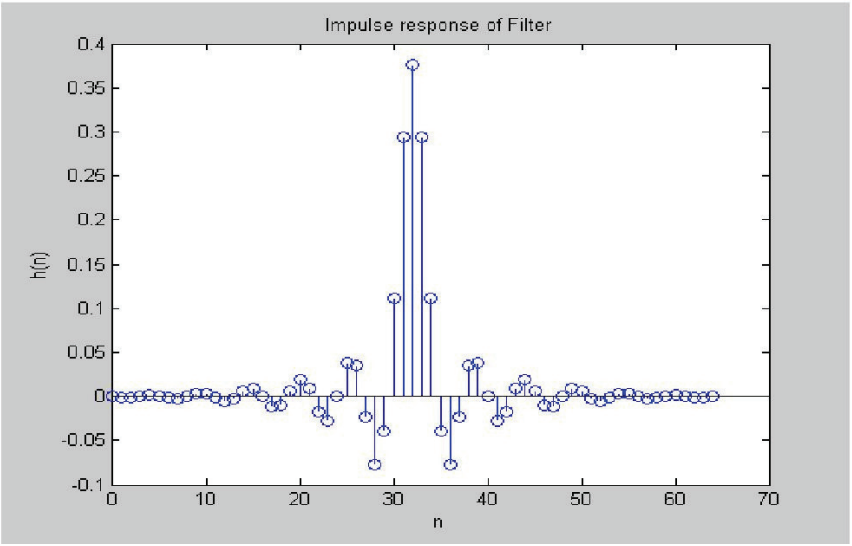
```
ws = input('Enter the Stop band edge in radians = ');
wt = ws-wp;
n1 = ceil (8*pi/wt);                                    % calculate the order of filter
N = n1 + rem(n1-1, 2);
Disp('order of the FIR filter N = ');
Disp(N);
wn = (hamming(N));                                      % calculate the filter coefficients

Wc1 = wp + wt/2 ;
Wc = Wc1/pi;                                            % calculate the cutoff frequency
Disp(' cut off frequency = ');
Disp(Wc);
h = fir1(N-1,Wc, wn);                                   % calculate the response of the filter
disp('Impulse Response of    FIR filter=');
disp(h);
fiqure(1);
freqz(h);                                               % plot the frequency response
figure(2);
n = 0:1:N-1;
stem(n,h);                                              % plot the impulse response
xlabel('n');
ylabel('h(n)');
title('Impulse Response of Filter');
```

OUTPUT:

```
enter the  Pass band edge in radians  =    0.375*pi
enter the  stop band edge in radians =    0.5*pi
order of the FIR filter N = 65
Impulse response of FIR filter =
  -0.0000      -0.0008  -0.0007   0.0004   0.0013    0.0006  -0.0014   -0.0022      0.0000
0.0032         0.0029  -0.0019  -0.0058  -0.0026    0.0056  0.0086      -0.0000  -0.0115  -
0.0101       0.0063   0.0190    0.0084    -0.0179   -0.0272  0.0000   0.0377       0.0346  -0.0231
```

-0.0769 -0.0398 0.1117 0.2938 0.3754 0.2938 0.1117 -0.0398 -0.0769 -0.0231
0.0346 0.0377 0.0000 -0.0272 -0.0179 0.0084 0.0190 0.0063 -0.0101 -0.0115
-0.0000 0.0086 0.0056 -0.0026 -0.0058 -0.0019 0.0029 0.0032 0.0000 -0.0022
-0.0014 0.0006 0.0013 0.0004 -0.0007 -0.0008 -0.0000





OUTCOME: Design and implementation of FIR filter for the given specifications is done and the desired frequency response is obtained

VIVA QUESTIONS WITH ANSWERS

1. What is a digital filter?

Digital filters are a very important part of DSP. Filters have two uses: signal separation and signal restoration. These problems can be attacked with either analog or digital filters. Digital filters are used for two general purposes: separation of signals that have been combined, and restoration of signals that have been distorted in some way.

2. What is the difference between FIR and IIR filter?

IIR is infinite and used for applications where linear characteristics are not of concern.FIR filters are Finite IR filters which are required for linear-phase characteristics.3. IIR is better for lower-order tapping, whereas the FIR filter is used for higher-order tapping.FIR filters are preferred over IIR because they are more stable, and feedback is not involved.IIR filters are recursive and used as an alternate, whereas FIR filters have become too long and cause problems in various applications.

3. What is Finite Impulse Response?

When length of h[n] is finite it is called finite impulse response.

4. How to find output of the FIR filter using FFT?

In FIR filter length of h[n] is finite. Output of the filter is always linear convolution of impulse response with the input of the signal. To find output i.e. to find LC by FFT

5. How to find output of the IIR filter using DFT / FFT?

Output of the filter is linear convolution of impulse response with the input of the signal. To find output means to find LC by DFT/FFT. Length of h[n] in IIR filter is infinite. So, DFT/FFT implementation of infinite length signals is not possible.

6. Why FFT is used to find output of FIR filter? Justify.

FFT produces fast results because in practical applications FFT algorithms are implemented using parallel processing techniques. Because in FFT calculations are done in parallel, FFT produces fast results.

7. What is frequency response?

Frequency response means magnitude response and phase response.

8. What are the limitations of filtering by FFT algorithms? Justify.

(i) NOT suitable for real time applications: FFT algorithms are implemented using parallel processing techniques. When FFT is used input is applied in parallel i.e., simultaneously. For real time applications entire input signal is not available. So FFT algorithms cannot be used. (ii) NOT suitable for Long Data Sequence. As the length of the input sequence increases, the no of stages in FFT will also increase proportionally and so the delay increases, processing time at each stage increases.

9. How to find output of FIR filter for long input sequence.

In FIR filter length of h[n] is finite. Output of the filter is always Linear Convolution of impulse response with the input of the signal. To find output of digital FIR filter FFT technique is used. But for Long data sequence, direct FFT technique is not suitable. For long data sequence, Overlap Add Method using FFT or Overlap Save Method using FFT is used.

10. How to find output of FIR filter for real time input signal?

In real time application entire input is not available and input signal has to be processed online. Length of input signal depends on application. It can be long sequence also. In FIR filter length of h[n] is finite. Output of the filter is always Linear Convolution of impulse response with the input of the signal. To find output of digital FIR filter, Overlap Add Method using FFT or Overlap Save Method using FFT is used.

EXPERIMENT NO-12:- DESIGN AND IMPLEMENTATION OF IIR FILTER

AIM: Design and implementation of IIR filter to meet given specifications.

OBJECTIVE:

1. To design the BUTTERWORTH filter to meet the given specification using the MATLAB functions BUTTORD and BUTTER
2. Bilinear transformation for analog-to-digital filter conversion using function "BILINEAR"
3. To design the CHEBYSHEV filter to meet the given specification using the MATLAB function cheb1ord and cheby1
4. Bilinear transformation for analog-to-digital filter conversion using function "BILINEAR
5. To verify the result by theoretical calculations.

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

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

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

ALGORITHM:

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

EXAMPLE:

Let's design an analog Butterworth lowpass filter.

Steps to design an analog Butterworth lowpass filter.

1. Get the pass band and stop band edge frequencies
2. Get the pass band and stop band ripples
3. Get the sampling frequency
4. From the given specifications find the order of the filter N.
5. Round off it to the next higher integer.
6. Find the transfer function H(s) for Ωc = 1rad/sec for the value of N.

7. Calculate the value of cutoff frequency Ωc
8. Find the transfer function Ha(s) for the above value of Ωc by substituting s→ (s/ Ωc) in H(s).

PROGRAM:DESIGN AND IMPLEMENTATION OF BUTTERWORTH FILTER

Design: step 1:

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

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

Step 2: T=1

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

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

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

Step 3: order of filter

$$N \geq \frac{log\frac{10^{0.1Ap}-1}{10^{0.1As}-1}}{2log\frac{\Omega_p}{\Omega_s}} \qquad N \geq \frac{log\frac{10^{0.301}-1}{10^{1.5}-1}}{2log\frac{2}{4.828}}$$

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

Step 4: cut off frequency

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

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

Step 5: poles

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

Where K=0 to N-1

Therefore $s_0 = -1.45 + j1.45$

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

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

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

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

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

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

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

PROGRAM:

```
clc;                                                    % clear screen
clear all;                                              % clear screen
close all;                                              % close all figure windows
fp = input('Enter the Pass band frequency in Hz = ' );   % input specifications
fs = input('Enter the Stop band frequency in Hz = ');
Fs = input('Enter the Sampling frequency in Hz = ');
Ap = input(' Enter the Pass band ripple in db:');
As = input('Enter theStop band ripple in db:');
wp=2*pi*Fp/ws;                                          % Analog frequency
ws=2*pi*Fs/ws;
Up = 2*tan(wp/2);% Prewrapped frequency
Us = 2*tan(ws/2);
[n,wn]= buttord (Up,Us,Ap,As,'s');                      %Calculate order and cutoff freq
Disp('order of the filter N =');
disp(N);
disp('Normalized cut off frequency = ');
```

```
disp(wn);
[num, den] = butter(n,wc,'s');                    % analog filter transfer
[b,a] = bilinear(num, den,1);          % conversion of analog filter to digital filter
Freqz(b,a,512,Fs);                     % frequency response of the filter
Printsys(b,a,'z');                     % print the H(z) equation obtained on screen
```

OUTPUT:

enter the Pass band edge frequency in Hz = 500

enter the stop band frequency in Hz = 750

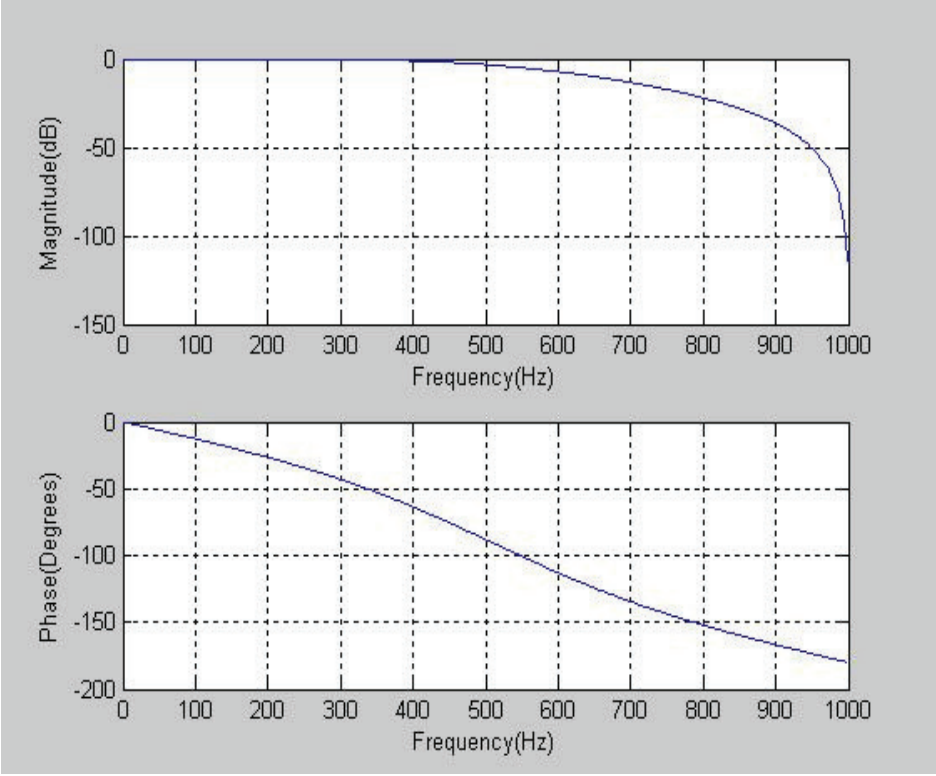
enter the sampling frequency in Hz = 2000

enter the pass band ripple n db = 3.01

enter the stop band attenuation in db =

15 order of the filter N = 2

Normalised cutoff frequency = 2.052



DESIGN OF CHEBYSHEV FILTER

step 1: $w_p = \frac{2\pi f_p}{F_s} = \frac{2\pi * 100}{4000} = 0.05\pi\ rad$

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

Step 2: T=1

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

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

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

Step 3: order of filter

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

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

Step 4: cut off frequency

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

Step 5: poles

$s_k = \sigma_k + j\Omega_k$

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

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

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

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

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

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

Therefore $s_1 = -0.4016 + j0.813$

$s_2 = -0.4016 - j0.813$

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

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

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

Step 6: analog to digital conversion

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

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

PROGRAM:

```
clc;                                                    % clear screen
clear all;                                              % clear work space
close all;                                              % close all figure windows
fp = input('Enter the Pass band frequency in Hz = ' );  % input specifications
fs = input('Enter the Stop band frequency in Hz = ');
Fs = input('Enter the Sampling frequency in Hz = ');
Ap = input(' Enter the Pass band ripple in db:');
As = input('Enter the Stop band ripple in db:');
wp=2*pi*Fp/ws;                                          % Analog frequency
ws=2*pi*Fs/ws;
Up = 2*tan(wp/2);                                       % Prewrapped frequency
Us = 2*tan(ws/2);
[N,wn]= cheb1ord (Up,Us,Ap,As,'s');                     %Calculate order and cutoff freq
Disp('order of the filter N =');
disp(N);
```

```
disp('Normalized cut off frequency =
'); disp(Wn);
[num, den] = cheby1(N, Ap,                  % analog filter transfer
Wn,'s'); [b,a] = bilinear(num,              %  conversion of analog filter to digital filter
den,1); Freqz(b,a,512,Fs);                  % frequency response of the filter
Printsys(b,a,'z');                          % print the H(z) equation obtained on screen
```

OUTPUT:

enter the Pass band edge frequency in Hz = 100

enter the stop band frequency in Hz = 500

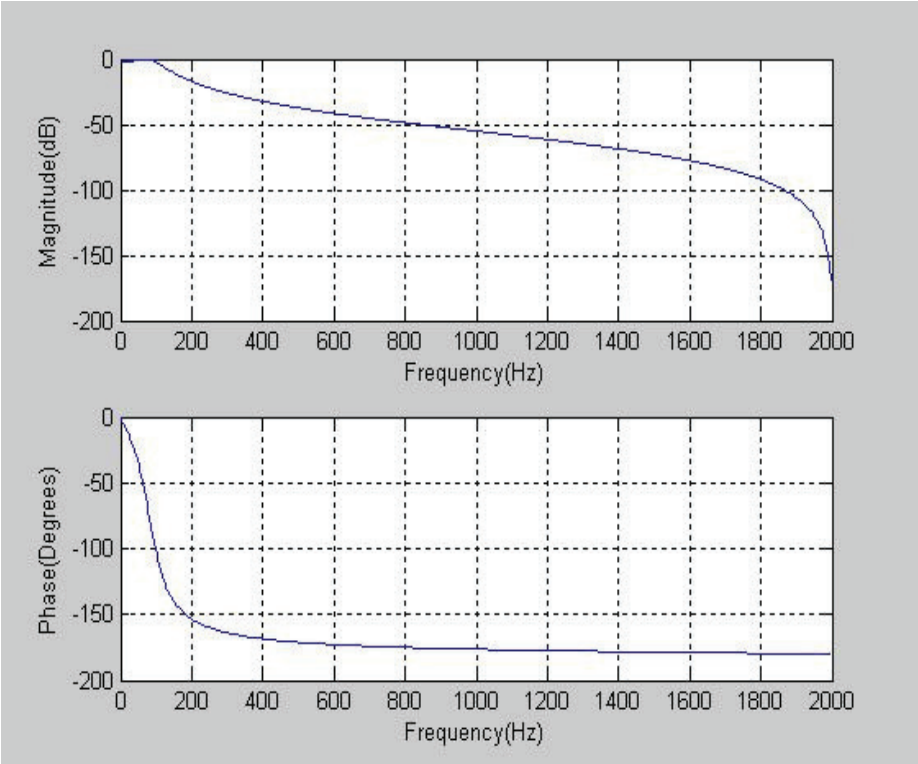
enter the sampling frequency in Hz = 4000

enter the pass band ripple n db = 2

enter the stop band attenuation in db =

20 order of the filter N = 2

Normalised cutoff frequency = 0.1574



OUTCOME:IIR filter for the given specifications is designed and the magnitude and phase plots are seen and verified

VIVA QUESTIONS WITH ANSWER

1) What is Infinite Impulse Response (IIR) filter?

If the impulse response of the system is of infinite duration, the system is said to be IIR filter system.

2) How to find output of IIR filter for real time input signal?

In real time application entire input is not available and input signal has to be processed online. Length of input signal depends on application. It can be long sequence also. In IIR filter length of h[n] is infinite. Output of the filter is always Linear Convolution of impulse response with the input of the signal. To find output of digital IIR filter, Overlap Add Method using FFT or Overlap Save Method using FFT cannot be used. Output of digital IIR filter is calculated using difference equation recursively.

3) How to find output of IIR filter for long input sequence?

In IIR filter length of h[n] is infinite. Output of the filter is always Linear Convolution of impulse response with the input of the signal. To find output of digital IIR filter, Overlap Add Method using FFT or Overlap Save Method using FFT cannot be used. Output of digital IIR filter is calculated using difference equation recursively.

4) Explain how to find output of digital IIR filter in real time application.

In real time applications, output of IIR filter can be obtained by evaluating difference equation.

5) Why IIR filters are called as recursive filters?

In IIR filter output depends on output values. e.g. y[n] = x[n] + x[n-1] + y[n] + y[n-1]. Therefore IIR filters are also called as Recursive Filters.

6) Explain how to find output of digital IIR filter in real time application.

In real time applications, output of IIR filter can be obtained byevaluating difference equation.

7) What are the advantages of IIR filters (compared to FIR filters)?

IIR filters can achieve a given filtering characteristic using less memory and calculations than a similar FIR filter.

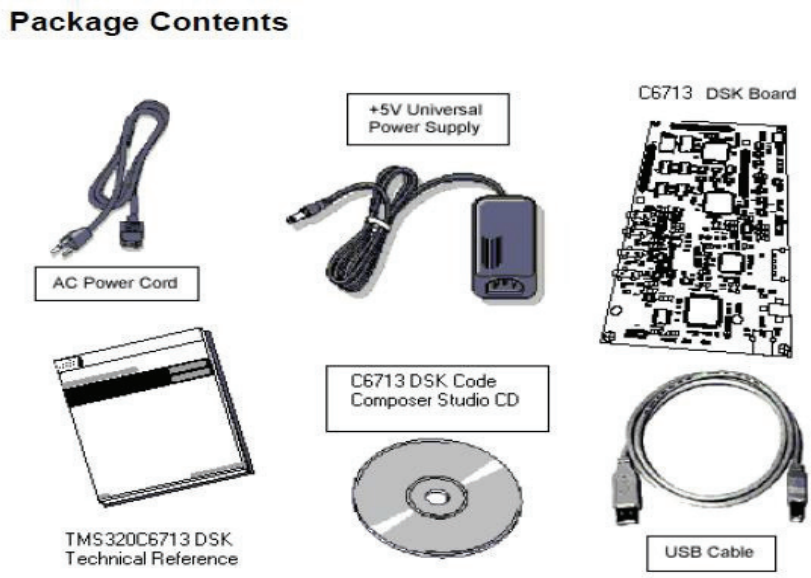
8) What are the disadvantages of IIR filters (compared to FIR filters)?

They are more susceptible to problems of finite-length arithmetic, such as noise generated by calculations, and limit cycles. They are harder to implement using fixed-point arithmetic. They don't offer the computational advantages of FIR filters for multirate (decimation and interpolation) application

PART-B

INTRODUCTION TO DSP PROCESSORS (TMS320C6713DSK)

The C6713™ DSK builds on TI's industry-leading line of low cost, easy-to-use DSP Starter Kit (DSK) development boards. The high-performance board features the TMS320C6713 floating-point DSP. Capable of performing 1350 million floating-point operations per second (MFLOPS), the C6713 DSP makes the C6713 DSK the most powerful DSK development board. The DSK is USB port interfaced platform that allows efficiently developing and testing applications for the C6713. The DSK consists of a C6713-based printed circuit board that will serve as a hardware reference design for TI's customers' products. With extensive host PC and target DSP software support, including bundled TI tools, the DSK provides ease-of-use and capabilities that areattractive to DSP engineers.



The C6713 DSK has a TMS320C6713 DSP onboard that allows full-speed verification of code with Code Composer Studio. The C6713 DSK provides:

- A USB Interface
- SDRAM and ROM
- An analog interface circuit for Data conversion (AIC)
- An I/O port
- Embedded JTAG emulation support

Connectors on the C6713 DSK provide DSP external memory interface (EMIF) and peripheral signals that enable its functionality to be expanded with custom or third party daughter boards. The DSK provides a C6713 hardware reference design that can assist you in the development of your own C6713-based products. In addition to providing a reference for interfacing the DSP to various types of memories and peripherals, the design also addresses power, clock, JTAG, and parallel peripheral interfaces.

The C6713 DSK includes a stereo codec. This analog interface circuit (AIC) has the following characteristics:

- High-Performance Stereo *Codec*
  - 90-dB SNR Multibit Sigma-Delta ADC (A-weighted at 48 kHz)
  - 100-dB SNR Multibit Sigma-Delta DAC (A-weighted at 48 kHz)
  - 1.42 V – 3.6 V Core Digital Supply: Compatible With TI C54x DSP
- Core Voltages
  - 2.7 V – 3.6 V Buffer and Analog Supply: Compatible Both TI C54x
- DSP Buffer Voltages
  - 8-kHz – 96-kHz Sampling-Frequency Support
- Software Control Via TI McBSP-Compatible Multiprotocol Serial Port
  - I 2 C-Compatible and SPI-Compatible Serial-Port Protocols
  - Glueless Interface to TI McBSPs
- Audio-Data Input/Output Via TI McBSP-Compatible Programmable Audio Interface
  - I 2 S-Compatible Interface Requiring Only One McBSP for both ADC and DAC
  - Standard I 2 S, MSB, or LSB Justified-Data Transfers
  - 16/20/24/32-Bit Word Lengths

The 6713 DSK is a low-cost standalone development platform that enables customers to evaluate and develop applications for the TI C67XX DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market. The DSK uses the 32-bit EMIF for the SDRAM (CE0) and daughter card expansion interface (CE2 and CE3). The Flash is attached to CE1 of the EMIF in 8-bit mode.

An on-board AIC23 codec allows the DSP to transmit and receive analog signals. McBSP0 is used for the codec control interface and McBSP1 is used for data. Analog audio I/O is done through four 3.5mm audio jacks that correspond to microphone input, line input, line output and headphone output. The codec can select the microphone or the line input as the active input. The analog output is driven to both the line out (fixed gain) and headphone (adjustable gain) connectors. McBSP1 can be re-routed to the expansion connectors in software.

A programmable logic device called a CPLD is used to implement glue logic that ties the board components together. The CPLD has a register based user interface that lets the user configure the board by reading and writing to the CPLD registers. The registers reside at the midpoint of CE1.The DSK includes 4 LEDs and 4 DIP switches as a simple way to provide the user with interactive feedback. Both are accessed by reading and writing to the CPLD registers.

An included 5V external power supply is used to power the board. On-board voltage regulators provide the 1.26V DSP core voltage, 3.3V digital and 3.3V analog voltages. A voltage supervisor monitors the internally generated voltage, and will hold the boards in reset until the supplies are within operating specifications and the reset button is released. If desired, JP1 and

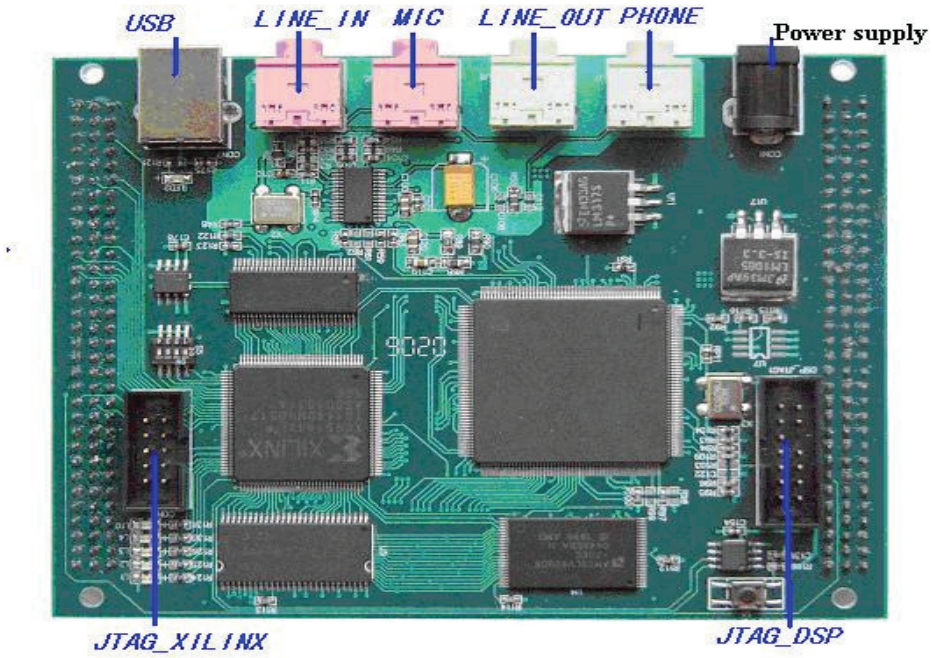
JP2 can be used as power test points for the core and I/O power supplies. Code Composer communicates with the DSK through an embedded JTAG emulator with a USB host interface. The DSK can also be used with an external emulator through the external JTAG connector.

TMS320C6713 DSP Features

- Highest-Performance Floating-Point Digital Signal Processor (DSP):
- Eight 32-Bit Instructions/Cycle
- 32/64-Bit Data Word
- 300-, **225**-, 200-MHz (GDP), and **225-**, 200-, 167-MHz (PYP) Clock Rates
- 3.3-, 4.4-, 5-, 6-Instruction Cycle Times
- 2400/1800, 1800/1350, 1600/1200, and 1336/1000 MIPS /MFLOPS

- Rich Peripheral Set, Optimized for Audio
- Highly Optimized C/C++ Compiler
- Extended Temperature Devices Available
- Advanced Very Long Instruction Word (VLIW) TMS320C67x™ DSP Core
- Eight Independent Functional Units:
- Two ALUs (Fixed-Point)
- Four ALUs (Floating- and Fixed-Point)
- Two Multipliers (Floating- and Fixed-Point)
- Load-Store Architecture With 32 32-Bit General-Purpose Registers
- Instruction Packing Reduces Code Size
- All Instructions Conditional
- Instruction Set Features
- Native Instructions for IEEE 754
- Single- and Double-Precision
- Byte-Addressable (8-, 16-, 32-Bit Data)
- 8-Bit Overflow Protection
- Saturation; Bit-Field Extract, Set, Clear; Bit-Counting; Normalization
- L1/L2 Memory Architecture
- 4K-Byte L1P Program Cache (Direct-Mapped)
- 4K-Byte L1D Data Cache (2-Way)
- 256K-Byte L2 Memory Total: 64K-Byte L2 Unified Cache/Mapped RAM, and 192K-Byte Additional L2 Mapped RAM
- Device Configuration
- Boot Mode: HPI, 8-, 16-, 32-Bit ROM Boot
- Endianness: Little Endian, Big Endian
- 32-Bit External Memory Interface (EMIF)
- Glue less Interface to SRAM, EPROM, Flash, SBSRAM, and SDRAM
- 512M-Byte Total Addressable External Memory Space
- Enhanced Direct-Memory-Access (EDMA) Controller (16 Independent Channels)
- 16-Bit Host-Port Interface (HPI)

- Two Multichannel Audio Serial Ports (McASPs)
- Two Independent Clock Zones Each (1 TX and 1 RX)
- Eight Serial Data Pins Per Port: Individually Assignable to any of the Clock Zones
- Each Clock Zone Includes:
- Programmable Clock Generator
- Programmable Frame Sync Generator
- TDM Streams From 2-32 Time Slots
- Support for Slot Size:8, 12, 16, 20, 24, 28, 32 Bits
- Data Formatter for Bit Manipulation
- Wide Variety of I2S and Similar Bit Stream Formats
- Integrated Digital Audio Interface Transmitter (DIT) Supports:
- S/PDIF, IEC60958-1, AES-3, CP-430 Formats
- Up to 16 transmit pins
- Enhanced Channel Status/User Data
- Extensive Error Checking and Recovery
- Two Inter-Integrated Circuit Bus (I2C Bus™) Multi-Master and Slave
- Interfaces
- Two Multichannel Buffered Serial Ports:
- Serial-Peripheral-Interface (SPI)
- High-Speed TDM Interface
- AC97 Interface
- Two 32-Bit General-Purpose Timers
- Dedicated GPIO Module With 16 pins (External Interrupt Capable)
- Flexible Phase-Locked-Loop (PLL) Based Clock Generator Module
- IEEE-1149.1 (JTAG) Boundary-Scan-Compatible



INTRODUCTION TO CODE COMPOSER STUDIO

Code Composer is the DSP industry's first fully integrated development environment (IDE) with DSP-specific functionality. With a familiar environment liked MS-based C++TM, Code Composer lets you edit, build, debug, profile and manage projects from a single unified environment. Other unique features include graphical signal analysis, injection/extraction of data signals via file I/O, multi-processor debugging, automated testing and customization via a C-interpretive scripting language and much more.

Code composer features include:

- IDE
- Debug IDE
- Advanced watch windows
- Integrated editor
- File I/O, Probe Points, and graphical algorithm scope probes
- Advanced graphical signal analysis
- Interactive profiling
- Automated testing and customization via scripting
- Visual project management system

- Compile in the background while editing and debugging
- Multi-processor debugging
- Help on the target DSP

PROCEDURE TO SETUP EMULATOR

1. Open the "Setup CCStudio v3.3"
2. Choose c67xx in the "Family".
3. Choose AHxds510usb emulator in the "Platform".
4. Choose little in the "Endianness".
5. Now you are left with two options under Available Factory Boards, Choose C671X AHXDS510 USB Emulator, right click and "Add to system…"
6. Now the Emulator and the processor both are selected under "system configuration".
7. Choose file and click on "save".
8. Choose file and click on "exit" , Click on yes.
9. Go to Debug and select the option connect.
10. Now Target is connected.

PROCEDURE TO CREATE NEW PROJECT

- To create project, Go to Project and Select New.
- Give project name and click on finish.

  Note: Location must be c:\CCStudio_v3.3\MyProjects).
- Click on File ⇨ New ⇨ Source File, To write the Source Code.
- Enter the source code and save the file with "**.C"**extension.
- Right click on source, Select add files to project and Choose "**.C**"file Saved before.
- Right Click on libraries and select add files to Project and choose C:\CCStudio_v3.3\C6000\cgtools\lib\rts6700.lib and click open
- Go to Project to Compile. Build, Rebuild All
- Go to file and load program and load **".out"**file into the board.
- Go to Debug and click on run to run the program.
- Observe the output in output window.
- To see the Graph go to View and select time/frequency in the Graph, and give the correct Start address provided in the program, Display data can be taken as per user.

EXPERIMENT NO 1: LINEAR CONVOLUTION

AIM: Linear Convolution of the two given sequences

OBJECTIVE: After completing this lab, student will be able to find the linear convolution of two given sequences by implementing the C-code on the TMS320C6713 DSP processor.

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

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

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

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

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

In linear convolution length of output sequence is,Length (y(n)) = length(x(n)) + length(h(n)) – 1

PROGRAM: LINEAR CONVOLUTION

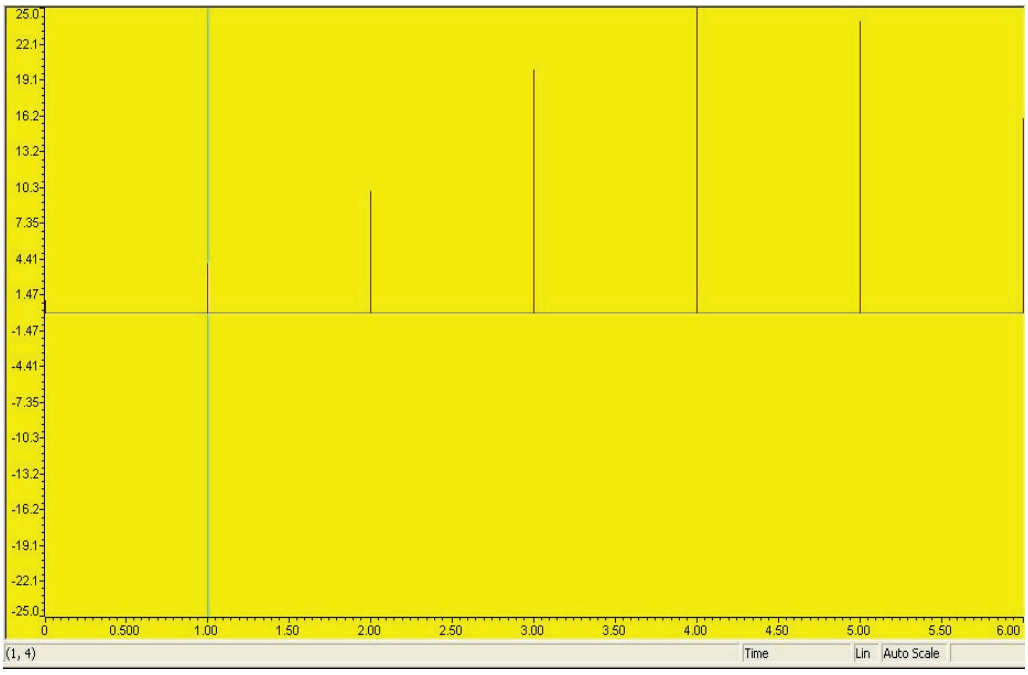
```
#include<stdio.h>
#include<math.h>
float x[4] = {1, 2, 3, 4};          /* first sequence*/
float h[4] = {1, 2, 3, 4};          /* Second sequence*/
float y[7];
voidmain()
{
intxlen=4;                          /*Lengh of i/p samples sequence*/
inthlen=4;
int N = xlen + hlen - 1;            /*Lengh of output sequence */

inrn,k;
```

```
for(n=0; n<N;i++)      /* loop to calculate the output according to convolution equation*/
        {
        y[n]=0;
        for(k=0; k<hlen; k++)
        {
        if(((n – k) >= 0) & ((n – k) <xlen))
        y[n] = y[n] + h[k]* x[n - k];
        }
        printf("%f \t",y[i]);
        }
}
```

OUTPUT:

1.000000 4.000000 10.000000 20.000000 25.000000 24.000000 16.000000

GRAPH:



OUTCOME: Linear convolution of the sequence is found and the code is implemented on the DSP processor to verify the results.

VIVA QUESTIONS WITH ANSWER

1. What do you mean by invariant?

   Invariant means, Not variant, i.e., doesn't change.

2. What is a linear phase filter?

   "Linear Phase" refers to the condition where the phase response of the filter is a linear (straight- line) function of frequency

3. What is the advantage of Linear Phase?

   This results in the delay through the filter being the same at all frequencies. Therefore, the filter does not cause "phase distortion" or "delay distortion".

4. What is a causal and non-causal system?

   A system is said to be causal if the output does not depends on future inputs and outputs. A system is said to be non-causal if the output does not depends on future inputs and outputs.

5. What is the importance of causality?

   Only the causal systems can be realized in real time. Since non-causal systems depends on future inputs and outputs they cannot be realized in real time.

EXPERIMENT NO 2: CIRCULAR CONVOLUTION

AIM: To implement circular convolution of two sequences.

OBJECTIVE: To find the circular convolution of the given sequence by implementing C code on TMS320C6713 DSP processor.

THEORY: Let x1(n) and x2(n) are finite duration sequences both of length N with DFT's X1(k) andX2(k). Convolution of two given sequences x1(n) and x2(n) is given by the equation,

$$x3(n) = IDFT[X3(k)]$$

$$X3(k) = X1(k)\ X2(k)$$

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)x_2(n-m)\ N$$

PROGRAM: CIRCULAR CONVOLUTION

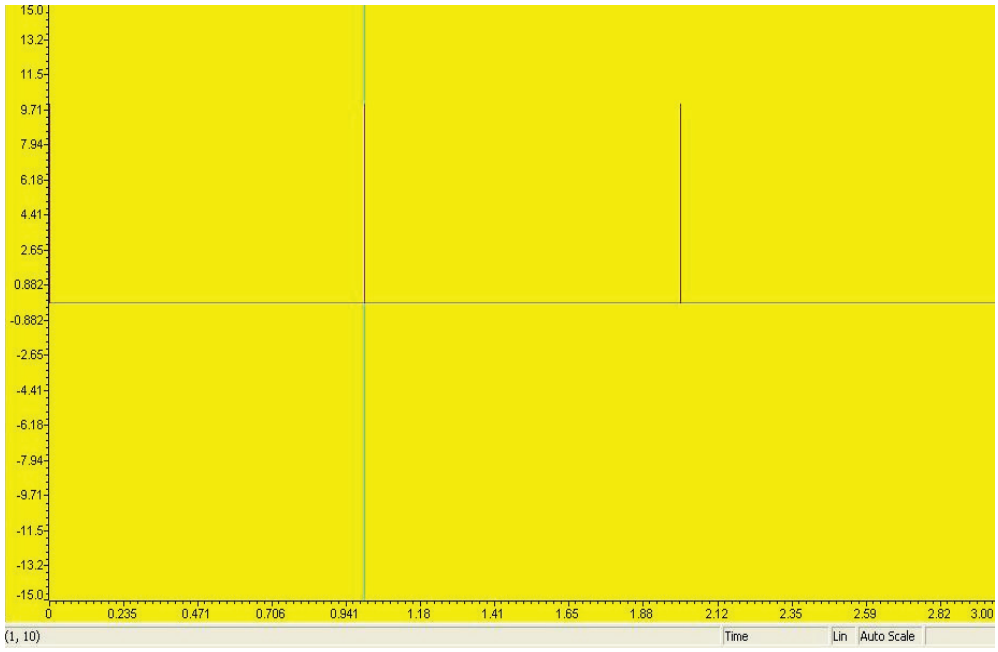
```
#include<stdio.h>
#include<math.h>
float x[4] = {4, 3, 2, 1};                    /* First sequence*/
float h[4] = {1, 1, 1, 1};                    /* second sequence*/
float y[4];
void main()
{
int N = 4;
intn,k, i;
for(n=0; n<N; i++)                    /* loop to calculate circular convolution */
        {
        y[n]=0;
        for(k=0; k<N; k++)
        {
        i = (n- k)%N;
        if (i<0)
        i= i + N;
```

```
        y[n] = y[n] + h[k] * x[i];
        }
        Printf("%f \t",y[n]);
        }
}
```

OUTPUT:

10.000000 10.000000 10.000000 10.000000

GRAPH:



OUTCOME: Circular convolution of the sequence is found and the code is implemented on the DSP processor to verify the results.

VIVA QUESTIONS WITH ANSWER

1. How to find Circular Convolution using FFT?

To find Circular Convolution of x[n] and h[n] using FFT, (i) Select N Let N = max(L,M) where L is the length of x[n] and M is length of h[n], (ii) Append x[n] by (N-L) zeros and Append h[n] by (N-M) zeros

2. List the features of General purpose MPU and DSP

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

Data manipulation

Math calculations

Control systems

They run large blocks of software

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

DSPs are single-minded, dedicated to:

Perform mathematical calculations

Small blocks of software

Have a predictable execution time

Real-time only

Could assist a general-purpose host MPU

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

MPU	DSP
1. General purpose	1. Arithmetic
2. Fixed internal format	2. Varying internal format
3. Single memory access	3. Multiple memory access
4. General addressing mode	4. Special addressing mode
5. Very large external memory	5. Very large internal memory

EXPERIMENT NO 3: N-POINT DFT

AIM: To compute n-point DFT of a given sequence and to plot

OBJECTIVE: To find the N point DFT of the sequence using TMS320C6713 DSP processor.

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

The sequence of N complex numbers $x0,...,xN-1$ is transformed into the sequence of N complex numbers $X0, ..., XN-1$ by the DFT according to the formula:

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

PROGRAM: N POINT DFT

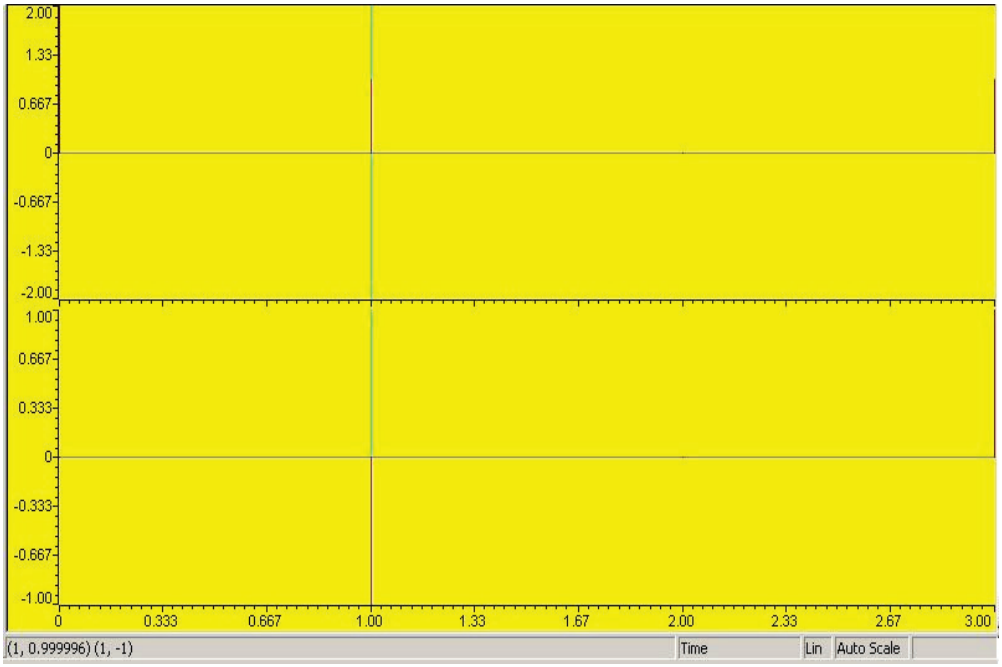
```
#include<stdio.h>
#include<math.h>
float y[8];
float x[4] = {1, 1, 0, 1};                    /* input sequence */
float w;
voidmain()
{
int n, k, k1, N = 4, xlen = 4;
for(k=0; k < 2*N; k+2)                        /*loop to calculate N point DFT */
        {
        y[k]=0;
        y[k+1]=0;
        k1 = k/2;
        for(n=0; n<xlen; n++)
        w = - 2*3.14*k1*n/N;                  /* calculation of twiddle factor*/
```

```
        y[k] = y[k] + x[n]* cos(w);                    /* real components of DFT*/
        y[k+1] = y[k+1] + x[n]*sin(w);                 /* imaginary components of DFT*/
        }
        Printf ("%f + j %f \t",y[k], y[k+1]);
        }
}
```

OUTPUT:

2.000000 0.000000 0.999996 -1.000000 0.000000 0.000007 1.000011 1.000000

GRAPH:



OUTCOME: DFT of the sequence is found and the code is implemented on the DSP processor to verify the results.

VIVA QUESTIONS WITH ANSWER

1. What is an FFT "radix"?

The "radix" is the size of FFT decomposition. In the example above, the radix was 2. For single-radix FFT's, the transform size must be a power of the radix. In the example above, the size was 32, which is 2 to the 5th power.

2. What is an "in place" FFT?

An "in place" FFT is simply an FFT that is calculated entirely inside its original sample memory. In other words, calculating an "in place" FFT does not require additional buffer memory (as some FFT's do.)

3. What is "bit reversal"?

"Bit reversal" is just what it sounds like: reversing the bits in a binary word from left to write. Therefore the MSB's become LSB's and the LSB's become MSB's.

4. But what does Bit-reversal have to do with FFT's?

The data ordering required by radix-2 FFT's turns out to be in "bit reversed" order, so bit-reversed indexes are used to find order of input and output.

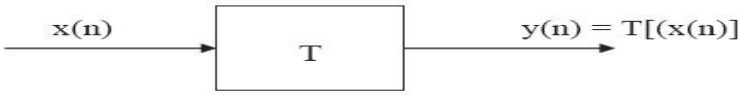
EXPERIMENT 4: IMPULSE RESPONSE

AIM: To find Impulse response of a first order and second order system.

OBJECTIVE: To find the impulse response for the system using TMS320C6713 DSP processor.

THEORY:

A discrete time system performs an operation on an input signal based on predefined criteria to produce a modified output signal. The input signal x(n) is the system excitation, and y(n) is the system response. The transform operation is shown as,



The convolution sum can be represented by, y(n) = x(n) * h(n)

For Example let's find out an impulse response of a difference equation.

The general form of difference equation is,

$$y(n) = \sum_{k=1}^{m} a_k\, y(n-k) + \sum_{\square=0}^{\square} b_k\, x(n-k)$$

Find out the impulse response of second order difference equation.

PROGRAM: N POINT DFT

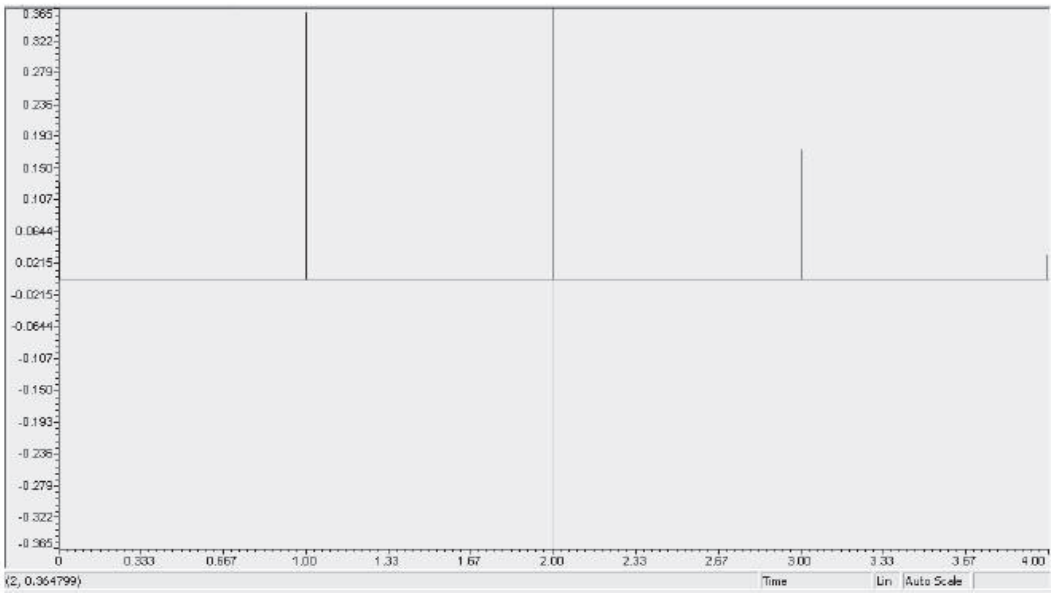
```
#include<stdio.h>
#include<math.h>
float x[60], y[60];
voidmain()
{
        float a0, a1, a2, b0, b1, b2;
        int i, j, N=7;
        b0 = 0.1311, b1 = 0.2622, b2 = 0.1311;          /* coefficients of x(n)*/
        a0 = 1, a1 = -0.7478, a2 = 0.2722;              /* coefficients of y(n) */
        x[0]= 1;
        for (i=1; i< N; i++)
        x[1] = 0;
```

```
        for(j=0;j<N;j++)                    /* loop to calculate impulse response */
        {
        y[j] = b0 * x[j];
        if(j>0)
        y[j] = y[j] + b1* x[j-1] – a1 * y[j-1];
        if((j – 1)>0)
        y[j] = y[j] + b2 * x[j-2] – a2 * y[j-2];
        }
        Printf ("%f \t",y[j]);
        }
}
```

OUTPUT:

0.131100 0.360237 0.364799 0.174741 0.031373

GRAPH:



OUTCOME: Impulse response and the output response of the first order and second order system is found using TMS320C6713 DSK.

VIVA QUESTIONS WITH ANSWER

1. List the different types of structures for realization of IIR filters

The different types of structures for realizing IIR systems are i) Direct form – I, ii) Direct form- II iii) Cascade form, iv) Parallel form

2. What are the factors that influence the choice of structure for realization of an LTI system?

The factors that influence the choice of realization structure are computational complexity, memory requirements, finite word length effects, parallel processing and pipelining of computations

3. List the different types of structures for realizing FIR systems?

The different types of structures for realizing FIR systems are i) Direct form realization, ii) Cascade realization, iii)Linear phase realization.

4. What is Recursive system?

If the output y(n) at time of a system depends on past output values then the system is called Recursive system. The recursive systems are functions of past outputs, present and past inputs.

5. What is Non-recursive system?

If the output y(n) at time of a system does not depends on past output values then the system is called Recursive system. The recursive systems are functions of present and past inputs.

EXPERIMENT NO 5: FIR FILTER

AIM: Realization of fir filter (any type) to meet given specifications. The input can be a signal from function generator/speech signal

OBJECTIVE: To design and realize FIR filter using CODEC and TMS3206713 DSP processor.

THEORY: The FIR filters are of non-recursive type, whereby the present output sample is depending on the present input sample and previous input samples. The transfer function of a FIR causal filter is given by,

$$H(Z) = \sum_{n=0}^{N-1} h(n) Z^{-n}$$

Where h(n) is the impulse response of the filter.

The Fourier transform of h(n) is

$$H(e^{jw}) = \sum_{n=0}^{N-1} h(n) e^{-j\omega n}$$

In the design of FIR filters most commonly used approach is using windows.The desired frequency response $H_d(e^{jw})$ of a filter is periodic in frequency and can be expanded in Fourier series. The resultant series is given by,

$$h_d(n) = (1/2\pi) \int_{-\pi}^{\pi} H(ejw) e^{jwn} dw$$

This is known as Fourier series coefficients having infinite length. One possible way of obtaining FIR filter is to truncate the infinite Fourier series at n = ± [(N-1)/2]Where N is the length of the desired sequence.

The Fourier coefficients of the filter are modified by multiplying the infinite impulse response with a finite weighing sequence w(n) called a window.

Where w(n)= w(-n) ≠ 0 for |n| ≤ [(N-1)/2]

= 0 for |n| > [(N-1)/2]

After multiplying w(n)with $h_d(n)$, we get a finite duration sequence h(n) that satisfies the desired magnitude response,

h(n) = $h_d(n)$ w(n) for |n| ≤ [(N-1)/2]

= 0 for |n| > [(N-1)/2]

The frequency response $H(e^{jw})$ of the filter can be obtained by convolution of $H_d(e^{jw})$ and $W(e^{jw})$ is given by,

$H(e^{jw}) = (1/2\pi) \int_{-\pi}^{\pi} Hd(ej\theta)\, W(ej(w-\theta))\, d\theta$

$H(e^{jw}) = H_d(e^{jw}) * W(e^{jw})$

PROGRAM: FIR FILTER

#include <stdio.h> #include

"c6713dsk.h" #include

"master.h" #include

"aic23cfg.h" #include

"dsk6713_aic23.h" #define

FilLen 32

// LowPass Filter

/*float FilCo[32]= { -0.0006,-0.0056,-0.0129,-0.0193,-0.0176,-0.0043, 0.0156, 0.0282, 0.0190,-0.0129,0.0483,-0.0545,-0.0065, 0.0927, 0.2064, 0.2822, 0.2822, 0.2064, 0.0927,-0.0065,-0.0545,-0.0483,-0.0129, 0.0190, 0.0282, 0.0156,-0.0043,-0.0176,-0.0193,-0.0129,-0.0056,-0.0006};*/

// HighPass Filter

float FilCo[32] = { -0.0128,-0.0007,0.0068,0.0150, 0.0173, 0.0091,-0.0082,-0.0260,-0.0317,-0.0160, 0.0200, 0.0611,0.0808,0.0481,-0.0803,-0.5875, 0.5875, 0.0803,-0.0481,-0.0808,-0.0611,-0.0200,0.0160,0.0317, 0.0260, 0.0082,-0.0091,-0.0173,-0.0150,-0.0068, 0.0007, 0.0128};

DSK6713_AIC23_Config config = { \

0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Left line input channel volume */ \

0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume */\

0x00ff, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */ \

0x00ff, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */ \

0x0010, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */ \

0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */ \

0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */ \

0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */ \

```
0x008c, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */ \
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
DSK6713_AIC23_CodecHandle hCodec;
Int32 LInc = 4;
Int32 RInc = 3;
Int32 LData;
Int32 RData;
Int32 InChannel =0;
Int32 OtChannel =0;
Int32 LIndex = 0;
Int32 RIndex = 0;
Int32 LDataArr[128];
Int32 RDataArr[128];
Int32 LWP=0;
Int32 LRP=100;
Int32 RWP=0;
Int32 RRP=100;
float LPrBuf[150];
float RPrBuf[150];
floatFilOut= 0;
main()
{
int i;
LED=0x0;
for( i = 0 ; i < 100; i++ ) LPrBuf[i]=0.0;                    // Filter Initialization
for( i = 0 ; i < 100; i++ ) RPrBuf[i]=0.0;
hCodec = DSK6713_AIC23_openCodec(0, &config);                 // Initialize codec
IRQ_globalEnable();
IRQ_enable(IRQ_EVT_RINT1);
IRQ_enable(IRQ_EVT_XINT1); }
void led()
```

```
{ static int cc = 1;
LED = cc;
if (cc == 0x03) cc = 0x05;                                  // To Shift Pattern
else if (cc == 0x05) cc = 0x06;
else if (cc == 0x06) cc = 0x03;
else cc = 0x03; }
setpll200M(){
}
void read(){
int i = 0;
if (InChannel ==0){
InChannel =1;
LData=MCBSP_read(DSK6713_AIC23_DATAHANDLE);          // Process Left Channel
for( i = 0; i <= FilLen-2; i++) LPrBuf[i] = LPrBuf[i+1];
LPrBuf[FilLen-1] = (float) LData;
FilOut = 0.0;
for( i = 0 ; i <= FilLen-1; i++ ) FilOut += (FilCo[i] * LPrBuf[i]);
LData = (int) (FilOut/16);
LDataArr[LWP++] = LData;
if (LWP >= 128) LWP =0;
}
else{
InChannel =0;
RData=MCBSP_read(DSK6713_AIC23_DATAHANDLE);          // Process Right Channel
RDataArr[RWP++] = RData;
if (RWP >= 128) RWP =0;
}
}
void write(){
if (OtChannel == 1){
MCBSP_write(DSK6713_AIC23_DATAHANDLE,LDataArr[LRP++]); if
(LRP >= 128) LRP = 0;
```

```
OtChannel = 0;
}
Else
{
MCBSP_write(DSK6713_AIC23_DATAHANDLE,RDataArr[RRP++]);
if (RRP >= 128) RRP = 0;
OtChannel = 1;
}
}
```

OUTCOME: Thus FIR filter is realized for given specification and the output is verified through CRO for a given input signal.

VIVA QUESTIONS WITH ANSWER

1. What are Advantages of FIR Filters-?

   The advantages of FIR filters are

   1) They can easily be designed to be "linear phase"
   2) They are suited to multi-rate applications.
   3) They have desirable numeric properties.
   4) They can be implemented using fractional arithmetic.
   5) They are simple to implement.

2. What are the disadvantages of FIR Filters (compared to IIR filters)?

   Compared to IIR filters, FIR filters sometimes have the disadvantage that they require more memory and/or calculation to achieve a given filter response characteristic.

3. Explain the concept of Linear Phase and its importance.

   If the Phase Response is Linear the output of the Filter during pass-band is delayed input. II. If the phase Response is non Linear the output of the filter during pass-band is distorted one. The linear Phase characteristic is important when the phase distortion is not tolerable. FIR Filter can be designed with linear phase characteristic. In application like data

transmission, speech processing etc. phase distortion cannot be tolerated and here linear phase characteristic of FIR filter is useful.

4. What is the role of window in the design of FIR filter? Name the few types of windows.
FIR filter is designed by truncating infinite samples of hd[n] by using window function. Examples of window function include, Hamming window,Bartlet Window, Hanning window, Blackman window etc.

5. Why rectangular window is not preferred for FIR filter design?

Rectangular window function has As = 21 db which is very small compared to other window function. Larger value of as desired.

6. Explain how to find output of digital FIR filter in real time application.

In real time applications, output of FIR filter is obtained using overlap add method / overlap save method.

EXPERIMENT 6: NOISE REMOVAL

AIM: Noise removal in a given mixed signal.

OBJECTIVE: After completing this lab, student will be able to learn the technique for the removal of nose from the mixed signals.

THEORY: In the real time systems every signal will get corrupted by the noise. To analyze the noise we need to add the noise, a noise will be generated and added to the signal. A noise of 3KHz is added to a tone of 400KHz then the noise is removed by using an high pass filter.

PROGRAM: NOISE REMOVAL

```
#include <stdio.h> #include
"c6713dsk.h" #include
"master.h" #include
"aic23cfg.h" #include
"dsk6713_aic23.h" #define
FiltLen 47
#define SINE_TABLE_SIZE 48
float PrNoise[600]={0.3509,0.1118,0.3631,0.2673,0.2826,0.1909,0.2901,0.1378,0.1270,
0.1798, 0.2925,0.3730,0.1307,0.1380,0.2859,0.1738,0.3278,0.0355,0.3177,0.2905,
0.1473,0.2380,0.2112,0.2662,0.3228,0.0264,0.2991,0.3027,0.1220,0.1676,0.2360,0.2942,
0.3058,0.3013,0.1423,0.1478,0.3479,0.2017,0.3278,0.1164,0.2747,0.2752,0.4239,0.0666,
0.2130,0.2124,0.0124,0.3306,0.3134,0.1420,0.2837,0.0446,0.2438,0.1069,0.1902,0.3497,
0.0962,0.1404,0.2046,0.1419,0.3231,0.3962,0.0549,0.0875,0.2742,0.1418,0.3797,0.3756,
0.2441,0.3271,0.1456,0.2332,0.3789,0.2250,0.3798,0.3153,0.2295,0.2950,0.3924,0.0982,
0.1493,0.3649,0.1159,0.2095,0.3088,0.3656,0.2539,0.0606,0.0548,0.2847,0.2610,0.4005,
0.2985,0.2459,0.1789,0.0824,0.3774,0.2256,0.0309,0.2949,0.2133,0.0136,0.3288,0.0923,
0.0491,0.0178,0.1205,0.1107,0.3042,0.0761,0.1512,0.1302,0.3434,0.1384,0.2454,0.1873,
0.2804,0.0407,0.4164,0.3858,0.0678,0.2275,0.1430,0.2304,0.0500,0.1793,0.1887,0.2067,
0.1776,0.0374,0.2449,0.2093,0.0772,0.1982,0.1787,0.2021,0.2916,0.2557,0.3304,0.0184,
0.0892,0.1823,0.3762,0.2984,0.1576,0.1120,0.0088,0.3026,0.2274,0.1223,0.2151,0.1131,
0.3248,0.1441,0.3475,0.0471,0.2749,0.1925,0.1708,0.2431,0.2975,0.0634,0.3891,0.0372,
0.1486,0.0943,0.3835,0.0355,0.2320,0.2466,0.2428,0.3175,0.0105,0.2891,0.1764,0.2621,
```

0.0814,0.2239,0.3074,0.4196,0.2065,0.0613,0.0321,0.2495,0.3044,0.2513,0.1193,0.3635,
0.2031,0.3801,0.0303,0.0733,0.3001,0.3507,0.2985,0.1186,0.0656,0.3612,0.3397,0.1294,
0.1102,0.1194,0.3025,0.2892,0.1845,0.1956,0.1073,0.2981,0.3682,0.2311,0.2244,0.2099,
0.0990,0.2220,0.2894,0.2813,0.2316,0.0653,0.2872,0.1048,0.1335,0.1639,0.1335,0.2752,
0.0262,0.2958,0.2226,0.2916,0.1142,0.2017,0.3252,0.2093,0.1280,0.4335,0.0627,0.1850,
0.4388,0.1821,0.1451,0.0544,0.3462,0.0154,0.0822,0.3031,0.1478,0.2130,0.2230,0.2897,
0.0397,0.2436,0.0428,0.3551,0.1458,0.3382,0.0957,0.2303,0.3804,0.0262,0.0356,0.0130,
0.2607,0.2102,0.0020,0.2581,0.1933,0.0047,0.0244,0.0922,0.3805,0.1092,0.1670,0.1447,
0.0477,0.3077,0.2124,0.2124,0.1879,0.1623,0.1219,0.2904,0.0953,0.1132,0.0502,0.4002,
0.3765,0.0096,0.1440,0.2291,0.1027,0.3114,0.2580,0.2089,0.1434,0.3168,0.3503,0.2551,
0.1064,0.3127,0.0588,0.1926,0.3867,0.1418,0.1538,0.4241,0.0659,0.1438,0.2286,0.0508,
0.2072,0.2740,0.1688,0.2416,0.0827,0.2314,0.3496,0.1934,0.3886,0.0894,0.0704,0.1493,
0.1843,0.1859,0.1252,0.2559,0.0035,0.1217,0.0717,0.0912,0.0251,0.2405,0.1436,0.1074,
0.0493,0.1552,0.2456,0.3565,0.0167,0.3025,0.0187,0.2772,0.2816,0.3582,0.0750,0.2422,
0.2169,0.1210,0.2634,0.2424,0.0600,0.1631,0.3293,0.1383,0.2371,0.1551,0.0338,0.1593,
0.3720,0.1812,0.0607,0.1999,0.1206,0.2411,0.1635,0.2727,0.2958,0.0758,0.0701,0.3565,
0.2880,0.0363,0.3726,0.2002,0.2003,0.2127,0.2768,0.3146,0.1400,0.0291,0.2936,0.1341,
0.3375,0.1544,0.3321,0.0716,0.0532,0.3473,0.0176,0.2671,0.2772,0.1617,0.1264,0.3688,
0.1475,0.1768,0.0764,0.1556,0.1539,0.3684,0.0979,0.1012,0.1261,0.2401,0.0153,0.3234,
0.0373,0.2052,0.0465,0.3405,0.3056,0.0901,0.2585,0.1746,0.2905,0.1795,0.3366,0.1744,
0.1618,0.2372,0.1421,0.3937,0.1927,0.0760,0.1248,0.2367,0.1159,0.0431,0.3350,0.2452,
0.1800,0.1234,0.1133,0.2164,0.1742,0.2783,0.0053,0.3180,0.2518,0.0386,0.3270,0.0609,
0.2273,0.2060,0.0477,0.0029,0.3182,0.3218,0.1980,0.0483,0.2532,0.0704,0.2688,0.1805,
0.0634,0.3304,0.2816,0.3470,0.0396,0.0822,0.3077,0.2812,0.2906,0.1659,0.1466,0.2278,
0.3560,0.3095,0.2671,0.1798,0.3339,0.1612,0.1967,0.2755,0.2225,0.2483,0.3013,0.2941,
0.0201,0.0807,0.1395,0.2114,0.0911,0.2137,0.1130,0.1435,0.2817,0.0310,0.0678,0.3107,
0.1429,0.0814,0.2429,0.3712,0.1587,0.2622,0.2169,0.1503,0.1834,0.3897,0.0846,0.1703,
0.2341,0.2631,0.3412,0.1344,0.0903,0.1993,0.0939,0.2847,0.1697,0.2312,0.2761,0.1753,
0.1190,0.0868,0.3864,0.0813,0.3023,0.2145,0.2764,0.3148,0.0086,0.1329,0.3952,0.2684,
0.3843,0.2181,0.0226,0.2681,0.1080,0.2249,0.2832,0.2395,0.2193,0.1590,0.0663,0.1474,
0.0163,0.2260,0.3727,0.0303,0.0235,0.3627,0.1062,0.0084,0.3246,0.0568,0.3485,0.0407,
0.1353,0.2338,0.2315,0.2545,0.2482,0.1510,0.0363,0.2882,0.0925,0.2603,0.3444,0.0445,

0.4034,0.0162,0.3319,0.3212,0.0066,0.2010,0.3604,0.1120,0.3155,0.2390,0.3231,0.3301,
0.0269,0.1138,0.2520,0.1875,0.3778,0.2939,0.2133,0.1170,0.0977,0.3524,0.1664,0.3211,
0.0252,0.3361,0.0023,0.3513,0.3686,0.2151,0.0581,0.0777,0.1664,0.3211,0.0252,0.3361,
0.0023,0.3513,0.3686,0.2151};

floatRemNoise[47]={ -2.181030629062908e-002,8.917428211564256e-
003,1.182863130963947e-002,1.428453375748106e-002,1.403350814552515e-
002,1.039856067649067e-002,5.093274200275827e-003,1.421581262318843e-
003,2.515430936577368e-003,9.363053213111126e-003,1.949152112446623e-002,
2.722897698684972e-002,2.579943237700517e-002,1.050645191333675e-002,-
1.832163998604397e-002,-5.406194181033640e-002,-8.514869723584616e-002,-
9.887618777194764e-002,-8.633153232820758e-002,-4.651515024517261e-
002,1.217011610629542e-002,7.394838432088444e-002,1.206802616409422e-001,
1.380624377729094e-001,1.206802616409422e-001,7.394838432088444e-002,
1.217011610629542e-002,-4.651515024517261e-002,-8.633153232820758e-002,-
9.887618777194764e-002,-8.514869723584616e-002,-5.406194181033640e-002,-
1.832163998604397e-002,1.050645191333675e-002,2.579943237700517e-002,
2.722897698684972e-002,1.949152112446623e-002,9.363053213111126e-003,
2.515430936577368e-003,1.421581262318843e-003,5.093274200275827e-
003,1.039856067649067e-002,1.403350814552515e-002,1.428453375748106e-002,
1.182863130963947e-002,8.917428211564256e-003,-2.181030629062908e-002};

DSK6713_AIC23_Config config = { \

0x0017, /* 0 DSK6713_AIC23_LEFTINVOL Left line input channel volume */ \
0x0017, /* 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume */\

0x00d8, /* 2 DSK6713_AIC23_LEFTHPVOL Left channel headphone volume */ \
0x00d8, /* 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume */
0x0011, /* 4 DSK6713_AIC23_ANAPATH Analog audio path control */ \

0x0000, /* 5 DSK6713_AIC23_DIGPATH Digital audio path control */ \
0x0000, /* 6 DSK6713_AIC23_POWERDOWN Power down control */ \
0x0043, /* 7 DSK6713_AIC23_DIGIF Digital audio interface format */ \

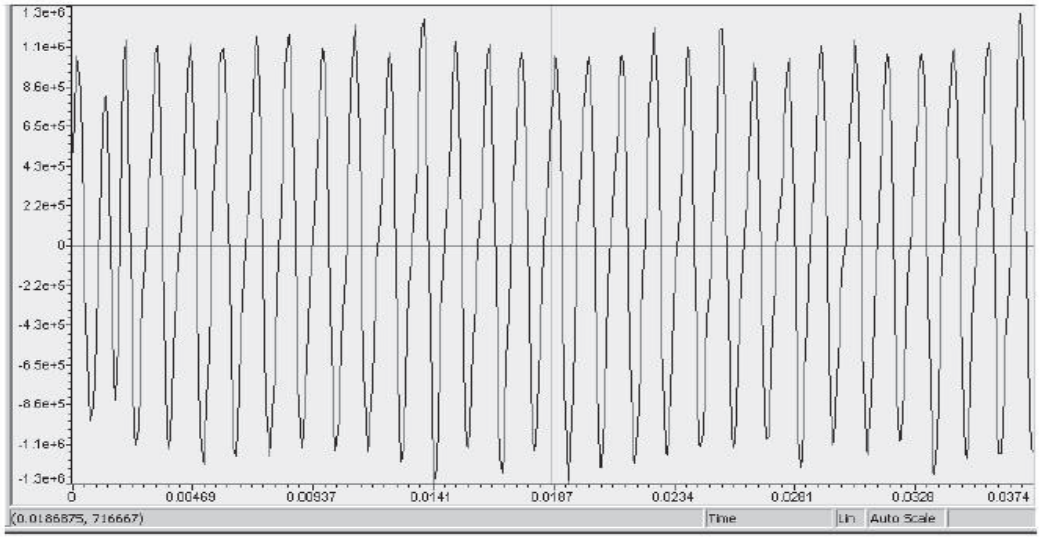
```
0x008c, /* 8 DSK6713_AIC23_SAMPLERATE Sample rate control */ \
0x0001 /* 9 DSK6713_AIC23_DIGACT Digital interface activation */ \
};
DSK6713_AIC23_CodecHandle hCodec;


Int32 InitWait =1;
Int32 data;
floatin_buffer[100];
floatInSigBuffer[600];
floatNoiseBuffer[600];
floatCorrSigBuffer[600];
floatRecovSigBuffer[600];
intLogIndex =0;
main(){
int i;
LED=0x0;
for( i = 0 ; i < 100; i++ ) in_buffer[i]=0.0;
hCodec = DSK6713_AIC23_openCodec(0,
&config); IRQ_globalEnable();
IRQ_enable(IRQ_EVT_RINT1);
IRQ_enable(IRQ_EVT_XINT1);
}
void led(){
staticint cc = 1;
LED = cc;
if (cc == 0x03) cc = 0x05; else
if (cc == 0x05) cc = 0x06; else
if (cc == 0x06) cc = 0x03; else
cc = 0x03;
}
setpll200M(){
}
```

```
void read(){
int i = 0;
float result;
floatInData;
floatInNoise;
floatInCorrup;
staticintNoiseIndex=0;
data=MCBSP_read(DSK6713_AIC23_DATAHANDLE);
if(data>=32768) data= data|0xffff0000;
InData = (Int32) (data)/16;
InData=data;
InNoise = PrNoise[NoiseIndex++]*4096 - 1024;
if (NoiseIndex == 600)
NoiseIndex = 0;
InCorrup = InData + InNoise;
for( i = 0; i <= (FiltLen-2); i++) in_buffer[i] = in_buffer[i+1];
in_buffer[FiltLen-1]=InCorrup;
result = 0;
for( i = 0 ; i <= (FiltLen-1); i++ ) result += RemNoise[i] *
in_buffer[i]; data = (Int32)(result*512);
InSigBuffer[LogIndex]= InData;
NoiseBuffer[LogIndex]= InNoise ;
CorrSigBuffer[LogIndex]= InCorrup;
RecovSigBuffer[LogIndex] = (float)data;
LogIndex++;
if (LogIndex ==
600) LogIndex =0;
}
void write(){
if (InitWait<1000){
InitWait++;
MCBSP_write(DSK6713_AIC23_DATAHANDLE, 0);
```

```
MCBSP_write(DSK6713_AIC23_DATAHANDLE, 0);
}
else{
MCBSP_write(DSK6713_AIC23_DATAHANDLE, data);
MCBSP_write(DSK6713_AIC23_DATAHANDLE, data);
}
```

OUTPUT:

Recovered signal can be seen as shown below



OUTCOME: Additive noise is removed from the corrupted signal using TMS320C6713 DSK.

VIVA QUESTIONS WITH ANSWER

1. What are the desirable characteristics of window Function?

(i) . The Fourier Transform of the window function W(ejw) should have a small width of the linear Phase characteristic is important when the phase distortion is not tolerable. FIR Filter can be designed with linear phase characteristic. In application like data transmission, speech processing etc. phase distortion cannot be tolerated and here linear phase characteristic of FIR filter is useful

2. Why rectangular window is not preferred for FIR filter design?

Rectangular window function has As = 21 db which is very small compared to other window function.

3. What is the advantage of frequency sampling realization?

The frequency sampling realization of filter is computationally more efficient than the direct form realization. Justification: When the desired frequency response characterization of the FIR filter is narrowband, most of the coefficients H[k] are zero. The corresponding filter sections can be eliminated and only the filters with non-zero coefficients need to be retained. The net result is a filter that requires fewer computations (multiplications and additions) than the corresponding direct form realization. Thus frequency Sampling realization is more efficient realization.

4. Why FIR filters are called as Non-recursive filters?

In FIR filter output depends only on input values. It doesn't depend on output values. e.g. y[n] = x[n] + x[n-1] Therefore FIR filters are also called as Non-Recursive Filters.

5. Explain how to find output of digital FIR filter in real time application.

In real time applications, output of FIR filter is obtained using overlap add method / overlap save method.