

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

L.B REDDY NAGAR, MYLAVARAM-521230

Digital Signal Processing Lab

B. Tech IV SEM (ECE)

COURSE CODE: 20EC55



DEPARTMENT OF ECE

A.Y:2021-22

Prepared by

Mr. T Anil Raju

Mr. M K Linga Murthy

Verified by

Dr. M V Sudhakar

Mrs. B Rajeswari

1. Generation of Discrete Time (DT) signals and Operations on DT signals

EXPT. NO: 1

DATE:

Aim:

To generate Discrete Time (DT) signals and operations on DT signals using MATLAB.

Equipment Required:

1. Personal Computer with MATLAB software.

Theory:

Generation of Discrete Time signals

A discrete time signal has a value defined only at discrete points in time and a discrete – time system operates on and produces discrete time variables. A discrete time signal is a sequence which is a function defined on positive and negative integers, that is

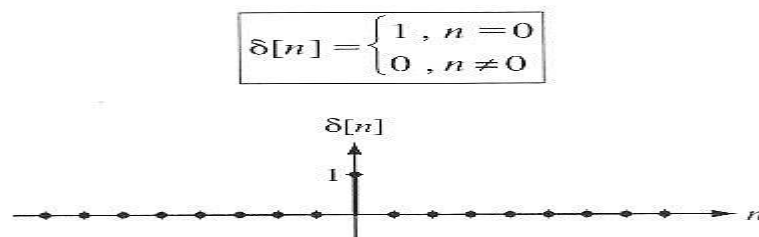
$$x(n) = \{x(n)\} = \{ \dots x(-1), x(0), x(1), \dots \}$$

Where UP arrow represents the sample at $n=0$. Here 'n' is an integer indicating the sample numbers as counted from a chosen time origin i.e. $n=0$. The negative values of 'n' corresponds to negative time. The function of 'n' is referred to as a sequence of samples (or) sequences in short.

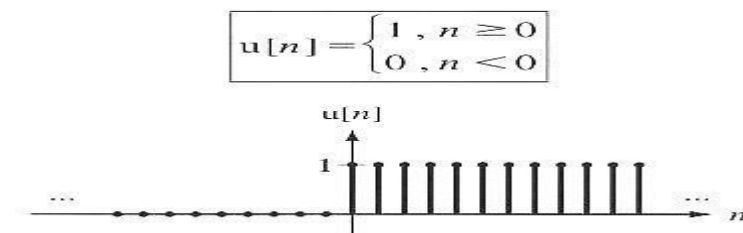
If a continuous time signal $x(t)$ is sample every T seconds, a sequence $x(nT)$ results. In general, the sequence values are called samples and the interval between them is called sample interval, T. For convenience, the sample interval T is taken as one second and hence $x(n)$ represents the sequence.

The Basic Discrete time sequences are

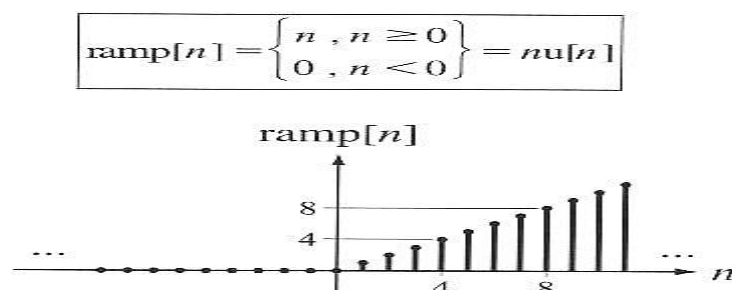
1. Unit- Impulse sequence (or) Unit- Sample sequence



2. Unit-Step sequence



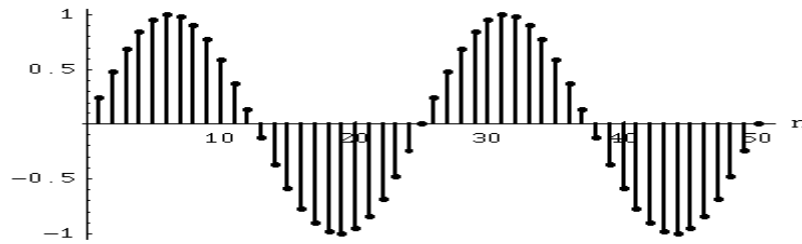
3. Unit-ramp sequence



4. Sinusoidal sequence

A discrete sinusoidal signal may be expressed as

$$x(n) = A \cos(\omega n + \theta) \quad -\infty < n < \infty$$



Operations on DT signals

When we process a sequence, this sequence may undergo several manipulations involving the independent variable or the amplitude of the signal.

The basic operations on sequences are as follows:

1. Time shifting
2. Time reversal
3. Time scaling
4. Amplitude scaling
5. Signal addition
6. Signal multiplication

The first three operations correspond to transformation in independent variable n of a signal. The last three operations correspond to transformation on amplitude of a signal.

(1) Time Shifting

The time shifting of a signal may result in time delay or time advance. The time shifting operation of a discrete-time signal $x(n)$ can be represented by

$$y(n) = x(n - k)$$

This shows that the signal $y(n)$ can be obtained by time shifting the signal $x(n)$ by k units. If k is positive, it is delay and the shift is to the right, and if k is negative, it is advance and the shift is to the left.

(2) Time Reversal

The time reversal also called time folding of a discrete-time signal $x(n)$ can be obtained by folding the sequence about $n = 0$. The time reversed signal is the reflection of the original signal. It is obtained by replacing the independent variable n by $-n$.

(3) Time Scaling

Time scaling may be time expansion or time compression. The time scaling of a discrete time signal $x(n)$ can be accomplished by replacing n by an in it. Mathematically, it can be expressed as: $y(n) = x(an)$

When $a > 1$, it is time compression and when $a < 1$, it is time expansion.

(4) Amplitude Scaling

The amplitude scaling of a discrete-time signal can be represented by

$$y(n) = ax(n) \text{ where } a \text{ is a constant.}$$

The amplitude of $y(n)$ at any instant is equal to a times the amplitude of $x(n)$ at that instant. If $a > 1$, it is amplification and if $a < 1$, it is attenuation. Hence the amplitude is rescaled. Hence the name amplitude scaling.

(5) Signal Addition and Subtraction

In discrete-time domain, the sum of two signals $x_1(n)$ and $x_2(n)$ can be obtained by adding the corresponding sample values and the subtraction of $x_2(n)$ from $x_1(n)$ can be obtained by subtracting each sample of $x_2(n)$ from the corresponding sample of $x_1(n)$ as illustrated below.

If $x_1(n) = \{1, 2, 3, 1, 5\}$ and $x_2(n) = \{2, 3, 4, 1, -2\}$

Then $x_1(n) + x_2(n) = \{1 + 2, 2 + 3, 3 + 4, 1 + 1, 5 - 2\} = \{3, 5, 7, 2, 3\}$

and $x_1(n) - x_2(n) = \{1 - 2, 2 - 3, 3 - 4, 1 - 1, 5 + 2\} = \{-1, -1, -1, 0, 7\}$

(6) Signal Multiplication

The multiplication of two discrete-time sequences can be performed by multiplying their values at the sampling instants as shown below.

If $x_1(n) = \{1, -3, 2, 4, 1.5\}$ and $x_2(n) = \{2, -1, 3, 1.5, 2\}$, Then

$x_1(n) \cdot x_2(n)$ is $\{1 \cdot 2, 3 \cdot 1, 2 \cdot 3, 4 \cdot 1.5, 1.5 \cdot 2\} = \{2, 3, 6, 6, 3\}$

Description of basic functions:

1. Sine of argument in radians and **sin(X)** is the sine of the elements of X.
2. Cosine of argument in radians and **cos(X)** is the cosine of the elements of X.
3. **rectpuls(T)** generates samples of a continuous, aperiodic, unity-height rectangle at the points specified in array T, centered about T=0. By default, the rectangle has width 1.
4. **tripuls(T)** generates samples of a continuous, aperiodic, unity-height triangle at the points specified in array T, centered about T=0. By default, the triangle is symmetric and has width 1.
5. **sawtooth(T)** generates a sawtooth wave with period 2π for the elements of time vector T. **sawtooth(T)** is like **SIN(T)**, only it creates a sawtooth wave with peaks of +1 to -1 instead of a sine wave.
6. **square(T)** generates a square wave with period 2π for the elements of time vector T.
7. **fliplr** Flip array in left/right direction. **Y = fliplr(X)** returns X with the order of elements flipped left to right along the second dimension.
8. **xlabel('text')** adds text beside the X-axis on the current axis.
9. **ylabel('text')** adds text beside the Y-axis on the current axis.
10. **title('text')** adds text at the top of the current axis.
11. Discrete sequence or "stem" plot. **stem(Y)** plots the data sequence Y as stems from the x axis terminated with circles for the data value.
12. Linear plot. **plot(X,Y)** plots vector Y versus vector X. If X or Y is a matrix, then the vector is plotted versus the rows or columns of the matrix, whichever line up.
13. subplot Create axes in tiled positions. **H = subplot(m,n,p)**, or **subplot(mnp)**, breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program :

% Experiment 1(a) : Generation of Discrete Time Signals

```
clc;
clear all;
close all;
figure(1)
n = -10:1:10;
```

% Unit Impulse Sequence

```
impulse = [zeros(1,10), ones(1,1), zeros(1,10)];  
subplot(2,2,1);  
stem(n,impulse);  
xlabel('Discrete time n --->');  
ylabel('Amplitude--->');  
title('Unit Impulse Sequence');
```

% Unit Step Sequence

```
step = [zeros(1,10), ones(1,11)];  
subplot(2,2,2);  
stem(n,step);  
xlabel('Discrete time n --->');  
ylabel('Amplitude--->');  
title('Unit Step Sequence');
```

% Unit Ramp Sequence

```
n1 = 0:1:10;  
ramp = n1;  
subplot(2,2,3);  
stem(n1,ramp);  
xlabel('Discrete time n --->');  
ylabel('Amplitude--->');  
title('Unit Ramp Sequence');
```

% Unit Parabolic Sequence

```
n1 = 0:1:10;  
parabola = 0.125*(n1.^3);  
subplot(2,2,4);  
stem(n1,parabola);  
xlabel('Discrete time n --->');  
ylabel('Amplitude--->');  
title('Unit parabola Sequence');
```

% Generation of Discrete time exponential sequence

```
figure(2)  
n3 = -10:1:10;  
% for  $0 < a < 1$   
a = 0.8;  
x1 = a.^n3;  
subplot(2,2,1);  
stem(n3,x1);  
xlabel('Discrete time n --->');  
ylabel('Amplitude--->');  
title('x1(n) for  $0 < a < 1$  (Decaying Exponential)');
```

```

% for a > 1
a = 1.5;
x2 = a.^n3;
subplot(2,2,2);
stem(n3,x2);
xlabel('Discrete time n --->');
ylabel('Amplitude--->');
title('x2(n) for a > 1 (Increasing Exponential)');
% for -1 < a < 0
a = -0.8;
x3 = a.^n3;
subplot(2,2,3);
stem(n3,x3);
xlabel('Discrete time n --->');
ylabel('Amplitude--->');
title('x3(n) for -1 < a < 0 (Alternating Decreasing Exponential)');
% for a < -1
a = -1.5;
x4 = a.^n3;
subplot(2,2,4);
stem(n3,x4);
xlabel('Discrete time n --->');
ylabel('Amplitude--->');
title('x3(n) for a<-1 (Alternating Increasing Exponential)');

```

```

x=0:0.1:(4*pi);           % defining the time instants
x1=-2*pi:0.1:2*pi;        % defining the time range from -pi to pi
y=sin(x);                 % obtain the amplitudes of sin signal
z=cos(x);                 % obtain the amplitudes of cos signal
k=square(x,50);           % obtain the square wave
b=tripuls(x1);             % obtain the triangular wave
i=sawtooth(x1);            % obtain the saw tooth signal
c=rectpuls(x1);            % obtain the rectangular signal

```

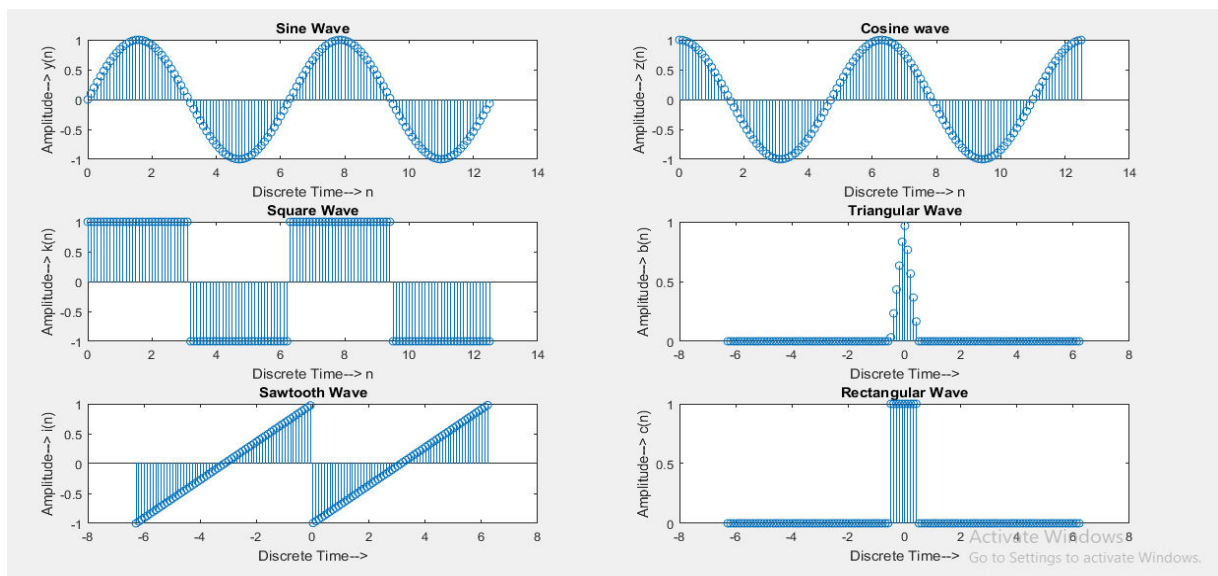
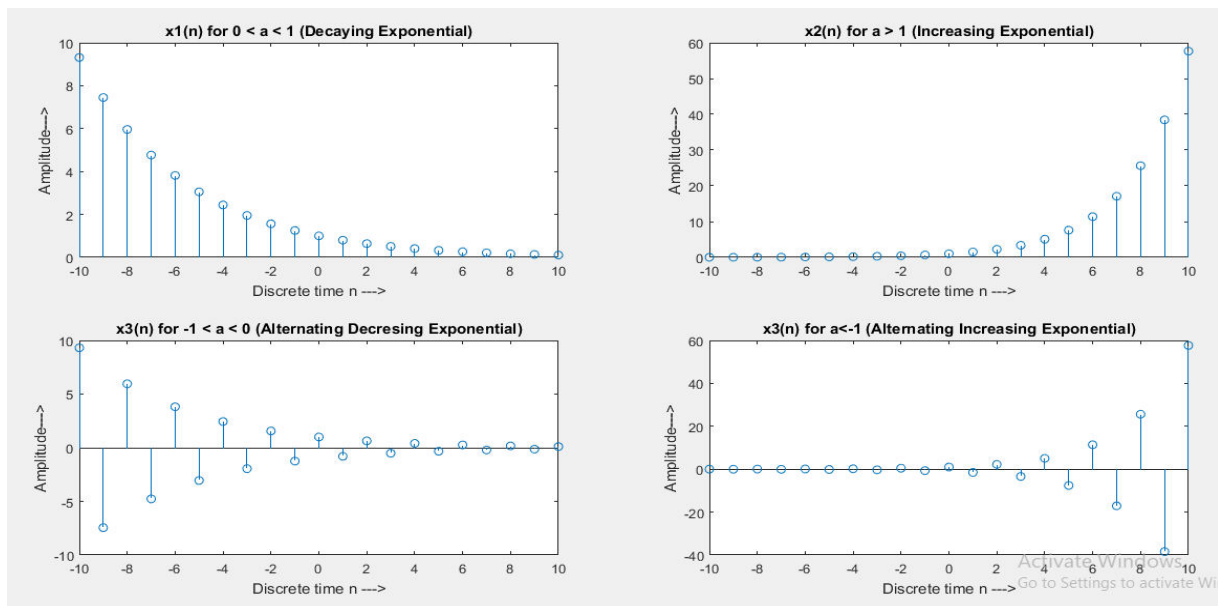
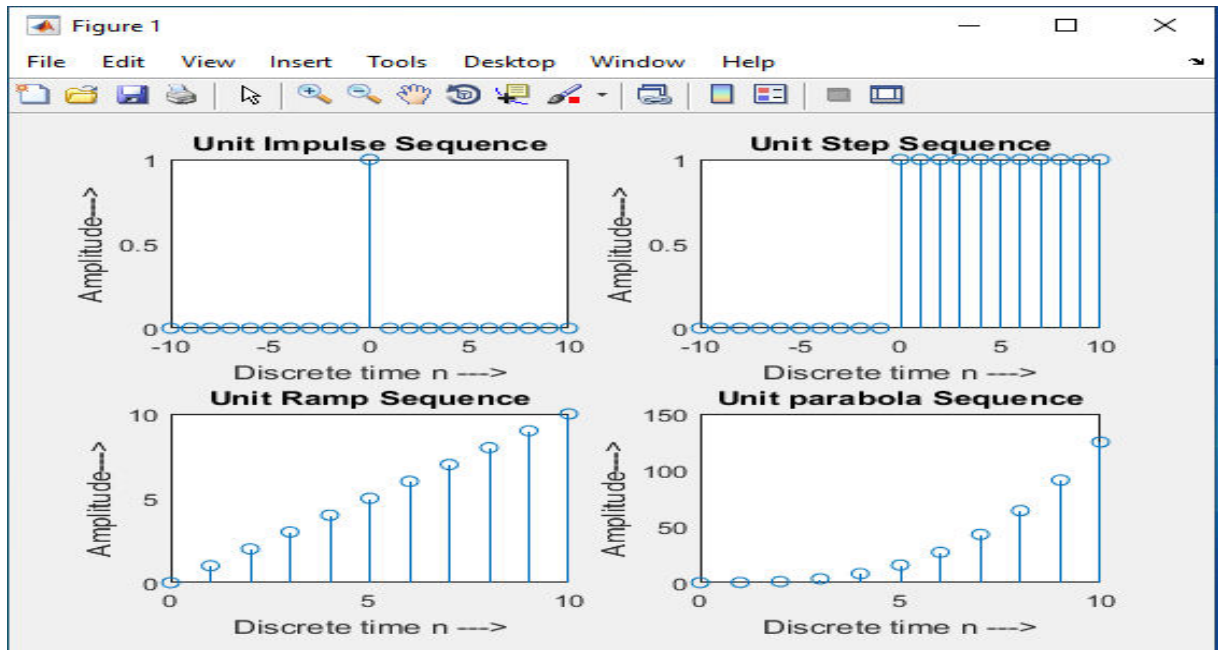
% Generation of Discrete Time Sequences

```

figure(3);
subplot(3,2,1), stem(x,y); xlabel('Discrete Time--> n'); ylabel('Amplitude--> y(n)');
title('Sine Wave');
subplot(3,2,2), stem(x,z); xlabel('Discrete Time--> n'); ylabel('Amplitude--> z(n)');
title('Cosine wave');
subplot(3,2,3), stem(x,k); xlabel('Discrete Time--> n'); ylabel('Amplitude--> k(n)');
title('Square Wave');
subplot(3,2,4), stem(x1,b); xlabel('Discrete Time-->'); ylabel('Amplitude--> b(n)');
title('Triangular Wave');
subplot(3,2,5), stem(x1,i); xlabel('Discrete Time-->'); ylabel('Amplitude--> i(n)');
title('Sawtooth Wave');
subplot(3,2,6), stem(x1,c); xlabel('Discrete Time-->'); ylabel('Amplitude--> c(n)');
title('Rectangular Wave');

```

Output 1(a):



% Experiment 1(b) : Operation on Discrete Time Sequences

```
clc;
clear all;
close all;
n1 = -2:1                                % Time instants for first sequence
x = [1 2 3 4]                            % Amplitudes of first sequence
subplot (3,3,1);
stem (n1,x);
title('input signal X');
axis([-3 3 0 5]);
n2 = 0:3;                                % Time instants for second sequence
y = [1 1 1 1];                          % Amplitudes of second sequence
subplot (3,3,2);
stem (n2,y);
title('input signal Y');
axis([-3 3 0 5]);
% Time instants for output sequence
n3 = min(min(n1),min(n2)):max(max(n1),max(n2));
s1 = zeros(1, length(n3));
s2 = s1;
s1(find((n3>=min(n1))&(n3<=max(n1))==1))=x;
s2(find((n3>=min(n2))&(n3<=max(n2))==1))=y;
% Addition of two signals S1 and S2
A = s1+s2;
subplot(3,3,3);
stem(n3,A);
title('Addition of both input signals A=X+Y');
axis([-3 3 0 5]);
% Subtraction of two signals S1 and S2
S = s1-s2;
subplot(3,3,4);
stem(n3,S);
title('subtraction of signal Y from X , S=X-Y');
axis([-4 4 -5 5]);
% Multiplication of two signals S1 and S2
M = s1.*s2;
subplot(3,3,5);
stem(n3,M);
title('Multiplication of two signals M=X*Y');
axis([-4 4 -5 5]);
% Amplitude Scaling by 2
SC = 2*s1;
% Time Reversal of a Sequence or Folding
c = fliplr(x);
y = fliplr(-n1);
subplot(3,3,6);
stem(y,c);
axis([-3 3 0 5]);
title('Reversed input signal x(-n)');
```


% Time Shifting of a Sequence or Delay the sequence by 3 Units

$m = n1+3$

$p = x$

subplot(3,3,7);

stem(m,p);

title('delayed signal x(n-3)');

% Time Shifting of a Sequence or Advance the sequence by 3 Units

$t = n1-3$

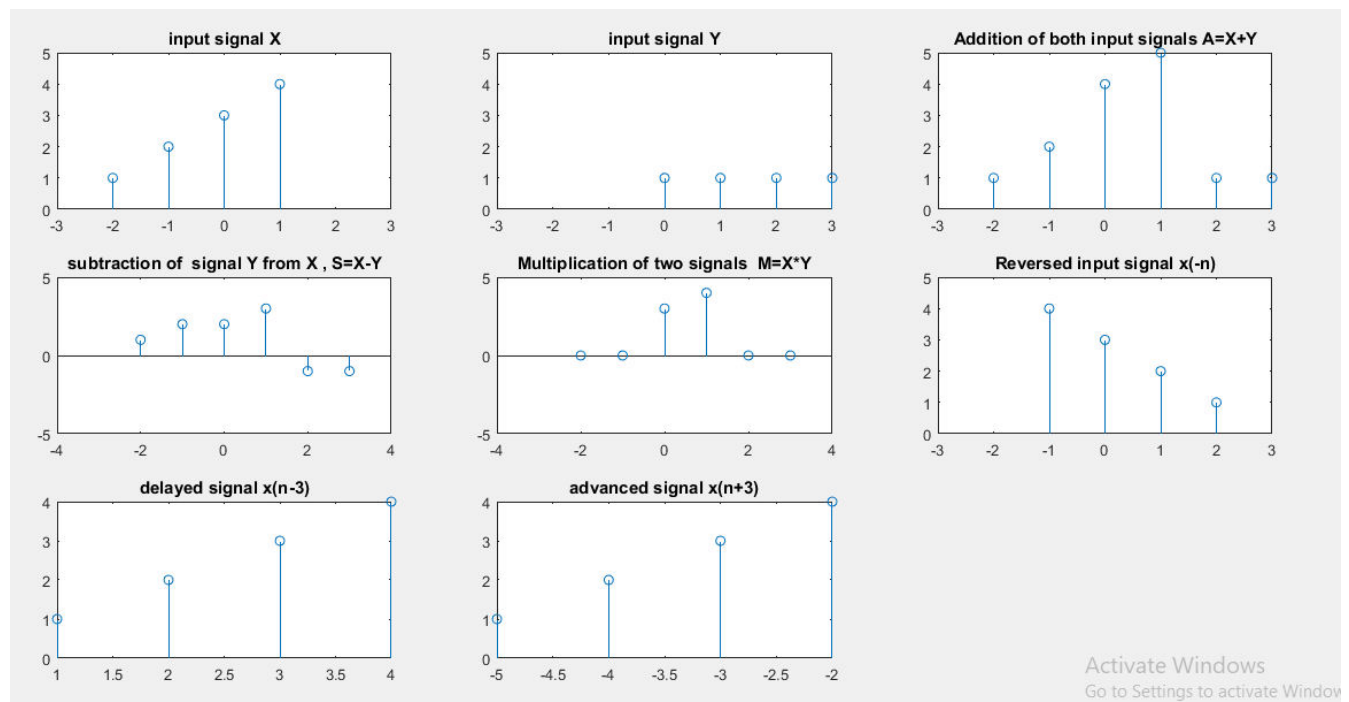
$z = x$

subplot(3,3,8);

stem(t,z);

title('advanced signal x(n+3)');

Output (2b):



Experimental Observations:

First Sequence $x = \{1 \ 2 \ 3 \ 4\}$ and time instants $\{-2, -1, 0, 1\}$

Second Sequence $y = \{1 \ 1 \ 1 \ 1\}$ and time instants $\{0, 1, 2, 3\}$

Addition of two signals S1 and S2

$A = \{1 \ 2 \ 4 \ 5 \ 1 \ 1\}$

Subtraction of two signals S1 and S2

$S = \{1 \ 2 \ 2 \ 3 \ -1 \ -1\}$

Multiplication of two signals S1 and S2

$M = \{0 \ 0 \ 3 \ 4 \ 0 \ 0\}$

Amplitude Scaling by 2

$SC = \{2 \ 4 \ 6 \ 8 \ 0 \ 0\}$

Folding or Time Reversal and its time instants

$c = \{4 \ 3 \ 2 \ 1\}$ and $y = \{-1 \ 0 \ 1 \ 2\}$

Time shifting or Delay by 3 Units and its time instants

$m = \{ 1 \quad 2 \quad 3 \quad 4 \}$ and $p = \{ 1 \quad 2 \quad 3 \quad 4 \}$

Time shifting or Advance by 3 Units and its time instants

$t = \{ -5 \quad -4 \quad -3 \quad -2 \}$ and $z = \{ 1 \quad 2 \quad 3 \quad 4 \}$

Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. What are the different types of representations of discrete time signal?
2. How discrete time signals are differed from continuous time signals?
3. Describe the basic discrete time signals along with their expressions?
4. What are the basic operations used on discrete time signals?
5. How discrete time signal differed from a digital signal?
6. What is meant by Time Scaling?
7. Describe the Time Scaling operation along with the expressions?
8. Describe the Folding operation along with the expressions?
9. Explain Addition, Subtraction and Multiplication operations used on discrete time signals?
10. How addition of two signals operation is carried out, if two signals are not having the same length?

Result:

The basic discrete time signals were generated and operations on discrete time signals were verified using MATLAB.

Aim:

To verify the linear convolution between input sequence $x(n)$ and impulse sequence $h(n)$ for a given discrete LTI system using MATLAB.

Equipment Required:

1. Personal Computer with MATLAB software.

Theory:**Linear Convolution**

Convolution is an important operation in DSP, because convolving two sequences in time domain is equivalent to multiplying the sequences in frequency domain. Convolution mainly used in processing signals especially analysing the output of a system. Convolution of two signals $x(n)$ and $h(n)$ is given as

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

Where the symbol $[*]$ is used to denote convolution

In practice, we often deal with sequences of finite length, and their convolution may be found by several methods. The convolution $y(n)$ of two finite-length sequences $x(n)$ and $h(n)$ is also of finite length and is subject to the following rules, which serve as useful consistency

checks:

1. The starting index of $y(n)$ equals the sum of the starting indices of $x(n)$ and $h(n)$.
2. The ending index of $y(n)$ equals the sum of the ending indices of $x(n)$ and $h(n)$.
3. The length N of $y(n)$ is related to the lengths N_1 and L_h of $x(n)$ and $h(n)$ by $N_2 = N_1 + N_2 - 1$.

The various steps involved in finding out convolution sum are

Step 1: Choose the starting time n for evaluating the output sequence $y(n)$. If $x(n)$ starts at $n = n_1$ and $h(n)$ starts at $n = n_2$, then $n = n_1 + n_2$ is a good choice.

Step 2: Express both the sequences $x(n)$ and $h(n)$ in terms of the index k .

Step 3: Fold $h(k)$ about $k = 0$ to obtain $h(-k)$ and shift by n to the right if n is positive and to the left if n is negative to obtain $h(n - k)$.

Step 4: Multiply the two sequences $x(k)$ and $h(n - k)$ element by element and sum the products to get $y(n)$.

Step 5: Increment the index n , shift the sequence $h(n - k)$ to the right by one sample and perform Step 4.

Step 6: Repeat Step 5 until the sum of products is zero for all remaining values of n .

Description of basic functions:

1. **C = conv(A, B)** convolves vectors A and B. The resulting vector is length MAX([LENGTH(A)+LENGTH(B)-1,LENGTH(A),LENGTH(B)]). If A and B are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials.
2. **disp(X)** displays the array, without printing the array name
3. **xlabel('text')** adds text beside the X-axis on the current axis.
4. **ylabel('text')** adds text beside the Y-axis on the current axis.
5. **title('text')** adds text at the top of the current axis.
6. Discrete sequence or "stem" plot. **stem(Y)** plots the data sequence Y as stems from the x axis terminated with circles for the data value.
7. subplot Create axes in tiled positions. **H = subplot(m,n,p)**, or subplot(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 2: Linear Convolution

```
clc;  
clear all;  
close all;
```

% Define the Input Sequence x(n)

```
x = input('enter the input sequence x(n)');  
subplot(3,1,1);  
stem(x);  
xlabel('Discrete time-->');  
ylabel('Amplitude-->');  
title('First Sequence x(n)');
```

% Define the Impulse Sequence h(n)

```
x = input('enter the impulse sequence h(n)');  
subplot(3,1,2);  
stem(h);  
xlabel('Discrete time-->');  
ylabel('Amplitude-->');  
title('Second Sequence h(n)');
```

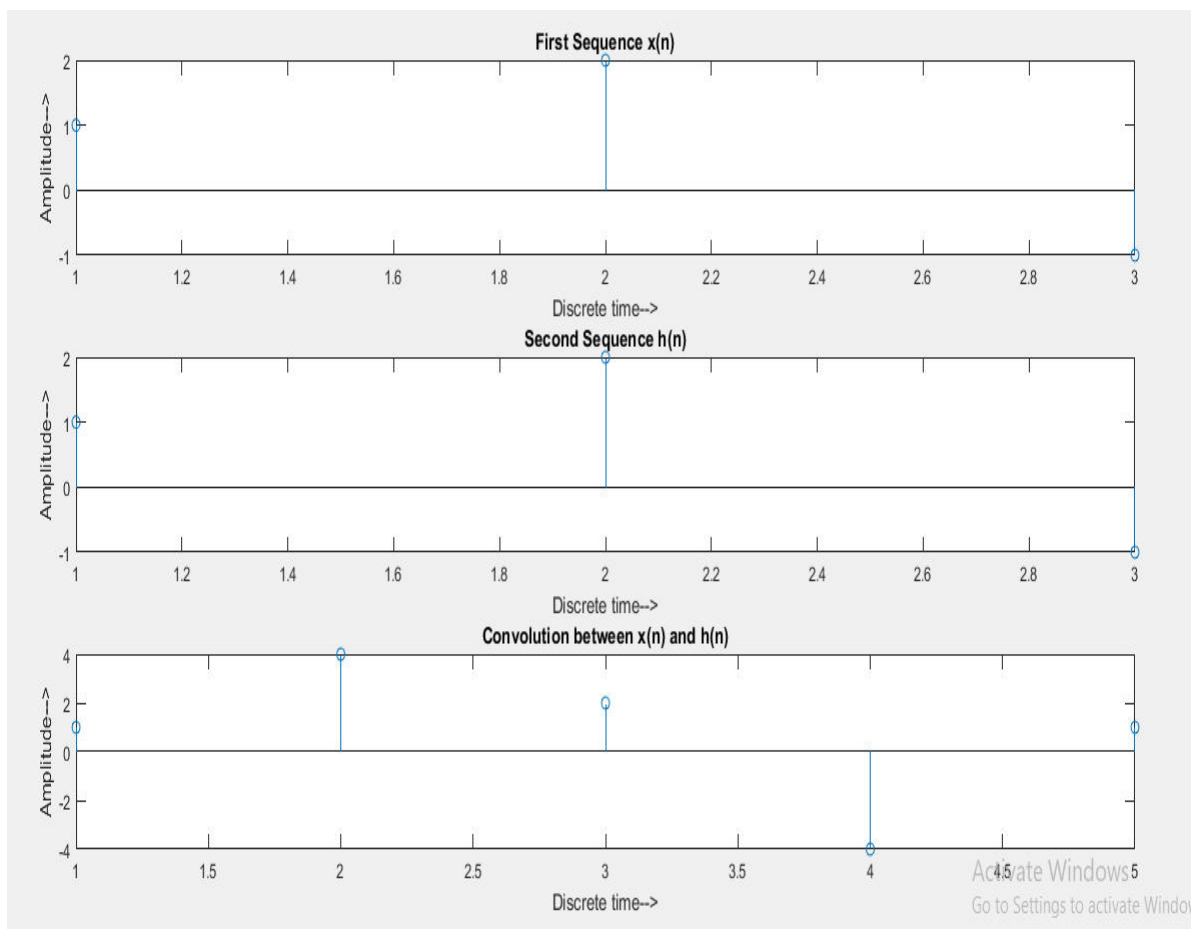
```

% Convolution between x(n) and h(n)
y = conv(x,h);
subplot(3,1,3);
stem(y);
xlabel('Discrete time-->');
ylabel('Amplitude-->');
title('Convolution between x(n) and h(n)');

disp('Input Sequence is x(n) is');
disp(x);
disp('Impulse Sequence is h(n) is');
disp(h);
disp('Convolution between x(n) and h(n) is');
disp(y);

```

Output 2:



Experimental Observations for 3(a):

First Sequence is $x(n)$ is : 1 2 -1
 Second Sequence is $h(n)$ is : 1 2 -1
 Convolution between $x(n)$ and $h(n)$ is : 1 4 2 -4 1

Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. What is convolution?
2. What are the applications of convolution?
3. Which command is used to find convolution between two sequences?
4. What are the various steps involved in finding out convolution?
5. What are the properties of Twiddle factor?
6. Explain DFT and IDFT with their expressions?
7. What is a command are used to perform DFT and IDFT?
8. How linear convolution operation is carried out through DFT and IDFT?

Result:

Linear convolution response of discrete LTI system with input sequence $x(n)$ and impulse response $h(n)$ was verified using MATLAB.

Aim:

To verify the circular convolution between two sequences $x_1(n)$ and $x_2(n)$ using MATLAB.

Equipment Required:

1. Personal Computer with MATLAB software.

Theory**Circular Convolution**

- The circular convolution of two sequences requires that at least one of the two sequences should be periodic. If both the sequences are non-periodic, then periodically extend one of the sequences and then perform circular convolution.
- The circular convolution can be performed only if both the sequences consists of the same number of samples. If the sequences have different number of samples, then convert the smaller size sequence to the size of larger size sequence by appending zeros. The circular convolution produces a sequence whose length is same as that of input sequences.
- Circular convolution basically involves the same four steps as linear convolution namely folding one sequence, shifting the folded sequence, multiplying the two sequences and finally summing the value of the product sequences.
- The difference between the two is that in circular convolution the folding and shifting (rotating) operations are performed in a circular fashion by computing the index of one of the sequences by modulo- N operation.
- In circular convolution, any one of the sequence is folded and rotated without changing the result of circular convolution

Concentric Circle Method

Let $x_1(n)$ and $x_2(n)$ be two given sequences.

The steps followed for circular convolution of $x_1(n)$ and $x_2(n)$ are

- Take two concentric circles. Plot N samples of $x_1(n)$ on the circumference of the outer circle (maintaining equal distance successive points) in anti-clockwise direction.
- For plotting $x_2(n)$, plot N samples of $x_2(n)$ in clockwise direction on the inner circle, starting sample placed at the same point as 0^{th} sample of $x_1(n)$.
- Multiply corresponding samples on the two circles and add them to get output.
- Rotate the inner circle anti-clockwise with one sample at a time.

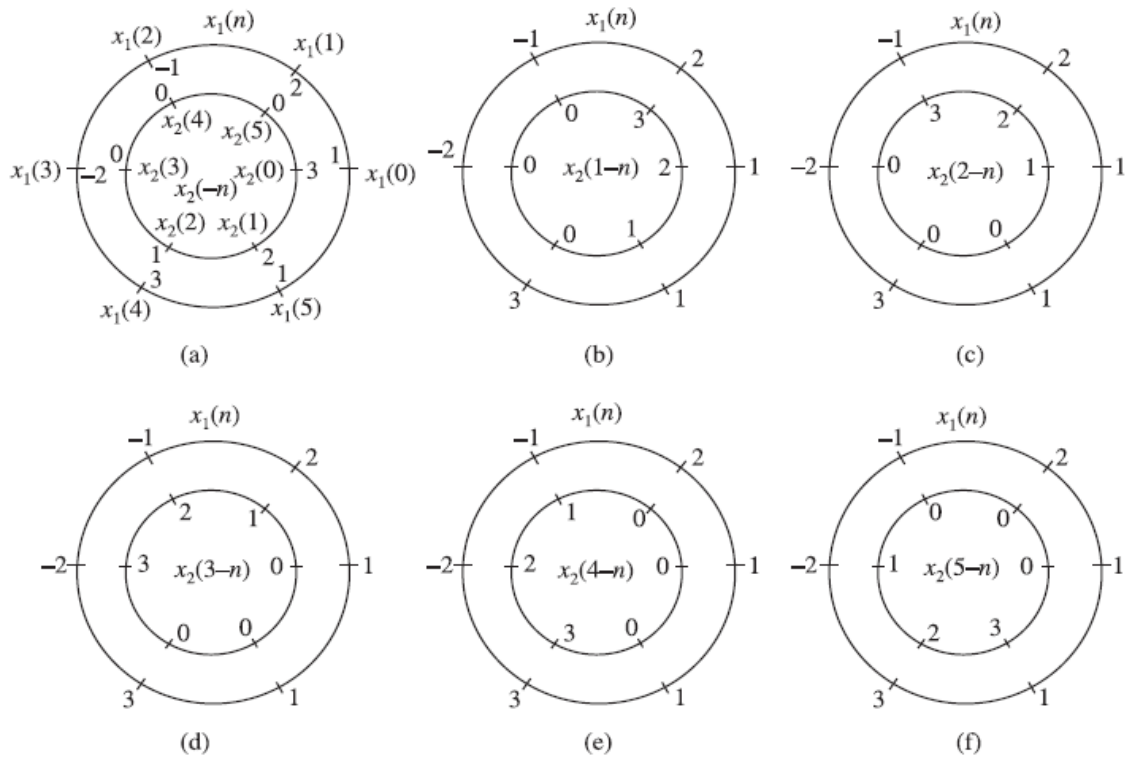
$$x_3(n) = \sum_{m=0}^{N-1} x_1(m)x_2((n-m))$$

Example Find the circular convolution of two finite duration sequences:

$$x_1(n) = \{1, 2, -1, -2, 3, 1\}, x_2(n) = \{3, 2, 1\}$$

Solution:

- Let $x_3(n)$ be the circular convolution of $x_1(n)$ and $x_2(n)$.
- To find the circular convolution, both sequences must be of same length.
- Here $x_1(n)$ is of length 6 and $x_2(n)$ is of length 3. Therefore, we append three zeros to the sequence $x_2(n)$ and use concentric circle method to find circular convolution.
- So we have $x_1(n) = \{1, 2, -1, -2, 3, 1\}$, $x_2(n) = \{3, 2, 1, 0, 0, 0\}$
From Figure $x_3(0) = (1)(3) + (2)(0) + (-1)(0) + (-2)(0) + (3)(1) + (1)(2) = 8$
From Figure $x_3(1) = (1)(2) + (2)(3) + (-1)(0) + (-2)(0) + (3)(0) + (1)(1) = 9$
From Figure $x_3(2) = (1)(1) + (2)(2) + (-1)(3) + (-2)(0) + (3)(0) + (1)(0) = 2$
From Figure $x_3(3) = (1)(0) + (2)(1) + (-1)(2) + (-2)(3) + (3)(0) + (1)(0) = -6$
From Figure $x_3(4) = (1)(0) + (2)(0) + (-1)(1) + (-2)(2) + (3)(3) + (1)(0) = 4$
From Figure $x_3(5) = (1)(0) + (2)(0) + (-1)(0) + (-2)(1) + (3)(2) + (1)(3) = 7$
Therefore, the circular convolution of $x_1(n)$ and $x_2(n)$ is: $x_3(n) = \{8, 9, 2, -6, 4, 7\}$



Description of basic functions:

1. **disp(X)** displays the array, without printing the array name
2. **xlabel('text')** adds text beside the X-axis on the current axis.
3. **ylabel('text')** adds text beside the Y-axis on the current axis.
4. **title('text')** adds text at the top of the current axis.
5. Discrete sequence or "stem" plot. **stem(Y)** plots the data sequence Y as stems from the x axis terminated with circles for the data value.
6. subplot Create axes in tiled positions. **H = subplot(m,n,p)**, or subplot(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.
7. The general form of a for statement is

for variable = expr, statement, ..., statement END

9. if Conditionally execute statements. The general form of the if statement is

if expression

statements

ELSEIF expression

statements

ELSE

statements

END

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 3 : Circular Convolution

```
clc;
```

```
clear all;
```

```
close all;
```

% Define the First Sequence x1(n)

```
x1 = [1 2 -1 -2 3 1];
```

```
N1 = length(x1);
```

```
subplot(3,1,1);
```

```
stem(x1);
```

```
xlabel('Discrete time-->');
```

```
ylabel('Amplitude-->');
```

```
title('First Sequence x1(n)');
```

% Define the Second Sequence x2(n)

```
x2 = [3 2 1];
```

```
N2 = length(x2);
```

```
subplot(3,1,2);
```

```
stem(x2);
```

```
xlabel('Discrete time-->');
```

```
ylabel('Amplitude-->');
```

```
title('Second Sequence x2(n)');
```

% Output Sequence length

```
N=max(N1,N2);
```

```
% Zero padding for two sequences
```

```
x1=[x1 zeros(1,N-N1)];
```

```
x2=[x2 zeros(1,N-N2)];
```

% Logic for Circular Convolution

```
for m=1:N
```

```
  y(m)=0;
```

```
  for n=1:N
```

```
    i=m-n+1;
```

```
    if(i<=0)
```

```
      i=N+i;
```

```
    end
```

```
    y(m)=y(m)+x1(n)*x2(i);
```

```
  end
```

```
end
```

```
subplot(3,1,3);
```

```
stem(y);
```

```
xlabel('Discrete time-->');
```

```
ylabel('Amplitude-->');
```

```
title('Circular Convolution between x1(n) and x2(n)');
```

```
disp('First Sequence x1(n) is ');
```

```
disp(x1);
```

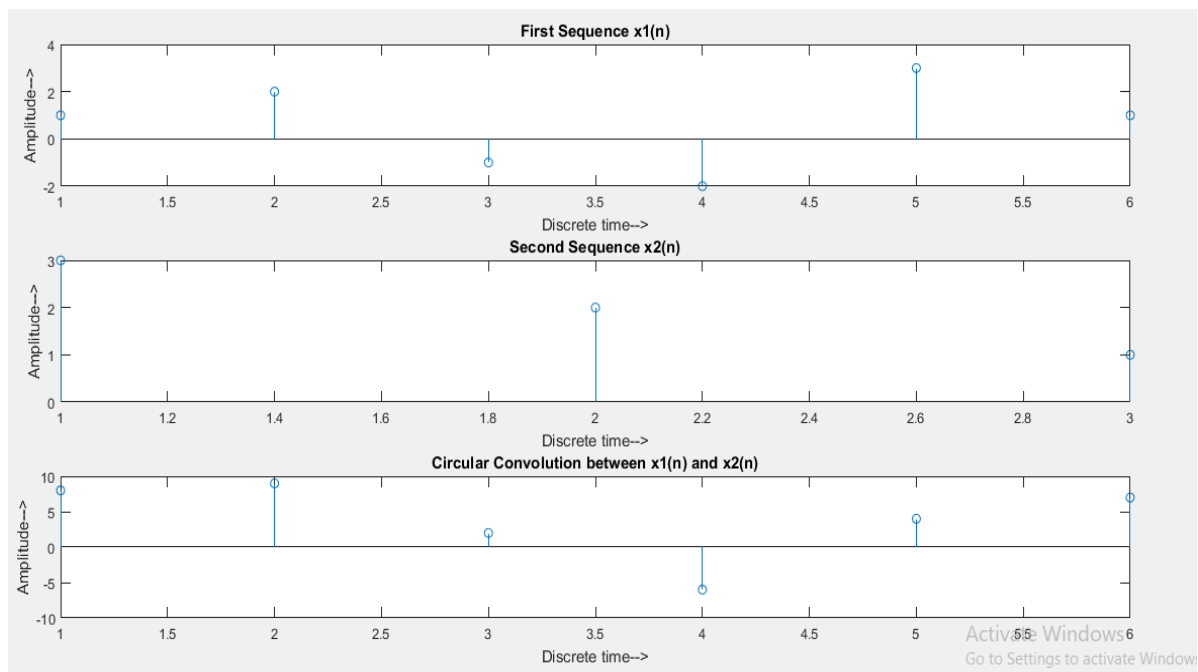
```
disp('Second Sequence x2(n) is ');
```

```
disp(x2);
```

```
disp('Circular Convolution between x1(n) and x2(n)is ');
```

```
disp(y);
```

Output 3:



Experimental Observations for 3:

First Sequence $x_1(n)$ is : 1 2 -1 -2 3 1

Second Sequence $x_2(n)$ is : 3 2 1 0 0 0

Circular Convolution between $x_1(n)$ and $x_2(n)$ is : 8 9 2 -6 4 7

Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. What is the need of circular convolution?
2. What are the basic operations involved in circular convolution?
3. What are the differences between linear and circular convolution?
4. What is meant by padding with zeros?
5. List the methods used for circular convolution

Result:

Circular convolution between $x_1(n)$ and $x_2(n)$ using MATLAB was verified.

Aim:

To compute Discrete Fourier Transform and Inverse Discrete Fourier Transform for given N-point sequence.

Equipment Required:

1. Personal Computer with MATLAB software.

Theory:

The DTFT of a sequence is periodic and continuous in frequency in the range from 0 to 2π . There are infinitely many ω in this range. If we use a digital computer to compute N equally spaced points over the interval $0 \leq \omega \leq 2\pi$, then the N-points should be located at

$$\omega_k = \frac{2\pi}{N} k, \quad k=0,1,2, \dots, N-1$$

These N equally spaced frequency samples of the DTFT are known as DFT of sequence and it is denoted by $X(k)$ is

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

Let $x(n)$ is a causal, finite duration sequence containing L samples, then its Fourier transform is given by

$$X(e^{j\omega}) = \sum_{n=0}^{L-1} x(n) e^{-j\omega n}$$

If we samples $X(e^{j\omega})$ at N equally spaced points over $0 \leq \omega \leq 2\pi$, we obtain

$$X(k) = X(e^{j\omega}) \Big|_{\omega = \frac{2\pi}{N}k} = \sum_{n=0}^{L-1} x(n) e^{-j\frac{2\pi}{N}kn}$$

Since time domain aliasing occurs if $N \leq L$, we increase the duration of sequence $x(n)$ from L to N samples by appending appropriate zeros, which is known as zero padding.

Since zero valued elements contribute nothing to sum, hence the above equation can be written as $X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}$ $0 \leq k \leq N-1$ which is called N-point DFT of sequence $x(n)$.

Since $x_p(n)$ is periodic extension of $x(n)$ with period N, it can be expressed in Fourier series expansion $x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}$ $0 \leq n \leq N-1$ is called N-point IDFT.

Example: Find the DFT of the sequence $x(n) = \{1, -1, 2, -2\}$

Solution: Given sequence is $x(n) = \{1, -1, 2, -2\}$. Here the DFT $X(k)$ to be found is $N = 4$ -point and length of the sequence $L = 4$. So no padding of zeros is required.

The N – Point DFT is given by $X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}$ $0 \leq k \leq N-1$

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)nk} = \sum_{n=0}^3 x(n) e^{-j(\pi/2)nk}, \quad k = 0, 1, 2, 3$$

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

$$\begin{aligned} X(1) &= \sum_{n=0}^3 x(n) e^{-j(\pi/2)n} = x(0) + x(1) e^{-j(\pi/2)} + x(2) e^{-j\pi} + x(3) e^{-j(3\pi/2)} \\ &= 1 + (-1)(0 - j) + 2(-1 - j0) - 2(0 + j) \\ &= -1 - j \end{aligned}$$

$$\begin{aligned}
 X(2) &= \sum_{n=0}^3 x(n) e^{-j\pi n} = x(0) + x(1) e^{-j\pi} + x(2) e^{-j2\pi} + x(3) e^{-j3\pi} \\
 &= 1 - 1(-1 - j0) + 2(1 - j0) - 2(-1 - j0) = 6 \\
 X(3) &= \sum_{n=0}^3 x(n) e^{-j(3\pi/2)n} = x(0) + x(1) e^{-j(3\pi/2)} + x(2) e^{-j3\pi} + x(3) e^{-j(9\pi/2)} \\
 &= 1 - 1(0 + j) + 2(-1 - j0) - 2(0 - j) = -1 + j \\
 X(k) &= \{0, -1 - j, 6, -1 + j\}
 \end{aligned}$$

Description of basic functions:

1. **disp(X)** displays the array, without printing the array name
2. **xlabel('text')** adds text beside the X-axis on the current axis.
3. **ylabel('text')** adds text beside the Y-axis on the current axis.
4. **title('text')** adds text at the top of the current axis.
5. Discrete sequence or "stem" plot. **stem(Y)** plots the data sequence Y as stems from the x axis terminated with circles for the data value.
6. subplot Create axes in tiled positions. **H = subplot(m,n,p)**, or subplot(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.
7. The general form of a for statement is
for variable = expr, statement, ..., statement END
8. **sqrt(X)** is the square root of the elements of X
9. **abs(X)** is the absolute value of the elements of X
10. **angle(H)** returns the phase angles, in radians, of a matrix with complex elements.

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 4: Computation of DFT and IDFT

```
clc;
```

```
close all;
```

```
clear all;
```

% Define the input sequence x(n) and its length N

```
xn=input('Enter the sequence x(n)');
```

```
N=length(xn);
```

% Initialize an array of same size as that of input sequence

```
Xk=zeros(1,N);
```

% Initialize an array of same size as that of input sequence

```
ixk=zeros(1,N);
```

% This code block is describes, to find the DFT of the sequence X(k)

```
i=sqrt(-1);  
  
for k=0:N-1  
    for n=0:N-1  
        Xk(k+1)=Xk(k+1)+(xn(n+1)*exp((-i)*2*pi*k*n/N));  
    end  
end
```

% Plot the input sequence x(n)

```
n=0:N-1;  
subplot(2,2,1);  
stem(n,xn);  
xlabel ('Discrete Time Index -->n');  
ylabel ('Amplitude');  
title('Input Sequence x(n)');
```

%Magnitudes of individual DFT points

```
Magnitude = abs(Xk);  
t=0:N-1;  
subplot(2,2,2);  
stem(n,Magnitude);  
xlabel ('Discrete Frequency Index -->K');  
ylabel ('Amplitude');  
title('Magnitude Response |X(k)|');
```

% Phases of individual DFT points

```
Phase = angle(Xk);  
n=0:N-1;  
subplot(2,2,3);  
stem(n,Phase);  
xlabel ('Discrete Frequency Index -->K');  
ylabel ('Phase');  
title('Phase Response angle(X(k))');
```

% This code block is describes, to find the IDFT of the sequence

```
for n=0:N-1  
    for k=0:N-1  
        ixk(n+1)=ixk(n+1)+(Xk(k+1)*exp(i*2*pi*k*n/N));  
    end  
end  
ixk=ixk ./ N
```

% Plot the Result of Inverse DFT sequence

```
n=0:N-1;  
subplot(2,2,4);  
stem(n,ixk);  
xlabel ('Discrete Time Index -->n');  
ylabel ('Amplitude');  
title('IDFT of the sequence x(n)');
```

Experimental Observations

Input Sequence $x(n)$ is : 1 -1 2 -2

DFT of sequence $x(n)$ is :

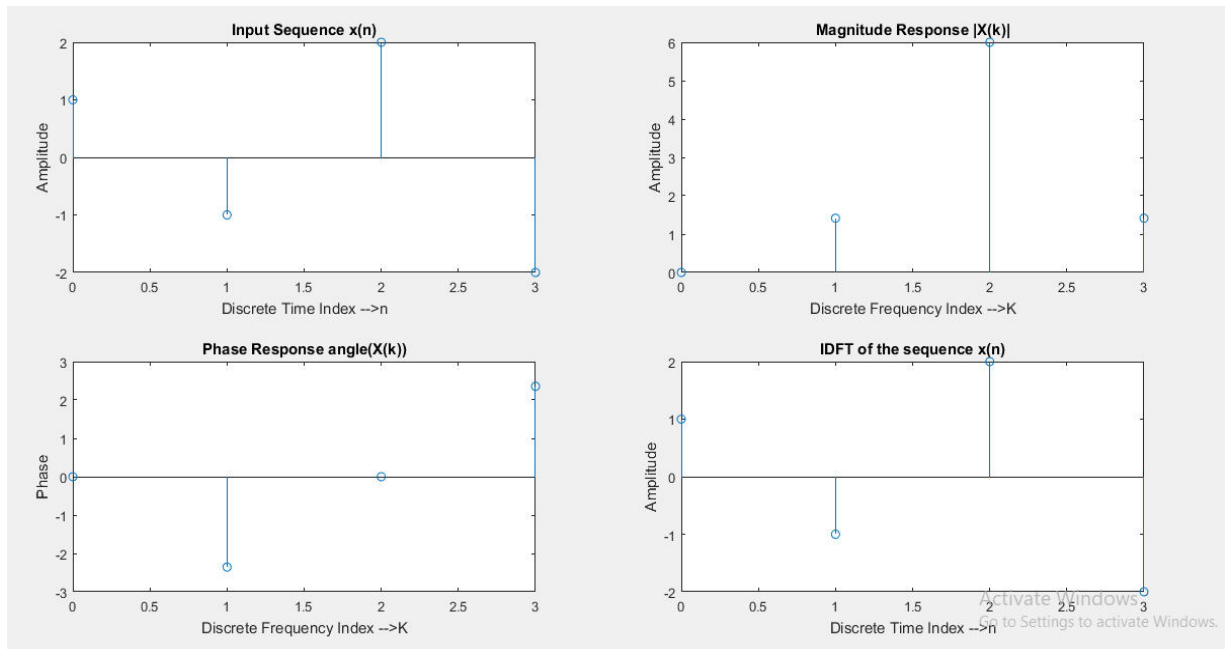
X_k = 0.0000 + 0.0000i -1.0000 - 1.0000i 6.0000 + 0.0000i -1.0000 + 1.0000i

Magnitude Response of DFT of $X(k)$ is : 0 1.4142 6.0000 1.4142

Phase Response of DFT of $X(k)$ is : 0 -2.3562 0.0000 2.3562

IDFT of Sequence $X(K)$ is $x(n)$: 1.0000 -1.0000 2.0000 -2.0000

Output



Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. Write the expressions for N –Point DFT and IDFT.
2. What are different commands available in MATLAB for the computation of DFT?
3. Differentiate between DTFT and DFT.
4. What are the advantages to use DFT in computers rather than DTFT?
5. State any two DFT properties.

Result: The N-Point DFT and IDFT of the given sequence were obtained using MATLAB.

Aim:

To verify the Linear Convolution and Circular Convolution through DFT and IDFT using MATLAB.

Equipment Required:

1. Personal Computer with MATLAB software.

Theory:**Linear Convolution and Circular Convolution through DFT and IDFT approach:**

Convolution is an important operation in DSP, because convolving two sequences in time domain is equivalent to multiplying the sequences in frequency domain. Convolution mainly used in processing signals especially analysing the output of a system. Convolution of two signals $x(n)$ and $h(n)$ is given as

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

Where the symbol $[*]$ is used to denote convolution

An interesting property of the Discrete Fourier Transforms is the effect it has on convolution. Convolution of two signals in the time domain translates to a multiplication of their Fourier transforms in the frequency domain.

Where DFT and IDFT is defined by

Steps involved in Linear Convolution through DFT and IDFT approach:

1. Zero pad $x[n]$ to $N \geq N_1 + N_2 - 1$ elements.
2. Zero pad $h[n]$ to $N \geq N_1 + N_2 - 1$ elements.
3. Compute N-Point DFT $X[k]$ of $x(n)$.
4. Compute N-Point DFT $H[k]$ of $h(n)$.
5. Multiply $Y[k] = X[k]H[k]$, $k = 0, 1, 2, \dots, N-1$
6. Compute N-Point Inverse DFT of $Y[k]$ to get linear convolution between $x(n)$ and $h(n)$

Steps involved in Circular Convolution using DFT and IDFT approach

1. Zero pad $x_1[n]$ to $N \geq \text{Max}(N_1, N_2)$ elements.
2. Zero pad $x_2[n]$ to $N \geq \text{Max}(N_1, N_2)$ elements.
3. Compute N-Point DFT $X_1[k]$ of $x_1(n)$.
4. Compute N-Point DFT $X_2[k]$ of $x_2(n)$.
5. Multiply $X_3[k] = X_1[k]X_2[k]$, $k = 0, 1, 2, \dots, N-1$
6. Compute N-Point Inverse DFT of $X_3[k]$ to get circular convolution between $x_1(n)$ and $x_2(n)$

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 5(a): Linear Convolution using DFT and IDFT

```
clc;
clear all;
close all;
% Define the First Sequence x1(n)
% x1 = [1 2 -1]
x1 = input('Enter the First Sequence x1(n)');
N1 = length(x1);
subplot(3,2,1);
stem(x1);
xlabel('Discrete time-->');
ylabel('Amplitude-->');
title('First Sequence x1(n)');

% Define the Second Sequence x2(n)
% x2 = [1 2 -1]
x2 = input('Enter the Second Sequence x2(n)');
N2 = length(x2);
subplot(3,2,2);
stem(x2);
xlabel('Discrete time-->');
ylabel('Amplitude-->');
title('Second Sequence x2(n)');

% Output Sequence length
N = N1 + N2 - 1;

% Zero Padding for increasing two sequences up to length N
x3=[x1 zeros(1,N-N1)];
x4=[x2 zeros(1,N-N2)];
subplot(3,2,3);
stem(x3);
xlabel('Discrete time-->');
ylabel('Amplitude-->');
title('First Sequence after zero padding x3(n)');
subplot(3,2,4);
stem(x4);
xlabel('Discrete time-->');
ylabel('Amplitude-->');
title('Second Sequence after zero padding x4(n)');

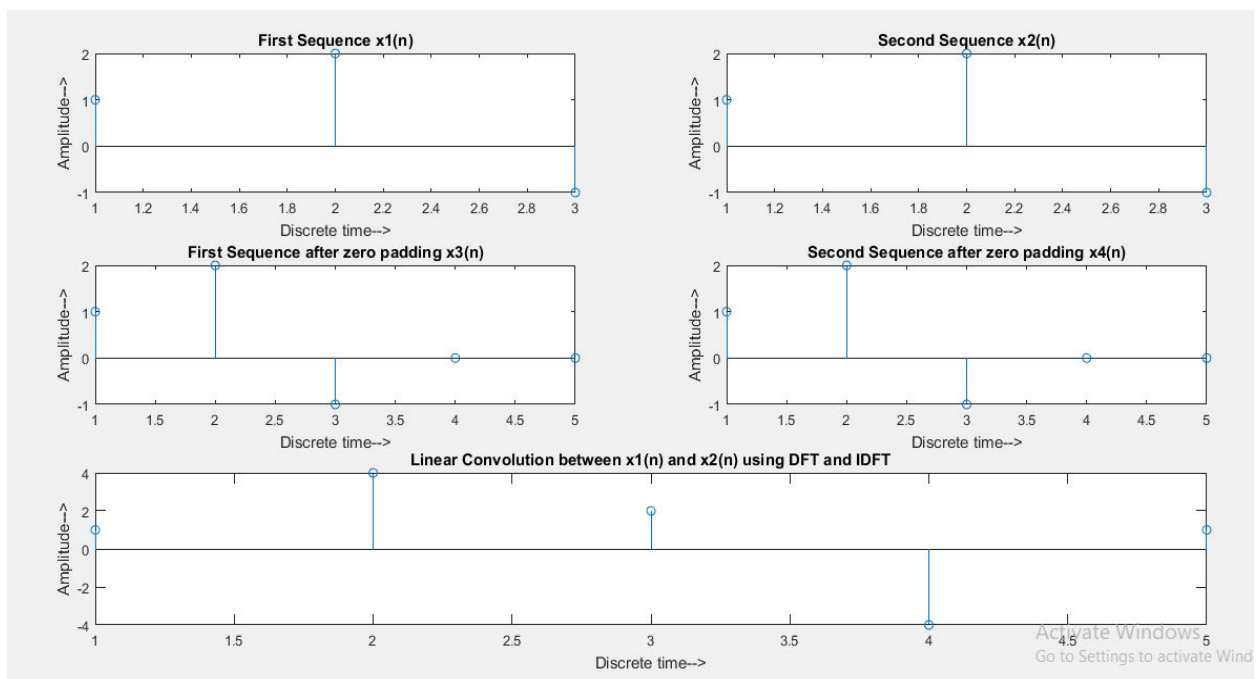
% Calculating the Frequency domains of x3(n) and x4(n) using DFT
X3 = fft(x3,N);
X4 = fft(x4,N);

% According to convolution theorem  $x3(n)*x4(n) \longleftrightarrow X3(k).X4(k)$ 
Y=X3.*X4;
```


% Linear Convolution between $x_1(n)$ and $x_2(n)$ using IDFT

```
y=ifft(Y,N);  
subplot(3,1,3);  
stem(y);  
xlabel('Discrete time-->');  
ylabel('Amplitude-->');  
title('Linear Convolution between  $x_1(n)$  and  $x_2(n)$  using DFT and IDFT');  
  
disp('First Sequence  $x_1(n)$  is ');  
disp(x1);  
disp('Second Sequence  $x_2(n)$  is ');  
disp(x2);  
  
disp('First Sequence after zero padding  $x_3(n)$  is ');  
disp(x3);  
disp('Second Sequence after zero padding  $x_4(n)$  is ');  
disp(x4);  
  
disp('DFT of  $x_3(n)$  is ');  
disp(X3);  
disp('DFT of  $x_4(n)$  is ');  
disp(X4);  
  
disp('Multiplication of  $X_3(k)$  and  $X_4(k)$  is ');  
disp(Y);  
disp('Linear Convolution between  $x_1(n)$  and  $x_2(n)$  using DFT and IDFT is');  
disp(y);
```

Output for Linear Convolution using DFT and IDFT 5(a):



Experimental Observations 5(a):

Enter the First Sequence $x_1(n)$ [1 2 -1]

Enter the Second Sequence $x_2(n)$ [1 2 -1]

First Sequence $x_1(n)$ is : 1 2 -1

Second Sequence $x_2(n)$ is : 1 2 -1

First Sequence after zero padding $x_3(n)$ is : 1 2 -1 0 0

Second Sequence after zero padding $x_4(n)$ is : 1 2 -1 0 0

DFT of $x_3(n)$ is

2.0000, 2.4271 - 1.3143i, -0.9271 - 2.1266i, -0.9271 + 2.1266i, 2.4271 + 1.3143i

DFT of $x_4(n)$ is

2.0000, 2.4271 - 1.3143i, -0.9271 - 2.1266i, -0.9271 + 2.1266i, 2.4271 + 1.3143i

Multiplication of $X_3(k)$ and $X_4(k)$ is

4.0000, 4.1631 - 6.3799i, -3.6631 + 3.9430i, -3.6631 - 3.9430i, 4.1631 + 6.3799i

Linear Convolution between $x_1(n)$ and $x_2(n)$ using DFT and IDFT is

1.0000 4.0000 2.0000 -4.0000 1.0000

Program:

% Experiment 5(b): Circular Convolution using DFT and IDFT

clc;

clear all;

close all;

% Define the First Sequence $x_1(n)$

% $x_1 = [1 \ 2 \ 1 \ 2]$;

$x_1 = \text{input}(\text{'Enter the First Sequence } x_1(n)\text{'})$;

$N_1 = \text{length}(x_1)$;

$\text{subplot}(3,2,1)$;

$\text{stem}(x_1)$;

$\text{xlabel}(\text{'Discrete time-->'})$;

$\text{ylabel}(\text{'Amplitude-->'})$;

$\text{title}(\text{'First Sequence } x_1(n)\text{'})$;

% Define the Second Sequence $x_2(n)$

% $x_2 = [4 \ 3 \ 2 \ 1]$

$x_2 = \text{input}(\text{'Enter the Second Sequence } x_2(n)\text{'})$;

$N_2 = \text{length}(x_2)$;

$\text{subplot}(3,2,2)$;

$\text{stem}(x_2)$;

$\text{xlabel}(\text{'Discrete time-->'})$;

$\text{ylabel}(\text{'Amplitude-->'})$;

$\text{title}(\text{'Second Sequence } x_2(n)\text{'})$;

% Output Sequence length

$N = \max(N_1, N_2);$

% Zero Padding for increasing two sequences up to length N

$x_3 = [x_1 \text{ zeros}(1, N-N_1)];$

$x_4 = [x_2 \text{ zeros}(1, N-N_2)];$

subplot(3,2,3);

stem(x3);

xlabel('Discrete time-->');

ylabel('Amplitude-->');

title('First Sequence after zero padding $x_3(n)$ ');

subplot(3,2,4);

stem(x4);

xlabel('Discrete time-->');

ylabel('Amplitude-->');

title('Second Sequence after zero padding $x_4(n)$ ');

% Calculating the Frequency domains of $x_3(n)$ and $x_4(n)$

$X_3 = \text{fft}(x_3, N);$

$X_4 = \text{fft}(x_4, N);$

% According to convolution theorem $x_3(n)*x_4(n) \longleftrightarrow X_3(k).X_4(k)$

$Y = X_3.*X_4;$

% Circular Convolution between $x_1(n)$ and $x_2(n)$ using DFT and IDFT

$y = \text{ifft}(Y, N);$

subplot(3,1,3);

stem(y);

xlabel('Discrete time-->');

ylabel('Amplitude-->');

title('Circular Convolution between $x_1(n)$ and $x_2(n)$ using DFT and IDFT');

disp('First Sequence $x_1(n)$ is ');

disp(x1);

disp('Second Sequence $x_2(n)$ is ');

disp(x2);

disp('First Sequence after zero padding $x_3(n)$ is ');

disp(x3);

disp('Second Sequence after zero padding $x_4(n)$ is ');

disp(x4);

disp('DFT of $x_3(n)$ is ');

disp(X3);

disp('DFT of $x_4(n)$ is ');

disp(X4);

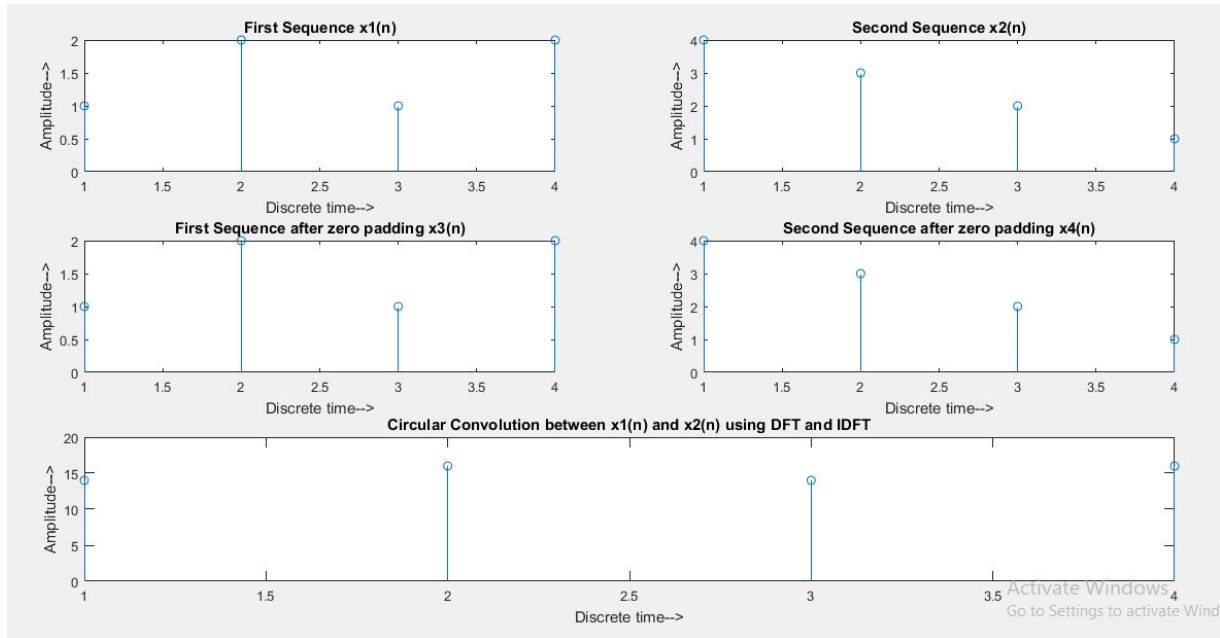
disp('Multiplication of $X_3(k)$ and $X_4(k)$ is ');

disp(Y);

disp('Circular Convolution between $x_1(n)$ and $x_2(n)$ using DFT and IDFT is');

disp(y);

Output 5(b):



Experimental Observations 5(b)

Enter the First Sequence $x_1(n)$ [1 2 1 2]

Enter the Second Sequence $x_2(n)$ [4 3 2 1]

First Sequence $x_1(n)$ is : 1 2 1 2

Second Sequence $x_2(n)$ is : 4 3 2 1

First Sequence after zero padding $x_3(n)$ is : 1 2 1 2

Second Sequence after zero padding $x_4(n)$ is : 4 3 2 1

DFT of $x_3(n)$ is : 6 0 -2 0

DFT of $x_4(n)$ is : 10.0000, 2.0000 - 2.0000i, 2.0000, 2.0000 + 2.0000i

Multiplication of $X_3(k)$ and $X_4(k)$ is : 60 0 -4 0

Circular Convolution between $x_1(n)$ and $x_2(n)$ using DFT and IDFT is : 14 16 14 16

Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Result:

Linear convolution and Circular convolution through DFT and IDFT was verified.

Aim:

To compute power spectral density (PSD) for sinusoidal signal using MATLAB.

Equipment Required:

Personal Computer with MATLAB software.

Theory:

Power spectral density function (PSD) shows the strength of the variations (energy) as a function of frequency. In other words, it shows at which frequencies variations are strong and at which frequencies variations are weak. The unit of PSD is energy (variance) per frequency (width) and you can obtain energy within a specific frequency range by integrating PSD within that frequency range.

Power Spectral Density (PSD) is the frequency response of a random or periodic signal. It tells us where the average power is distributed as a function of frequency. The PSD is deterministic, and for certain types of random signals is independent of time. This is useful because the Fourier transform of a random time signal is itself random, and therefore of little use calculating transfer relationships (i.e., finding the output of a filter when the input is random). The PSD of a random time signal $x(t)$ can be expressed in one of two ways that are equivalent to each other.

The PSD is the average of the Fourier transform magnitude squared, over a large time interval

$$S_x(f) = \lim_{T \rightarrow \infty} E \left\{ \frac{1}{2T} \left| \int_{-T}^T x(t) e^{-j2\pi ft} dt \right|^2 \right\}$$

The PSD is the Fourier transform of the auto-correlation function.

$$S_x(f) = \int_{-T}^T R_x(\tau) e^{-j2\pi f\tau} d\tau$$

$$R_x(\tau) = E\{x(t)x^*(t + \tau)\}$$

The power can be calculated from a random signal over a given band of frequencies as follows:

$$P = \int_{-\infty}^{\infty} S_x(f) df = R_x(0)$$

1. Total Power in $x(t)$:

$$P_{12} = \int_{f_1}^{f_2} S_x(f) df = R_x(0)$$

2. Power in $x(t)$ in range $f_1 - f_2$:

The signal has to be stationary, which means that its statistics do not change as a function of time.

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 6: Power Spectral Density

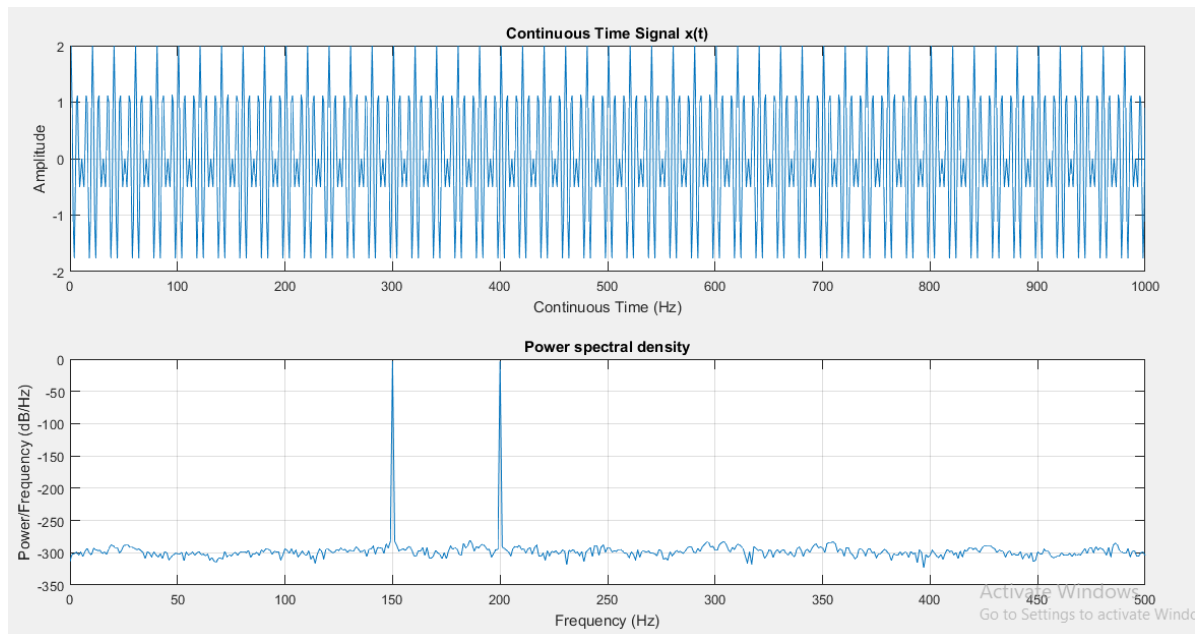
```
clc;
clear all;
close all;

% Define the Sampling Frequency
Fs = 1000;
% Define the Time instants of x(t)
t = 0:1/Fs:1-1/Fs;
% Define the CT signal x(t)
x = cos(2*pi*200*t) % + cos(2*pi*150*t);
% Plot the input signal x(t)
subplot(2,1,1);
plot(x);
grid on
xlabel('Continuous Time (Hz)');
ylabel('Amplitude');
title('Continuous Time Signal x(t)');

% Computation of Power Spectral Density of x(t)
N = length(x);
xdft = fft(x);
xdft = xdft(1:N/2+1);
psdx = (1/(Fs*N)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);

% Plot the PSD
freq = 0:Fs/length(x):Fs/2;
subplot(2,1,2);
plot(freq,10*log10(psdx));
grid on
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
title('Power spectral density');
```

Output



Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Result:

Power Spectral Density for a given sinusoidal signal was computed using MATLAB.

.

Design of Digital IIR Butterworth filter using Bi-linear Transformation.	EXPT. NO: 7
	DATE:

Aim:

To design the Digital IIR Low Pass Filters using bilinear transformation technique for a given specification using Butterworth approximation.

Equipment Required:

Personal Computer with MATLAB software.

Theory:

IIR filters are of recursive type, where the present output samples depends on the present input, past input samples and output samples.

The IIR digital filters have the transfer function of the form

$$H(z) = \frac{\sum_{k=0}^M b_k Z^{-k}}{1 + \sum_{k=0}^N a_k Z^{-k}}$$

Designing Procedure for IIR Low Pass filter using Butterworth Approximation and Bi-linear Transformation

For a given Digital filter specifications ω_p , ω_s , α_p , α_s

ω_p = Pass Band Frequency

ω_s = Stop Band Frequency

α_p = Pass Band Ripple

α_s = Stop Band Ripple

- Convert the Digital frequencies into analog frequencies by using Bilinear transformation

$$\Omega'_p = \frac{2}{T} \tan\left(\frac{\omega_p}{2}\right) \quad \text{and} \quad \Omega'_s = \frac{2}{T} \tan\left(\frac{\omega_s}{2}\right)$$

- Convert the all frequencies into radians per seconds using

$$\Omega_p = 2 \frac{2\pi\Omega'_p}{f_{smp}} \quad \text{and} \quad \Omega_s = 2 \frac{2\pi\Omega'_s}{f_{smp}}$$

- The order of the filter is determined by using

$$N = \frac{\log \sqrt{\frac{10^{0.1\alpha_s} - 1}{10^{0.1\alpha_p} - 1}}}{\log \frac{\Omega_s}{\Omega_p}} \quad \text{or} \quad \frac{\log \sqrt{\frac{A_s^{-2} - 1}{A_p^{-2} - 1}}}{\log \frac{\Omega_s}{\Omega_p}}$$

- The Cut-off frequency Ω_c is given by

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

- The Poles of the Butterworth filter is given by

$$S_K = e^{i\Phi_K}, \quad K = 1, 2, \dots, N$$

$$\Phi_K = \frac{\pi}{2} + \frac{(2K-1)\pi}{2N}$$

The poles of the Filter are given by $P_K = S - S_K$

(OR)

- The Normalized Denominator polynomials are given by for various values of N

N	D(s) = Denominator Polynomial of H(s)
1	$S+1$
2	$S^2 + \sqrt{2}S + 1$
3	$(S+1)(S^2 + S + 1)$
4	$(S^2 + 0.76537S + 1)(S^2 + 1.8477S + 1)$
5	$(S+1)(S^2 + 0.61803S + 1)(S^2 + 1.61803S + 1)$
6	$(S^2 + 1.931855S + 1)(S^2 + \sqrt{2}S + 1)(S^2 + 0.51764S + 1)$

- Determine the transfer function of Analog low pass Butterworth filter

$$H(s) = \frac{1}{D(s)}, \quad \text{where } S = \frac{S}{\Omega_C}$$

(OR)

Case (i): if N = Even

$$H(s) = \prod_{K=1}^{\frac{N}{2}} \frac{\Omega_C^2}{S^2 + b_K \Omega_C S + \Omega_C^2}$$

Case (i): if N = Odd

$$H(s) = \prod_{K=1}^{\frac{N-1}{2}} \left[\frac{\Omega_C^2}{S^2 + b_K \Omega_C S + \Omega_C^2} \right] \left[\frac{\Omega_C}{S + \Omega_C} \right]$$

$$\text{Where } b_K = 2 \sin\left(\frac{(2k-1)\pi}{2N}\right)$$

- Determine the transfer function H(z) from Analog low pass Butterworth filter transfer function H(s) using

$$H(z) = H(s) \text{ at } s \rightarrow \frac{2}{T} \left[\frac{1 - Z^{-1}}{1 + Z^{-1}} \right]$$

Description of basic functions:

1. [N, Wn] = buttord(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Butterworth filter which has a pass band ripple of no more than Rp dB and a stop band attenuation of at least Rs dB. Wp and Ws are the pass band and stop band edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample) and **[N, Wn] = buttord(Wp, Ws, Rp, Rs, 's')** does the computation for an analog filter, in which case Wp and Ws are in radians/second.

2. butter Butterworth digital and analog filter design **[B,A] = butter(N,Wn)** designs an Nth order low pass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be $0.0 < Wn < 1.0$, with 1.0 corresponding to half the sample rate.

[B,A] = butter(N,Wn,'high') designs a high pass filter.

[B,A] = butter(N,Wn,'low') designs a low pass filter.

[B,A] = butter(N,Wn,'bandpass') is a band stop filter if $W_n = [W1 \ W2]$.

[B,A] = butter(N,Wn,'stop') is a band stop filter if $W_n = [W1 \ W2]$.

3. **bilinear** Bilinear transformation with optional frequency prewarping

[NUMd, DEND] = **bilinear**(NUM,DEN,Fs), where NUM and DEN are row vectors containing numerator and denominator transfer function coefficients, NUM(s)/DEN(s), in descending powers of s, transforms to z-transform coefficients NUMd(z)/DEND(z)

4. **freqz** Frequency response of digital filter

[H,W] = **freqz**(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter and given numerator and denominator coefficients in vectors B and A.

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 7: Design of Digital IIR Low Pass filter using Butterworth Approximation and Bi-linear Transformation

% % Design of Digital IIR filter using Bilinear Transformation % %

clc;

clear all;

close all;

% % Read the Digital Filter Specifications % %

disp('Enter the IIR Digital Filter Design Specifications');

Ap = input('Enter the Pass Band Attenuation in dB');

As = input('Enter the Stop Band Attenuation in dB');

wp = input('Enter the Pass Band Edge (Digital) Frequency in Hz');

ws = input('Enter the Stop Band Edge (Digital) Frequency in Hz');

fs = input('Enter the Sampling Frequency in Hz');

% % Conversion of Digital Specifications into radians per second % %

wp1 = 2*wp/fs **% Pass Band Frequency in terms of radians per second**

ws2 = 2*ws/fs **% Stop Band Frequency in terms of radians per second**

% % Conversion of Digital Specifications into Analog Specifications using Bilinear Transformation % %

w1 = 2*fs*tan(wp1/2)

w2 = 2*fs*tan(ws2/2)

% % Compute the Order, Cutoff frequency and Analog filter coefficients using Butterworth Approximation % %

```
[N,W] = buttord(w1,w2,Ap,As,'s')
```

```
c = input('Enter the Choice of Analog Filter 1. LPF 2. HPF 3.BPF 4.BSF \n ');
```

% % Determine the Filter Coefficients % %

```
if(c==1)
```

```
disp('Frequency Response of IIR LPF is:');
```

```
[nums,dens]=butter(N,W,'low','s')
```

```
end
```

```
if(c==2)
```

```
disp('Frequency Response of IIR HPF is:');
```

```
[nums,dens]=butter(N,W,'high','s')
```

```
end
```

```
if(c==3)
```

```
disp('Frequency Response of IIR BPF is:');
```

```
[nums,dens]=butter(N,[w1,w2],'bandpass','s')
```

```
end
```

```
if(c==4)
```

```
disp('Frequency Response of IIR BSF is:');
```

```
[nums,dens]=butter(N,[w1,w2],'stop','s')
```

```
end
```

% % Analog IIR filter transfer function (H(s)) converted to Digital IIR Filter transfer function (H(z)) using Bilinear Transformation % %

```
[numd dend]=bilinear(nums, dens, fs);
```

% % Obtaining the Magnitude and Phase Response of filter % %

```
w = 0:0.1:pi;
```

```
[h,om] = freqz(numd,dend,w);
```

% % Obtaining the Magnitude Response of filter % %

```
m=20*log10(abs(h));
```

% % Obtaining the Phase Response of filter % %

```
an=angle(h);
```

% % Plotting the Magnitude and Phase Response of the Filter % %

```
figure;
```

```
subplot(2,1,1);
```

```
plot(om/pi,m);
```

```
xlabel('(a) Normalized freq. -->');
```

```
ylabel('Gain in dB-->');
```

```
title('Magnitude Response of IIR Digital Filter');
```

```
subplot(2,1,2);
```

```
plot(om/pi,an);
```

```
xlabel('(b) Normalized freq. -->');
```

```
ylabel('Phase in radians-->');
```

```
title('Phase Response of IIR Digital Filter');
```

Input and Output:

Enter the IIR Digital Filter Design Specifications

Enter the Pass Band Attenuation in dB 20

Enter the Stop Band Attenuation in dB 50

Enter the Pass Band Edge (Digital) Frequency in Hz 400

Enter the Stop Band Edge (Digital) Frequency in Hz 800

Enter the Sampling Frequency in Hz 2000

Digital Specifications into radians per second

$\omega_{p1} = 0.4000 \text{ rad/s}$ and $\omega_{s2} = 0.8000 \text{ rad/s}$

Analog filter specifications

$\omega_1 = 810.8401$ and $\omega_2 = 1.6912 \times 10^3$

Order of the filter $N = 5$

Cutoff frequency $W = 534.7964$

Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Result:

The Butterworth Digital IIR Low Pass Filters were designed for given specifications using Bilinear transformation.

Design of Digital IIR Chebyshev filter using Bi-linear Transformation	EXPT. NO: 8
	DATE:

Aim:

To design the Digital IIR Low Pass Filters using bilinear transformation technique for a given specification using Butterworth approximation.

Equipment Required:

Personal Computer with MATLAB software.

Theory:

IIR filters are of recursive type, where the present output samples depends on the present input, past input samples and output samples.

The IIR digital filters have the transfer function of the form

$$H(z) = \frac{\sum_{k=0}^M b_k Z^{-k}}{1 + \sum_{k=0}^N a_k Z^{-k}}$$

Designing Procedure for IIR Low Pass filter using Chebyshev Approximation

The Magnitude Square response of Nth order type-I Chebyshev filter is given by

$$|H(j\Omega)| = \frac{1}{\left[1 + \varepsilon^2 C_N^2\left(\frac{\Omega}{\Omega_p}\right) \right]}, \quad N = 1, 2, 3, \dots$$

Where ε is a parameter of the filter related to the ripple in the pass band

and $C_N(x)$ is Nth order Chebyshev polynomial and is defined as

$$C_N(x) = \cos(N \cos^{-1} x), \quad |x| \leq 1 \quad (\text{Pass Band})$$

$$C_N(x) = \cosh(N \cosh^{-1} x), \quad |x| > 1 \quad (\text{Stop Band})$$

- Type – I Chebyshev filters are all pole filters that exhibits equi ripple behaviour in the pass band and a monotonic characteristics in the stop band.
- Type – II Chebyshev filters contains both poles and zeros and exhibits a monotonic behaviour in the pass band and equi ripple behaviour in the stop band.
- The poles of the Chebyshev filters lie on an ellipse.
- The number of poles are less compared to Butterworth filter for same order, so that less number of discrete components are required to construct the filter.

For a given filter specifications $\Omega_p, \Omega_s, \alpha_p, \alpha_s$

Ω_p = Pass Band Frequency

Ω_s = Stop Band Frequency

α_p = Pass Band Ripple

α_s = Stop Band Ripple

- The order of the filter is determined by using the below equation and round off to nearest integer value.

$$N \geq \frac{\cosh^{-1} \sqrt{\frac{10^{0.1\alpha_s} - 1}{10^{0.1\alpha_p} - 1}}}{\cosh^{-1} \frac{\Omega_s}{\Omega_p}} \quad N \geq \frac{\cosh^{-1} A}{\cosh^{-1} \frac{1}{K}}$$

- The minor and major axis values a & b of ellipse are given by

$$a = \Omega_p \left[\frac{\mu^{1/N} - \mu^{-1/N}}{2} \right] \quad b = \Omega_p \left[\frac{\mu^{1/N} + \mu^{-1/N}}{2} \right]$$

$$\mu = \varepsilon^{-1} + \sqrt{\varepsilon^{-2} + 1} \quad \text{and} \quad \varepsilon = \sqrt{10^{0.1\alpha_p} - 1}$$

- The Poles of the Chebyshev filter is given by

$$S_K = a \cos \Phi_K + j b \sin \Phi_K, \quad K = 1, 2, \dots, N$$

$$\Phi_K = \frac{\pi}{2} + \frac{(2K-1)\pi}{2N}$$

- The denominator polynomial D(s) is given by D(s) = (S-S_K)

- The numerator Polynomial N(s) is given by

(a). For N Odd, substitute S = 0 in the denominator polynomial D(s) and find the the numerator N(s) value of the filter transfer function.

(b). For N Even, Substitute S = 0 in the denominator polynomial D(s) and divide the result by $\sqrt{1 + \varepsilon^2}$

- Determine the transfer function of H(s) Analog low pass Chebyshev filter

$$H(s) = \frac{N(s)}{D(s)}$$

(OR)

Case(i): if N = Even

$$H(s) = \prod_{K=1}^{\frac{N}{2}} \frac{B_K \Omega_C^2}{S^2 + b_K \Omega_C S + C_K \Omega_C^2}$$

Case(i): if N = Odd

$$H(s) = \prod_{K=1}^{\frac{N-1}{2}} \left[\frac{B_K \Omega_C^2}{S^2 + b_K \Omega_C S + C_K \Omega_C^2} \right] \left[\frac{B_0 \Omega_C}{S + C_0 \Omega_C} \right]$$

Where $b_K = 2 y_n \sin\left(\frac{(2k-1)\pi}{2N}\right)$, $C_K = y_n^2 + \cos^2\left[\frac{(2k-1)\pi}{2N}\right]$, $C_0 = y_n$ and

$$y_n = \frac{1}{2} \left[\left[\left(\frac{1}{\varepsilon^2} + 1 \right)^{1/2} + \frac{1}{\varepsilon} \right]^{1/N} - \left[\left(\frac{1}{\varepsilon^2} + 1 \right)^{1/2} + \frac{1}{\varepsilon} \right]^{-1/N} \right]$$

To get B_K values, use

Case(i): if N = Even $B_0 = H(0) = H(s) : s=0 = A_p = \frac{1}{\sqrt{1 + \varepsilon^2}} = B_K$

Case(ii): if N = Odd $B_0 = H(0) = H(s) : s=0 = 1_K$

$$B_0 = B_1 = B_2 = \dots = B_K$$

Description of basic functions:

Type I Filter Design

1. **[N, Wp] = cheb1ord(Wp, Ws, Rp, Rs)** returns the order N of the lowest order digital chebyshev Type-I filter which has a pass band ripple of no more than Rp dB and a stop band attenuation of at least Rs dB. Wp and Ws are the pass band and stop band edge frequencies, normalized from 0 to 1 (where 1 corresponds to π radians/sample) and

[N, Wp] = cheb1ord(Wp, Ws, Rp, Rs, 's') does the computation for an analog filter, in which case Wp and Ws are in radians/second.

2. **cheby1** Chebyshev Type-I digital and analog filter design **[B,A] = cheby1(N,Wp,Rp)** designs an Nth order low pass Chebyshev type-I filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be $0.0 < Wn < 1.0$, with 1.0 corresponding to half the sample rate.

[B,A] = cheby1(N,Wp,Rp,'high') designs a high pass filter.

[B,A] = cheby1(N,Wp,Rp,'low') designs a low pass filter.

[B,A] = vheby1(N,[Wp, Ws], Rp,'bandpass') is a band stop filter

[B,A] = cheby1(N,[Wp, Ws],'stop') is a band stop filter

Type II Filter Design

3. **[N, Ws] = cheb2ord(Wp, Ws, Rp, Rs)** returns the order N of the lowest order digital Chebyshev Type-II filter which has a pass band ripple of no more than Rp dB and a stop band attenuation of at least Rs dB. Wp and Ws are the pass band and stop band edge frequencies, normalized from 0 to 1 (where 1 corresponds to π radians/sample) and

[N, Ws] = cheb2ord(Wp, Ws, Rp, Rs, 's') does the computation for an analog filter, in which case Wp and Ws are in radians/second.

4. **cheby2** Chebyshev Type-II digital and analog filter design **[B,A] = cheby2(N,Ws,Rs)** designs an Nth order low pass Chebyshev type-II filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be $0.0 < Wn < 1.0$, with 1.0 corresponding to half the sample rate.

[B,A] = cheby2(N,Ws,Rs,'high') designs a high pass filter.

[B,A] = cheby2(N,Ws,Rs,'low') designs a low pass filter.

[B,A] = vheby2(N, Rs, [Wp, Ws], Rp,'bandpass') is a band stop filter

[B,A] = cheby2(N, Rs,[Wp, Ws],'stop') is a band stop filter

5. **bilinear** Bilinear transformation with optional frequency prewarping

[NUMd, DEND] = bilinear(NUM,DEN,Fs), where NUM and DEN are row vectors containing numerator and denominator transfer function coefficients, NUM(s)/DEN(s), in descending powers of s, transforms to z-transform coefficients NUMd(z)/DEND(z)

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 8: Design of Digital IIR Low Pass filter using Chebyshev (Type-I)

Approximation and Bi-linear Transformation

% % Design of Digital IIR filter using Bilinear Transformation % %

clc;

clear all;

close all;

% % Read the Digital Filter Specifications % %

disp('Enter the IIR Digital Filter Design Specifications');

Ap = input('Enter the Pass Band Attenuation in dB');

As = input('Enter the Stop Band Attenuation in dB');

wp = input('Enter the Pass Band Edge (Digital) Frequency in Hz');

ws = input('Enter the Stop Band Edge (Digital) Frequency in Hz');

fs = input('Enter the Sampling Frequency in Hz');

% % Conversion of Digital Specifications into radians per second % %

wp1 = 2*wp/fs **% Pass Band Frequency in terms of radians per second**

ws2 = 2*ws/fs **% Stop Band Frequency in terms of radians per second**

% % Conversion of Digital Specifications into Analog Specifications using Bilinear Transformation % %

w1 = 2*fs*tan(wp1/2)

w2 = 2*fs*tan(ws2/2)

% % Compute the Order, Cutoff frequency and Analog filter coefficients using Chebyshev-I Approximation % %

[N,W] = cheb1ord(w1,w2,Ap,As,'s')

c = input('Enter the Choice of Analog Filter 1. LPF 2. HPF 3.BPF 4.BSF \n ');

% % Determine the Filter Coefficients % %

if(c==1)

disp('Frequency Response of IIR LPF is:');

[nums,dens]=cheby1(N,W,'low','s')

end

if(c==2)

disp('Frequency Response of IIR HPF is:');

[nums,dens]=cheby1(N,W,'high','s')

end

if(c==3)

disp('Frequency Response of IIR BPF is:');

[nums,dens]=cheby1(N,[w1,w2],'bandpass','s')

end

if(c==4)

disp('Frequency Response of IIR BSF is:');

[nums,dens]=cheby1(N,[w1,w2],'stop','s')

end

% % Analog IIR filter transfer function (H(s)) converted to Digital IIR Filter transfer function (H(z)) using Bilinear Transformation % %

[numd dend]=bilinear(nums, dens, fs);

% % Obtaining the Magnitude and Phase Response of filter % %

w = 0:0.1:pi;

[h,om] = freqz(numd,dend,w);

% % Obtaining the Magnitude Response of filter % %

m=20*log10(abs(h));

% % Obtaining the Phase Response of filter % %

an=angle(h);

% % Plotting the Magnitude and Phase Response of the Filter % %

figure;

subplot(2,1,1);

plot(om/pi,m);

xlabel('(a) Normalized freq. -->');

ylabel('Gain in dB-->');

title('Magnitude Response of IIR Digital Filter');

subplot(2,1,2);

plot(om/pi,an);

xlabel('(b) Normalized freq. -->');

ylabel('Phase in radians-->');

title('Phase Response of IIR Digital Filter');

Input and Output:

Enter the IIR Digital Filter Design Specifications

Enter the Pass Band Attenuation in dB 20

Enter the Stop Band Attenuation in dB 50

Enter the Pass Band Edge (Digital) Frequency in Hz 400

Enter the Stop Band Edge (Digital) Frequency in Hz 800

Enter the Sampling Frequency in Hz 2000

Digital Specifications into radians per second

wp1 = 0.4000 rad/s and ws2 = 0.8000 rad/s

Analog filter specifications

w1 = 810.8401 and w2 = 1.6912e+03

Order of the filter N = 4

Cutoff frequency W = 534.7964

Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Result:

The Chebyshe-I approximation of Digital IIR Low Pass Filters were designed for given specifications using Bilinear transformation.

<p align="center">11. Design of Digital FIR Filters (LPF, HPF, BPF AND BSF)</p>	<p>EXPT. NO: 11</p> <hr/> <p>DATE:</p>
---------------------------------------------------------------------------------------------------	------------------------------------------------------

Aim:

To design the Digital FIR Low Pass Filters using rectangular window technique for a given specification.

Equipment Required:

Personal Computer with MATLAB software.

Theory:

FIR filters are of recursive type, where the present output samples depends on the present input, past input samples.

The FIR digital filters have the transfer function of the form

$$H(z) = \sum_{k=0}^M b_k Z^{-k}$$

Designing Procedure for FIR filter using window technique

- Choose the desired frequency response $H_d(e^{j\omega})$
- Apply Inverse DTFT to obtain the impulse response of the desired FIR filter

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

- Choose the window sequence $w(n)$ and multiply with $h_d(n)$, to obtain the impulse response of the FIR filter $h(n)$

$$h(n) = h_d(n) w(n) \quad , \quad 0 \leq n \leq N-1$$

- Determine the transfer function of FIR filter by applying the Z-Transforms

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

- Realize the Transfer function $H(z)$ using Direct Form or Cascaded form or Linear Phase Realization.
- Determine the frequency response of FIR filter

$$H(e^{j\omega}) = H(z) \Big|_{z=e^{j\omega}}$$

Description of basic functions:

1. **disp(X)** displays the array, without printing the array name
2. **xlabel('text')** adds text beside the X-axis on the current axis.
3. **ylabel('text')** adds text beside the Y-axis on the current axis.
4. **title('text')** adds text at the top of the current axis.
5. Discrete sequence or "stem" plot. **stem(Y)** plots the data sequence Y as stems from the x axis terminated with circles for the data value.
6. subplot Create axes in tiled positions. **H = subplot(m,n,p)**, or subplot(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.
7. **abs(X)** is the absolute value of the elements of X.

8. (a) **W = rectwin(N)** returns the N-point rectangular window.
- (b). **W = triang(N)** returns the N-point triangular window.
- (c). **W = hanning(N)** returns the N-point symmetric Hanning window
- (d). **W = hamming(N)** returns the N-point symmetric Hamming window
- (e). **W = kaiser(N)** returns an N-point Kaiser window
9. **fir1** FIR filter design using the window method.
B = fir1(N,Wn) designs an N'th order lowpass FIR digital filter and returns the filter coefficients in length N+1 vector B. The cut-off frequency Wn must be between $0 < Wn < 1.0$
B = fir1(N,Wn,'high') designs an N'th order high pass filter.
B = fir1(n,[w1,w2],'bandpass',window) designs a Bandpass filter
B = fir1(n,[w1,w2],'stop',window) designs a Band stop filter
10. **freqz** Frequency response of digital filter
[H,W] = freqz(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter and given numerator and denominator coefficients in vectors B and A.
11. **angle(H)** returns the phase angles, in radians, of a matrix with complex elements.

Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window.

Program:

% Experiment 11: Design of Digital FIR filter using window techniques (Rectangular Window)

% Design of Digital FIR filters using Rectangular Window Method

```

clc;
clear all;
close all;
warning off;
disp('Enter the FIR filter Design Specifications');

%% % Read the Filter Specifications %%
n = input('Enter the order or no of samples');
wp = input('Enter the Pass Band Edge Frequency');
ws = input('Enter the Stop Band Edge Frequency');
fs = input('Enter the Sampling Frequency');
w1 = 2*wp/fs           % Pass Band Edge Frequency in terms of radians per second
w2 = 2*ws/fs           % Stop Band Edge Frequency in terms of radians per second
%% % Window Function Definition %%
window = rectwin(n+1);
c=input('Enter the Choice of Digital FIR Filter 1. LPF 2. HPF 3.BPF 4.BSF \n ');
%% % Determine the Filter Coefficients %%
if(c==1)
disp('Frequency Response of FIR LPF is:');
b = fir1(n,w1,'low',window);

```

```

end
if(c==2)
disp('Frequency Response of FIR HPF is:');
b = fir1(n,w1,'high',window);
end
if(c==3)
disp('Frequency Response of FIR BPF is:');
b = fir1(n,[w1,w2],'bandpass',window);
end
if(c==4)
disp('Frequency Response of FIR BSF is:');
b = fir1(n,[w1,w2],'stop',window);
end

%% Obtaining the Magnitude and Phase Response of filter %%
w = 0:0.01:pi;
[h,om]=freqz(b,1,w);

%% Obtaining the Magnitude Response of filter %%
m = 20*log10(abs(h));

%% Obtaining the Phase Response of filter %%
an = angle(h);
figure;
subplot(2,1,1);
plot(om/pi,m);
xlabel('(a) Normalized Frequency -->');
ylabel('Gain in dB-->');
title('Magnitude Response of FIR filter');
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b) Normalized Frequency -->');
ylabel('Phase in Radians-->');
title('Phase Response of FIR filter');

```

Input and Output:

Enter the FIR Digital Filter Design Specifications

Enter the order or no of samples 20

Enter the Pass Band Edge Frequency 100

Enter the Stop Band Edge Frequency 200

Enter the Sampling Frequency 1000

w1 = 0.2000 rad /sec

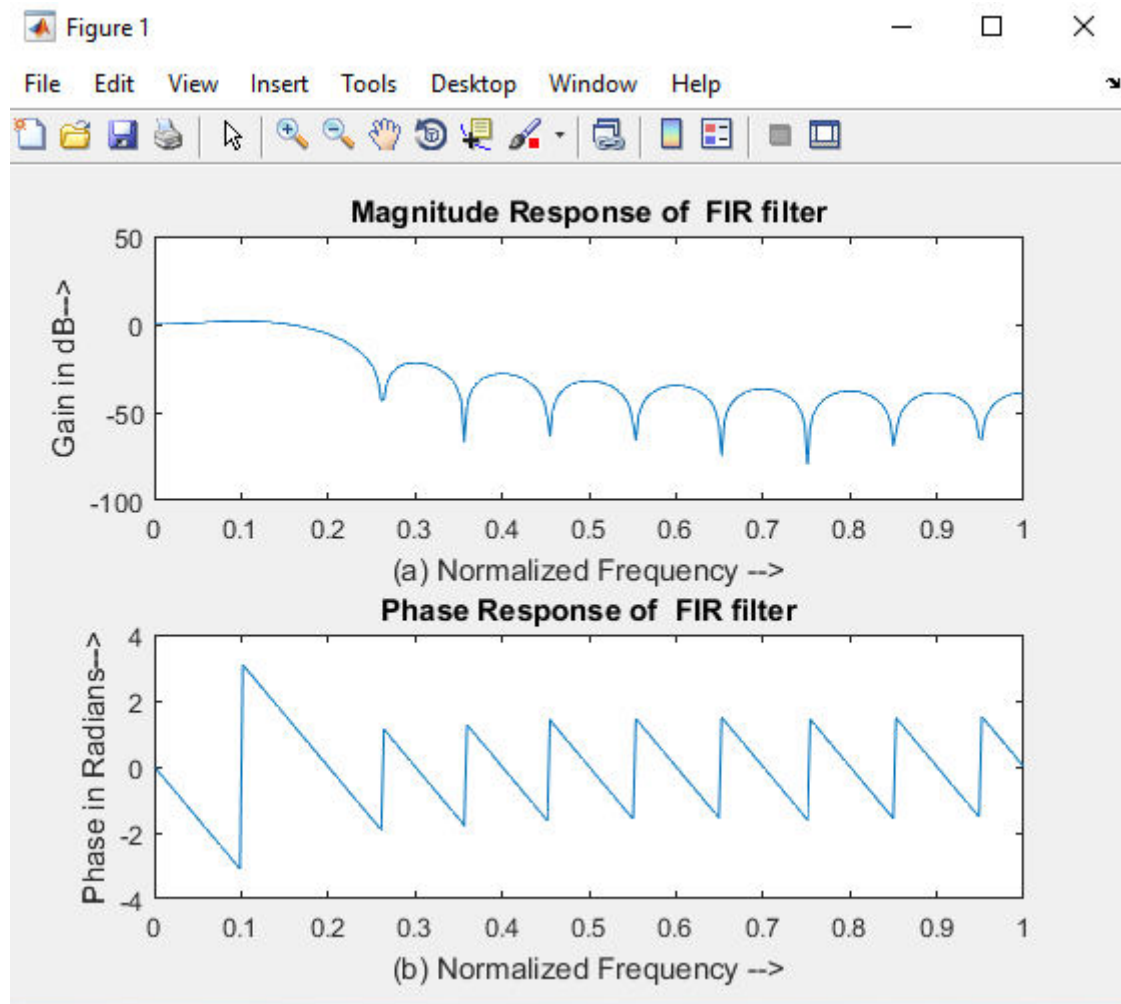
w2 = 0.4000 rad /sec

Enter the Choice of Digital FIR Filter 1. LPF 2. HPF 3.BPF 4.BSF

1

Frequency Response of FIR LPF is:

-0.0000 -0.0229 -0.0418 -0.0477 -0.0344 0.0000 0.0516 0.1114 0.1670 0.2065
0.2207 0.2065 0.1670 0.1114 0.0516 0.0000 -0.0344 -0.0477 -0.0418 -0.0229
-0.0000



Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. What are advantages of FIR Filter?
2. What are different windows for fir filter?
3. What is the need of usage of windows?
4. What are different steps required to execute the DSP program in Matlab.
5. What is the command used for FIR filter implementation.
6. Write the expressions for Rectangular window, Hanning window, Hamming window, Triangular window.
7. Explain the steps used in design of FIR filter using Fourier series method.

Result:

The FIR Filters were designed for given specifications using windows

Aim:

To verify the linear convolution between input sequence $x(n)$ and impulse sequence $h(n)$ for a given discrete LTI system on TMS320C6748 DSP Processor using Code Composer Studio (CCS).

Equipment Required:

1. Personal Computer with CCS software.
2. TMS320C6748 DSP Processor.
3. XDS110 USB Debug Probe.

Theory:

Convolution is an important operation in DSP, because convolving two sequences in time domain is equivalent to multiplying the sequences in frequency domain. Convolution mainly used in processing signals especially analysing the output of a system. Convolution of two signals $x(n)$ and $h(n)$ is given as

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

In practice, we often deal with sequences of finite length, and their convolution may be found by several methods. The convolution $y(n)$ of two finite-length sequences $x(n)$ and $h(n)$ is also of finite length and is subject to the following rules, which serve as useful consistency

Procedure for Linear Convolution:

1. The starting index of $y(n)$ equals the sum of the starting indices of $x(n)$ and $h(n)$.
2. The ending index of $y(n)$ equals the sum of the ending indices of $x(n)$ and $h(n)$.
3. The length N of $y(n)$ is related to the lengths N_1 and L_h of $x(n)$ and $h(n)$ by
$$N_2 = N_1 + N_2 - 1.$$

The various steps involved in finding out convolution sum are

Step 1: Choose the starting time n for evaluating the output sequence $y(n)$. If $x(n)$ starts at $n = n_1$ and $h(n)$ starts at $n = n_2$, then $n = n_1 + n_2$ is a good choice.

Step 2: Express both the sequences $x(n)$ and $h(n)$ in terms of the index k .

Step 3: Fold $h(k)$ about $k = 0$ to obtain $h(-k)$ and shift by n to the right if n is positive and to the left if n is negative to obtain $h(n-k)$.

Step 4: Multiply the two sequences $x(k)$ and $h(n-k)$ element by element and sum the products to get $y(n)$.

Step 5: Increment the index n , shift the sequence $h(n-k)$ to the right by one sample and perform Step 4.

Step 6: Repeat Step 5 until the sum of products is zero for all remaining values of n .

Procedure:

1. Open Code Composer Studio and Click on Launch to open the CCS V8
2. Click on Project - New CCS Project – Type Project Name.
3. Select the Target as TMS320C674X floating point and Processor as TMS320C6748 - Connections as Texas Instruments XDS110 USB Debug Probe – Compiler version as TI v8.2.5 then click on Finish to load the main.c editor window and Project with above mentioned project name.
4. Type the C - Source code in main.c file.
5. After Creating the Source file then click on the Build to check and verify the errors in the Source code.
6. After Build finished, then click on the Debug and Resume the program main.c to get the outputs.
7. To plot the graph – tools – graph - Single time.

Program:

```
/* program to implement linear convolution */
#include<stdio.h>
#include<math.h>
int y[20];
main()
{
    int m = 6; /*Length of i/p samples sequence*/
    int n = 6; /*Length of impulse response Co-efficients */
    int i = 0,j;
    int x[15]={ 1,2,3,4,5,6,0,0,0,0,0,0}; /*Input Signal Samples*/
    int h[15]={ 1,2,3,4,5,6,0,0,0,0,0,0}; /*Impulse Response Coefficients*/
    for(i=0;i<m+n-1;i++)
    {
        y[i]=0;
        for(j=0;j<=i;j++)
            y[i]+=x[j]*h[i-j];
    }
    printf("Linear Convolution\n");
    for(i=0;i<m+n-1;i++)
        printf("%d\n",y[i]);
}
```

Input and Output:

Input

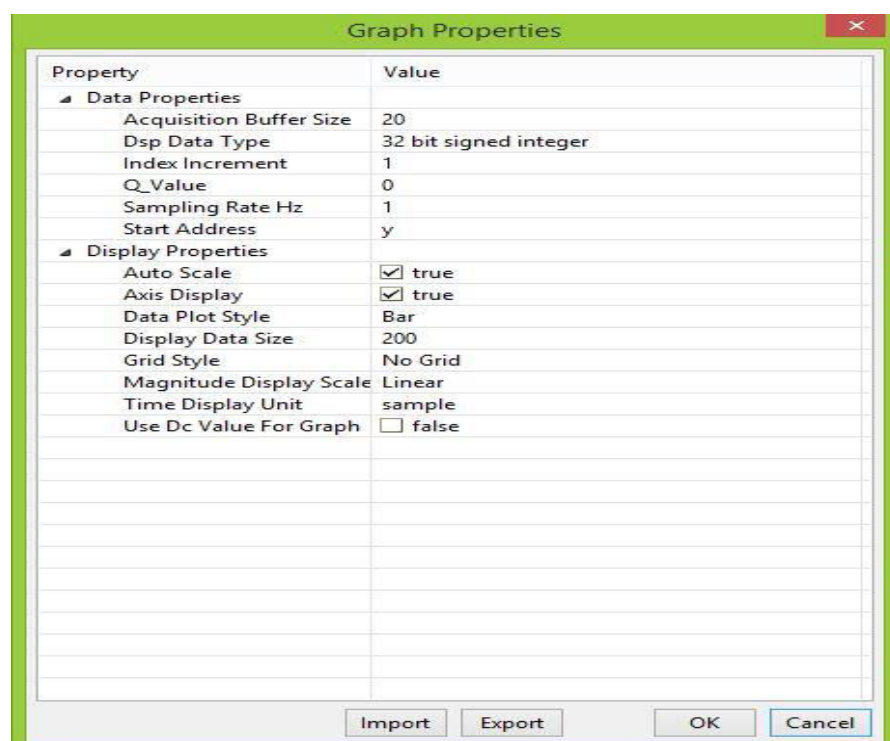
x[15]={ 1,2,3,4,5,6,0,0,0,0,0,0}

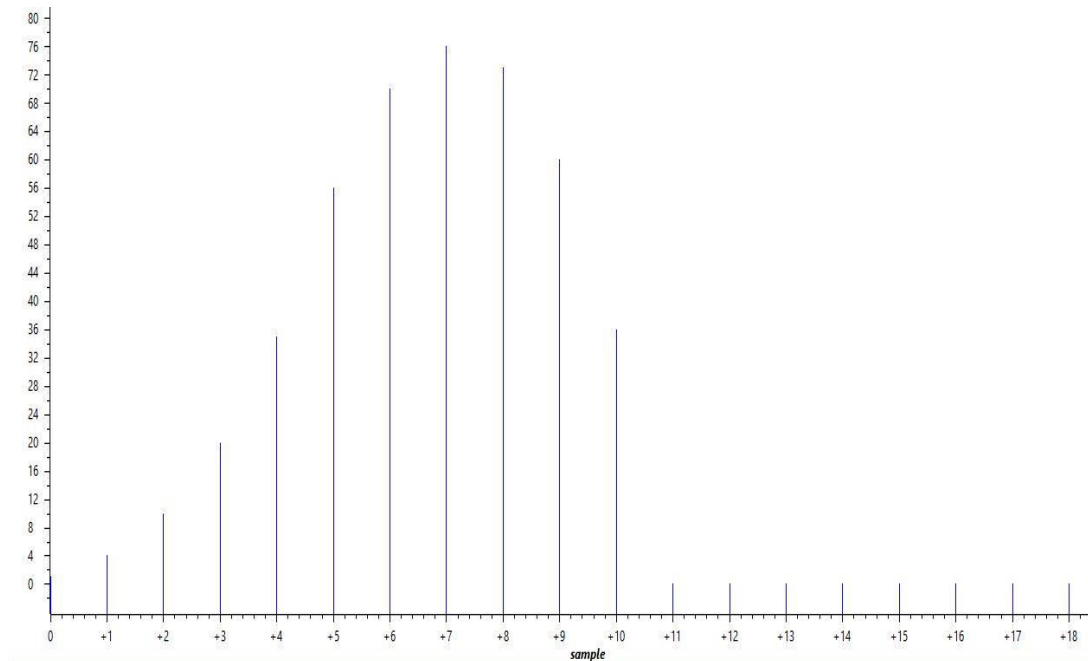
h[15]={ 1,2,3,4,5,6,0,0,0,0,0,0}

Output

Y[15]={ 1, 4, 10, 20, 35, 56, 70, 76, 73, 60, 36, 0, 0, 0, 0}

Graph Properties to display the output graphically





Precautions:

1. Check out source file is with '.c' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. What is convolution?
2. What are the applications of convolution?
3. What are the various steps involved in finding out convolution?
4. What is the expression for Linear Convolution?
5. List the methods involved in finding the linear convolution.

Result:

Linear convolution response of discrete LTI system with input sequence $x(n)$ and impulse response $h(n)$ was verified on TMS320C6748 DSP Processor using CCS Software.

Aim:

To verify the circular convolution between $x_1(n)$ and $x_2(n)$ on TMS320C6748 DSP Processor using Code Composer Studio (CCS).

Equipment Required:

1. Personal Computer with CCS software.
2. TMS320C6748 DSP Processor.
3. XDS110 USB Debug Probe.

Theory:

The circular convolution of two sequences requires that at least one of the two sequences should be periodic. If both the sequences are non-periodic, then periodically extend one of the sequences and then perform circular convolution.

- The circular convolution can be performed only if both the sequences consists of the same number of samples. If the sequences have different number of samples, then convert the smaller size sequence to the size of larger size sequence by appending zeros. The circular convolution produces a sequence whose length is same as that of input sequences.
- Circular convolution basically involves the same four steps as linear convolution namely folding one sequence, shifting the folded sequence, multiplying the two sequences and finally summing the value of the product sequences.
- The difference between the two is that in circular convolution the folding and shifting (rotating) operations are performed in a circular fashion by computing the index of one of the sequences by modulo- N operation.
- In circular convolution, any one of the sequence is folded and rotated without changing the result of circular convolution

Concentric Circle Method

Let $x_1(n)$ and $x_2(n)$ be two given sequences.

The steps followed for circular convolution of $x_1(n)$ and $x_2(n)$ are

- Take two concentric circles. Plot N samples of $x_1(n)$ on the circumference of the outer circle (maintaining equal distance successive points) in anti-clockwise direction.
- For plotting $x_2(n)$, plot N samples of $x_2(n)$ in clockwise direction on the inner circle, starting sample placed at the same point as 0^{th} sample of $x_1(n)$.
- Multiply corresponding samples on the two circles and add them to get output.
- Rotate the inner circle anti-clockwise with one sample at a time.

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

Procedure:

1. Open Code Composer Studio and Click on Launch to open the CCS V8
2. Click on Project - New CCS Project – Type Project Name.
3. Select the Target as TMS320C674X floating point and Processor as TMS320C6748 - Connections as Texas Instruments XDS110 USB Debug Probe – Compiler version as TI v8.2.5 then click on Finish to load the main.c editor window and Project with above mentioned project name.
4. Type the C - Source code in main.c file.
5. After Creating the Source file then click on the Build to check and verify the errors in the Source code.

6. After Build finished, then click on the Debug and Resume the program main.c to get the outputs.
7. To plot the graph – tools – graph - Single time.

Program:

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

/*displaying the result*/

```
printf(" the circular convolution is\n");  
for(i=0;i<n;i++)  
printf("%d \t",y[i]);  
}
```

Input and Output:

Input

Enter the length of the first sequence

4

Enter the length of the second sequence

4

Enter the first sequence

1 2 3 4

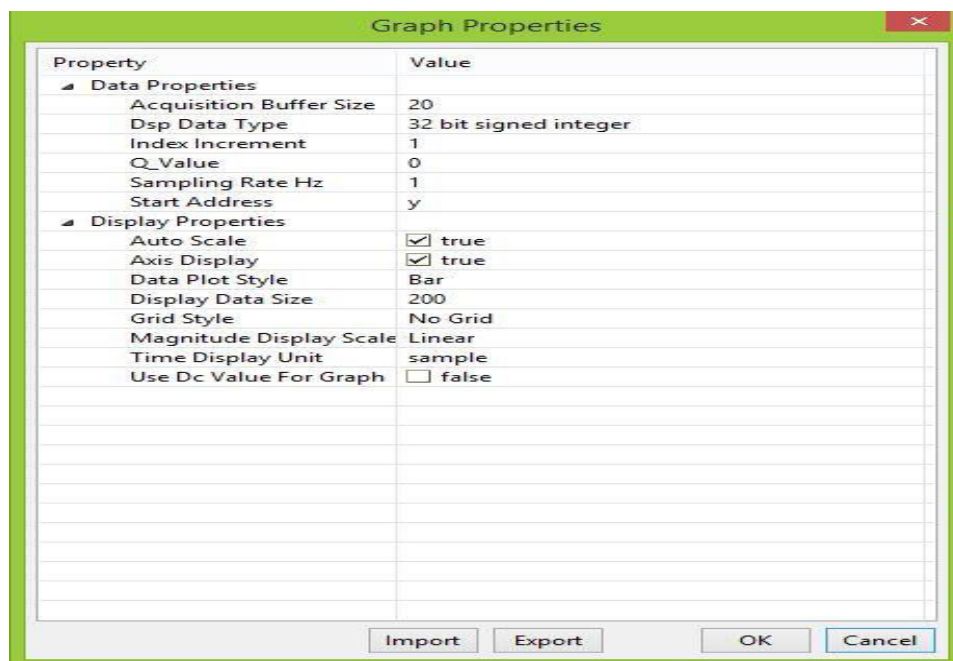
Enter the second sequence

4 3 2 1

Output

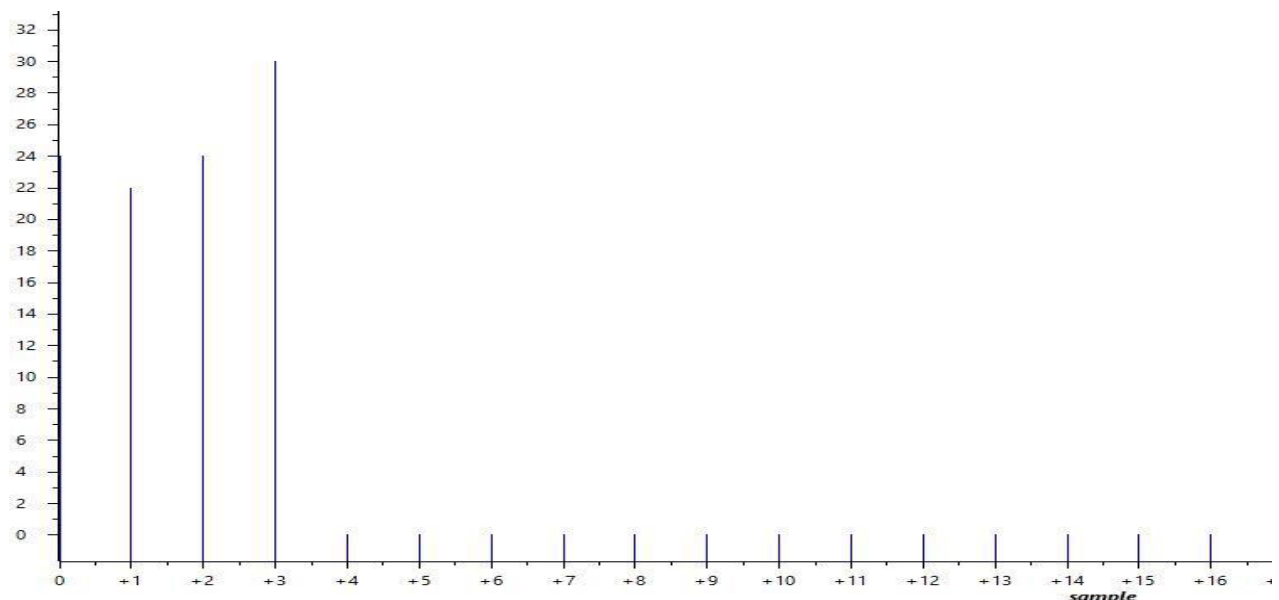
24 22 24 30

Graph Properties to display the output graphically



The image shows a 'Graph Properties' dialog box with a green title bar. It contains a table with two columns: 'Property' and 'Value'. The table is divided into two sections: 'Data Properties' and 'Display Properties'. The 'Data Properties' section includes 'Acquisition Buffer Size' (20), 'Dsp Data Type' (32 bit signed integer), 'Index Increment' (1), 'Q_Value' (0), 'Sampling Rate Hz' (1), and 'Start Address' (y). The 'Display Properties' section includes 'Auto Scale' (checked), 'Axis Display' (checked), 'Data Plot Style' (Bar), 'Display Data Size' (200), 'Grid Style' (No Grid), 'Magnitude Display Scale' (Linear), 'Time Display Unit' (sample), and 'Use Dc Value For Graph' (unchecked). At the bottom of the dialog are four buttons: 'Import', 'Export', 'OK', and 'Cancel'.

Property	Value
Data Properties	
Acquisition Buffer Size	20
Dsp Data Type	32 bit signed integer
Index Increment	1
Q_Value	0
Sampling Rate Hz	1
Start Address	y
Display Properties	
Auto Scale	<input checked="" type="checkbox"/> true
Axis Display	<input checked="" type="checkbox"/> true
Data Plot Style	Bar
Display Data Size	200
Grid Style	No Grid
Magnitude Display Scale	Linear
Time Display Unit	sample
Use Dc Value For Graph	<input type="checkbox"/> false



Precautions:

1. Check out source file is with '.c' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. What is convolution?
2. What are the applications of convolution?
3. What are the various steps involved in finding out convolution?
4. What is the expression for Circular Convolution?
5. List the methods involved in finding the circular convolution.

Result:

Circular convolution between $x_1(n)$ and $x_2(n)$ was verified on TMS320C6748 DSP Processor using CCS Software.

Aim:

To compute N-Point Discrete Fourier Transform for given N-point sequence. on TMS320C6748 DSP Processor using Code Composer Studio (CCS).

Equipment Required:

1. Personal Computer with CCS software.
2. TMS320C6748 DSP Processor.
3. XDS110 USB Debug Probe.

Theory:

The DTFT of a sequence is periodic and continuous in frequency in the range from 0 to 2π . There are infinitely many ω in this range. If we use a digital computer to compute N equally spaced points over the interval $0 \leq \omega \leq 2\pi$, then the N-points should be located at

$$\omega_k = \frac{2\pi}{N}k, \quad k=0,1,2, \dots, N-1$$

These N equally spaced frequency samples of the DTFT are known as DFT of sequence and it is denoted by $X(k)$ is

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

Let $x(n)$ is a causal, finite duration sequence containing L samples, then its Fourier transform is given by

$$X(e^{j\omega}) = \sum_{n=0}^{L-1} x(n) e^{-j\omega n}$$

If we samples $X(e^{j\omega})$ at N equally spaced points over $0 \leq \omega \leq 2\pi$, we obtain

$$X(k) = X(e^{j\omega}) \Big|_{\omega = \frac{2\pi}{N}k} = \sum_{n=0}^{L-1} x(n) e^{-j\frac{2\pi}{N}kn}$$

Since time domain aliasing occurs if $N \leq L$, we increase the duration of sequence $x(n)$ from L to N samples by appending appropriate zeros, which is known as zero padding.

Since zero valued elements contribute nothing to sum, hence the above equation can be written as $X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}$ $0 \leq k \leq N-1$ which is called N-point DFT of sequence $x(n)$.

Since $x_p(n)$ is periodic extension of $x(n)$ with period N, it can be expressed in Fourier series expansion $x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}$ $0 \leq n \leq N-1$ is called N-point IDFT.

Procedure:

1. Open Code Composer Studio and Click on Launch to open the CCS V8
2. Click on Project - New CCS Project – Type Project Name.
3. Select the Target as TMS320C674X floating point and Processor as TMS320C6748 - Connections as Texas Instruments XDS110 USB Debug Probe – Compiler version as TI v8.2.5 then click on Finish to load the main.c editor window and Project with above mentioned project name.
4. Type the C - Source code in main.c file.
5. After Creating the Source file then click on the Build to check and verify the errors in the Source code.
6. After Build finished, then click on the Debug and Resume the program main.c to get the outputs.
7. To plot the graph – tools – graph - Single time.

Program:

```
#include<stdio.h>
#include<math.h>
void main()
{
short N = 8;
short x[8] = {1,2,3,4,5,6,7,0}; // test data
float pi = 3.1416;
float sumRe = 0, sumIm = 0; // init real/imag components
float cosine = 0, sine = 0; // Initialise cosine/sine components
// Output Real and Imaginary components
float out_real[8] = {0.0}, out_imag[8] = {0.0};
int n = 0, k = 0;
for(k=0 ; k<N ; k++)
{
sumRe = 0;
sumIm = 0;
for (n=0; n<N ; n++)
{
cosine = cos(2*pi*k*n/N);
sine = sin(2*pi*k*n/N);
sumRe = sumRe + x[n] * cosine;
sumIm = sumIm - x[n] * sine;
}
out_real[k] = sumRe;
out_imag[k] = sumIm;
printf("[%d] %7.3f %7.3f \n", k, out_real[k], out_imag[k]);
}
}
```

Input and Output:

Input

N=8

X[8]= {1,2,3,4,5,6,7,0}

Output

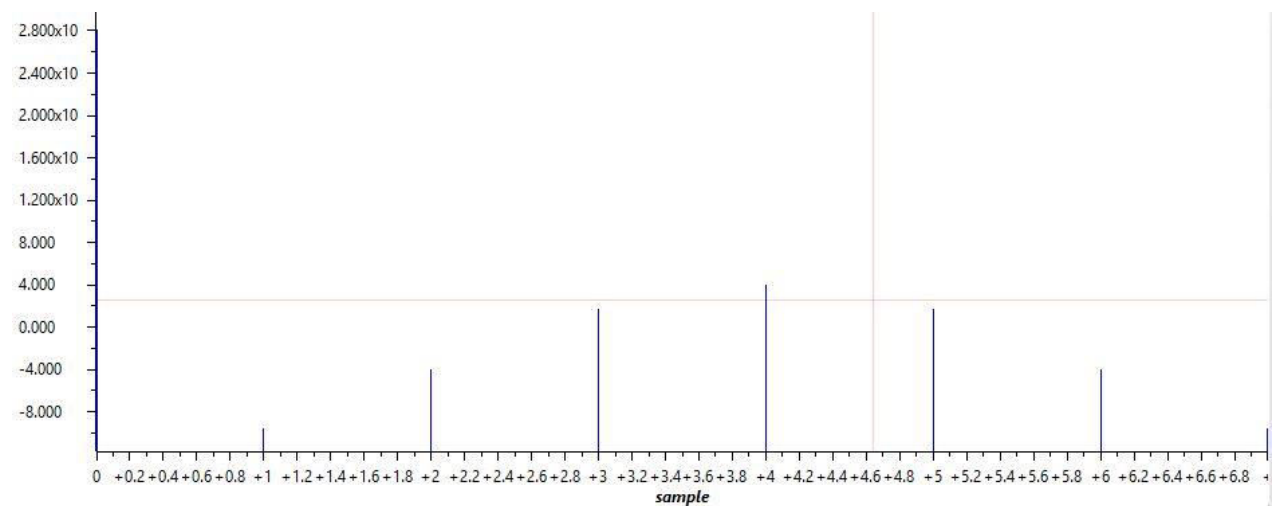
	Out_real	Out_imag
[0]	28.000	0.000
[1]	-9.657	4.000
[2]	-4.000	-4.000
[3]	1.657	-4.000
[4]	4.000	-0.000
[5]	1.657	4.000
[6]	-4.000	4.000
[7]	-9.657	-3.999

Graph Properties to display the output graphically

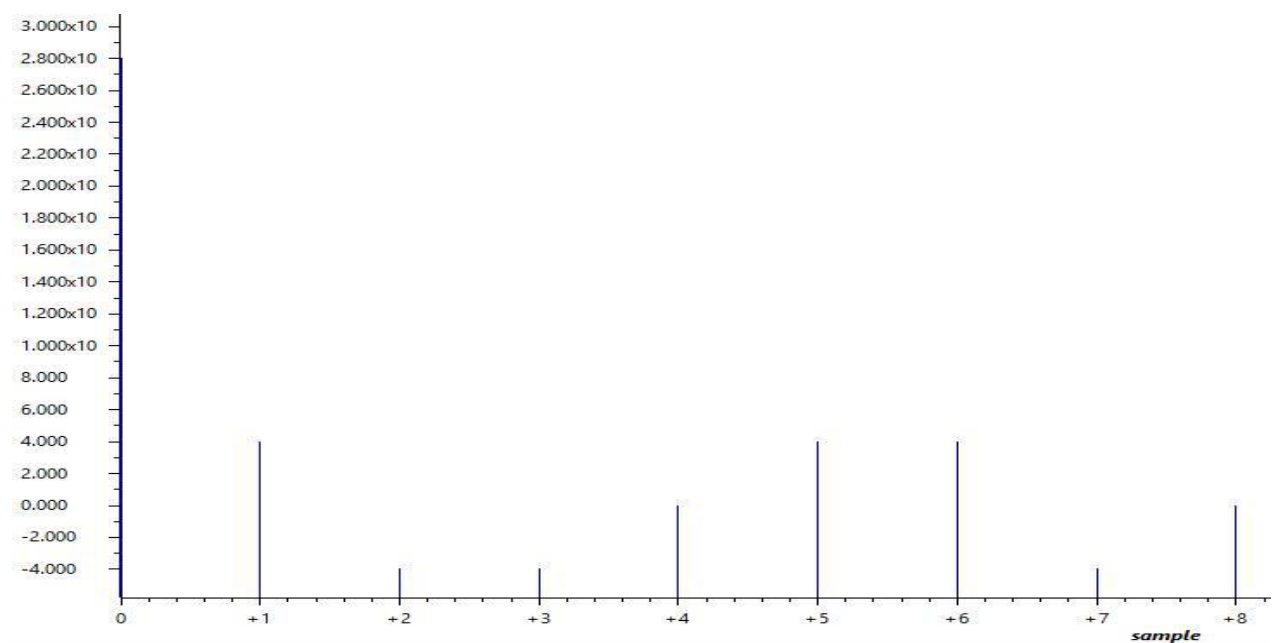
Property	Value
Data Properties	
Acquisition Buffer Size	20
Dsp Data Type	32 bit signed integer
Index Increment	1
Q_Value	0
Sampling Rate Hz	1
Start Address	y
Display Properties	
Auto Scale	<input checked="" type="checkbox"/> true
Axis Display	<input checked="" type="checkbox"/> true
Data Plot Style	Bar
Display Data Size	200
Grid Style	No Grid
Magnitude Display Scale	Linear
Time Display Unit	sample
Use Dc Value For Graph	<input type="checkbox"/> false

Import Export OK Cancel

Graph of DFT Real Part:



Graph of DFT Imaginary Part:



Precautions:

1. Check out source file is with '.c' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Don't delete built in functions and any file or folder without informing the system administrator or lab In-charge.

Viva -Voce Questions:

1. Write the expressions for N –Point DFT.
2. Differentiate between DTFT and DFT.
3. What are the advantages to use DFT in computers rather than DTFT?
4. State any two DFT properties.

Result:

The N-Point DFT of the given sequence was computed on TMS320C6748 DSP Processor using CCS Software.