Department of Electronics Engineering

Lab Manual

Of

Digital Signal Processing Lab (ECC 304)

Academic Complex, 2nd Floor, Room No.215

INDIAN INSTITUTE OF TECHNOLOGY (Indian School of Mines), DHANBAD

JHARKHAND-826004

Experiment List

Experiment-1

General Instructions for all the MATLAB programs

1. Click on the MATLAB icon on the desktop.
2. MATLAB window open.
3. Click on the 'FILE' Menu on the menu bar.
4. Click on NEW M-File from the File Menu.
5. An editor window opens, start typing commands.
6. Now SAVE the file in a directory in the format Name_Rollnumber.
7. Then Click on DEBUG from the Menu bar and Click Run

Aim:

(A.) Discrete-time signal generation (Square wave, Sinewave, & Impulse signal).

(B.) Impulse and Step response of an LTI system.

(C.) Step-response of an LTI system using convolution

Theory:

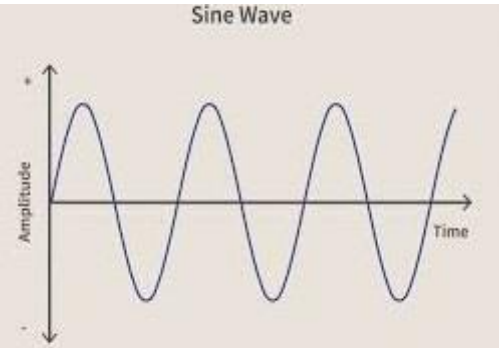
(A) Discrete-time signal

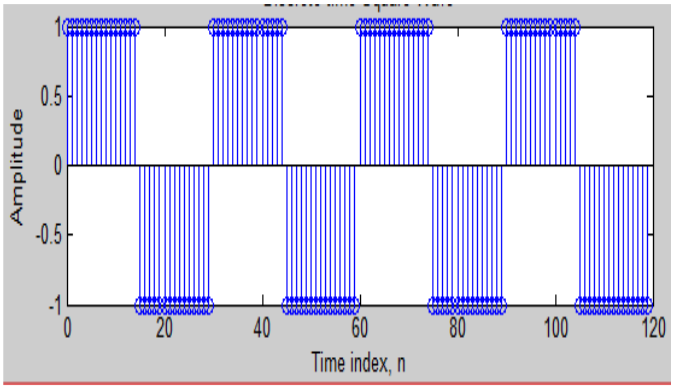
Signals are broadly classified into analog and discrete signals. An analogsignal will be denotedby $x_a(t)$, in which the variable t can represent anyphysical quantity, but we will assume that it represents time in seconds. Adiscrete signal will be denoted by $x(n)$, in which the variable n is integer-valuedand represents discrete instances in time. Therefore it is also calleda discrete-time signal, which is a *number sequence* and will be denoted byone of the following notations:

$x(n) = \{x(n)\} = \{\ldots, x(-1), \underset{\uparrow}{x(0)}, x(1), \ldots\}$

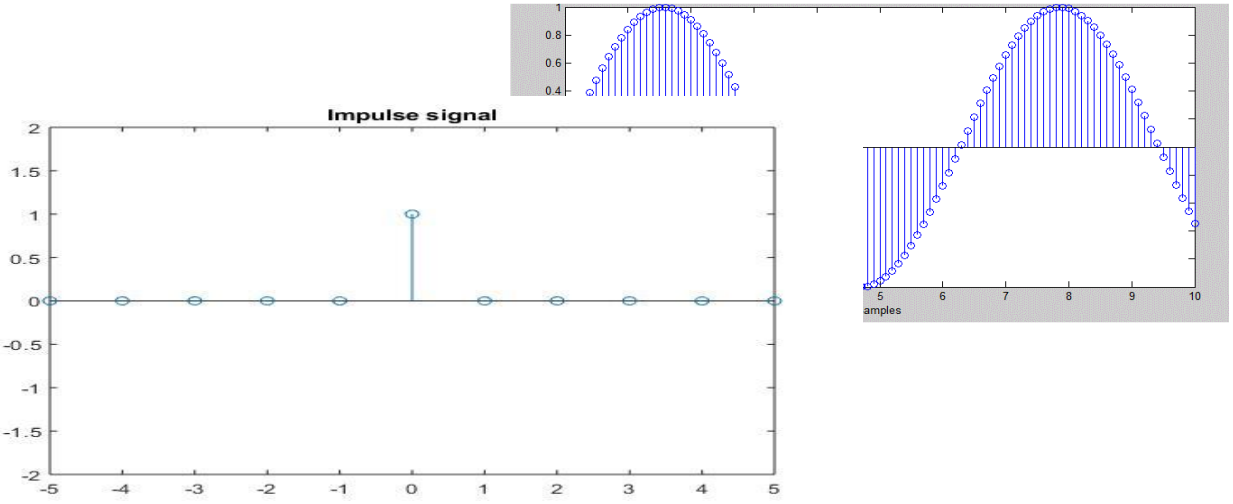
where the *up-arrow* indicates the sample at $n = 0$.

Square Wave:- The **square wave**, also called a pulse train, or pulse **wave** is a periodic **waveform** consisting of instantaneous transitions between two levels, with the same duration at minimum and maximum.





Sine wave:-A**sine wave** is a geometric waveform that oscillates (moves up, down or side-to-side) periodically, and isdefinedby the function y = sin x.

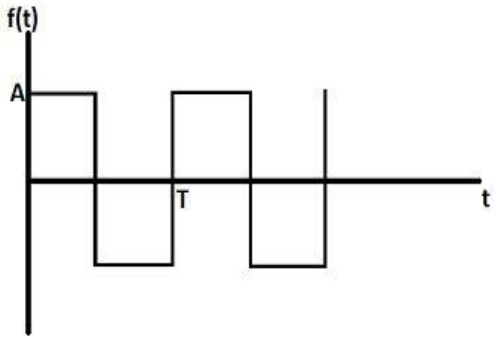


Impulse signal:-

The impulse signal is given as

δ[n]={1 if n=0

0 otherwise



Use MATLAB function

```
n1=-10;
n2=10;
n0=0;
n = [n1:n2];
```
x_impulse = [(n-n0) == 0]

B. Impulse and Step response of an LTI system
Find the impulse response and step response of the system

Y [n] $-\frac{1}{2}$ Y[n-1] = x[n] from the difference equation**.**

MATLAB Code

1. Generate input impulse :x=[1 zeros(1,9)]
2. Generate input step: x= ones(1,10)
3. Take h(1) = 1& y(1) = 1, (`The system is initially at rest`)
4. h(k) = x(k)+0.5 *h(k-1)
5. y(k) = x1(k)+0.5 *y(k-1)
6. Use **stem(n,h)** to generate a discrete signal of the impulse response
7. Use **stem(n,y)** to generate a discrete signal of the step response
8.

Assignment.

Q. Given the following difference equation (**Page 48**)

$$y(n) - y(n-1) + 0.9y(n-2) = x(n);\ \forall n$$

a. Calculate and plot the impulse response $h(n)$ at $n = -20, \ldots, 100$.
b. Calculate and plot the unit step response $s(n)$ at $n = -20, \ldots, 100$.

C. Find the step response of the system given in (B) using convolution

MATLAB code

1. Generate input impulse and input step
2. Take a=[1 -0.5] and b=1 (difference equation coefficients)
3. Use h=filter(b,a,x)to solve difference equations
4. Use conv(x,h) to find convolution

Experiment -2

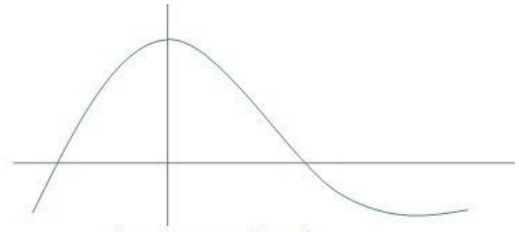
Aim:

- i)To represent a continuous-time signal using MATLAB
- ii) To obtain Fourier transform of this signal and plot it
- iii) To Sample the above signal and plot the Sampled Signal, and Obtain the Fourier transform of the discrete-time Signal and plot it, and observe the 'Aliasing' effect by sampling at different frequencies.
- iv) To obtain linear convolution of sequence.
- v) To obtain circular convolution

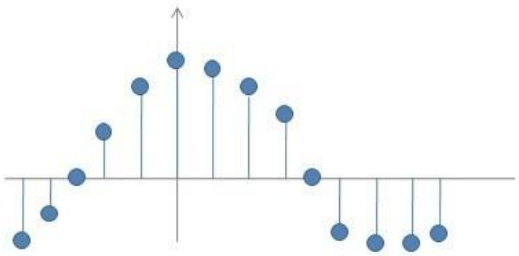
Theory:

A signal which is defined for all time t contained in some interval on the real line. All polynomial, Exponential functions can be represented as continuous time signals.

Similarly, a discrete time signal is a time series consisting of a sequence of quantities. Unlike a continuous-time signal, a discrete time signal is not a function of a continuous argument. However, it could be obtained by sampling the given continuous time signal at different frequencies. This process is known as Sampling.



Continuous Signal



Discrete Signal

In General Fourier transform is a mathematical transform that decomposes functions depending on time into functions depending on frequency. The Fourier transform is an operation that transforms data from the time (or spatial) domain into the frequency domain. A signal f(t) is a continuous time signal then its Fourier transform is

$$\mathbf{F(w)}=\int_{-\infty}^{+\infty} \boldsymbol{f(t)e^{-jwt}dt}$$

$$\boldsymbol{f(t)} = \frac{\mathbf{1}}{\boldsymbol{2\pi}}\int_{-\infty}^{+\infty} \boldsymbol{F(w)e^{jwt}dt}$$

Above two equations form the Fourier transform pair. For existence of Fourier transform, Dirichlet conditions are sufficient but not necessary conditions.

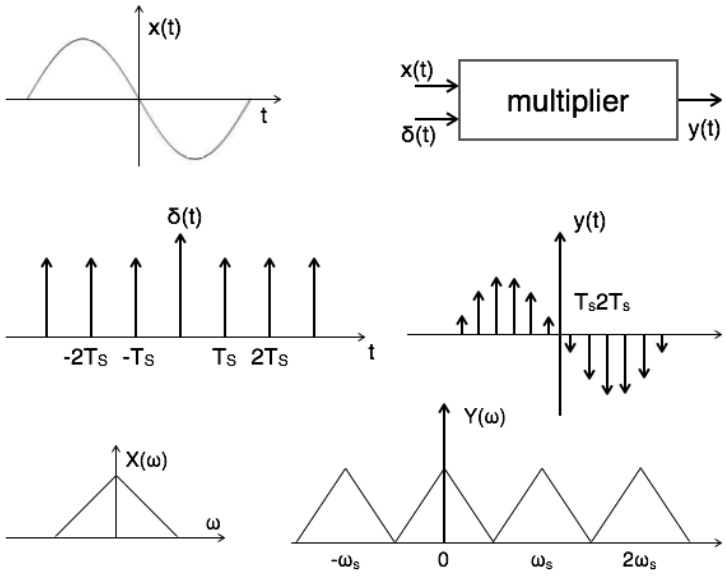
Sampling is the reduction of a continuous time signal to a discrete time signal by taking the values of continuous time signal at continuous time interval. The Sampling time interval is known as Sampling time period and its inverse is Sampling frequency.

In General, for a continuous time signal to be able to reconstruct properly, the sampling frequency must be greater than or equal to Nyquist frequency.

Fs >=FN and FN = 2*F

where F= frequency of the signal, Fs = Sampling frequency, FN = Nyquist frequency

Aliasing occurs when a signal is sampled less than Nyquist frequency. The Band of sampling gets overlapped leading to a loss of continuous time signal, making it unable to reconstruct. Hence it is always recommended to sample at frequency more than or equal to Nyquist frequency.



Convolution is an operator that takes an input signal and returns an output signal, based on knowledge about the systems unit response h[n].The convolution of function f and g is written as f*g.

It is defined as integral or product of two functions after one is reversed and shifted.

$$(\boldsymbol{f} * \boldsymbol{g})(\boldsymbol{t}) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} \boldsymbol{f}(\boldsymbol{\tau})\, \boldsymbol{g}(\boldsymbol{t}-\boldsymbol{\tau})\boldsymbol{d\tau} = \int_{-\infty}^{+\infty} \boldsymbol{f}(\boldsymbol{t}-\boldsymbol{\tau})\boldsymbol{g}(\boldsymbol{\tau})\boldsymbol{d\tau}$$

There are two types of convolution:**linear and circular**. In linear case, linear shifting is performed followed by basic operations of convolution. In circular shifting, length of two sequence should be same. If not, zero padding should be done.

General Instructions for all the MATLAB programs

1. Click on the MATLAB icon on the desktop.
2. MATLAB window open.
3. Click on the 'FILE' Menu on the menu bar.
4. Click on NEW M-File from the File Menu.
5. An editor window opens, start typing commands.
6. Now SAVE the file in a directory in the format Name_Roll number.
7. Then Click on DEBUG from the Menu bar and Click Run.

The given programs correspond to the signal

$$x_a(t) = e^{-1000|t|}$$

You need to first find F.T. of this continuous-time signal and verify that $Xa(j\Omega) = 0$ for $\Omega \geq 2\pi$ (2000) i.e., for $f \geq 2000$.

- The signal xa(t) with very small-time interval (much smaller than the Nyquist rate for the given signal i.e., 4000 Hz - you need to verify it in the program) and thus approximates a continuous-time signal. (Duration of the signal, considered in the program is [-5, 5] msec).
- The continuous-time signal (as approximated above) is sampled – this time to give a discrete-time signal (it's basically the signal xa(t) sampled now at a much lower rate than that was chosen earlier for smooth representation of the signal). Two different sampling frequencies are to be experimented with – 5000 samples/sec (higher than Nyquist rate for xa(t)) and 1000 samples/sec (lower than the Nyquist rate).

Hint:

i. Representation of continuous time signal.
a) Initialize the signal.
b) Give the time interval.
c) Use subplot function to generate continuous time signal.

Obtain Fourier transform of above obtained signal and plot it.

Steps:

- Define step size.
- Dt=0.0005; t=-0.0005:Dt:0.0005.
- **x(t)=exp(-1000*abs(t))**
- Define range of t.
- Write the signal expression.
- Go for continuous time Fourier transforms
  : Define maximum angular frequency.
  Wmax = 2*pi*2000;
  K = 500; k = 0:1:K;

W = k*Wmax/K;
Xa = xa * exp(-1i*t'*W) * Dt;
Xa = real(Xa);
W = [-fliplr(W), W(2:501)]; % Omega from -Wmax to Wmax
Xa = [fliplr(Xa), Xa(2:501)];
Now plot the continuous time Fourier transform.

Discrete time signal:

- Define the sampling interval.
- Instructions: Ts=0.0002; n=-25:1:25;
  x(n)= exp$^{(-1000*abs(nTs))}$
- Go for discrete time Fourier transform in a similar way.
- Now change the sampling frequency and check for aliasing.

Convolution:
Enter the first sequence: x[n]
Enter the second sequence: h[n]
Use in-built operator:
%For linear convolution:
Clin=conv(x,h);
%For circular convolution:
Ccir= cconv(x,h);

Functions used:
Plot: Plot (X, Y) creates a 2-D line plot of the data in Y verses the corresponding value in X.
STEM: stem(y) plots the data sequence, Y as stems that extend from a baseline along the x-axis.
Real: X=real(z) returns the real part of each element in array z.
Support: subplot(m,n,p) divides the current figure into an m-by-n-grid and creates axes in the position specified by p.
Flipr: B=flipr(A) returns A with its column flipped in the left right direction (that is vertical axis).
conv(x,h): performs linear convolution.
cconv(x,h): performs circular convolution.

Interpretation:

Results:

Experiment-3

AIM:-

(A) Reconstruction and Aliasing using sinc function.
(B) Reconstruction using stairs and plot function.
(C) Reconstruction using cubic spline.

Theory:-

Reconstruction of a continuous time signal from its samples.

$$x_r\,(t) = x_\delta(t) * h(t)$$

Different types of interpolation for reconstruction of signals:-

I. Ideal interpolation (using sinc function)

II. Zero-order hold interpolation

III. First-Order hold interpolation

IV. Cubic spline interpolation

i. Ideal Interpolation (using sinc function)

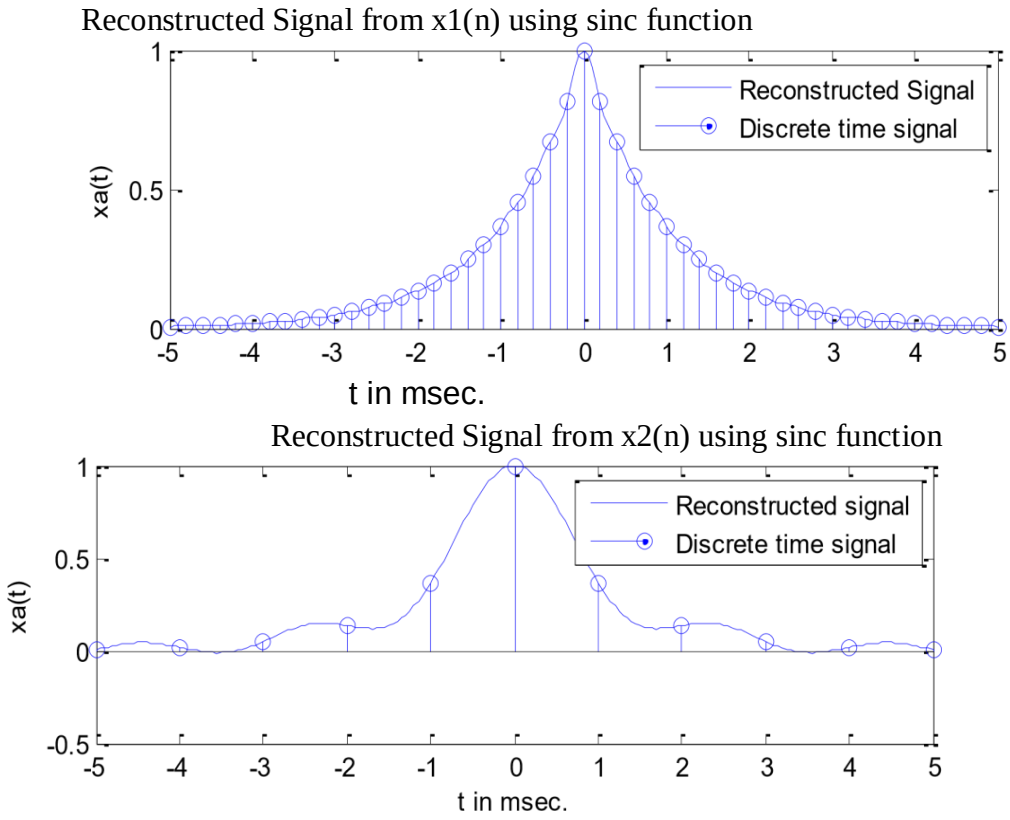
$$h(t) = h_r(t) = T_s sinc(f_s t)$$

$$x_r(t) = x_\delta(t) * h(t) = \sum_{n=-\infty}^{\infty} x[n] h_r t - nT_s)$$

- To get the continuous time signal from discrete time signal using ideal interpolation MATLAB function ‘sinc’ is used.

- sinc(X) returns a matrix whose elements are the sinc of the elements of X.

Plot of reconstruction using sinc function



ii. Zero-order hold interpolation

- It describes the effect of converting a discrete-time signal to a continuous-time signal by holding each sample value for one sample interval.

- To get the continuous time signal from discrete time signal using Zero-order hold filter MATLAB function 'stairs' is used.

- stairs(X,Y) draws a stair step graph of the elements in vector Y at the locations specified in X.

ii. First-order hold interpolation

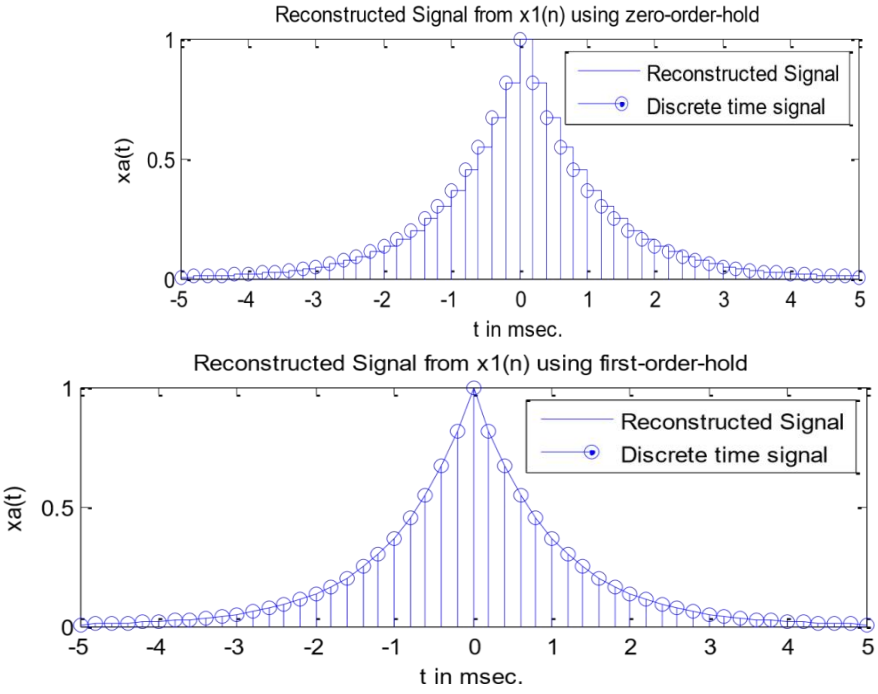
- In First order hold, the signal is reconstructed as a piece wise linear approximation to the original signal that was sampled.

- To get the continuous time signal from discrete time signal using Zero-order hold filter MATLAB function ‘plot’ is used.
- plot(X,Y) plots vector Y versus vector X

- Zero / First Order hold means the order of the Taylor Series of the function we use to interpolate.

- Zero Order means the function is constant, we interpolate the same value in the missing parts.
  First order means we can use linear function to interpolate (Line with a slope)

Plot of reconstruction using stairs and plot interpolation



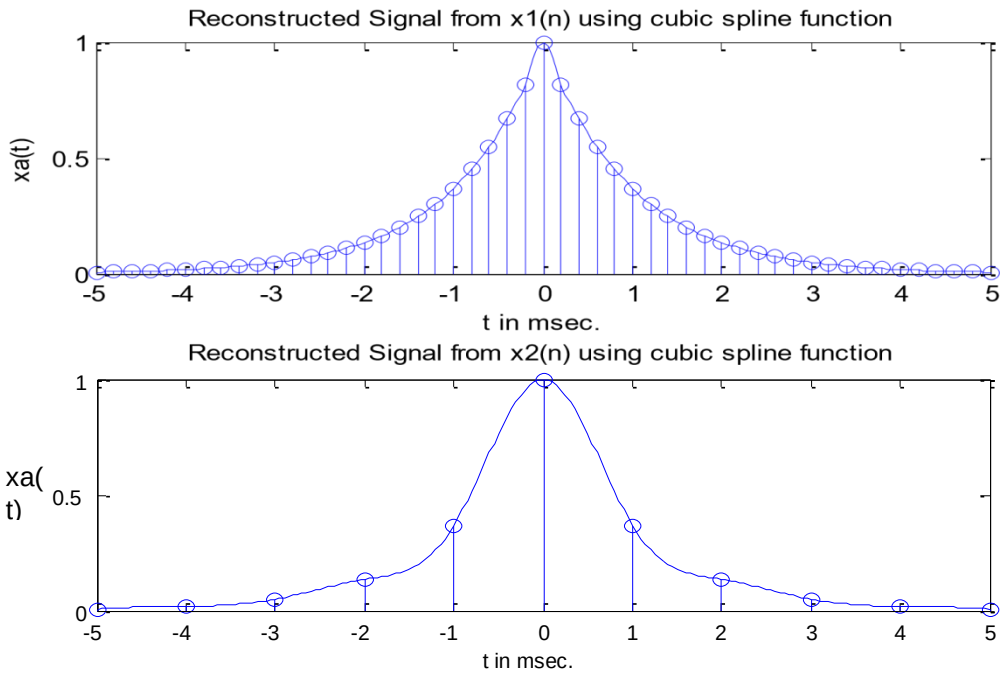
iv. Reconstruction using cubic spline.

- Cubic spline interpolation is a special case for Spline interpolation.

- To get the continuous time signal from discrete time signal using Zero-order hold filter MATLAB function ‘spline’ is used.

spline(X,Y) provides the piecewise polynomial form of the cubic spline interpolant to the data values Y at the data sites X,

Plot of reconstruction using cubic spline interpolation



Error Calculations:

Types of Reconstruction	Absolute Error	
	For Fs =5000 samples/sec	For Fs=1000 samples/sec
Sinc interpolation		
Zero-order Hold interpolation		
First-order Hold interpolation		
Cubic Spline interpolation		

Conclusion

- Reconstruction from sampled signal with higher sampling rate is free of aliasing.

Experiment -4

Aim(a):

(i)-To generate the discrete time signal

x[n]=0.8*sin(2*pi*0.0625*n)+0.3*sin(2*pi*0.2*n)

and plot the signal x[n] and its spectrum.

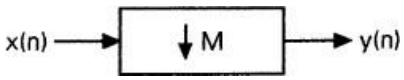
(ii)-Down sample given signal of x[n] by a factor of 2 and plot the down sample signal and its spectrum.

(iii)-Interpolate the down sampled signal by the same factor M=2 to obtain the signal observe the waveform and spectrum of the interpolated signal.

(iv)-Repeat the above for M=3.

Theory:

The decimator is a device that reduces the sampling rate by an integer factor of M,whereas the interpolator is used to increase the rate by L. In many applications the sampling rate of a system needs to be changed for a lower or higher sampling rate for an appropriate processing of the signal, for example in a digital mobile receiver system, the received signal often presents a high frequency which cannot be digitized by the ADC converter , so it needs to be first converted to a lower frequency using an RF analog down conversion, the resultant frequency is commonly known as intermediate frequency, after ADC a FIR filter is used to select a desired bandwidth and then it is downsampled or upsampled to an appropriate sampling rate. The process of downsampling consist in reducing the sampling rate of a signal that is already sampled, by an integer factor M, basically we are resampling the signal, but because we are reducing the sampling rate then we call this as subsampling. When a signal is downsampled we reduce the amount of data by taking only every M-th sample of the signal and discarding all others as we can see in fig1.



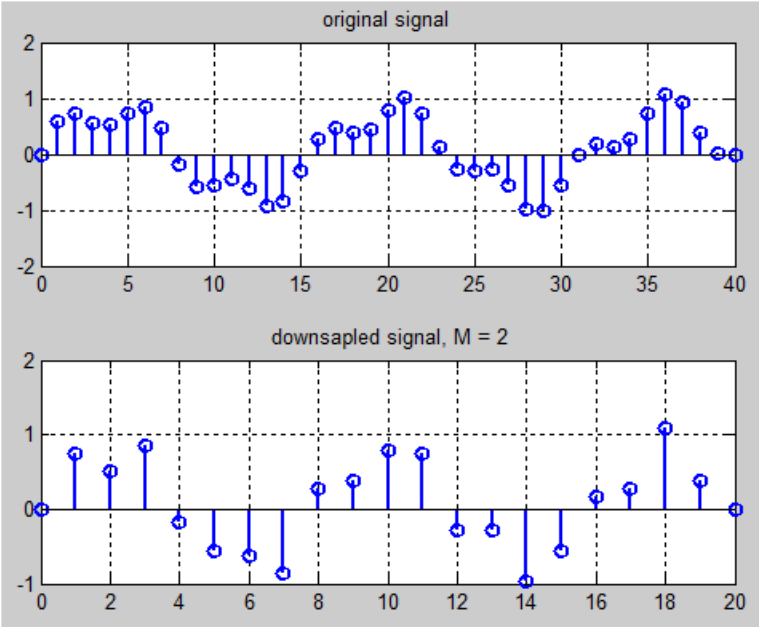


Fig.1

This operation introduce time scaling by a factor of 1/M. Due to the principle of duality in time-frequency, in time we get a compression so we expect an expansion in frequency as we can see in fig2 . The frequency is scaled by a factor of 2.

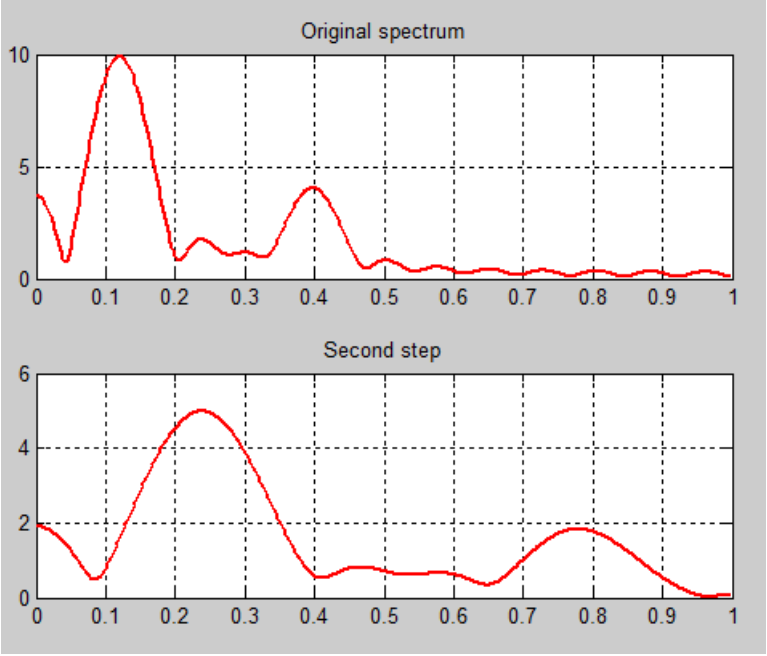


Fig.2

Interpolation:

interpolation always consists of two processes:

1. Inserting L-1 zero-valued samples between each pair of input samples. This operation is called “zero stuffing”.
2. Lowpass-filtering the result.

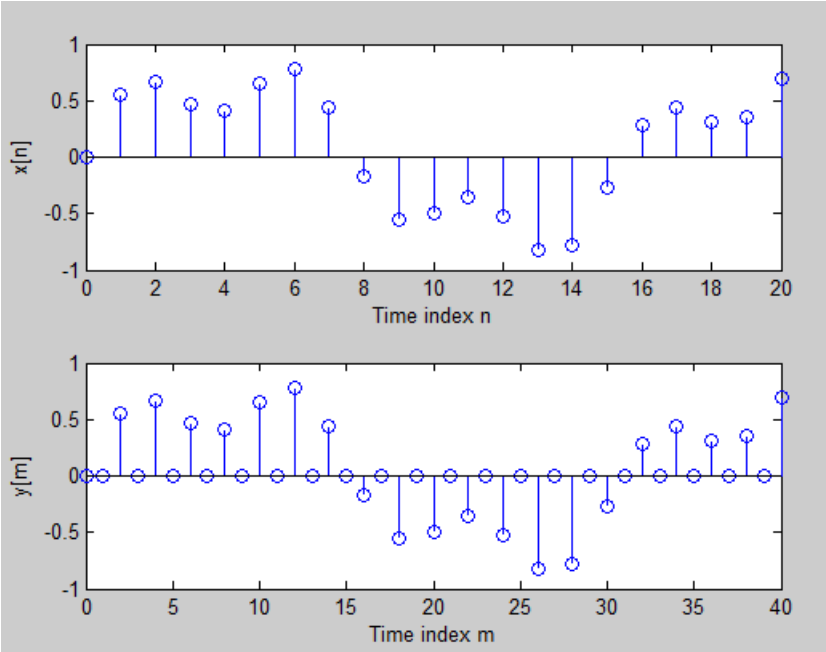


Fig.3

Step1.

Generate the discreate time signal and its spectrum.

```
n=0:99;
x=0.8*sin(2*pi*0.0625*n)+0.3*sin(2*pi*0.2*n);
x1=0:length(x)-1;
subplot(3,2,1);
stem(x1,x);
[xz w]=freqz(x,1,512);
subplot(3,2,2);
plot(w/pi,abs(xz));
```

step 2.

Downsampled the x signal by M=2 and observe its spectrum

```
y=downsample(x,2)
; y1=0:length(y)-
1;
subplot(3,2,3);
stem(y1,y);
[yz w]=freqz(y,1,512);
subplot(3,2,4);
plot(w/pi,abs(yz));
```

step3 .

Interpolate the signal and oberve the spectrum

```
re_xdown=interp(y,2);
re_xdown1=0:length(re_xdown)-1;
subplot(3,2,5);
stem(re_xdown1,re_xdown);
```

Aim (b):

(i)-To generate the discreate time signal x[n]=sin(2*pi*fm*n) at two different sampling rate N1=100, N2=1000.

(ii)-Take the DFT of both sampled signal.

(iii)-Oberve the difference in spike and conclude which one is more tending to impulse

Step1- Generate the discreate time signal and its spectrum with N=100

```
n=0:.01:1-.01;
fm=1;
y=sin(2*pi*fm*n)
; N=length(y);
p=0:N-1;
for k=0:N-1
    z(k+1)=sum(y.*exp(-j*2*pi/N*k*p));
end
```

Step2- Generate the same discreate time signal and its spectrum with N=1000

```
n=0:.001:1-.001;

fm=1;
y=sin(2*pi*fm*n)
; N=length(y);
p=0:N-1;

for k=0:N-1

    z(k+1)=sum(y.*exp(-j*2*pi/N*k*p));

end
```

Experiment -5

AIM:

(a) Decimation-in-Time FFT algorithm (DIT- FFT)

(b) *Decimation-in-Frequency FFT* algorithm (DIF- FFT)

Theory:

RADIX-2 FFT ALGORITHMS

The N -point DFT of an N -point sequence $x(n)$ is

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

Here for the computing of N – point DFT requires N^2 in complex multiplication and

N (N-1) in addition.

FFT algorithm is one of "divide and conquer." which involves decomposing an N-point DFT into successively smaller DFTs. Two algorithms are worked with FFT

Decimation-in-Time FFT :

The decimation-in-time FFT algorithm is based on splitting (decimating) x(n) into smaller sequences and finding ***X (k)*** from the DFTs of these decimated sequences.

Let x(n) be a sequence of length N, and suppose that x(n) is split (decimated) into two subsequences, with of length N/2, the first sequencing (n), is formed from the even-index terms and h(n), is formed from the odd-index terms

g(n) = $x(2n)$, n =0,1,……. N/2 -1

h(n) = $x(2n+1)$, n= 0,1, ……N/2 -1

The N -point DFT of x(n) is :$X(K) = \sum_{n=0}^{N-1} x(n)\omega_N^{nk}$

$$= \sum_{n\ even} x(n)\omega_N^{nk} + \sum_{n\ odd} x(n)\omega_N^{nk}$$

$$= \sum_{l=0}^{N/2-1} g(l)\omega_N^{2lk} + \sum_{n=0}^{N/2-1} h(l)\omega_N^{(2l+!)k}$$

$$= \sum_{l=0}^{N/2-1} g(l)\omega_{N/2}^{lk} + \omega_N^k \sum_{n=0}^{N/2-1} h(l)\omega_{N/2}^{lk}$$

Here the first term is the N/2 point DFT of g(n) and second term is the N/2 point DFT of h(n)

$X(K) = G(K) + \omega_N^k H(K)$. , K=0, 1 ,…. N-1

It is called a decimation in time because the time samples are rearranged in alternating groups, and a radix-2 algorithm because there are two groups

The basic computational unit of FFT is butterfly.

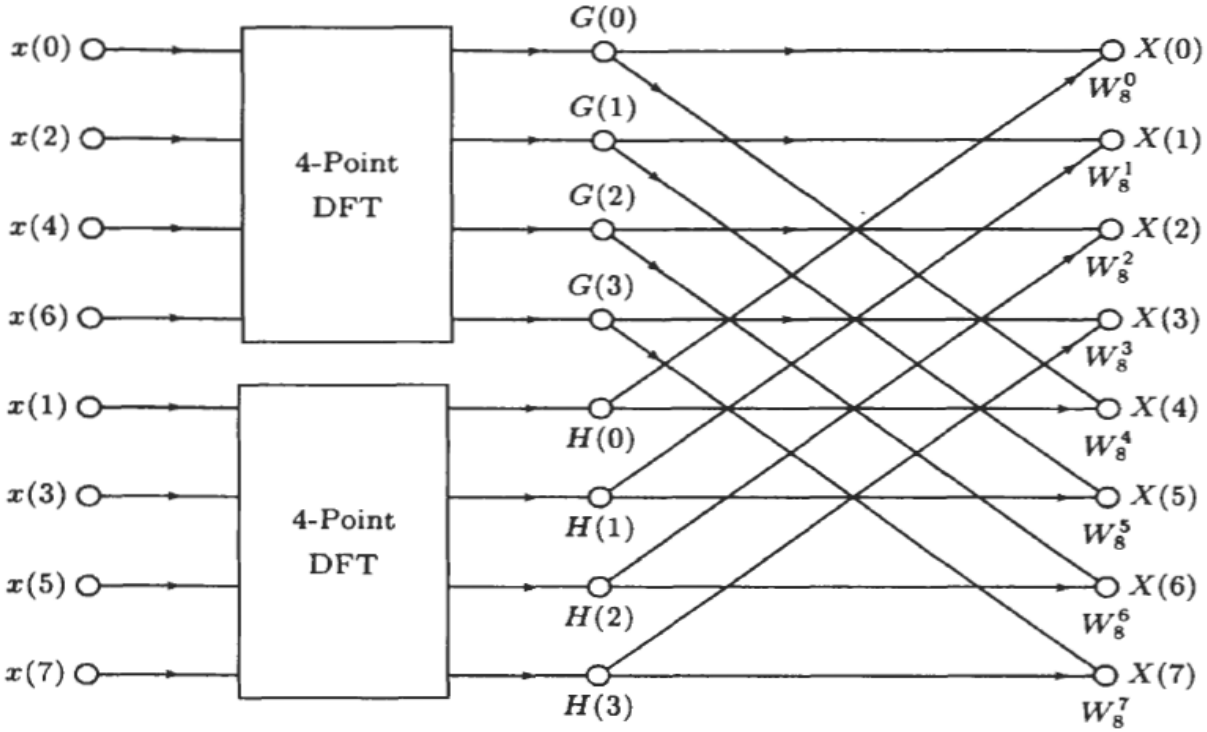


Fig (a): An Eight point Decimation in time -FFT

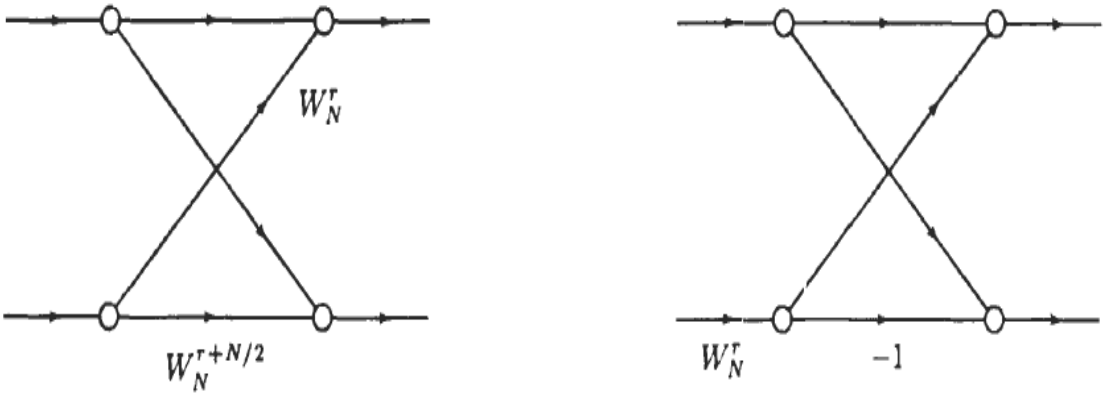
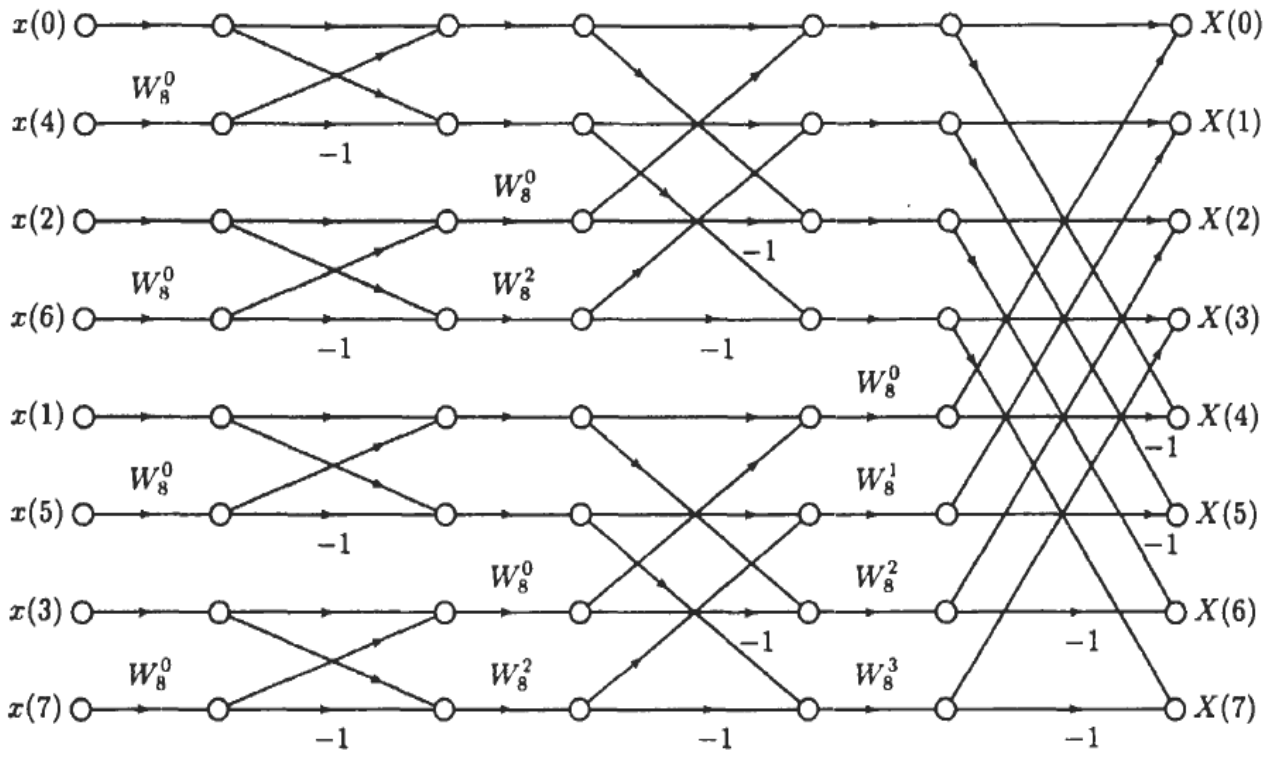


Fig (b): The butterfly, which is the basic computational clement of the FFT algorithm



Fig(c): A complete eight-point radix-2Decimation-in-time FFT.

Computational cost of radix-2 DIT FFT:

(a) $\frac{N}{2}\log_2 N$ Complex mutiplier

(b) $N\ \log_2 N$ *Complex addition*

❖ *Decimation-in-Frequency FFT (DIF – FFT) algorithm*:

Other types of FFT algorithms may be derived by decimating the output sequence X (k) into smaller and smaller subsequences. This is called decimation in frequency because the frequency samples are computed separately in alternating groups, and a radix-2 algorithm because there are two group.

Let N be a power of 2, N = $2^v$and consider separately evaluating the even-index and odd-index samples of X (k).

The even samples are

$$X(2K) = \sum_{n=0}^{N-1} x(n)\omega_N^{2nk}$$

$=\sum_{n=0}^{N/2-1} x\,n)\omega_{N/2}^{nk} + \sum_{n=0}^{N/2-1} x(n)\omega_{N/2}^{nk}$

With change in the indexing on the second

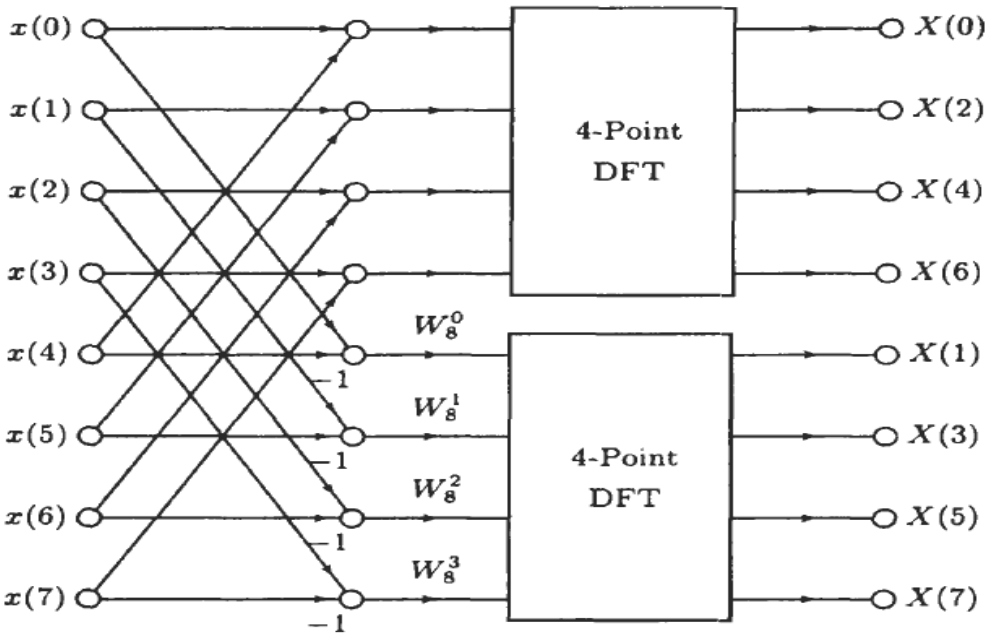
$$X(2K) = \sum_{n=0}^{N/2-1} x\,n)\omega_{N/2}^{nk} + \sum_{n=0}^{N/2-1} x(n+N/2)\omega_{N/2}^{(n+\frac{N}{2})k}$$

$= \sum_{n=0}^{N/2-1}[x\,n) + x(n+N/2]\omega_{N/2}^{nk}$

Now for the odd samples of X(K) is:

$$x(2K+1) = \sum_{n=0}^{N/2-1} \omega_N^n[x\,n) - x(n+N/2]\omega_{N/2}^{nk}$$

The complexity of the decimation-in-frequency FFT is the same as the decimation-in-time,



Fig(d) :An eight-point decimation-in-frequency FFT algorithm after the first stage of' decimation.

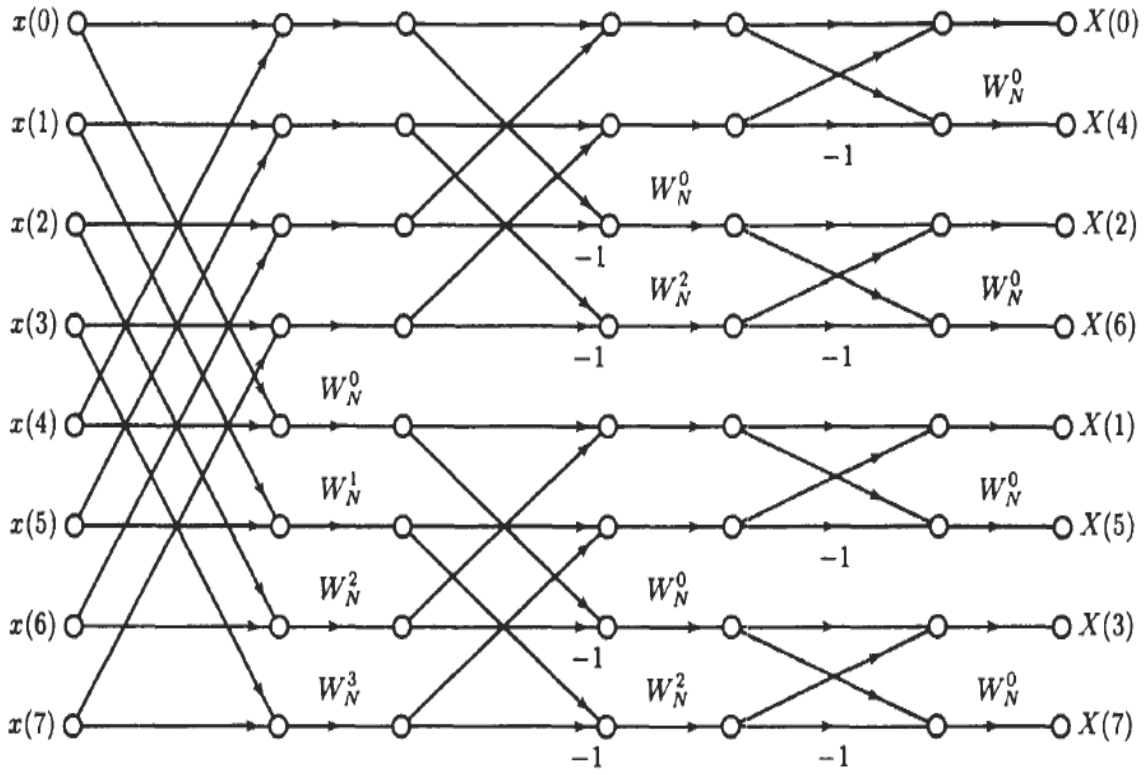


Fig (e) : Eight-point radix-2 decimation-in-frequency FFT.

Q1. Compute the DIT-FFT and DIF-FFT of the sequenses using MATLAB

a. x(n) = {-1, 0, 2, 0, -4, 0, 2, 0}

x(n) = {1, 0, -1, 2, 0, -4, 0, 0}

Experiment -6

- **Aim:**

I. To design a Digital IIR filter, using bilinear transformation method. The two filters that are to be used are Butterworth and Chebyshev filter. The given specifications are:

a) Sampling frequency: 2KHz
b) Low pass band edge frequency = 200Hz
c) Stop edge frequency = 600Hz
d) Maximum attenuation in pass band=1.938dB
e) Maximum attenuation in stop band=13.98dB

- **Theory:**

A digital filter is a system that performs mathematical operations on a discrete and sampled time signal. In contrast to an analog filter, design of digital filter requires some additional blocks namely C/D converter and D/C converter as shown in figure 1. Two major types of digital filters are finite impulse response digital filters (FIR filters) and infinite impulse response digital filters (IIR).

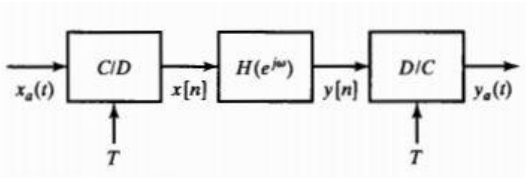


Fig.1. basic system for discrete time filtering of continuous time signals

There are two transformation techniques for designing Butterworth and Chebyshev filters in signal processing.

- Bilinear transformation.
- Impulse invariance.

In this experiment, filter design is performed using bilinear transformation method. This technique relies on one-to-one mapping technique.

Steps for designing any IIR filter:

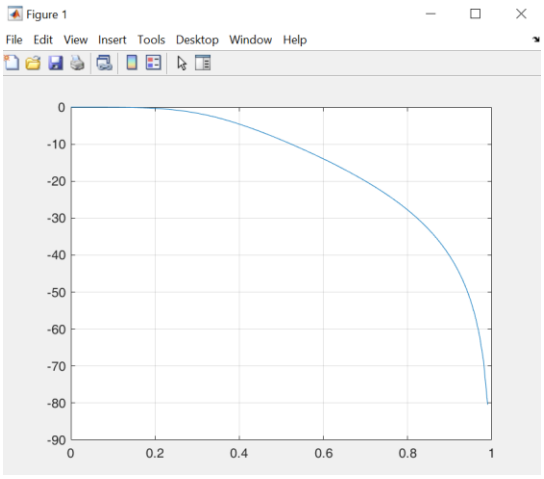
- Digital filter specifications will be known to us.
- From the given specifications we obtain specifications for the prototype continuous filter i.e Butterworth and Chebyshev filter.
- Once the specifications for the continuous time filter are known to us, we obtain the transfer function H(s).
- Finally, using suitable transformation i.e bilinear or impulse invariance, we obtain the transfer function H(z) for the desired digital filter.

- **Implementation of the experiment on MATLAB**

I. Design a Digital Butterworth filter, using Bilinear transformation, to lowpass filter an analog signal, sampled at 2kHz.The passband edge frequency is 200 Hz and the stopband edge frequency is 600 Hz. The ripple allowed in passband is 1.938 dB and minimum stopband attenuation is 13.98 dB. Plot the frequency response of your designed filter and check whether the given specifications are met.

MATLAB code

```
clc
clear all
close all
% Getting filter specifications from the user
FT=input('type in sampling freq=');
FP=input('type in passband edge freq=');
FS=input('type in stopband edge freq=');
RP=input('type in passband ripple=');
RS=input('type in minimum stopband attenuation=');
% Digital Filter passband and stopband edge frequencies
wp=FP*2/FT;
ws=FS*2/FT;
% Finding order of Butterworth filter
[N,wn]=buttord(wp,ws,RP,RS);
disp('order of the butterworth filter');disp(N);
disp('cutoff frequency');disp(wn);
%Finding system function of Butterworth filter
[b,a]=butter(N,wn);
disp('Numerator polynomial');disp(b);
disp('Denominator polynomial');disp(a);
% Obtaining and plotting frequency response of the designed filter
w=0:0.01*pi:pi;
h=freqz(b,a,w);
gain=20*log10(abs(h));
plot(w/pi,gain);
grid;
```



Frequency response of Butterworth Filter

- Design a digital Chebyshev filter to solve the above problem.

MATLAB code

```
% Getting filter specifications from the user
FT=2000;
FP=200;
FS=600;
RP=1.938;
RS=13.98;
% Digital Filter passband and stopband edge frequencies
wp=FP*2/FT;
ws=FS*2/FT;
% Finding order of chebyshev filter
[N,wp]=cheb1ord(wp,ws,RP,RS);

%Finding system function of chebyshev filter
[b,a]=cheby1(N,RP,wp);

% Obtaining and plotting frequency response of the designed filter
w=0:0.01*pi:pi;
h=freqz(b,a,w);
gain=20*log10(abs(h));
plot(w/pi,gain);
grid;
```

- **Interpretation:**

- **Assignment:**

I. Generate a signal, using MATLAB, consisting of two sinusoids of frequencies 100Hz and 1KHz. Use 'sound' command to play the signal. Now filter out the 1 KHz component and listen to the signal. Are you getting the desired signal? Try using the filter in 1.

MATLAB code

```
%filter specifications
ft=2000;
fp=200;
fs=600;
rp=1.938;
rs=13.98;
wp=(fp*2)/ft;
ws=fs*2/ft;

% finding the specification for the prototype continous time filter
[N,wn]=buttord(wp,ws,rp,rs);
[num,den]=butter(N,wn);  %finds the filter coefficients for H(s)
[b,a]=bilinear(num,den,ft);  %finds the filter coefficients for H(z)

%generating a continuous signal of 100Hz and 1KHz frequency
f1=100;
f2=1000;
```

```
t=0:1/ft:5-(1/ft);
y1=cos(2*pi*f1*t);
y2=cos(2*pi*f2*t);
y3=y1+y2;
y_f=filter(num,den,y3); %filters out the 1KHz component


sound (y3);  sound(y_f); %filters out the 1KHz component

analyses the frequency spectrum of y3
% nfft=length(y3);
% nfft2=2^nextpow2(nfft);
% ff=fft(y3,nfft2);
% fff=ff(1:nfft2/2);
% xfft=ft*(0:nfft2/2-1)/nfft2;
% plot(xfft,abs(fff));
%
% analyses the frequency spectrum of filtered signal y_f
% nfft_1=length(y_f);
% nfft_2=2^nextpow2(nfft_1);
% ff4=fft(y_f,nfft_2);
% fff4=ff4(1:nfft_2/2);
% xfft_1=ft*(0:nfft_2/2-1)/nfft_2;

% plot of all the signals

% subplot(3,2,1);
% plot(t,y1);
% subplot(3,2,2);
% plot(t,y2);
% subplot(3,2,3);
% plot(t,y3);
% subplot(3,2,4);
% plot(t,y_f);
% subplot(3,2,5);
% plot(xfft,abs(fff));
% subplot(3,2,6);
% plot(xfft_1,abs(fff4));
```

II. Generate a random signal, using MATLAB. Play the signal. Observe its spectrum. Pass it through a low pass filter with cut off frequency 1KHz. Observe the spectrum of the low pass filtered signal. Play this new signal. Comment on the result.

```
fs=2500;
t=0:1/fs:1-(1/fs);
u=0.2;
l=-.2;
%generating a random signal
x=l+(u-l)*rand(1,length(t));
y=sin(2*pi*1*t);
z=x+y;

%play the sound
sound(x);

subplot(2,3,1);
plot(t,x);

%creating a low pass filter with cut off frequency 1KHz
[b,a] = butter(2,0.8,'low');
```

```
%filtering the random signal
y_f=filter(b,a,x);

%play the sound of filtered signal
%sound(y_f);

subplot(2,3,2);
plot(t,y_f);

%spectrum of the random signal
l1=length(x);
neft=2^nextpow2(l1);
z_fft=abs(fft(x,neft));
freqaxis=fs/2*linspace(0,1,neft/2+1);

%plotting the spectrum of random signal
subplot(2,3,3);
plot(freqaxis,z_fft(1:length(freqaxis)));

%spectrum of filtered output signal
l2=length(y_f);
neft_1=2^nextpow2(l2);
z_fft_1=abs(fft(y_f,neft_1));
freqaxis_1=fs/2*linspace(0,1,neft_1/2+1);

%plotting the spectrum of filtered output signal
subplot(2,3,4);
plot(freqaxis_1,z_fft_1(1:length(freqaxis_1)));
```

- **Observation**

- The Butterworth filter and Chebyshev was successfully designed to low pass filter the sampled analog signal by using bilinear transformation.
- The Butterworth filter showed monotonic response throughout the passband and stopband whereas the Chebyshev filter showed an equiripple characterstics in the passband and the response was monotonic in the stopband.
- The two frequency components 100 Hz and 1 KHz was successfully played and the 1KHz component was filtered.
- The random signal was passed through a low pass filter and the frequencies concentrated between 0 to 1 KHz frequencies are retained while the rest are filtered out.

Experiment No.7

AIM:

(i) Design a Digital Butterworth filter, using Impulse invariance method, to low pass filter an Analog signal, sampled at 2kHz.The pass band edge frequency is 200 Hz and the stop bandedge frequency is 600 Hz. The ripple allowed in pass band is 1.938 dB and minimum stop band attenuation is 13.98 dB. Plot the frequency response of your designed filter and check whether the given specifications are met.

(ii) Design a digital Chebyshev filter to solve the above problem.

(iii) Generate a signal, using MATLAB, consisting of two sinusoids of frequencies 100Hz and 1KHz. Use 'sound' command to play the signal. Now filter out the 1 KHz component and listen to the signal. Are you getting the desired signal? Try using the filter in 1.

(iv) Generate a random signal, using MATLAB. Play the signal.Observe its spectrum. Pass it through a low pass filter with cut off frequency 1KHz. Observe the spectrum of the low passfiltered signal. Play this new signal. Comment on the result

Comparison between Butterworth filter and Chebyshev Filter

- For a particular desired specification of a digital filter, the order of Chebyshev filter will be lower as compared to Butterworth filter.
- For a particular specification, Chebyshev filter requires less hardware.
- For the same order, the transition band of Chebyshev filter is narrower as compared to Butterworth filter.
- All the poles of a Chebyshev filter lie on ellipse while in Butterworth filter, they lie on a circle.
- Chebyshev filters are of two types: Type 1 (has ripples in passband) and Type2 (ripples in stopband).

- Once the specifications for the continuous time filter are known to us, we obtain the transfer function H(s).
- Finally, using suitable transformation i.e bilinear or impulse invariance, we obtain the transfer function H(z) for the desireddigital filter.

Matlab code:

```
clear all;
close all;
clc;
ap=0.8;
as=.2;
p_d=0.2*pi;
s_d=0.6*pi;

t=.0005;
atp=-20*log10(ap);
ats=-20*log10(as);
wpp=p_d/t;
wss=s_d/t;
[n1 cf1]=buttord(wpp,wss,atp,ats,'s');
[bn1,an1]=butter(n1,1,'s');
hsn1=tf(bn1,an1);

[B1 A1]=butter(n1,cf1,'s');
hs1=tf(B1,A1);
[num1 dem1]=impinvar(B1,A1,1/t);
hz1=tf(num1,dem1,t);

[hw1 w]=freqz(num1,dem1,512);
hwman1=abs(hw1);
```

- subplot(3,1,1);
- plot(w/pi,20*log10(hwman1),'r');
- ap=0.8;
- as=.2;

- p_d=0.2*pi;
- s_d=0.6*pi;
- t=0.0005;
- atp=-20*log10(ap);
- ats=-20*log10(as);
- wpp=p_d/t;
- wss=s_d/t;

- [n2 cf2]=cheb1ord(wpp,wss,atp,ats,'s');
- [bn2,an2]=cheby1(n2,atp,1,'s');
- hsn2=tf(bn2,an2);
- [B2 A2]=cheby1(n2,atp,cf2,'s');
- hs2=tf(B2,A2);

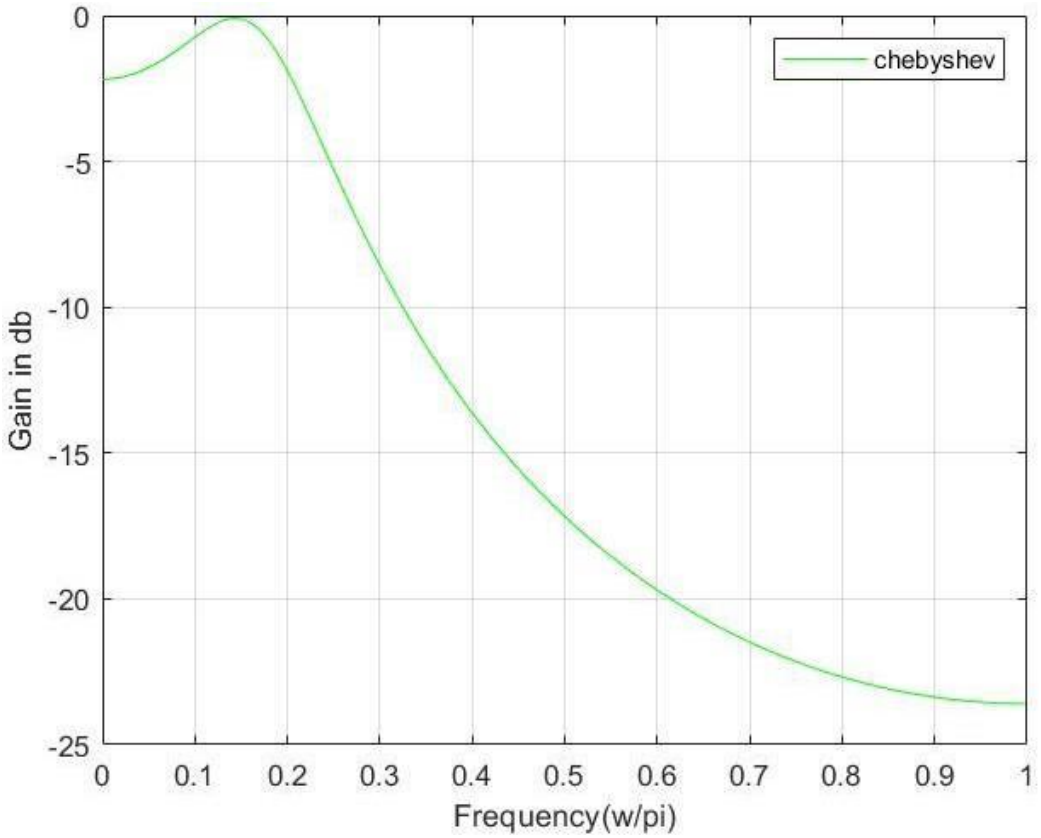
- [num2 dem2]=impinvar(B2,A2,1/t);
- hz2=tf(num2,dem2,t);
- [hw2 w]=freqz(num2,dem2,512);
- hwman2=abs(hw2);
- plot(w/pi,20*log10(hwman2),'g');
- grid on

- legend("butterworth","chebyshev"
- ts=.00048;
- f1=100;
- f2=1000;
- n=0:ts:1/f1-ts;
- p=0:length(n)-1;
- y1=sin(2*pi*f1*n);
- y2=sin(2*pi*f2*n);

- y3=y1+y2;
- [hw3 w]=freqz(y3,1,512);
- hw4=hw3.*hw1;
- y4=ifft(hw4);
- subplot(3,1,2)
- plot(w/pi,abs(hw3));
- subplot(3,1,3)
- plot(w/pi,abs(hw4));

Results:

Frequency response of Chebyshev filter, designed using impulse invariance



Experiment Number: - 08

Design of Finite Impulse Response Filter

AIM: -

(A) Generate different windows and plot their spectrum.

(B) Use fixed windows to design an FIR low pass filter.

(C) Use adjustable window (Kaiser) of different β-value to design an FIR low-pass filter.

Task to be performed.

1. Generate and plot the following windows for different values of M. Also compare their spectrums.

- Rectangular
- Barlett
- Hanning
- Hamming
- Blackman

2. Design a digital FIR low pass filter with the following specifications:

$\omega_p = 0.2\pi, \qquad R_p = o.25\ dB$

$\omega_s = 0.3\pi, \qquad A_s = 50dB$

Choose an appropriate window function from table. Determine the impulse response and provide a plot of the frequency response of the designed filter.

3. For the design specifications given above, choose the Kaiser window to design the necessary low pass filter.

Theory:

FIR Filter specifications:-

1. Absolute specifications

- band [0, ωp] is called the *pass band*, and $\delta 1$ is the acceptable tolerance (or ripple).
- band [ωs, π] is called the *stop band*, and $\delta 2$ is the corresponding tolerance (or ripple).

- band [ωp, ωs] is called the *transition band.*

2. Relative (dB) specifications

- Rp is the pass band ripple in dB.
- As is the stop band attenuation in dB.

Since $|H(e^{j\omega})|\max$ in absolute specifications is equal to (1 + δ1), we have

$$R_{p=-20log_{10}}\frac{(1-\delta 1)}{(1+\delta 1)} > 0\ (\approx 0)$$

$$A_s = -20log_{10}\frac{\delta 2}{(1+\delta 1)} > 0\ (>>1)$$

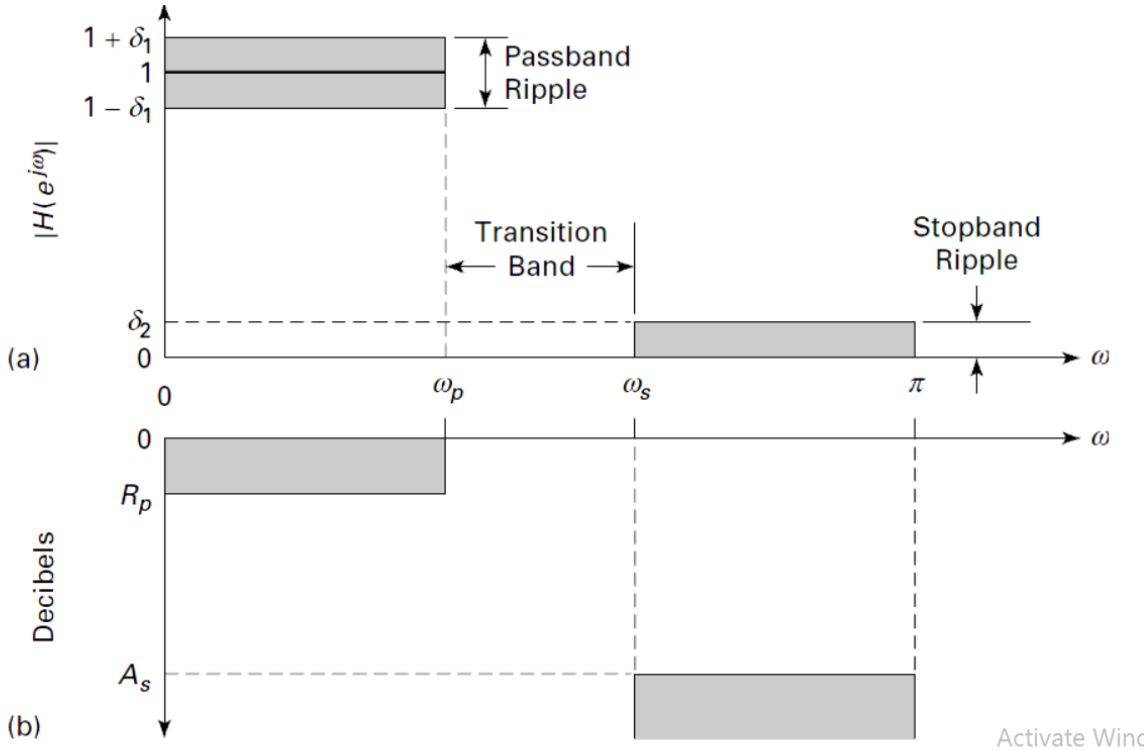


FIGURE *FIR filter specifications: (a) absolute (b) relative*

(A) Generate different windows and plot their spectrum.

To generate different windows MATLAB function required

Fixed Windows

- w=boxcar(M) or rectwin(M) returns the M-point rectangular window function in array w.
- w=bartlett(M) returns the M-point Bartlett window function in array w.

- w=hanning(M) returns the M-point Hanning window function in array w.
- w=hamming(M) returns the M-point Hamming window function in array w.
- w=blackman(M) returns the M-point Blackman window function in array w.

Adjustable Window

- w=kaiser(M,beta) returns the beta-valued M-point rectangular window

function in array w.

To generate the frequency response

- To display the frequency-domain plots, MATLAB provides the freqz function.
- **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:

Observation Table

Window Type	**Window Size(M)**	**Relative side lobe peak amplitude**	**Observed main lobe peak amplitude(dB)**	**Observed side lobe peak amplitude(dB)**	**Observed relative side lobe peak amplitude (dB)**	**Theoretical Width of major lobe**	**Observed width of the major lobe**

(B) Use windows to design a FIR low pass filter.

- The linear phase FIR filter h(n) of length M is

$$h(n) = h_d(n)\omega(n)$$

To design window filter based on window technique, an ideal low pass impulse response $h_d(n)$ needed.

Depending on how we define window w(n),we obtain different window design.

TABLE *Summary of commonly used window function characteristics*

Window Name	*Transition Width* $\Delta\omega$ *Approximate*	*Exact Values*	*Min. Stopband Attenuation*
Rectangular	$\frac{4\pi}{M}$	$\frac{1.8\pi}{M}$	21 dB
Bartlett	$\frac{8\pi}{M}$	$\frac{6.1\pi}{M}$	25 dB
Hann	$\frac{8\pi}{M}$	$\frac{6.2\pi}{M}$	44 dB
Hamming	$\frac{8\pi}{M}$	$\frac{6.6\pi}{M}$	53 dB
Blackman	$\frac{12\pi}{M}$	$\frac{11\pi}{M}$	74 dB

(C) Use Kaiser window of different β-value to design a FIR low-pass filter.

- This is an adjustable window function that is widely used in practice.

$$w(n) = \frac{I_0[\beta\sqrt{1-(1-\frac{2n}{M-1})^2}]}{I_0[\beta]}, \qquad 0 \le n \le M-1$$

Where $I_0[.]$ is the modified zero-order Bessel function given by

$$I_0(x) = 1 + \sum_{k=0}^{\infty}[\frac{(x/2)^k}{k!}]^2$$

which is positive for all real values of x.

β controls the minimum attenuation A_s.

This window can provide different transition width for the same M.

Experiment -09

Digital filter Structures

1. Consider a causal LTI system described by the difference equation Y[n]= (3/4) y[n-1]-(1/8) y[n-2] +x[n]+ (1/3)x[n-1]

(a) Find out System function H (z) for this system.

(b) Find out Impulse response sequence h[n] for this system.

(c) Obtain direct form 1 and II realization for H (z).

(d) Obtain Cascade and parallel realization for this system.

2. Write a.m. file that calculates impulse response sequence for this system, using impz function. Are you getting the same sequence as obtained in 1(b)?

3. Write a.m. file to obtained cascade and parallel realization for this system. Are you getting the same sequence as obtained in 1 (d)?

4. Use SIMULINK to create models for direct form I and II realization of H (z), using the parameters obtained in 1(c). Run the simulation for a specified stop time, say 10. Compare the impulse sequence obtained here with that obtained in 1(b) and 2.

5. Repeat for cascade and parallel realization of the system. Home Exercise

6. Obtained following structures for an FIR system with Impulse response given by h[0]=1/2 , h[1]= 1, h[2]= 1/2 , h[n]=0 otherwise. (a) Linear Phase Structure (b) Frequency Sampling Structure.

7. Construct a SIMULINK model for the system in 6 with linear phase structure. Run the simulation. Compare the impulse response sequence you obtained as result of simulation with the given sequence.

8. Repeat step 7 with frequency sampling structure.

Experiment- 10

Image Enhancement (Spatial domain)

- *For this experiment, download the images from the website of the book "Image Processing' by Gonzalez, 2nd ed. (You can experiment with images from the later editions of the book where fig/chapter nos. may be different and you need to compare the results with figs that have nos. different from the ones given in this instruction sheet).*

- *The tasks given below are to be carried out independently, by searching for the required functionsin Matlab. Purpose is to see your understanding of the topics and the effort you are putting to solve unknown problems, making use of the knowledge gained in the previous lab classes and through solving the home assignments.*

- *Write a report, after each of the tasks is performed. The report should clearly mention theobservations and their theoretical justifications.*

1. Download Fig 3.35(a). Obtain smoothing with square averaging filter masks of size n□n where n=3, 5, 9, 15 and 35. Display the processed images (compare these images with those given in the book. Are they same?). Comment on the results.

2. Download Fig 3.37(a). Obtain histogram of this image. What type of noise is it corrupted with? Give your answer by observing the histogram. Apply (i) 3□3 averaging mask and (ii) 3□3 median filter to this image. Which one is more acceptable to you? Comment on your results.

3. Obtain histogram of the gray levels in the image Fig. 3.10(b). Equalize this histogram and display the histogram equalized image. Also show the histogram of the final image.

4. Download Fig 3.20(a). Repeat the tasks given in (a). Are you satisfied with the enhancement achieved?

1. Program for expt. On rotation property of 2-d F.T.

```
name=input('give file name = ');
a=imread(name);
imshow(a);

b=imrotate(a,90);
figure;
imshow(b);
```

```
%fft_size=max(size(a));
%fft_a=fft2(a,fft_size,fft_size);
fft_a=fft2(a);
figure;
colormap(gray(256));
imagesc(log(abs(fft_a)+1));

figure;
shift_fft_a = fftshift(fft_a);
colormap(gray(256));
imagesc(log(abs(shift_fft_a)+1));

%repeat the above for rotated image

fft_b=fft2(b);
figure;
colormap(gray(256));
imagesc(log(abs(fft_b)+1));

figure;
shift_fft_b = fftshift(fft_b);
colormap(gray(256));
imagesc(log(abs(shift_fft_b)+1));
```

2. Program for expt. on filtering in spatial and frequency domain

```
%BLURRING OF IMAGES IN SPATIAL AND FREQUENCY DOMAIN (WITH DIFFERENT WINDOW SIZES)

%reads image
fid1=fopen('girl.img','rb');
img1=fread(fid1,[256,256]);
x=imrotate(img1,-90);

%displays image
figure;
colormap(gray(256));
image(x);

%generate FFT
fftx=fft2(x);

%blurring with 7 X 7 window
ker=ones(7,7)/49;

%FFT method
fftker=fft2(ker,256,256);
fft_blur1=fftx.*fftker;
blur11_x=ifft2(fft_blur1);
figure;
colormap(gray(256));
image(abs(blur11_x));

%Convolution method
blur1_x=conv2(x,ker);
```

```
figure;
colormap(gray(256));
image(blur1_x);

%blurring with 3 X 3 window
ker1=ones(3,3)/9;

%FFT method
fftker1=fft2(ker1,256,256);
fft_blur2=fftx.*fftker1;
blur21_x=ifft2(fft_blur2);
figure;
colormap(gray(256));
image(abs(blur21_x));

%Convolution method
blur2_x=conv2(x,ker1);
figure;
colormap(gray(256));
image(blur2_x);
```