DSP LAB MANUAL

Prepared by

Dr. Md. Ekramul Hamid

Department of CSE, RU

CONTENTS

Sl. No . Experiment

(d) Reconstruction of signal

(e)Add white and color noise to speech at particular SNR- show waveform, spectrogram, etc

(f)Show the FFT with changing different parameters.

(g) Show the effect of filters on noisy speech- adaptive

(h) Calculation of SNR

Experiment No 1

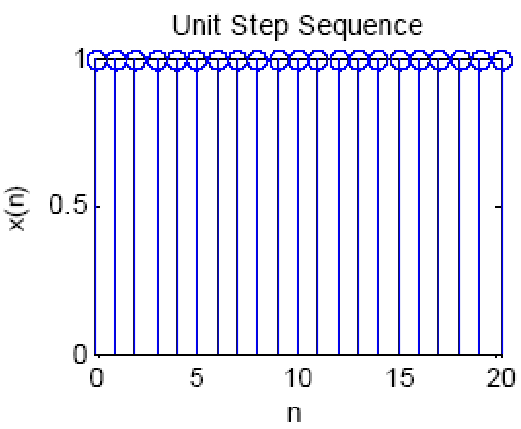
Aim of Experiment: -To develop programs for generating elementary signal functions like Unit Step, Ramp, Exponential, Sine and Cosine sequences.

Appartus: - PC installed with Matlab software.

Program: -

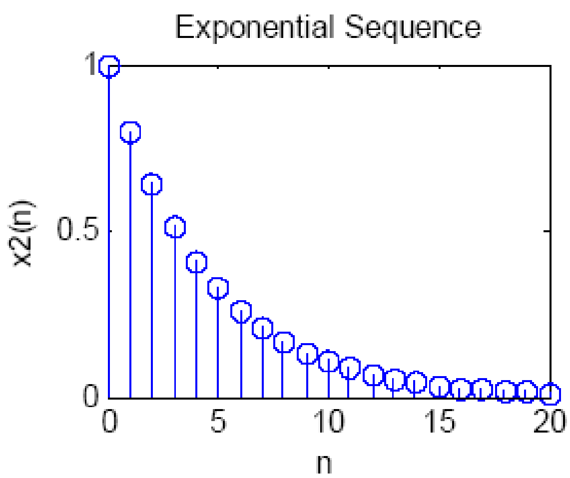
(a) %Unit Step Sequence:-

```
N=21;
x=ones(1,N);
n=0:1:N-1;
subplot(2,2,1);stem(n,x);
xlabel('n');ylabel('x(n)');
title('Unit Step Sequence');
```

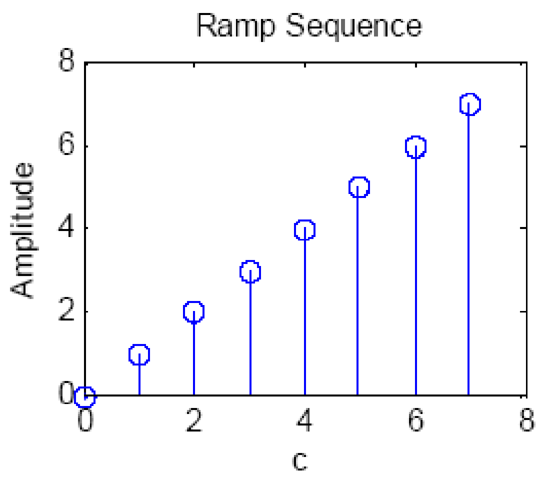


(b) %Exponential sequence: -

```
x2=0.8.^(n);
subplot(2,2,3);stem(n,x2);
xlabel('n');ylabel('x(n)');
title('Exponential Sequence');
```

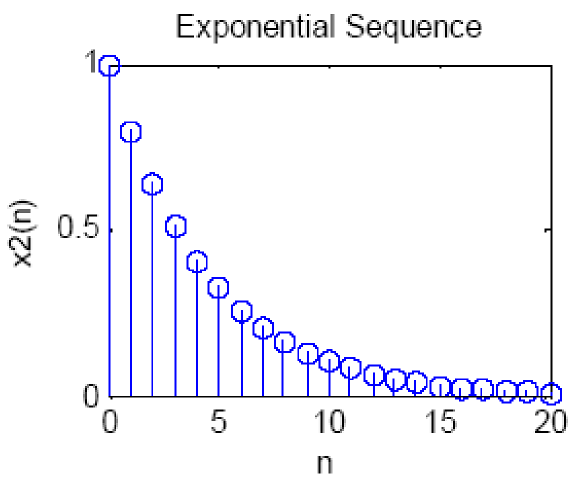


```
(c) % Ramp Sequence
x=input('enter the length of ramp sequence')
enter the length of ramp sequence
x =
7
t=0:7;
subplot(2,2,1);stem(t,t);
xlabel('c');
ylabel('Amplitude');
title(' Ramp Sequence');
```



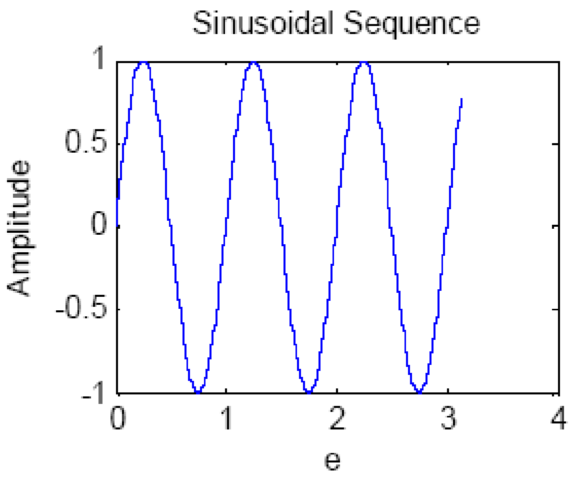
(d) %Exponential sequence: -

```
x2=0.8.^(n);
subplot(2,2,3);stem(n,x2);
xlabel('n');ylabel('x(n)');
title('Exponential Sequence');
```



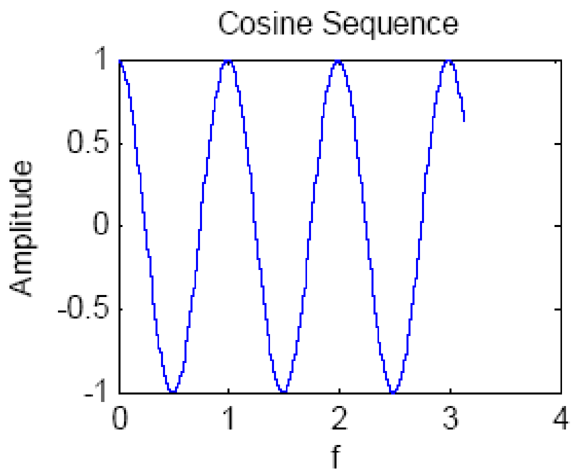
(e) %Sinusoidal sequence:-

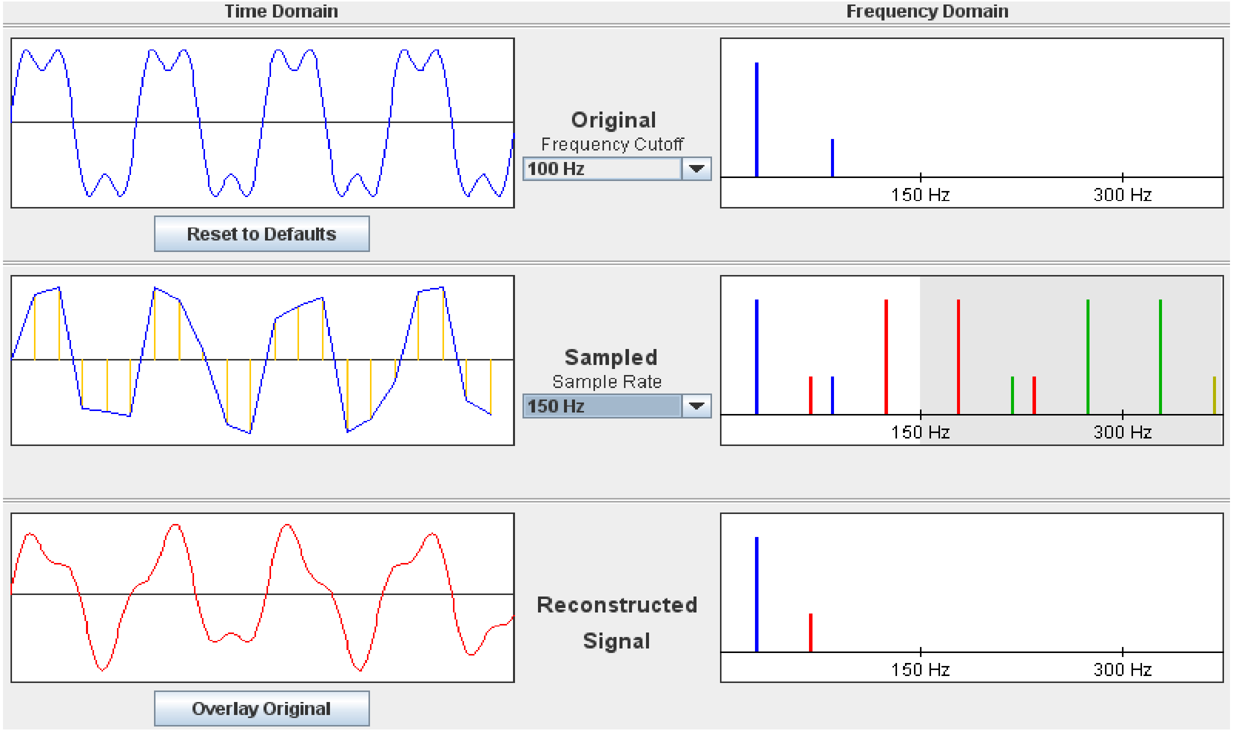
```
t=0:0.01:pi;
y=sin(2*pi*t);
subplot(2,2,1);
plot(t,y);
ylabel('Amplitude');
xlabel('e');
title('Sinusoidal Sequence');
```



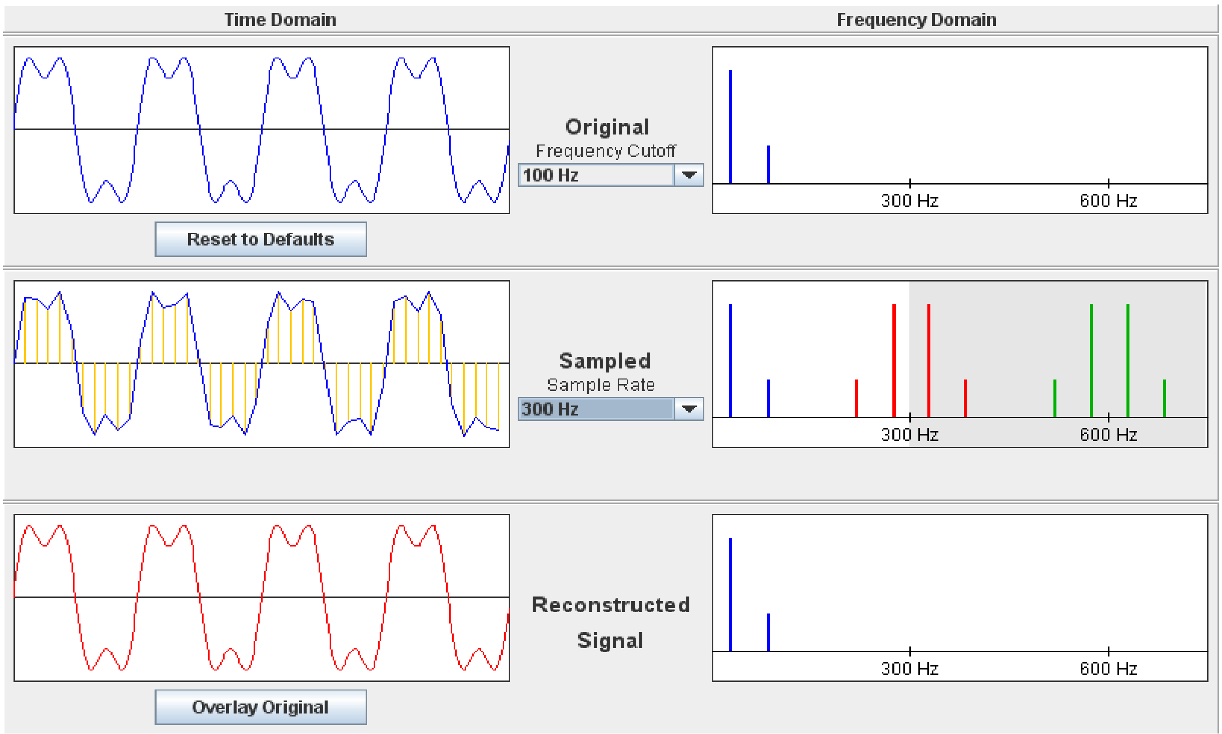
(f) % Cosine Sequence:-

```
t=0:0.01:pi;
y=cos(2*pi*t);
subplot(2,2,1);
plot(t,y);
ylabel('Amplitude');
xlabel('f');
title('Cosine Sequence');
```





Effect of under sampling



Effect of required sampling frequency

Experiment:-2

Aim of Experiment:- To develop the program for finding the convolution between two sequences.

Appartus:- PC installed with Matlab Software.

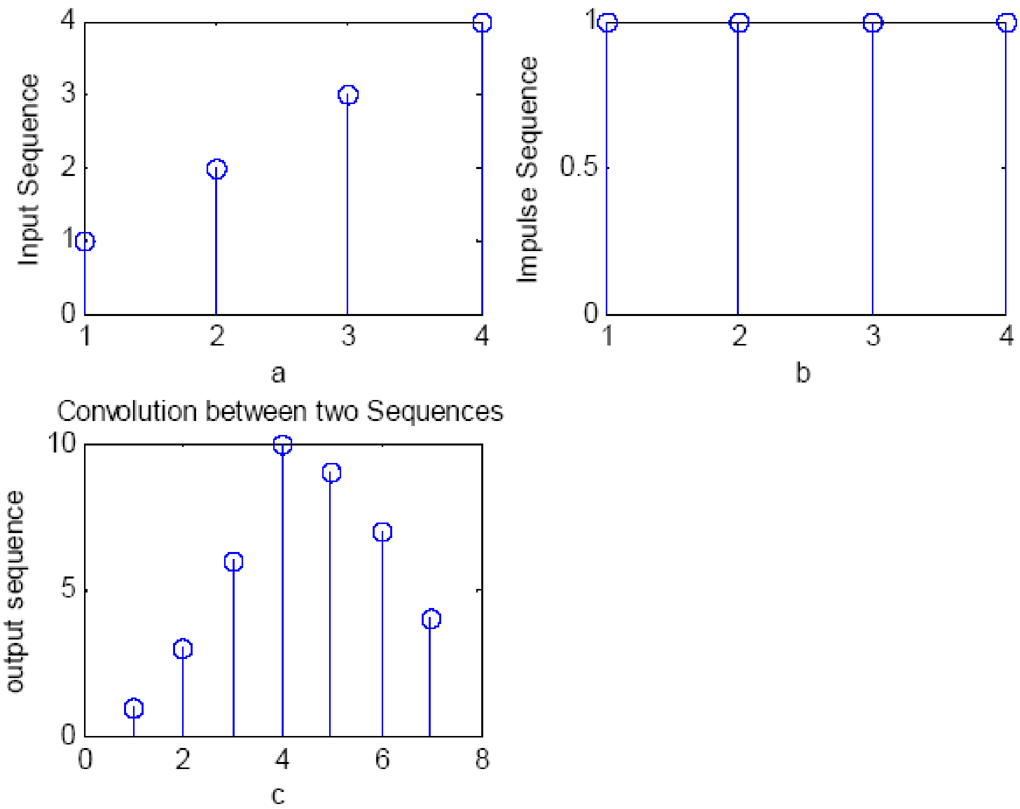
Program:-

```
x=input('enter the first sequence')
enter the first sequence[1 2 3 4]
h=input('enter the second sequence')
enter the second sequence[1 1 1 1]
y=conv(x,h);
subplot(2,2,1);
stem(x);
xlabel('a');
ylabel('Input Sequence');
subplot(2,2,2);
stem(h);
xlabel('b');
ylabel('Impulse Sequence');
subplot(2,2,3);
stem(y);
xlabel('c');
ylabel('output sequence');
title('Convolution between two Sequences');
```

Program 2:

```
clc;
close all;
clear all;
x=[1,2,1,1]; %first signal 0r input signal
h=[1,-1,1,-1]; %second signal
N1=length(x);
N2=length(h);
X=[x,zeros(1,N2)]; %padding of N2 zeros
H=[h,zeros(1,N1)]; %padding of N1 zeros
  for i=1:N1+N2-1
    y(i)=0;
    for j=1:N1
      if(i-j+1>0)
        y(i)=y(i)+X(j)*H(i-j+1);
      else
      end
```

```
    end
end
stem(y);
ylabel('y[n]');
xlabel('------>n');
title('convolution of two signal');
```



Use what is in the noisyC script to generate a noisy sine wave:

```
fs = 1e4;
t = 0:1/fs:5;
sw = sin(2*pi*262.62*t); % Middle C
n = 0.1*randn(size(sw));
swn = sw + n;
```

Use a simple lowpass (averaging) _lter:

```
b=[.25 .25 .25 .25];
a=[1 0 0 0];
y=filter(b,a,swn);
figure, plot(t,y), axis([0 0.04 -1.1 1.1])
h=impz(b,a);
y2=conv(swn,h);
figure, plot(t,y2(1:end-3)), axis([0 0.04 -1.1 1.1])
```

Q. How do the two outputs (y and y2) compare?

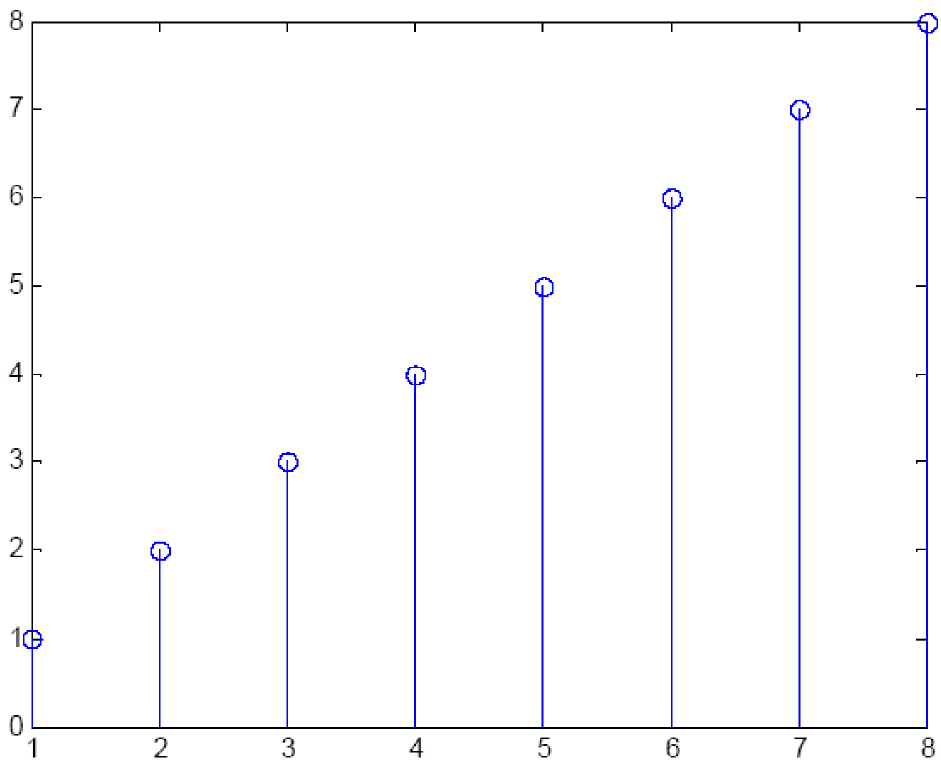
Experiment:-3

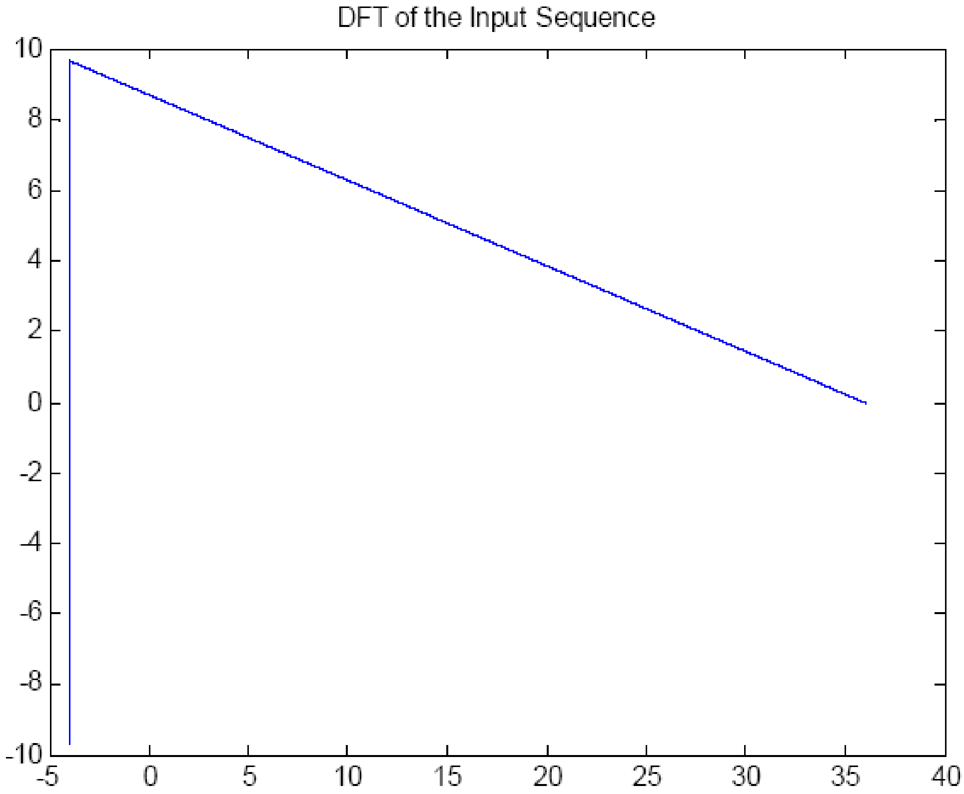
Aim of Experiment:- To develop the program for finding the DFT

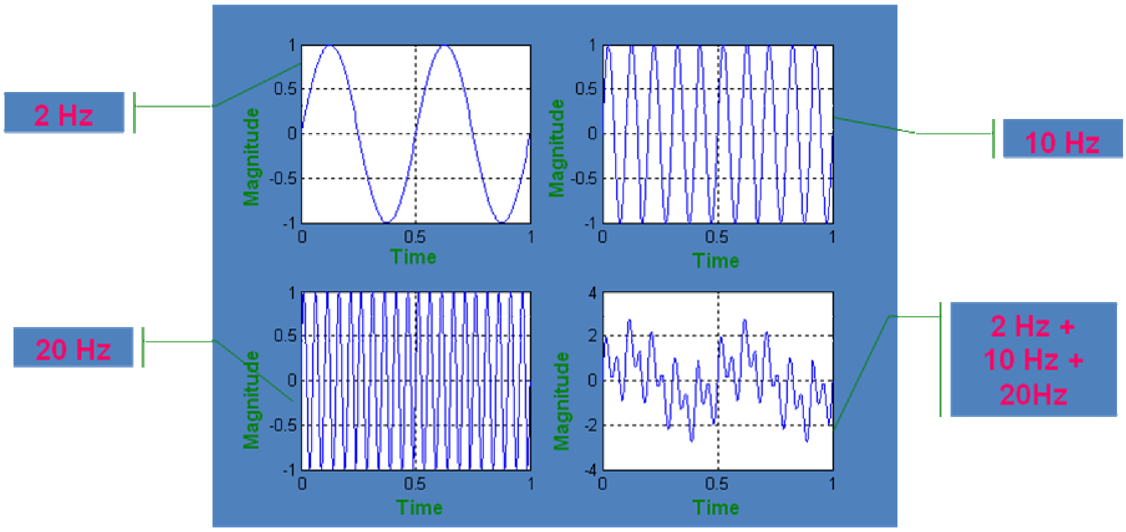
Appartus:- PC installed with Matlab Software

Program:-

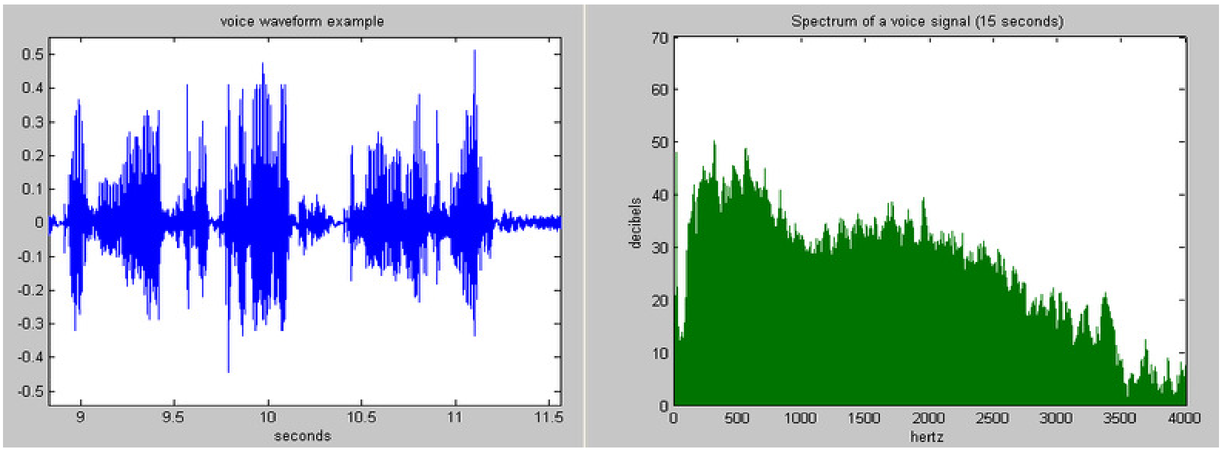
```
x=input('enter the input sequence');
enter the input sequence[1 2 3 4 5 6 7 8]
n=input('enter the length of sequence');
enter the length of sequence8
X=fft(x,n);
stem(x);
plot(x);
plot(X);
title('DFT of the Input Sequence');
```



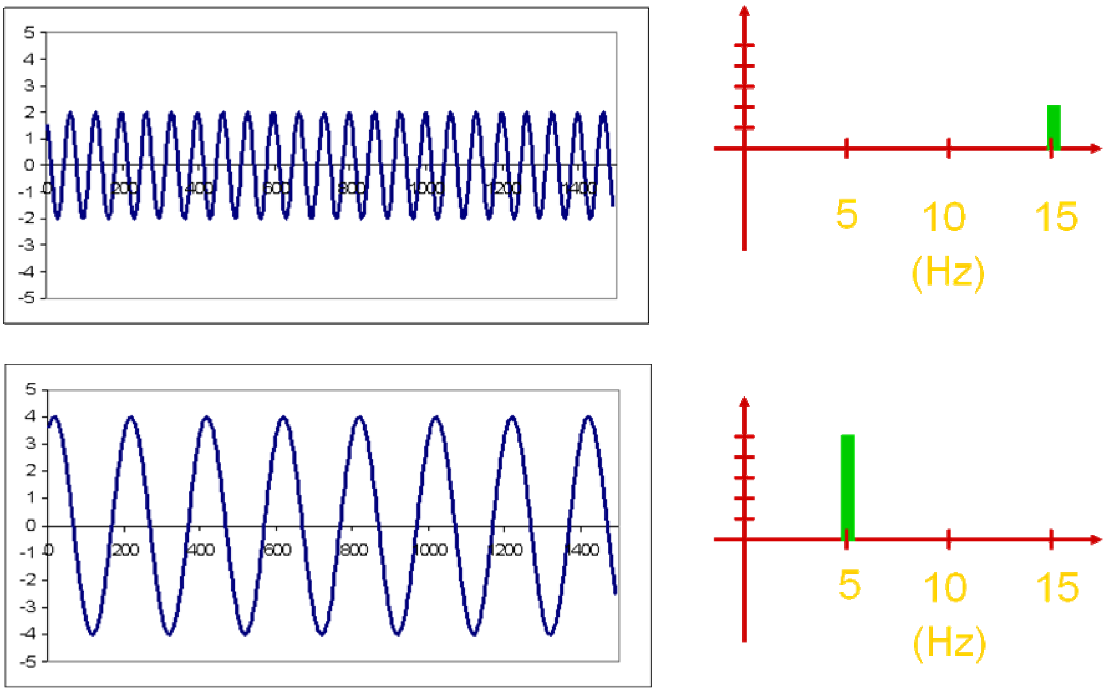




Demonstrate the addition of 3 TD signals at different frequencies, show the FFT output

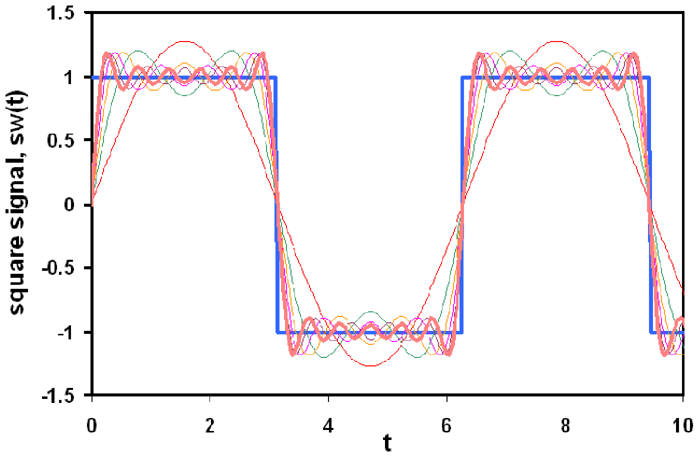


Demonstrate the power spectrum of speech signal

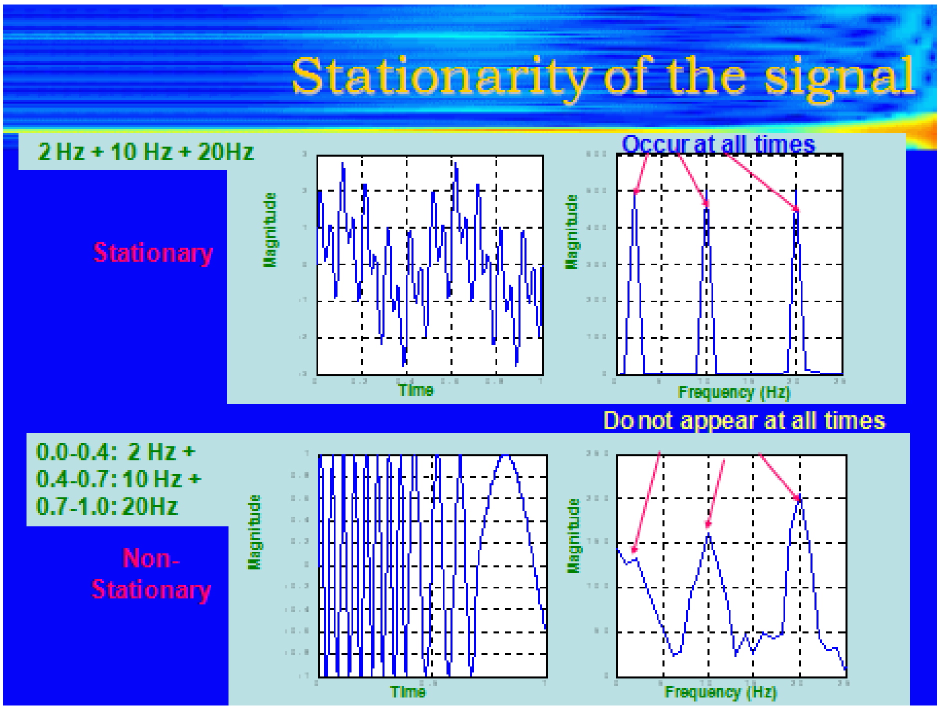


Demonstration of fourier transform

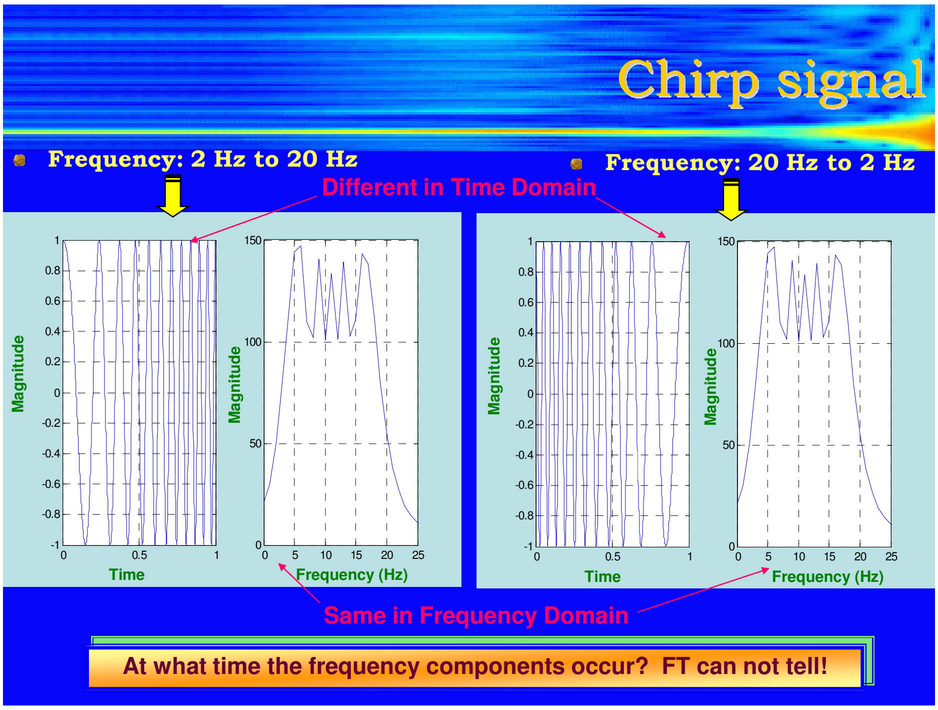
$$sw_{11}(t) = \sum_{k=1}^{11}[-b_k \cdot \sin(kt)]$$



Square wave reconstruction from spectral terms



Demonstration of stationary and non stationary signals



Chirp signal

Discrete Fourier Transform

A common use of Fourier transforms is to find the frequency components of a signal buried in a noisy time domain signal. In this exercise we will create a sound signal sampled at 1000 Hz containing 50 Hz and 120 Hz and corrupt it with some zero-mean random noise.

• Use SIN function to create a 5 second sound signal with 50Hz and 120 Hz

t = 0:0.001:5; number of sampling points at 1000Hz for 5 seconds
x = sin(2*pi*50*t)+sin(2*pi*120*t); Create a sound signal with 50Hz and 120Hz

• Use RANDN to create random noise to be added to the sound signal.
y = x + 2*randn(size(t));

• Use PLOT command to plot the corrupted sound signal
plot(1000*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')

• It is difficult to identify the frequency components by looking at the original signal. Using discrete Fourier transform to convert to the signal to frequency domain. We will perform a 512-point fast Fourier transform (FFT), calculate and plot the power spectrum.

1. Calculate FFT
Y = fft(y,512);

2. Calculate power spectrum (a measurement of the power at various frequencies)

Power spectrum = Y.* conj(Y) / 512;

3. Graph the first 257 points (the other 255 points are redundant) on a meaningful frequency axis:

f = 1000*(0:256)/512;
plot(f,Pyy(1:257))
title('Frequency content of y')
xlabel('frequency (Hz)')

Experiment:-4

Aim of Experiment:- To develop the program for finding the Autocorrelation of a sequence.

Appartus:- PC installed with Matlab Software.

Program:-

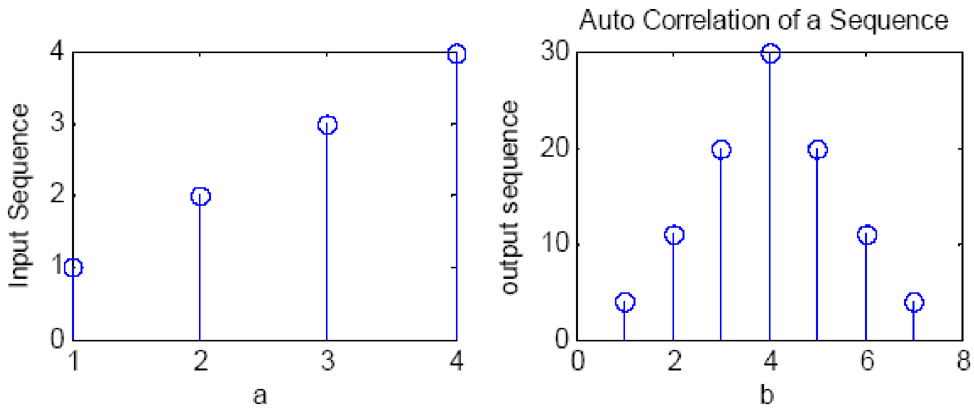
```
x=input('enter the first sequence')
enter the first sequence[1 2 3 4]
y=xcorr(x,x);
subplot(2,2,1);
stem(x);
xlabel('a');
ylabel('Input Sequence');
subplot(2,2,2);
stem(y);
xlabel('b');
ylabel('output sequence');
title('Auto Correlation of a Sequence ');
```

Program 2:

```
% correlation of two sequences

% x(n)=[3,11,7,0,-1,4,2], x=[-3:3]

% y(n)= x(n-2)+w(n), where w(n) is a random number

clc;

x=[3, 11, 7, 0, -1, 4, 2]; nx=[-3:3]; % given x(n)

% signal shift

ny=nx+2; y=x; % implement y(n)=x(n-2)

%%%%%%%%%%%%%%

w=rand(1,length(y)); nw=ny; % generate white noise w(n)

% signal addition

n=min(min(nx),min(nw)):max(max(nx),max(nw))% duration of y(n)

y1=zeros(1,length(n)); y2=y1; % initialization

y1(find((n>=min(nx))&(n<=max(nx))==1))=y; % y with duration of y
```

```
y2(find((n>=min(nw))&(n<=max(nw))==1))=w; % w with duration of y

ww=y1+y2; % sequence addition

%%%%%%%%%%%%%

% signal folding

x=fliplr(x); nx=-fliplr(nx);

% convolution of arbitrary support sequences

nyb=nx(1)+n(1);

nye=nx(length(x))+n(length(ww));

ny=[nyb:nye];

%%%%%%%%%%%%%%%%


% convolution


N1=length(x);

N2=length(ww);

X=[x,zeros(1,N2)];% padding of N2 zeros

H=[ww,zeros(1,N1)];% padding of N1 zeros

  for i=1:N1+N2-1

      y(i)=0;

    for j=1:N1

        if(i-j+1>0)

      y(i)=y(i)+X(j)*H(i-j+1);

        else

        end

    end

  end
```

```
%%%%%%%%%%%%%%%%%%%%
%subplot(1,1,1); subplot(2,1,1);
stem(ny,y);
%axis([-4,8,-50,250]); xlebel('lag variable 1')
%ylebel('ny');
%title('cross correlation with noise sequence')
```



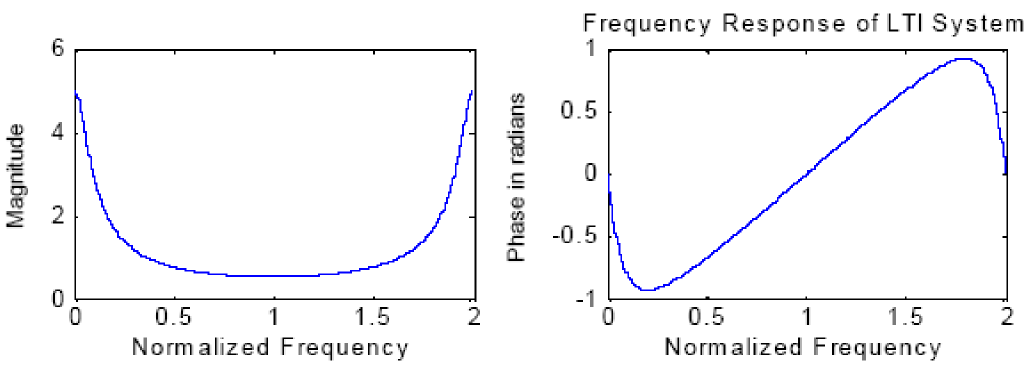
Experiment:-5

Aim of Experiment:- To develop the program for finding the magnitude and phase response of system described by system function H(s).

Appartus:- PC installed with Matlab Software.

Program:-

```
b=[1];
a=[1,-0.8];
h=freqz(b,a,w);
w=0:0.01:2*pi;
[h]=freqz(b,a,w);
subplot(2,2,1);
plot(w/pi,abs(h));
xlabel('Normalized Frequency');
ylabel('Magnitude');
subplot(2,2,2);
plot(w/pi,angle(h));
xlabel('Normalized Frequency');
ylabel('Phase in radians');
title('Frequency Response of LTI System');
```



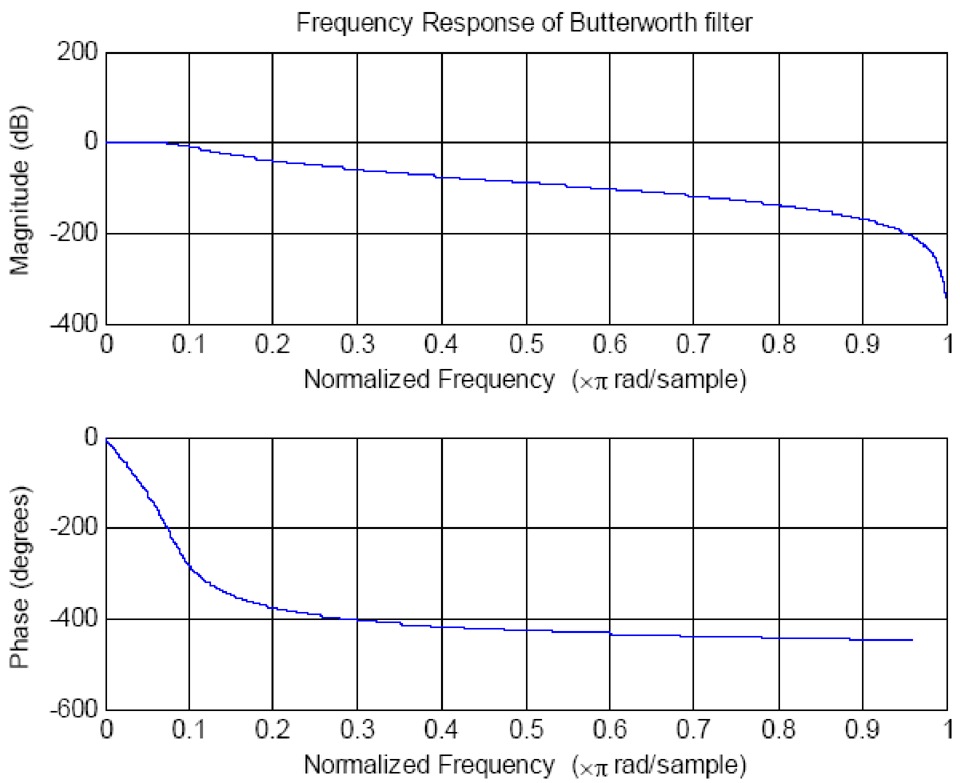
Experiment:-6

Aim of Experiment:- To develop the program for designing Low Pass Butterworth filter having passband defined from 0-40 Hz and stopband in the range of 150-500Hz having less than 3 dB of ripple in the passband and atleast 60dB of attenuation in the stopband.

Appartus:- PC installed with Matlab Software.

Program:-

```
wp=40/500;
ws=150/500;
[n,wn]=buttord(wp,ws,3,60);
n =
5
wn =
0.0810
(b,a)=butter(n,wn);
freqz(b,a)
title('Frequency Response of Butterworth filter')
```



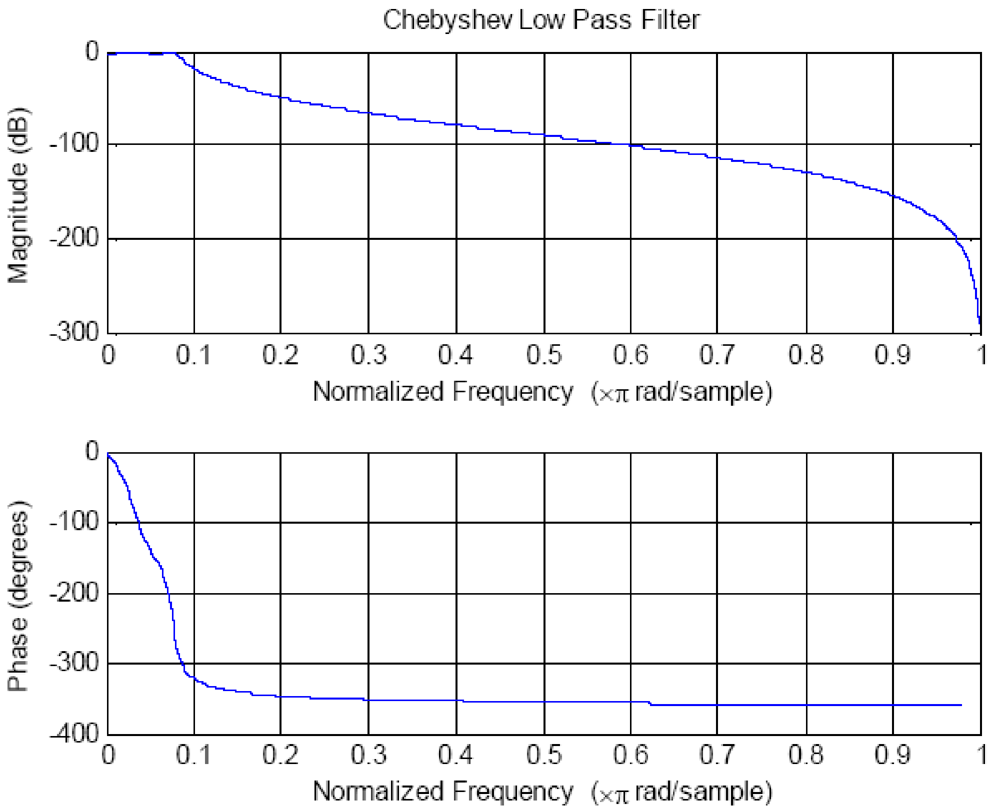
Experiment:-7

Aim of Experiment:- To develop the program for designing Low pass Type 1 Chebyshev filter having passband defined from 0-40 Hz and stopband in the range of 150-500Hz having less than 3 dB of ripple in the passband and atleast 60dB of attenuation in the stopband.

Appartus: - PC installed with Matlab Software.

Program: -

```
wp=40/500;
ws=150/500;
[n,wn]=cheb1ord(wp,ws,3,60)
n =
4
wn =
0.0800
[b,a]=cheby1(n,3,wn);
freqz(b,a)
title('Chebyshev Low Pass Filter');
```


Chebyshev Low Pass Filter
Magnitude (dB)
0
-100
-200
-300
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-100
-200
-300
-400
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)

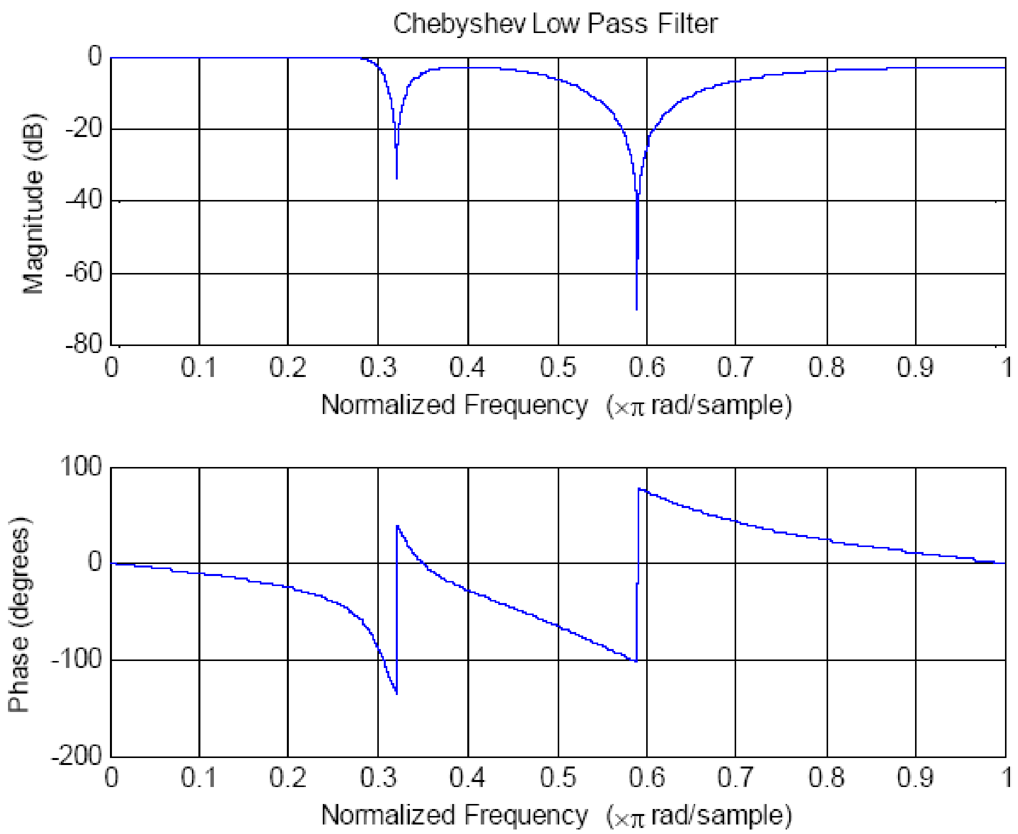
Experiment:-8

Aim of Experiment:- To develop the program for designing Low pass Type II Chebyshev filter having passband defined from 0-40 Hz and stopband in the range of 150-500Hz having less than 3 dB of ripple in the passband and atleast 60dB of attenuation in the stopband.

Appartus: - PC installed with Matlab Software.

Program: -

```
wp=40/500;
ws=150/500;
[n,wn]=cheb2ord(wp,ws,3,60)
n =
4
wn =
0.0800
[b,a]=cheby2(n,3,wn);
freqz(b,a)
title('Chebyshev Low Pass Filter');
```


Chebyshev Low Pass Filter
Magnitude (dB)
0
-20
-40
-60
-80
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
100
0
-100
-200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)

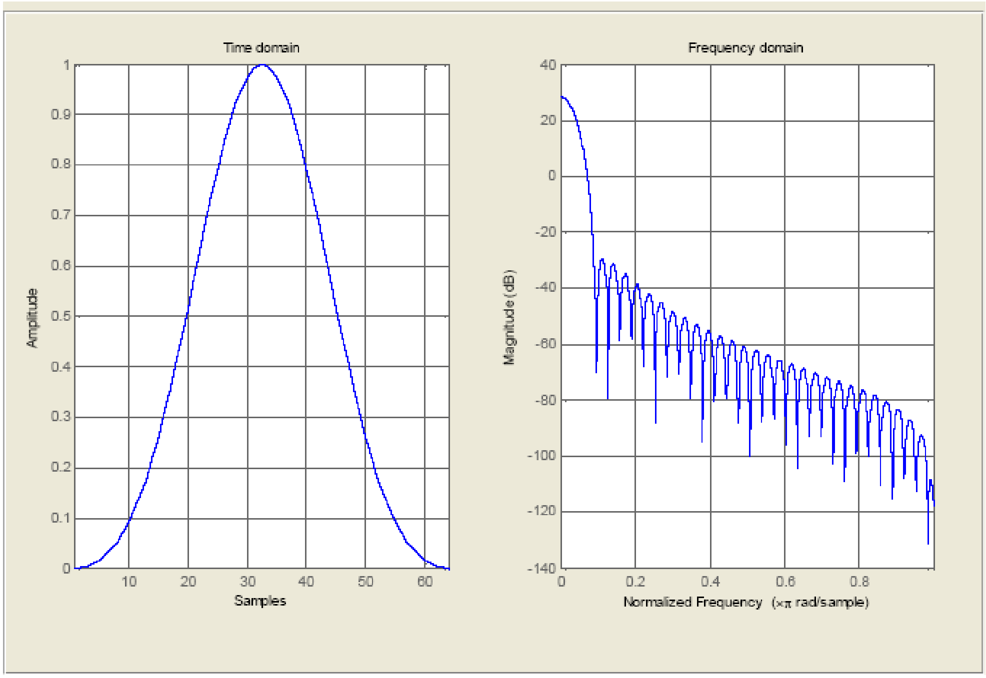
Experiment:-9

Aim of Experiment:- To develop a program for designing FIR Filters

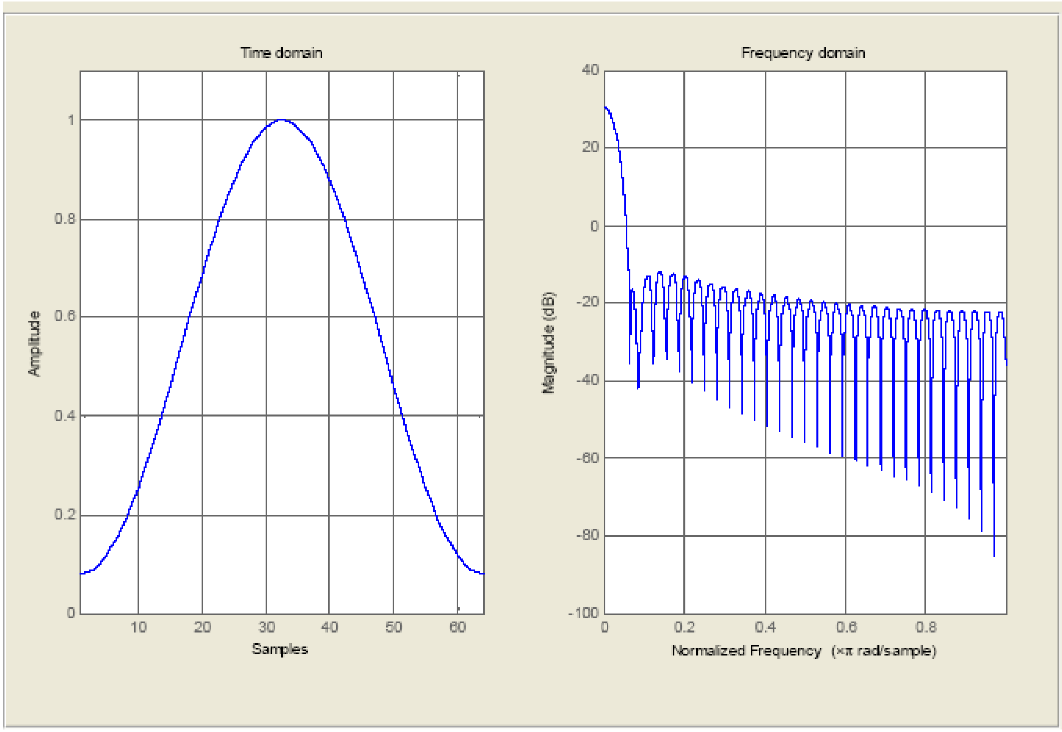
Appartus: - PC installed with Matlab Software.

Program:-

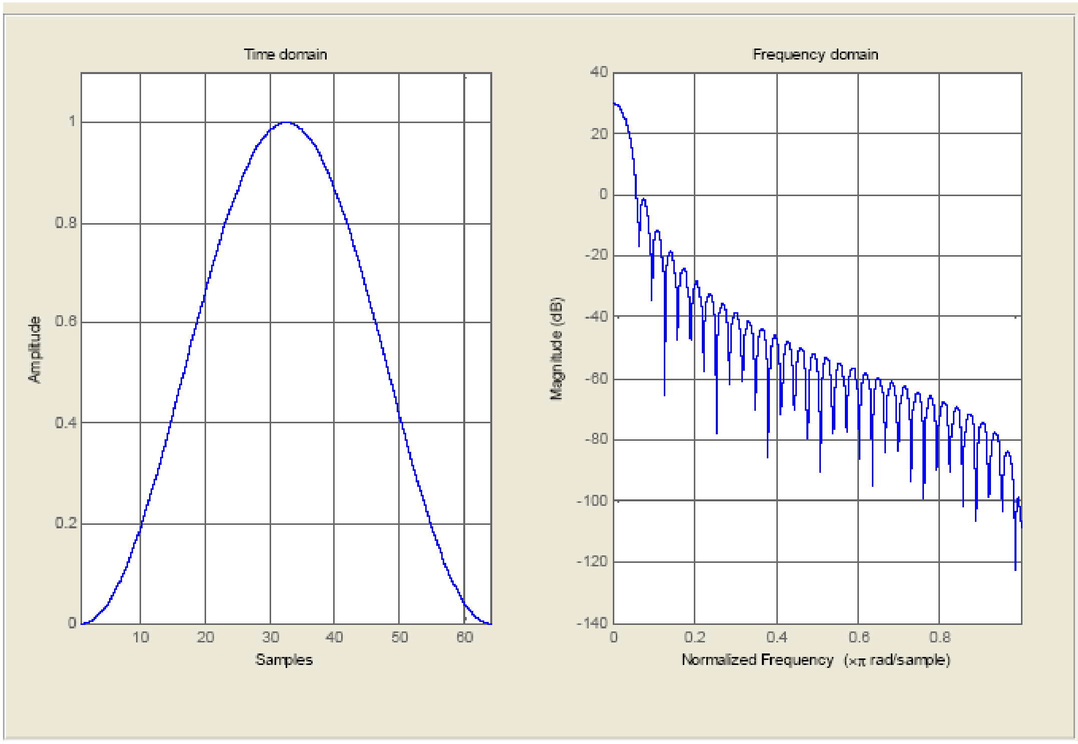
```
N=64;
wvtool(blackman(N))
wvtool(hamming(N))
wvtool(hann(N))
wvtool(gausswin(N))
wvtool(Blackman(N),hamming(N),hann(N),gausswin(N))
```



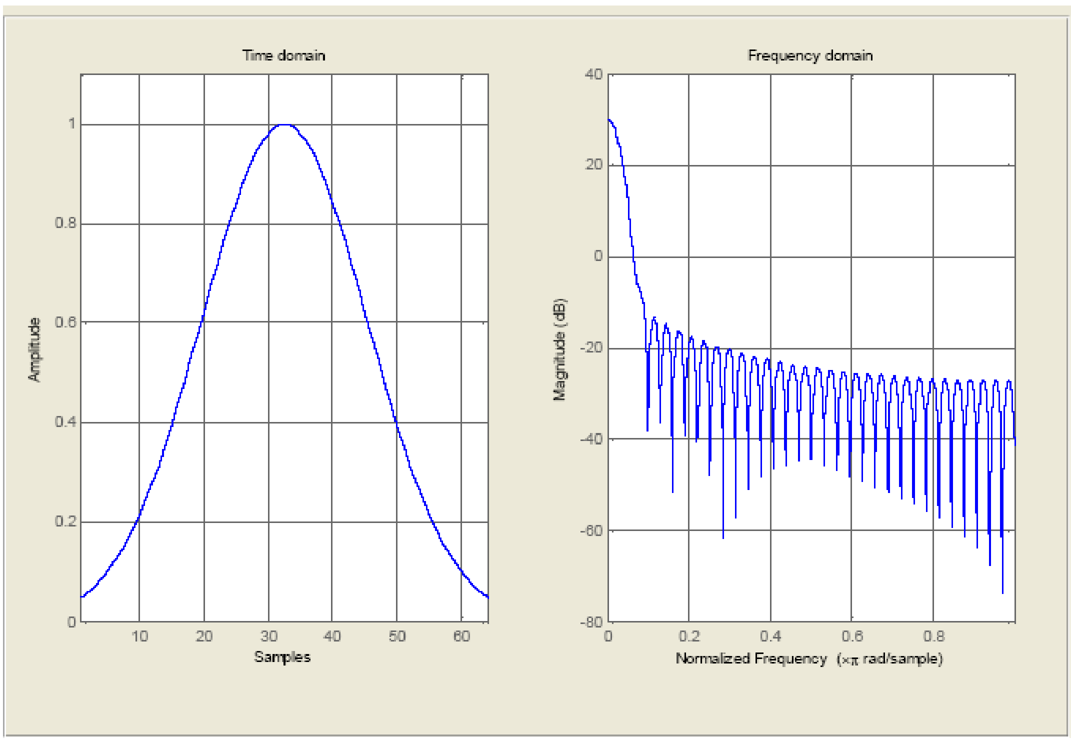
Blackman Window



Hamming Window



Hanning Window



Gaussian window

Experiment:-10

Aim of Experiment:- To develop a program for designing IIR Filters .

Appartus: - PC installed with Matlab Software.

Program:-

(A). **Using bilinear transformation.**

```
b=[2];
a=[1,3,2];
fs=1;
[B,A]=bilinear(b,a,fs);
B =0.1667 0.3333 0.1667
A =1.0000 -0.3333 0.0000
```

(B). **Using impulse invariant method**

```
b=[2];
a=[1,3,2];
fs=1;
[B,A]=impinvar(b,a,fs)
B =
0 0.4651
A =
1.0000 -0.5032 0.0498
```

Experiment:-11

Aim of Experiment:- Analysis of Z transform and Inverse Z Transform.

Appartus: - PC installed with Matlab Software

```
%Analysis of Z-Transforms %Definition of numerators and denominator
coefficients
num=[5 6 -44 21 32];
den=[5 13 15 18 -12]; %Conversion from rational to Factored form
[z,p,k]=tf2zp(num,den);
disp('Zeros are at');disp(z);
disp('Poles are at');disp(p);
disp('Gain Constant');disp(k); %Determination of radius of the poles
radius=abs(p);
disp('Radius of the poles');
disp(radius); %Pole Zero Map using numerator and denominator
coefficients
zplane(num,den) %Conversion from factored to secomd ordered factored
sos=zp2sos(z,p,k)
disp('Second Order Sections');disp(sos);
```

```
%Inverse Z-Transform using impz
%definition of numerator and denominator coefficients
num=[0.1+.4*i 5 .05];
den=[1 .9+0.3*i .12];
%Finding the inverse z transform of G(z)
[a,b]=impz(num,den);
%Evaluating on Unit Circle i.e. Fourier Transform
[c,d]=freqz(num,den);
% Plotting of x[n] and it's fourier transform
subplot(2,2,1)
stem(b,real(a))
title('Real Part of g[n]')
xlabel('Samples'); ylabel('Magnitude')
grid on
subplot(2,2,2)
stem(b,imag(a))
title('Imaginary Part of g[n]')
xlabel('Samples'); ylabel('Magnitude')
grid on
subplot(2,2,3)
plot(d/pi,abs(c))
title('Magnitude Spectrum of g[n]')
xlabel('\omega/\pi'); ylabel('Magnitude')
grid on
subplot(2,2,4)
plot(d/pi,angle(c))
title('Phase Spectrum of g[n]')
xlabel('\omega/\pi'); ylabel('Phase, radians')
grid on
```

Experiment:-12

Aim of Experiment:- To develop an application with speech signal .

Appartus: - PC installed with Matlab Software

```
clc
clear
%%%%%%%%%read of clean speech file%%%%%%%

[x, fs]=wavread('2.wav');  %fs=sampling frequency.  x=sample point(the
sampled data in y)
block=256;%floor(0.023*fs); block=number of sample data in each block.
n=floor(length(x)/block);% number of blocks

x=x(1:block*n);% x will start from 1 and go upto blockno*no of
block=total data

figure,plot(x);
title('Original signal');
%%%%%%%%%%%%%%% Generate noise using random number (white noise)
%%%%%%%%%%%%%%%%%%%

db=0; % SNR
%signal=x/max(abs(x));
sig_var=cov(x); % covariance of x
randn('state',0); % use default initial stage of random number
v1=randn(1,size(x)); % generate random number (noise), lenght is 1 to
size of x
noise_var=sig_var*10^(-db/10); % find noise variance for the particular
SNR
v=sqrt(noise_var)*v1; % modify noise with noise variance
xn=x+v'; % add noise with clean speech we have noisy speech
wavwrite(xn,11000,'nois.wav');% making wav file of noisy speech
figure,plot(xn);
```

THIS APPLICATION WILL EXTEND STEP BY STEP

SNR CALCULATION

```
function [overall_snr, segmental_snr] = snr(clean_speech,
processed_speech)

% ---------------------------------------------------------------------
-
% Check the length of the clean and processed speech.  Must be the
same.
% ---------------------------------------------------------------------
-

clean_length      = length(clean_speech);
processed_length  = length(processed_speech);

if (clean_length ~= processed_length)
  disp('Error: Both Speech Files must be same length.');
  return
end

% ---------------------------------------------------------------------
-
% Scale both clean speech and processed speech to have same dynamic
% range.  Also remove DC component from each signal
% ---------------------------------------------------------------------
-

clean_speech     = clean_speech     - mean(clean_speech);
processed_speech = processed_speech - mean(processed_speech);

processed_speech = processed_speech.*(max(abs(clean_speech))/...
                      max(abs(processed_speech)));

overall_snr = 10*log10(sum(clean_speech.^2)/sum((processed_speech-
clean_speech).^2));

% ---------------------------------------------------------------------
-
% Global Variables
% ---------------------------------------------------------------------
-

sample_rate =  16000;          % default sample rate
winlength   =  320;        % window length in samples
skiprate    =  100;        % window skip in samples
MIN_SNR     = -10;         % minimum SNR in dB
MAX_SNR     =  30;         % maximum SNR in dB

% ---------------------------------------------------------------------
-
% For each frame of input speech, calculate the Segmental SNR
% ---------------------------------------------------------------------
-
```

```
num_frames = clean_length/skiprate-(winlength/skiprate); % number of
frames
start      = 1;                  % starting sample
window     = 0.54 -0.46*cos(2*pi*(1:winlength)'/(winlength+1));
for frame_count = 1:num_frames

   % -----------------------------------------------------------
   % (1) Get the Frames for the test and reference speech.
   %     Multiply by Hanning Window.
   % -----------------------------------------------------------

   clean_frame = clean_speech(start:start+winlength-1);
   processed_frame = processed_speech(start:start+winlength-1);
   clean_frame = clean_frame.*window;
   processed_frame = processed_frame.*window;

   % -----------------------------------------------------------
   % (2) Compute the Segmental SNR
   % -----------------------------------------------------------

   signal_energy = sum(clean_frame.^2);
   noise_energy  = sum((clean_frame-processed_frame).^2);
   segmental_snr(frame_count) = 10*log10(signal_energy/noise_energy);
   segmental_snr(frame_count) =
max(segmental_snr(frame_count),MIN_SNR);
   segmental_snr(frame_count) =
min(segmental_snr(frame_count),MAX_SNR);

   start = start + skiprate;

end
```