Indian Institute of Engineering Science and Technology, Shibpur

Department of Electronics and Telecommunication Engineering

REPORT ON

Digital Signal Processing Lab Report

REPORT ON

Name : Shaik Umar Farooq

Date of Performance : 21/03/2023

Semester : Sixth Semester

Date of Submission : 11/04/2023

Branch : Electronics and Telecommunication

Roll No. : 2020ETB064

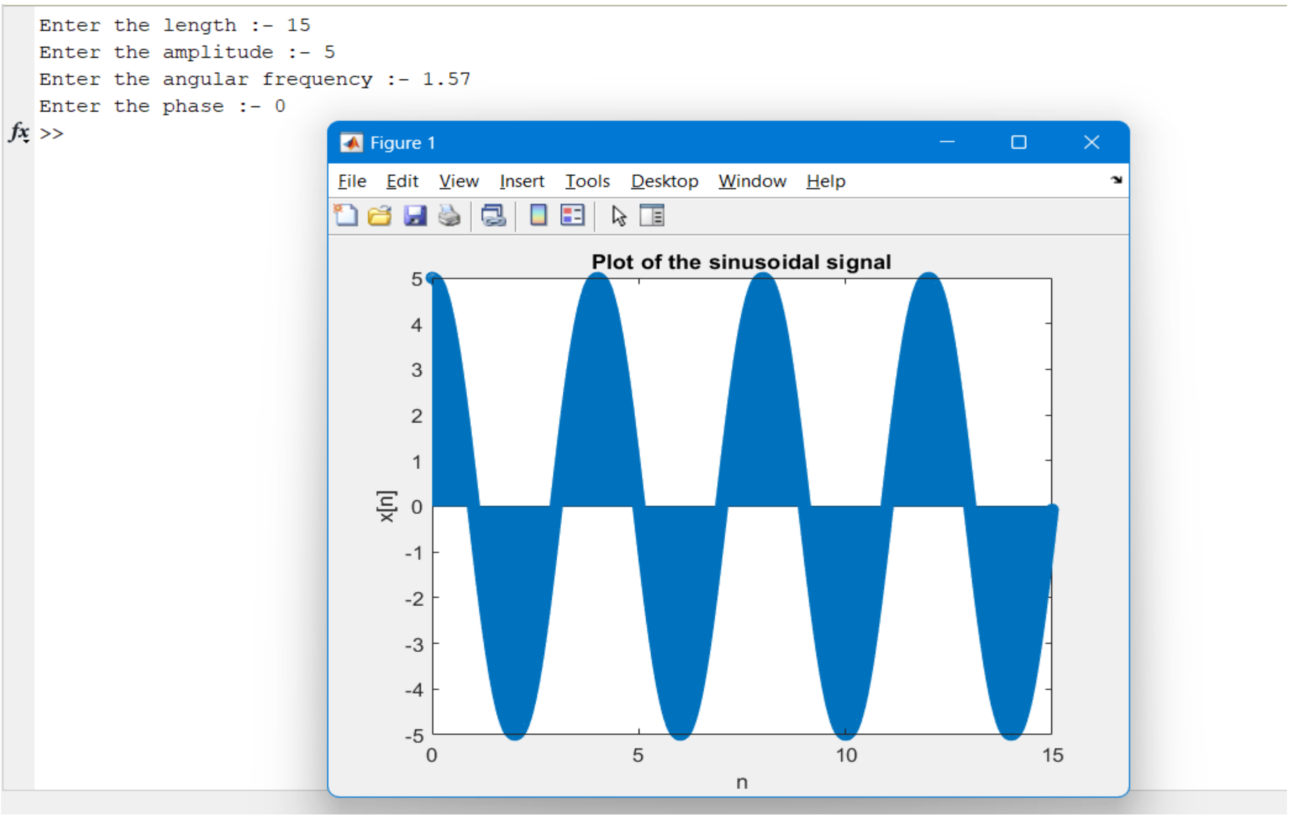
Examined By: ______________________________

Assignment– 1
Discrete time signals and systems

1. Write a MATLAB program to generate a sinusoidal sequence **x[n] = A cos(w0*n + phi)** , and plot the sequence using the 'stem' function. The input data specified by the user are the desired length L , amplitude A , the angular frequency w0 and the phase (phi) where 0<w0<π and 0<(phi)<π with a sampling rate of 20 KHz.

Ans :

```
clc
clear all
close all
l = input("Enter the length :- ");
A = input("Enter the amplitude :- ");
w0 = input("Enter the angular frequency :- ");
phi = input("Enter the phase :- ");
fs = 20000;
n = 0:1/fs:l;
x = A*cos(w0*n + phi);
stem(n,x);
title("Plot of the sinusoidal signal");
xlabel("n");
ylabel("x[n]");
```



2. A discrete-time system is represented by the following input output relation:

$$y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$$

This is an example of M-point moving average filter. Such a system is often used in smoothing random variations in data. Consider for example a signal corrupted by a noise whose minimum and maximum values are -0.5 and 0.5 respectively, i.e.

$$x[n] = s[n] + d[n]$$

Original uncorrupted signal is given by,

$$s[n] = 2[n(0.9)^n]$$

Investigate the effect of signal smoothing by a moving average filter of length 5, 7 and 9. Does the filtered signal improve with an increase in the filter-length? Is there any effect of the filter-length on the delay between the smoothed output and the noisy input?

Ans. :

```
clc
clear all
close all
n = 0:1:100;
s = 2*n.*((0.9).^n);
figure(1);
stem(n,s, "Linewidth",2);
title("Plot of the original uncorrupted signal");
xlabel("n");
ylabel("s[n]");

d = randi([-5,5],1,length(s))/10;
figure(2);
stem(n,d, "Linewidth", 2);
title("Plot of the noise signal");
xlabel("n");
ylabel("d [n]");

x=d+s;
figure(3);
stem(n,x, "Linewidth", 2);
title("Plot of the noise corrupted signal");
xlabel("n");
ylabel("y[n]")
```

```
f=4;
for m=5:2:9
    y=zeros(size(x));
    for i = 0 : m - 1
        for j = 1 : m - 1
            if (j - 1) > 0
                y(j)=y(j)+x(j-1);
            end
        end
        for j=m:100
            y(j)=y(j)+x(j-1);
        end
    end
    y=y/m;
    figure(f);
    stem(n,x, "Linewidth", 2);
    str = sprintf("Plot of the filtered signal with m=%d ",m);
    title(str);
    xlabel("n");
    ylabel("y[n]")
    f=f+1;
end
```

Figure : 1

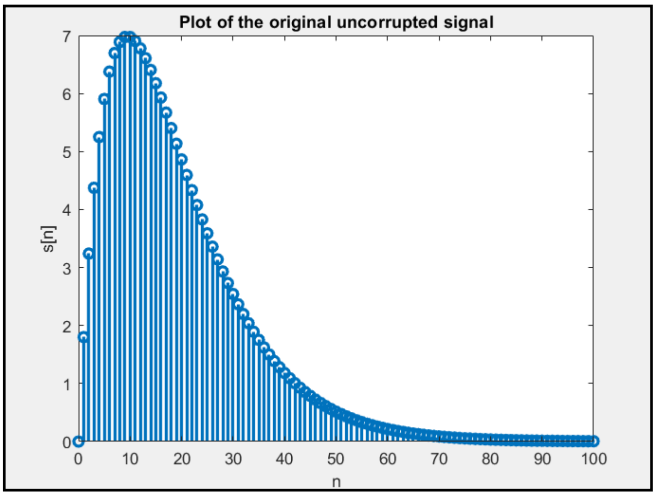


Figure : 2

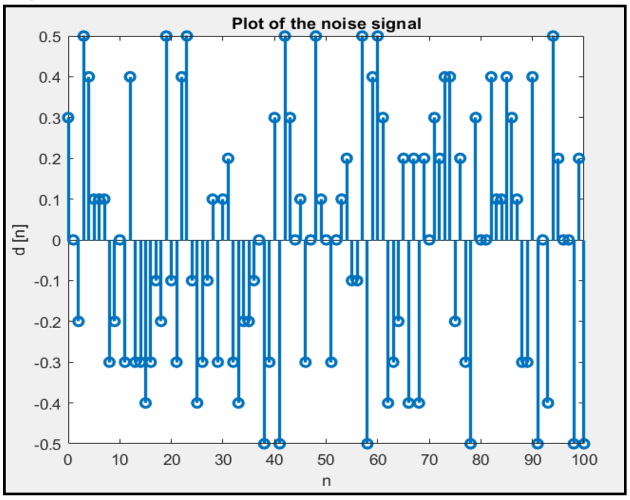


Figure : 3

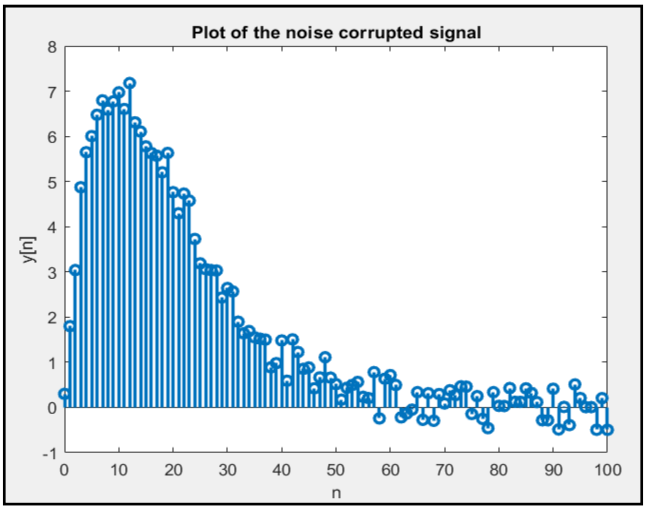


Figure : 4

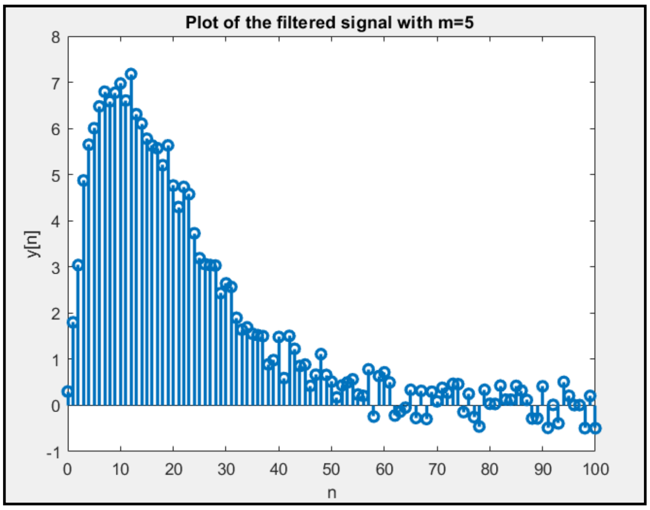


Figure : 5

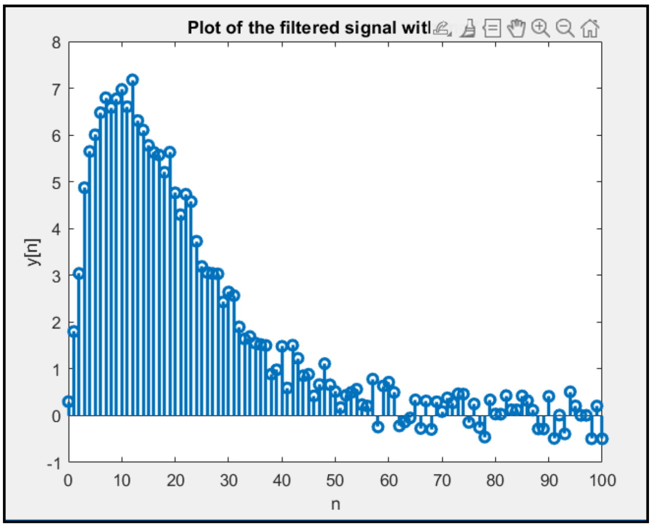
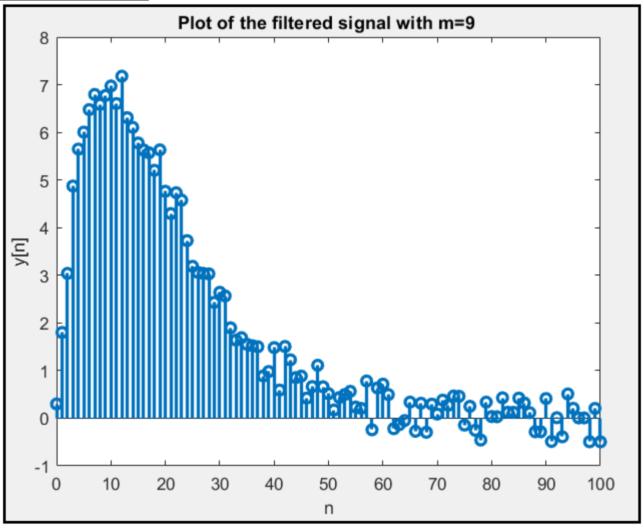


Figure : 6



3. Write a MATLAB program implementing the discrete-time system given by following input output relation,

$$y[n] = 0.5(y[n-1] + \frac{x[n]}{y[n-1]})$$

Show that the output y[n] of this system; for an input x[n] = αμ[n] with y[-1] = 1 converges to as where, is a positive number.

Ans. :

```
clc
clear all
close all
n=0:1:50;
a=100;
x=a*ones(size(n));
x=[0 x];
y=zeros(size(n));
y=[1 y];
for i=2:52
    y(i) = 0.5*(y(i-1)+(x(i)./y(i-1)));
end
stem(n,x(2:52), "Linewidth",2);
figure(1);
title("Plot of the input signal");
xlabel("n");
ylabel("x[n]");
figure(2);
stem(n,y(2:52),"Linewidth",2);
title("Plot of the output signal");
xlabel("n");
ylabel("x[n]");
hold on
plot (n,a^.5*ones(size(n)),'Linewidth',2);
legend('y[n]','Root of a');
```

Figure : 1

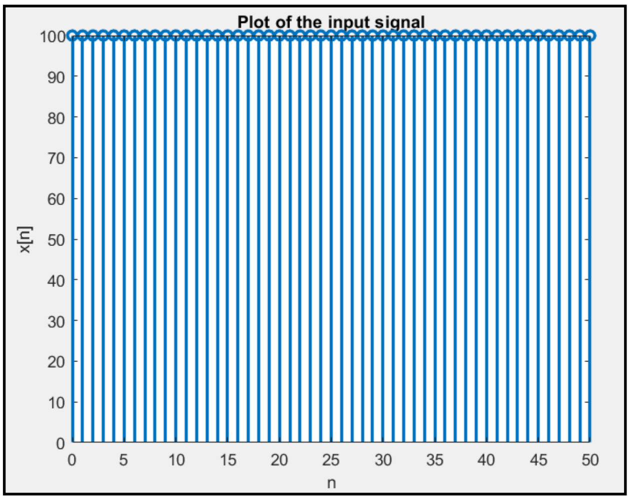
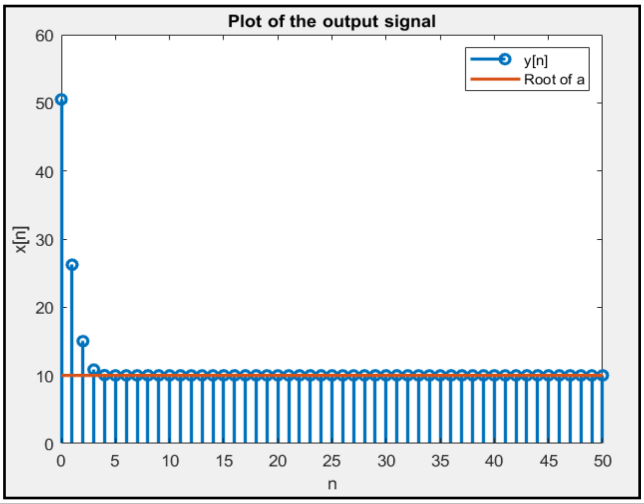


Figure : 2



4. Plot the given input speech file in MATLAB and write a program to implement a quantizer for the given speech file. Plot the variation of signal to quantization noise ratio against variation of number of bits/sample.

Ans. :

```
clc
clear all
close all
load mtlb;
x = mtlb;
t = (0:(length(x)-1))/Fs;
figure(1);
plot(t,x);
title("Original speech signal");
xlabel('Time');
ylabel('Audio Signal');
for j = 1:50
    b(j) = j;
    delta=(max(x)-min(x))/(2^j);
    Q  =  round(x/delta)*delta;
    ps = rms(x)^2;
    pn = rms(x-Q)^2;
    sqnr(j) = 10*log10(ps/pn);
end
figure(2);
plot(b,sqnr,"-o","Linewidth",2);
title("SQNR versus bits per sample");
xlabel("Bits per sample");
ylabel("SQNR (dB)");
```

Figure : 1

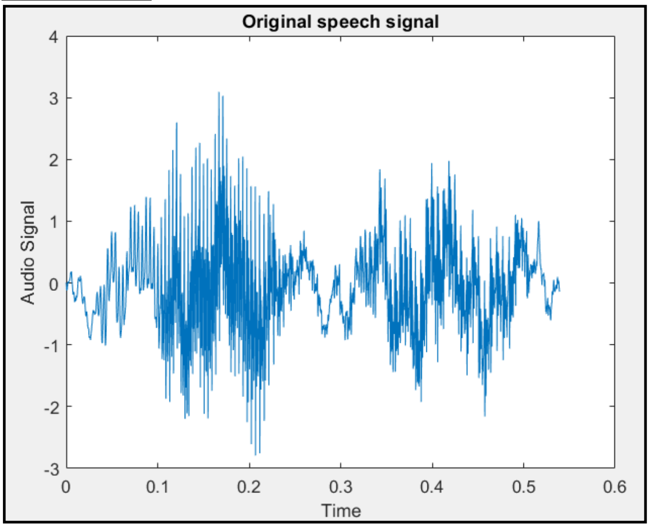
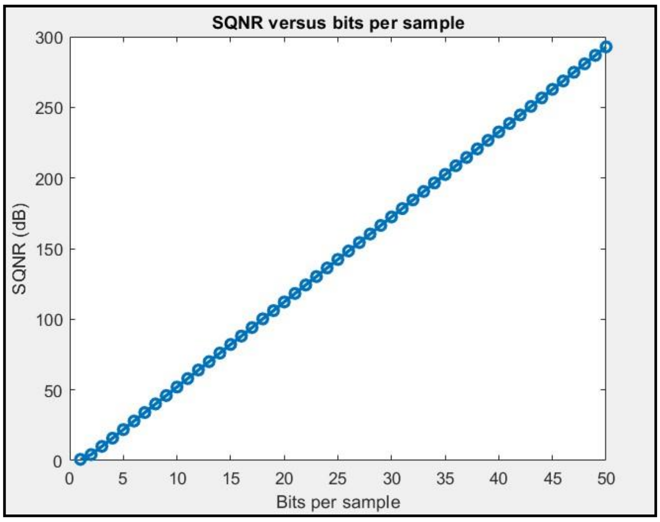


Figure : 2



Assignment – 2

Discrete time signals and systems in the transform domain

1. Compute an N-point DFT of the following sequences and plot its magnitude and phase spectrum.

$$x[n] = \begin{cases} A & \text{for } n = 0,1,2,\ldots\ldots, M-1 \\ 0 & \text{otherwise} \end{cases}$$

   where M=10 and (i) N=10 (ii) N=100 (iii) N=256

   Plot the magnitude spectrum of DTFT of and compare the plots for different lengths.

Ans. :

```
clc
clear all
close all
M = 10;
A = 10;
x= [zeros(1,length(-5:-1)) A*ones(1,length(0:M-1)) zeros(1,length(M:M+4))];
n = -5:M+4 ;
figure(1);
stem(n,x,"Linewidth",2);
N = 100;
w = -pi:(2*pi/N):pi ;
X = [x zeros(1,(N-length(x)))];
y = zeros(1, length(w));
for i = 1:length(w)
    temp = 0;
    for j = 1:N
        temp = temp + X(j)*exp(-sqrt(-1)*w(i)*(j-1));
    end
    y(i)= temp;
end
m = abs(y);
p = unwrap(angle(y));
figure;
subplot(2,1,1);
stem (w,m,"Linewidth",1.5);
str = sprintf("Magnitude spectrum for N = %d",N);
title(str);
subplot(2,1,2);
stem(w,p,"Linewidth",1.5);
str=sprintf("Phase spectrum for N = %d",N);
title(str);
```

Figure : 1

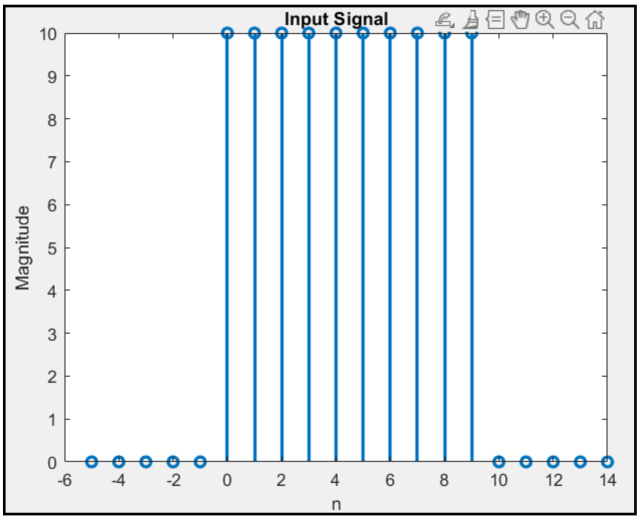
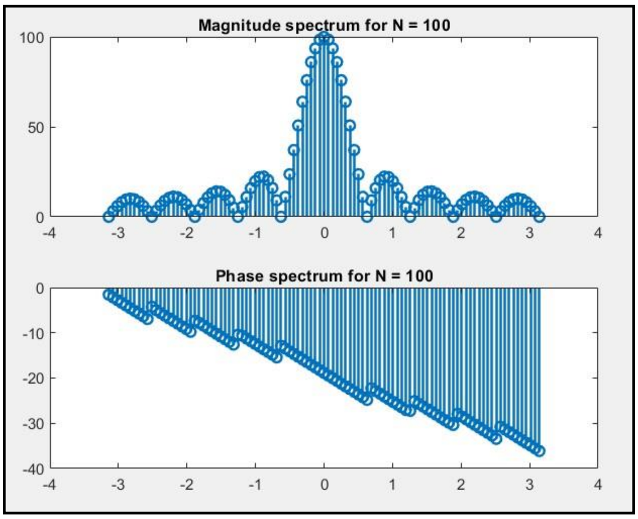


Figure : 2



2. Write a program to plot the magnitude and phase response of discrete-time system characterized by its impulse response:

$$h[n] = \begin{cases} 0.5 & \text{for } n = 0 \\ \dfrac{\sin 0.5\pi n}{\pi n} & \text{otherwise} \end{cases}$$

   i. n = -8:7
   ii. n = -16:15
   iii. n = -64:63

Ans. :

```
clc
clear all
close all
n = -8:7;
n2 = -16:15;
n3 = -64:63;
h = sin (0.5*pi*n) ./ (pi*n) ;
for a=1:3
    h = sin(0.5*pi*n)./(pi*n);
    h(abs(n(1))+1) = 0.5;
    figure(a);
    subplot (3, 1, 1)
    stem(n,h,"r","Linewidth",2);
    title('Plot of h with n');
    h = @(k)(k~=0).*(sin(0.5*pi*k)./(pi*k+eps))+(k==0)*0.5;
    N = 2^nextpow2(length(h(n)));
    w = -pi:2*pi/N:pi;
    x = [h(n) zeros(1,(N-length(h(n))))];
    y = zeros(1,length(w));
    for i=1:length(w)
        temp  =  0;
        for j=1:N
            temp = temp+x(j)*exp(-(sqrt(-1))*w(i)*(j-1));
        end
        y(i)=temp;
    end
    m = abs(y);
    p = unwrap (angle(y));
    subplot(3,1,2);
    stem(w,m,"Linewidth",2);
    title('Magnitude Response')
    subplot(3,1,3)
    stem(w,p,"Linewidth",2);
    title("Phase Response");
    if (a==1)
        n = n2;
    else
        n = n3;
    end
end
```

Figure : 1 - For (i) n=-8:7

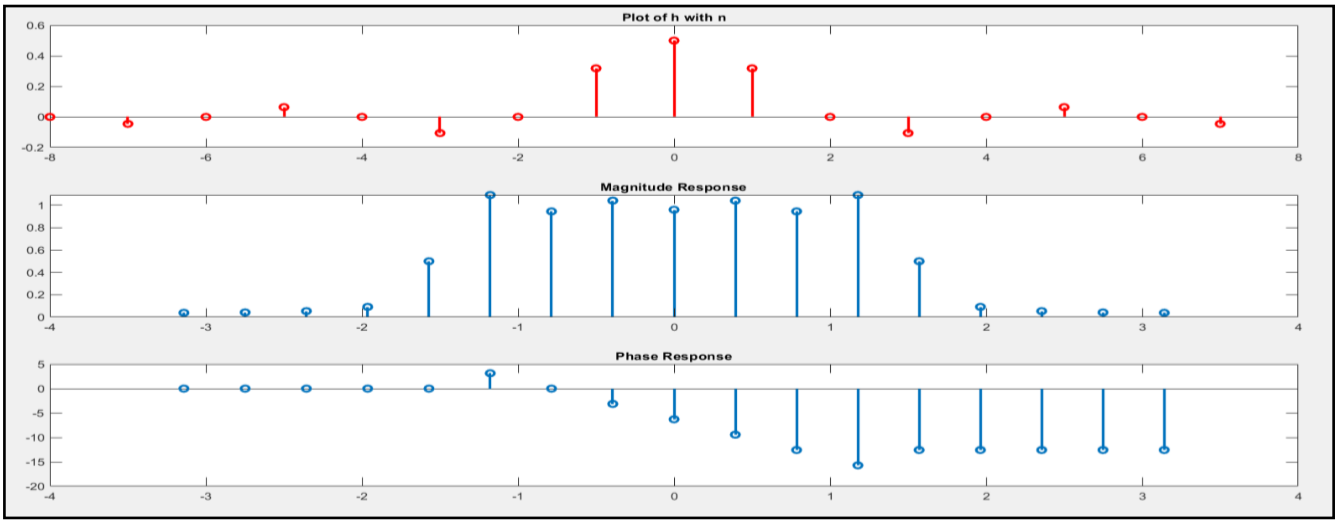


Figure : 2 – For (ii) n=-16:15

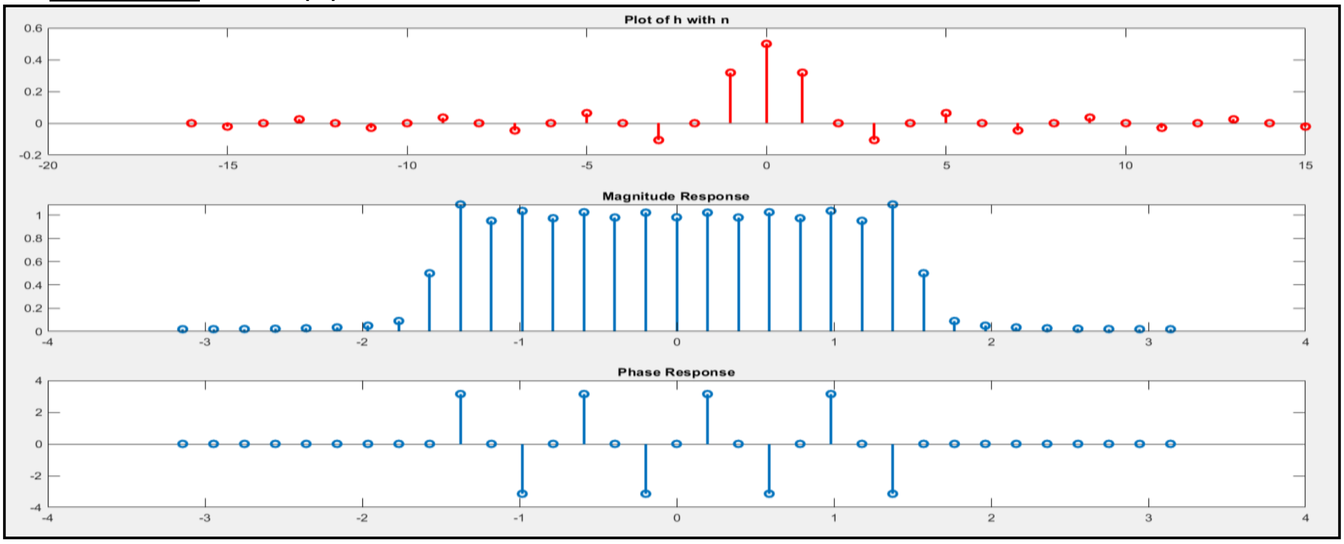
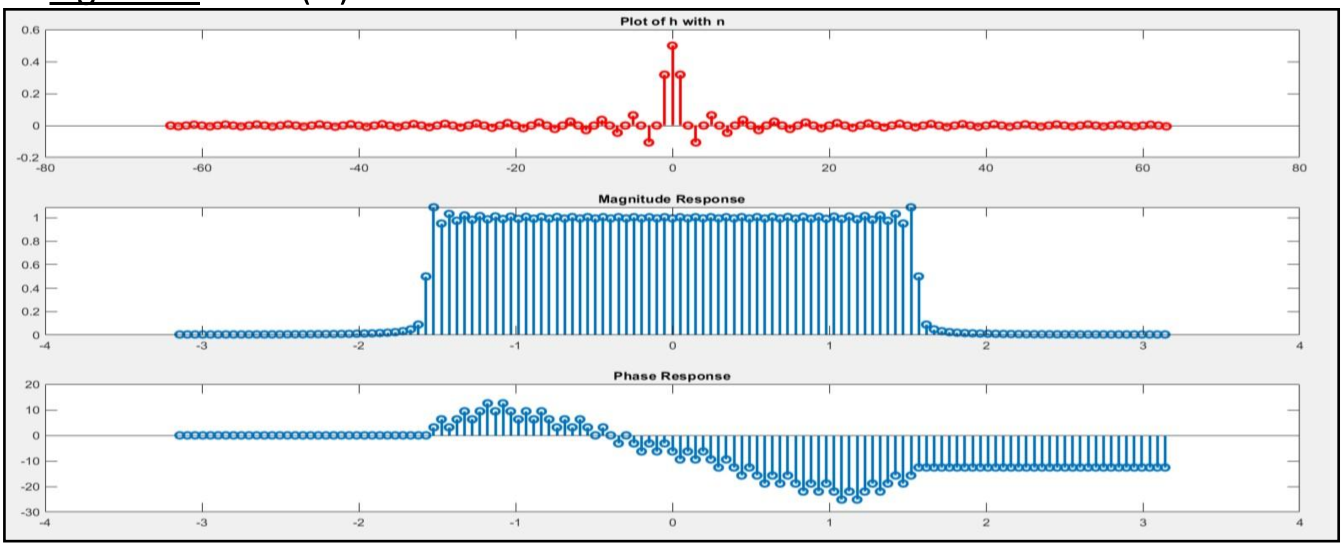


Figure : 3 – For (iii) n=-64:63



3. Evaluate and plot the spectrum of your own voice signal.

Ans. :

```
clc
clear all
close all
rec = audiorecorder;
disp("Begin Speaking");
recordblocking(rec,10);
disp("End Speaking");
x=getaudiodata(rec);
fs = 800;
t = (0:(length(x)-1))/fs;
figure(1)
plot(t,x);
title('Plot of original audio Signal');
xlabel('Time');
ylabel('Amplitude');
y = fft(x);
l  =  length(x);
p2  =  abs(y/l);
p1 = p2(1:l/2+1);
f = fs*(0:l/2)/l;
figure(2);
plot(f,p1);
title("Single-Sided Amplitude Spectrum of voice signal");
xlabel('Frequency(Hertz)');
ylabel ("|P(f)|");
```

Figure : 1

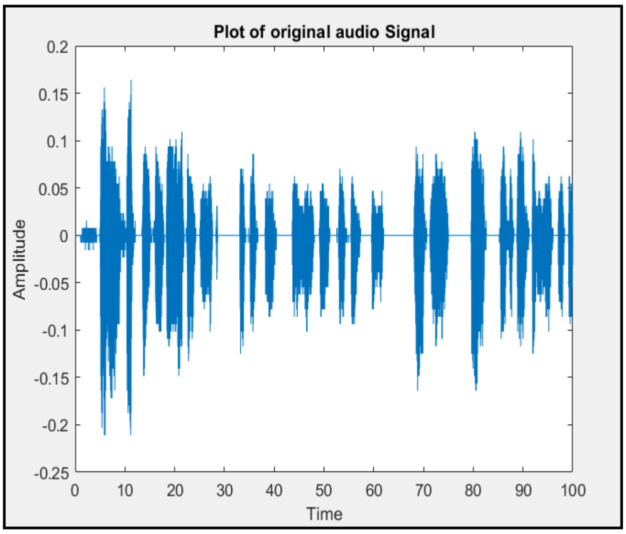
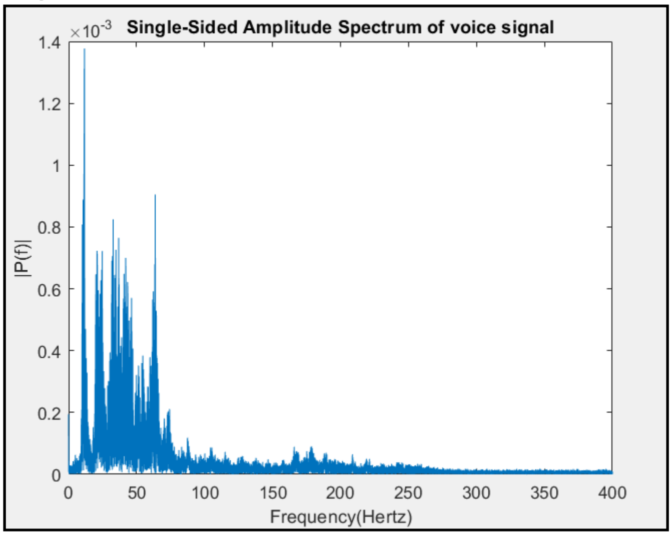


Figure : 2



4. Write a program to implement linear convolution via DFT-based approach, and compare your results using direct linear convolution.

Ans. :

```
clc
clear all
close all
f = [1 2 3 4 5];
g = [6 7 8 9 10];
n = 1:5 ;
figure(1);
subplot(2,1,1);
stem(n,f,"Linewidth",2);
title("First function")
subplot(2,1,2);
stem (n,g,"Linewidth",2);
title("Second function")
N = 2^nextpow2(max([length(f) length(g)]));
f_h = [f zeros(1, N-length(f))];
g_h = [g zeros(1, N-length(g))];
w = -pi:2*pi/N:pi;
yf = zeros(size(w));
yg = zeros(size(w));
for i=1:length(w)
    temp_f = 0;
    temp_g = 0;
    for j = 1:N
        temp_f = temp_f + f_h(j)*exp(-(sqrt(-1))*w(i)*(j-1));
        temp_g = temp_g + g_h(j)*exp(-(sqrt(-1))*w(i)*(j-1));
    end
    yf(i)=temp_f;
    yg(i)=temp_g;
end
yx = yf.*yg;
x = zeros(1,length(f)+length(g)-1);
for i = 1:length(yx)
    temp = 0;
    for j = 1:N
        temp = temp+(1/N)*yx(j)*exp((sqrt(-1))*w(j)*(i-1));
    end
    x(i) = temp;
end
c=conv(f,g);
figure(2);
subplot(2,1,1);
stem(x, "Linewidth",2);
title("Convolution of f and g by DFT method.")
subplot(2,1,2);
stem(c, "Linewidth",2);
title('Direct Linear Convolution of f and g');
```

Figure : 1

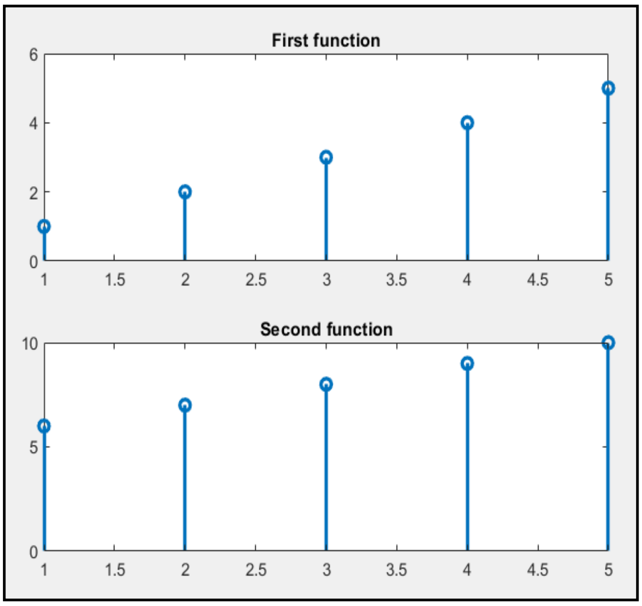
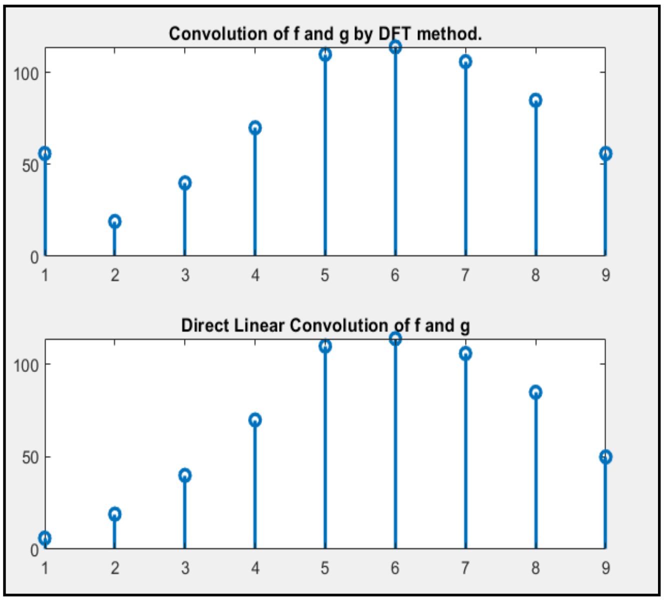


Figure : 2

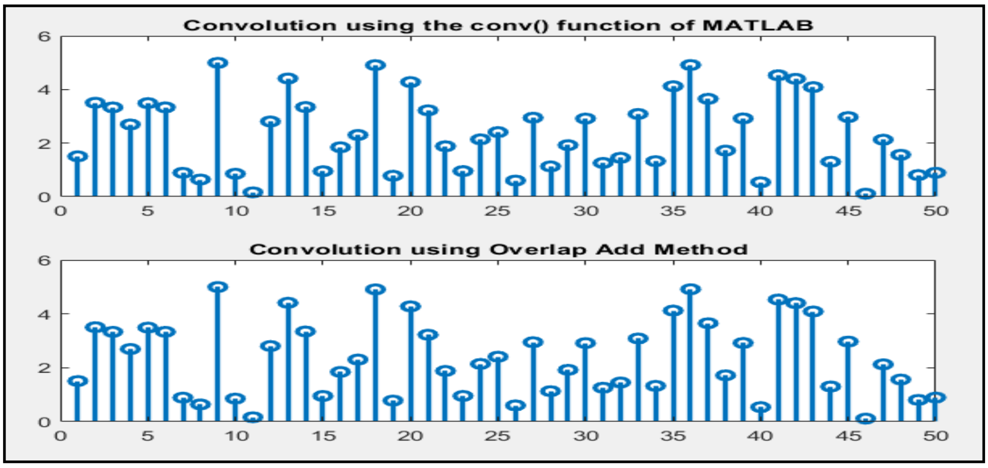


5. Write a program to realize linear convolution between a small and a long discrete-time sequence using overlap-add method. (Do not use 'fft filter' function of MATLAB)

Ans. :

```
clc
clear all
close all
x = rand(1,50);
h = 5;
l = 100;
n1 = length(x);
m = length(h);
c = conv(x,h);
x = [x zeros(1,mod(-n1,l))];
N2 = length(x);
h = [h zeros(1,l-1)];
H = fft(h,l+m-1);
S = N2/l;
index = 1:l;
X = [zeros(m-1)];
for s = 1:S
    xm = [x(index) zeros(1,m-1)];
    X1 = fft(xm, l+m-1);
    Y = X1.*H;
    Y=ifft(Y);
    Z = X((length(X)-m+2):length(X)) + Y(1:m-1);
    X = [X(1:(s-1)*l) Z Y(m:m+l-1)];
    index = s*l+1:(s+1)*l;
end
i= 1:n1+m-1;
X = X(i);
subplot(2, 1, 1)
stem(c,"Linewidth",2);
title('Convolution using the conv() function of MATLAB');
subplot(2,1,2)
stem(X,"Linewidth",2);
title('Convolution using Overlap Add Method');
```

Figure :



Assignment – 3
FIR Filter Design

1. Design a low-pass FIR filter of length 21 and 41 respectively with a cutoff frequency of 2 KHz using the following window functions. Assume the sampling frequency is 8 KHz.
   Window function:
   a. Rectangular window function
   b. Hamming window function
   c. Hanning window function
   d. Blackman window function

Ans. :

```
clc
clear all
close all
fs = 8000;      % Sampling frequency
fc = 2000;      % Cutoff frequency

% Design the filter using different window functions

% Rectangular window function
figure(1);
N = 21; % Filter length
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
w = rectwin(N);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Rectangular Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
N = 41; % Filter length
w = rectwin(N);
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Rectangular Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
```

```
% Hamming window function
figure(2);
N = 21; % Filter length
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
w = hamming(N);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Hamming Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
N = 41; % Filter length
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
w = hamming(N);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Hamming Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')

% Hanning window function:
figure(3);
N = 21; % Filter length
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
w = hanning(N);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Hanning Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
N = 41; % Filter length
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
w = hanning(N);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Hanning Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
```

```
% Blackman window function:
figure(4);
N = 21; % Filter length
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
w = blackman(N);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Blackman Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
N = 41; % Filter length
n = -floor(N/2):floor(N/2);        % Calculate the ideal low-pass filter
coefficients
hd = 2*fc/fs*sinc(2*fc*n/fs);
w = blackman(N);
h = hd .* w';
freqz(h,1,1024,fs)
title('Frequency Response - Blackman Window')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
```

Figure : 1

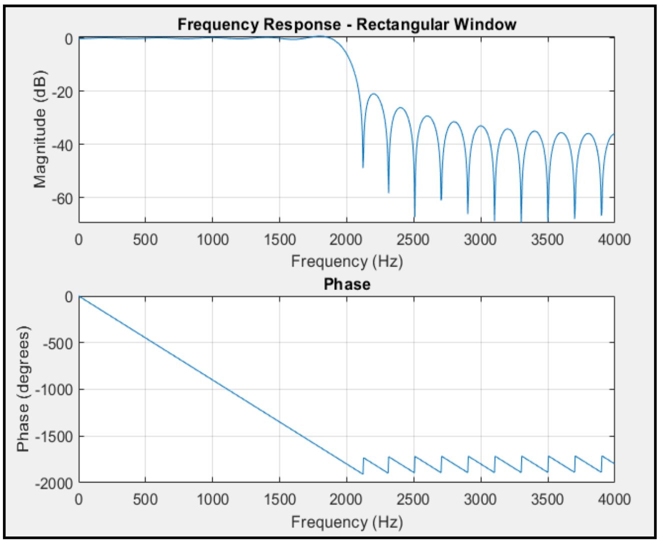


Figure :2

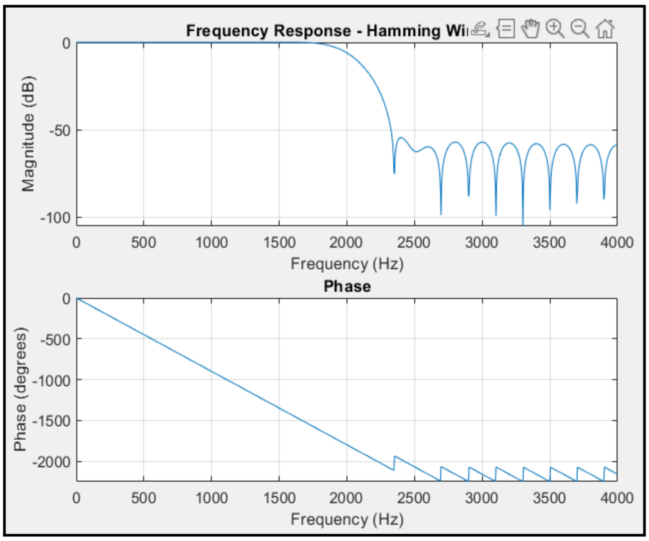


Figure : 3

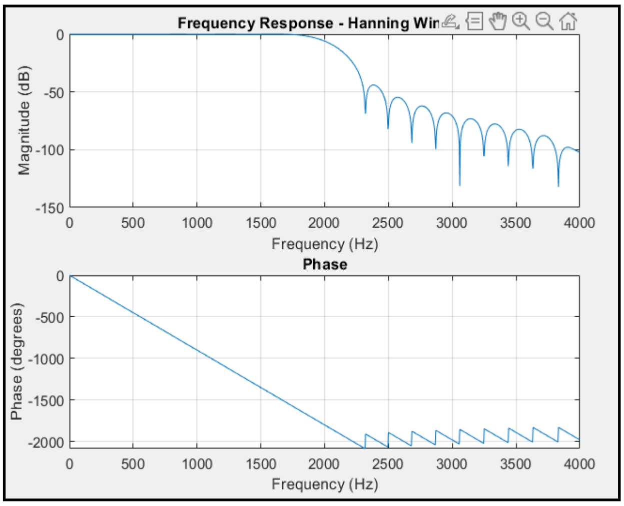
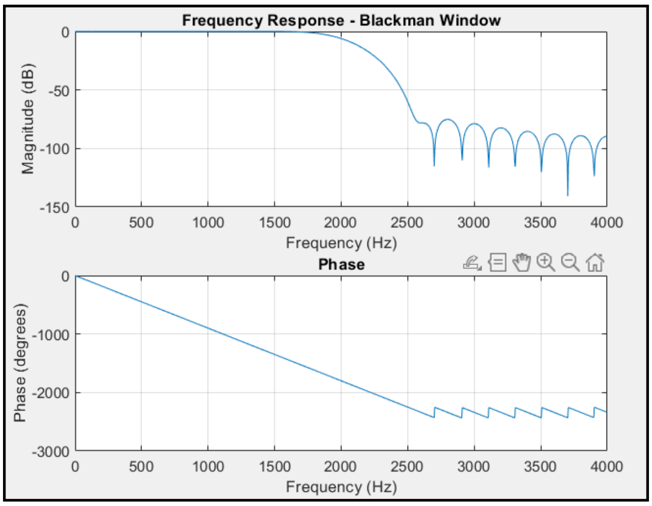


Figure : 4



2. Design 21-length and 41-length band-pass FIR filter with lower and upper cutoff frequency at 2.5 KHz and 3 KHz respectively using the following window functions. Assume the sampling frequency is 8 KHz.
   Window function:
   a. Rectangular window function
   b. Hamming window function
   c. Hanning window function
   d. Blackman window function

Ans. :

```
% Filter specifications
fs = 8000; % Sampling frequency
f1 = 2500; % Lower cutoff frequency
f2 = 3000; % Upper cutoff frequency
a1 = 0; % Desired amplitude at f1
a2 = 1; % Desired amplitude at f2
dev = 0.01; % Maximum allowable deviation

% Rectangular window function
figure(1);
% Filter design
N = 21; % Filter length
b = fir1(N-1, [2*f1/fs, 2*f2/fs], 'bandpass', rectwin(N));
freqz(b, 1, 1024, fs);
% Verification
x = sin(2*pi*f1*(0:1/fs:0.05)) + sin(2*pi*f2*(0:1/fs:0.05));
y = filter(b, 1, x);
```

```
% Hamming window function
figure(2);
% Filter design
N = 21; % Filter length
b = fir1(N-1, [2*f1/fs, 2*f2/fs], 'bandpass', hamming(N));
freqz(b, 1, 1024, fs);
% Verification
x = sin(2*pi*f1*(0:1/fs:0.05)) + sin(2*pi*f2*(0:1/fs:0.05));
y = filter(b, 1, x);

% Hanning window function:
figure(3);
% Filter design
N = 21; % Filter length
b = fir1(N-1, [2*f1/fs, 2*f2/fs], 'bandpass', hann(N));
freqz(b, 1, 1024, fs);
% Verification
x = sin(2*pi*f1*(0:1/fs:0.05)) + sin(2*pi*f2*(0:1/fs:0.05));
y = filter(b, 1, x);

% Blackman window function:
figure(4);
% Filter design
N = 21; % Filter length
b = fir1(N-1, [2*f1/fs, 2*f2/fs], 'bandpass', blackman(N));
freqz(b, 1, 1024, fs);
% Verification
x = sin(2*pi*f1*(0:1/fs:0.05)) + sin(2*pi*f2*(0:1/fs:0.05));
y = filter(b, 1, x);
```

Figure : 1

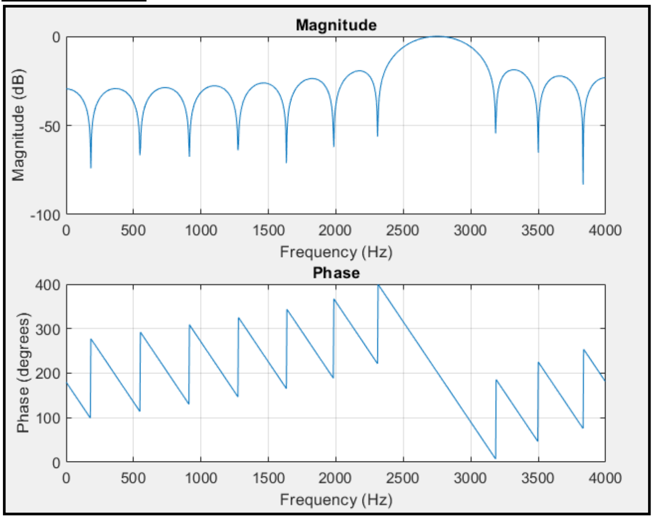


Figure : 2

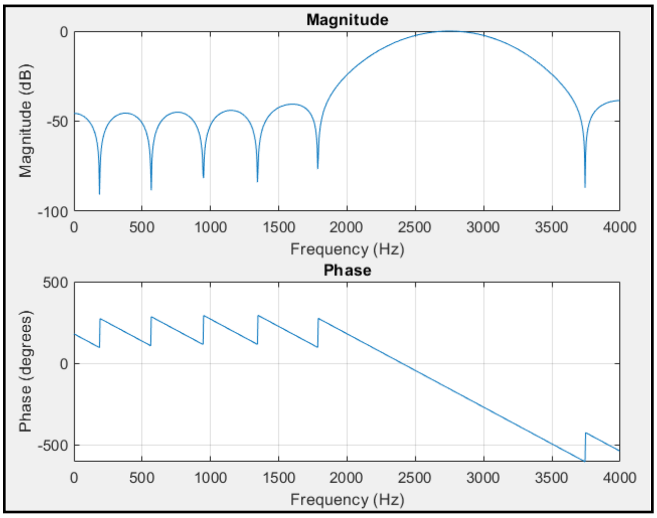


Figure : 3

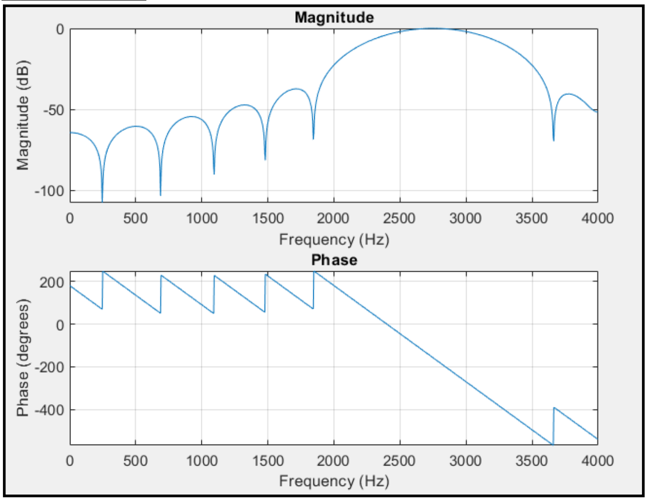
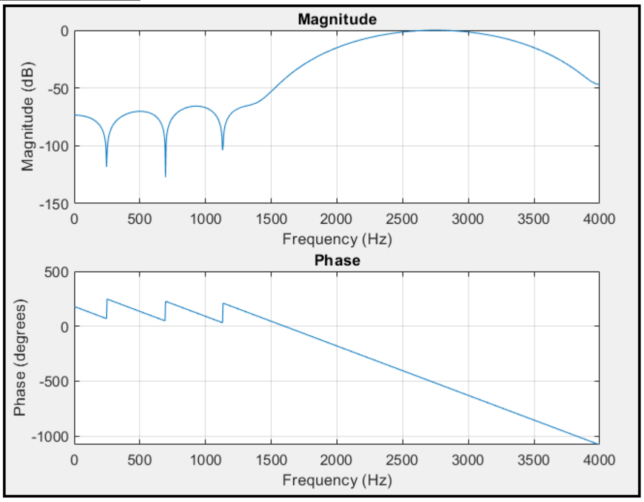


Figure : 4



3. Use the frequency sampling method to design a linear phase low-pass FIR filter of length 21 and 41 respectively. Let the cutoff frequency be 2 KHz and assume a sampling frequency of 8KHz. List FIR filter coefficients and plot the frequency responses.

Ans. :

```
clc
close all

% Filter specifications
fs = 8000; % Sampling frequency
f_cutoff = 2000; % Cutoff frequency
N = 21; % Filter length
n = 0:N-1; % Time samples
k = 0:N-1; % Frequency samples
w = 2*pi*k/N; % Normalized frequencies
H = (w <= pi*f_cutoff/fs); % Desired frequency response
figure(1);
% Filter design
h = real(ifft(H));
h = circshift(h, (N-1)/2); % Shift to center of impulse response
% Verification
freqz(h, 1, 1024, fs);
% Print coefficients
disp('Filter coefficients:');
disp(h);
% Filter specifications
fs = 8000; % Sampling frequency
f_cutoff = 2000; % Cutoff frequency
N = 41; % Filter length
```

```
n = 0:N-1; % Time samples
k = 0:N-1; % Frequency samples
w = 2*pi*k/N; % Normalized frequencies
H = (w <= pi*f_cutoff/fs); % Desired frequency response
figure(2);
% Filter design
h = real(ifft(H));
h = circshift(h, (N-1)/2); % Shift to center of impulse response
% Verification
freqz(h, 1, 1024, fs);
% Print coefficients
disp('Filter coefficients:');
disp(h);
```

Figure : 1 (For length 21)

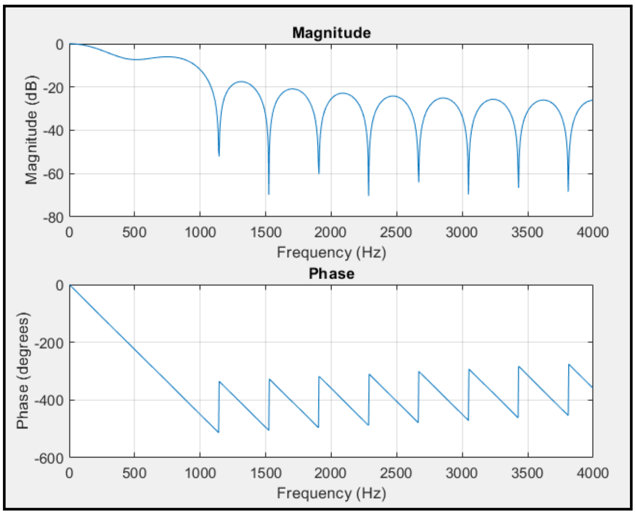
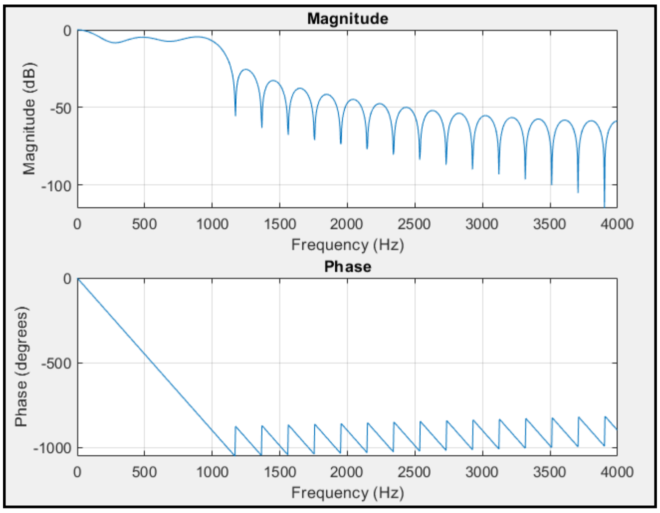


Figure : 2 (For length 41)




```
b=1x21
0.0460    0.0344    0.0163   -0.0000   -0.0059    0.0041    0.0301    0.0667
0.1044    0.1325     0.1429    0.1325    0.1044    0.0667    0.0301
0.0041   -0.0059    0.0000    0.0163    0.0344         0.0460

b=1x41
0.0011    0.0085    0.0185    0.0246    0.0225    0.0132    0.0026   -0.0023
0.0024    0.0147    0.0270    0.0307    0.0216    0.0033   -0.0137   -0.0165
0.0030    0.0430    0.0916    0.1311    0.1463    0.1311    0.0916    0.0430
0.0030   -0.0165   -0.0137    0.0033    0.0216    0.0307    0.0270    0.0147
0.0024   -0.0023    0.0026    0.0132    0.0225    0.0246    0.0185    0.0085
0.0011
```

4. Use the frequency sampling method to design a linear phase band-pass FIR filter of length 21 and 41 respectively. Let the upper and lower cutoff frequency be 2.5 KHz and 3 KHz respectively and assume a sampling frequency of 8KHz. List FIR filter coefficients and plot the frequency responses.

Ans. :

```
clc
clear all
close all
i=1;
for n=[21,41]
    flo=2500;
    fhi=3000;
    fs=8000;
    nr=fs/2;
    f=[0 flo/nr-0.1 flo/nr fhi/nr fhi/nr+0.1 1];
    m=[0 0 1 1 0 0];
    b=fir2(n-1,f,m);
    [h,w]=freqz(b);
    figure(i)
    subplot(2,1,1);
    plot(w/pi,20*log10(abs(h)),"Linewidth",2);
    title(["Magnitude response for filter of length ",n]);
    subplot(2,1,2);
    plot(w/pi, angle(h),"Linewidth",2);
    title(["Phase response for filter of length ",n]);
    i=i+1;
end
```

Figure : 1

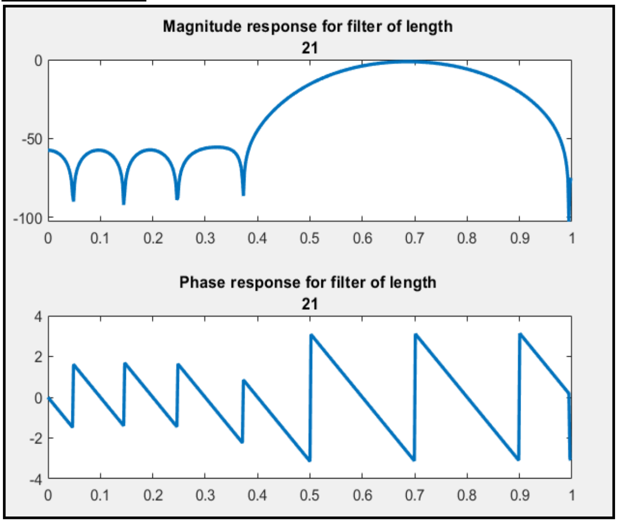
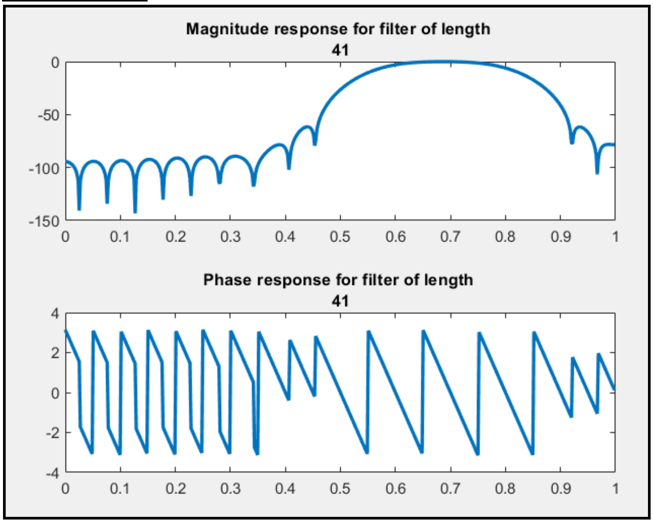


Figure : 2



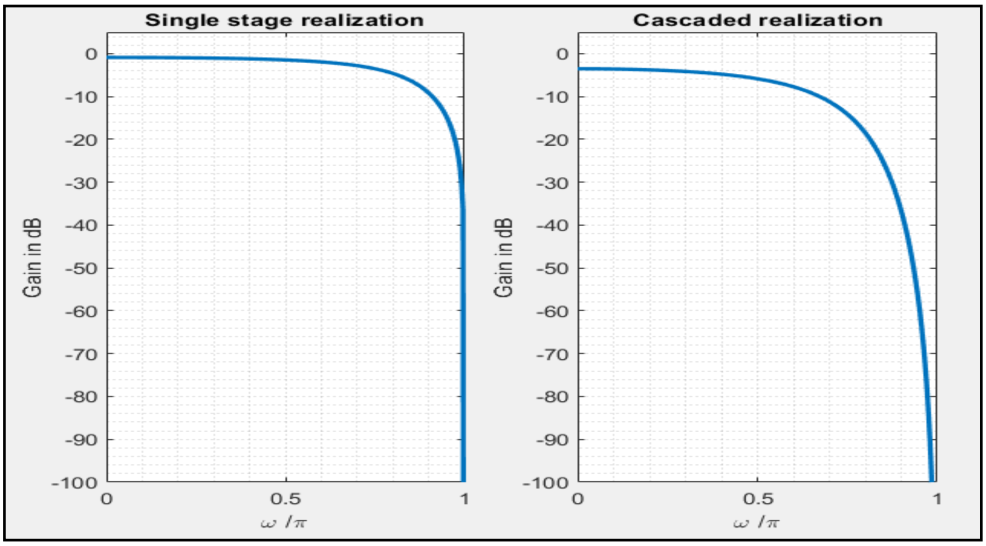
Assignment – 4
IIR Filter Design

1. Design an IIR low-pass filter with 3-dB cut-off frequency at $w_c = 0.45\pi$ using a single stage realization and a cascade of four first-order low pass filters and compare their gain responses.

Ans. :

```
clc
clear
close all
w = 0:pi/255:pi;
% For Single Stage Realization
num1=[1 1]/1.5267; % the /1.5267 is to make static gain = 1 (0 dB)
den1=[1 0.45]; % Since 0.45pi is given as cutoff frequency
num=num1;
den=den1;
h1 = freqz(num, den, w);
g1 = 20*log10(abs(h1));
subplot(1,2,1)
plot(w/pi,g1,"Linewidth",2);
grid minor; axis ([0 1 -100 5]);
xlabel("\omega /\pi");ylabel("Gain in dB");
title("Single stage realization");
for i=1:3 % For cascaded realization
    num = conv (num,num1);
    den = conv (den,den1);
end
h1 = freqz(num, den, w);
g1 = 20*log10(abs(h1));
subplot(1,2,2)
plot(w/pi,g1,"Linewidth",2); % PLOT CODE For Cascaded version
grid minor; axis ([0 1 -100 5]);
xlabel("\omega /\pi");ylabel("Gain in dB");
title("Cascaded realization");
```

Figure :

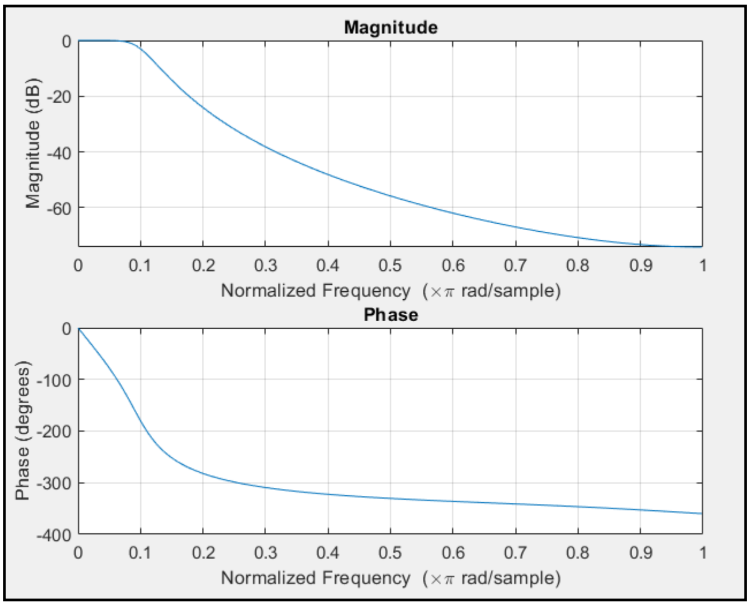


2. Write a MATLAB program to design a digital Butterworth low-pass filter using impulse invariance method. Determine the order of the analog prototype filter for this purpose. The input data required for your program are sampling frequency (F_S), pass-band edge frequency (F_p), stop-band edge frequency (F_s), maximum pass-band ripple (δ_p) and minimum stop- band attenuation (δ_s). Plot the gain response of the designed filter for the flowing inputs:
   F_S = 80KHz , F_p = 4 KHz , F_s = 20 KHz , δ_p = 0.5 dB , δ_s = 45 dB .
   You may use the M-file impinvar of MATLAB.

Ans. :

```
clc
clear
close all
% Input parameters
FS = 80000; % sampling frequency in Hz
Fp = 4000; % passband edge frequency in Hz
Fs = 20000; % stopband edge frequency in Hz
deltap = 0.5; % maximum passband ripple in dB
deltas = 45; % minimum stopband attenuation in dB
% Determine the order of the analog prototype filter
N = ceil(log10((10^(0.1*deltas)-1)/(10^(0.1*deltap)-1))/(2*log10(Fs/Fp)));
% Design the analog prototype filter
[b,a] = butter(N,2*pi*Fp,'s');
% Convert the analog prototype filter to a digital filter using the impulse
invariance method
[bd,ad] = impinvar(b,a,FS);
% Plot the gain response of the designed filter
freqz(bd,ad);
```

Figure :

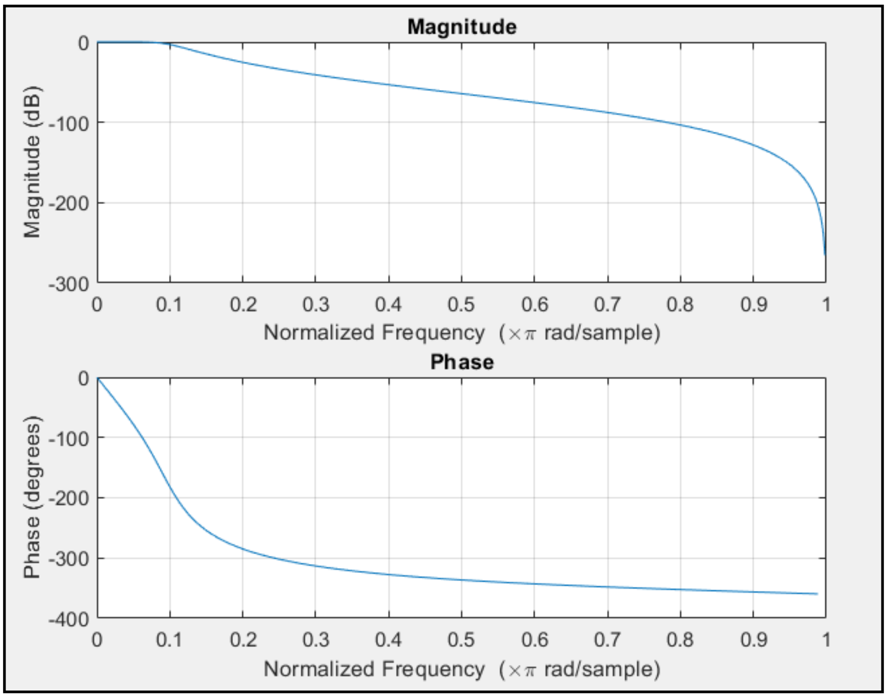


3. Repeat the above problem using bilinear transformation method. You may specifically use the M-file of MATLAB.

Ans. :

```
clc
clear
close all
% Input parameters
FS = 80000; % sampling frequency in Hz
Fp = 4000; % passband edge frequency in Hz
Fs = 20000; % stopband edge frequency in Hz
deltap = 0.5; % maximum passband ripple in dB
deltas = 45; % minimum stopband attenuation in dB
% Determine the order of the analog prototype filter
N = ceil(log10((10^(0.1*deltas)-1)/(10^(0.1*deltap)-1))/(2*log10(Fs/Fp)));
% Design the analog prototype filter
[b,a] = butter(N,2*pi*Fp,'s');
% Convert the analog prototype filter to a digital filter using the bilinear transformation method
[bd,ad] = bilinear(b,a,FS);
% Plot the gain response of the designed filter
freqz(bd,ad);
```

Figure :



Teacher's Signature