VINAYAKA MISSION'S RESEARCH FOUNDATION

(Deemed to be University under section 3 of the UGC Act 1956)

VINAYAKA MISSION'S KIRUPANANDA VARIYAR ENGINEERING COLLEGE

VINAYAKA MISSION'S KIRUPANANDA VARIYAR ENGINEERING COLLEGE, SALEM

(A Constituent College of Vinayaka Mission's Research Foundation, Deemed to be University, Salem)

(An ISO 9001:2000 Certified, NAAC Accredited and AICTE Approved)

NH-47, Sankari Main Road, Periya Seeragapadi, Salem – 636 308

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

____Laboratory Record

University Reg. No.______________________

Name:___________________________ Batch:_________________

Course Name :___________________________ Course Code :__________________

CERTIFIED THAT THIS BONAFIDE RECORD OF WORK DONE BY

Mr / Ms ______________________________________

Staff-in-charge **Head Of the Department**

Salem: 636308

Date: __________

Submitted to the
Vinayaka Missions Research Foundation (Deemed to be University)
Practical Examination held on__________20

Internal Examiner **External Examiner**

Sl No.	Date	List of Experiments	Remarks	Signature

Exp No: 1
Date:

GENARATION OF SIGNALS

Aim:

To write MATLAB programs to generate the following signals

1. Unit impulse function
2. Unit step function.
3. Unit ramp
4. Exponential function

Essentials required:

Hardware: IBM PC or compatible

Software: MATLAB v5.1 or higher

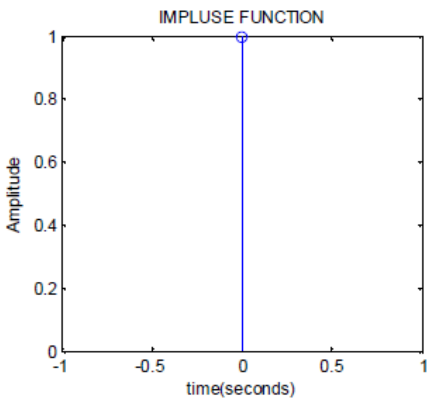
Program:

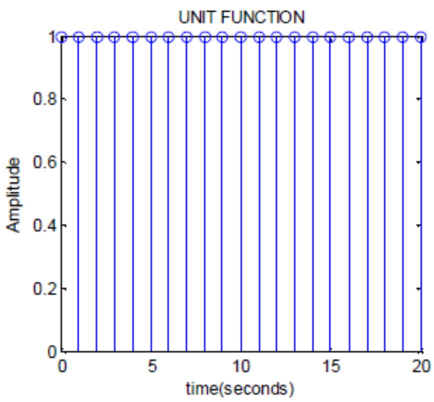
```
close all;
clear all;
n=0:.1:5;
% RAMP FUNCTION
figure(1);
subplot(3,3,7);
plot(n);
title('RAMP FUNCTION');
xlabel('time(seconds)');
ylabel('Amplitude');
% IMPLUSE FUNCTION
n:0:.2:5;
figure(1);
subplot(3,3,8);
stem(0,1);
title('IMPLUSE FUNCTION');
```

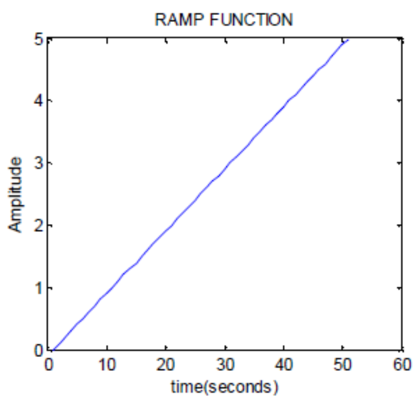
```
xlabel('time(seconds)');
ylabel('Amplitude');
% UNIT FUNCTION
N=21;
X=ones(1,N);
n=0:1:N-1;
figure(1);
subplot(3,3,9);
stem(n,X);
title('UNIT FUNCTION');
xlabel('time(seconds)');
ylabel('Amplitude');
% EXPONENTIAL FUNCTION
X2=exp(n);
figure(2);
subplot(2,2,1);
plot(n,X2);
title('EXPONENTIAL FUNCTION');
xlabel('time(seconds)');
ylabel('Amplitude');
X3=exp(-n);
figure(2);
subplot(2,2,2);
plot(n,X3);
title('EXPONENTIAL FUNCTION');
xlabel('time(seconds)');
ylabel('Amplitude');
X4=-exp(n);
figure(2);
subplot(2,2,3);
```

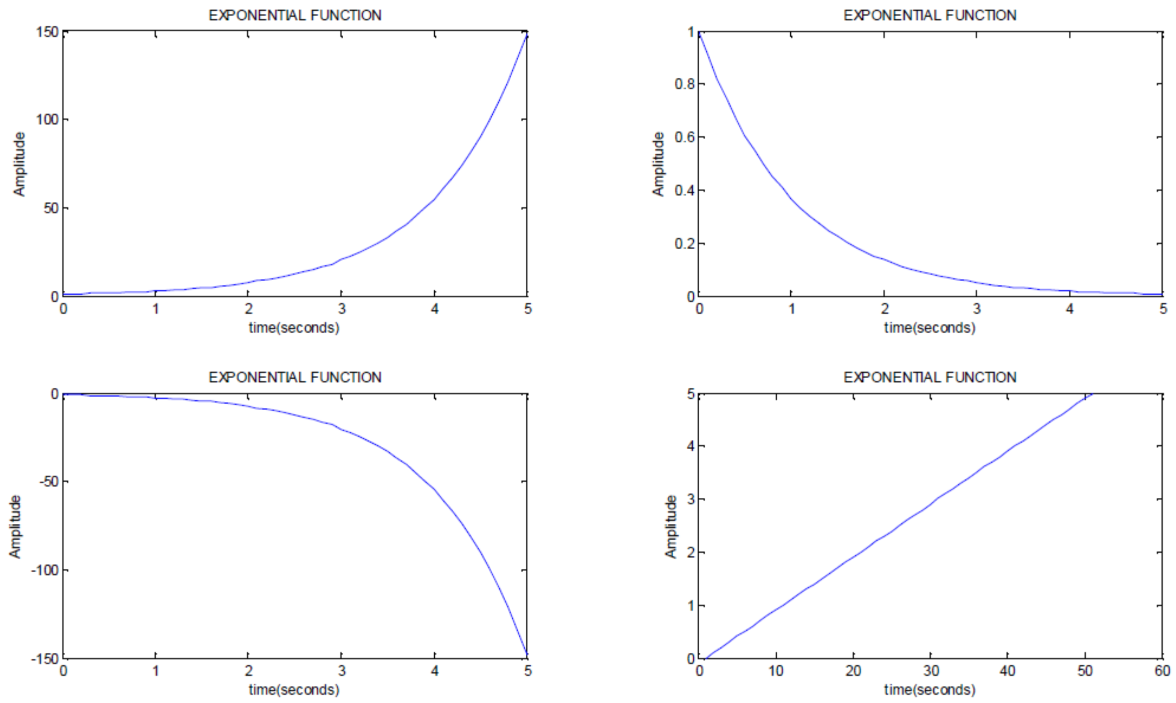
```
plot(n,X4);
title('EXPONENTIAL FUNCTION');
xlabel('time(seconds)');
ylabel('Amplitude');
X5=-exp(-n);
figure(2);
subplot(2,2,4);
plot(n);
title('EXPONENTIAL FUNCTION');
xlabel('time(seconds)');
ylabel('Amplitude');
```

Output:









Result:

The MATLAB programs to generate various sequences and waves are written and the results are plotted.

Exp No: 2
Date:

LINEAR CONVOLUTION

Aim:

To write MATLAB programs for the following:

1. Linear convolution

Essentials required:

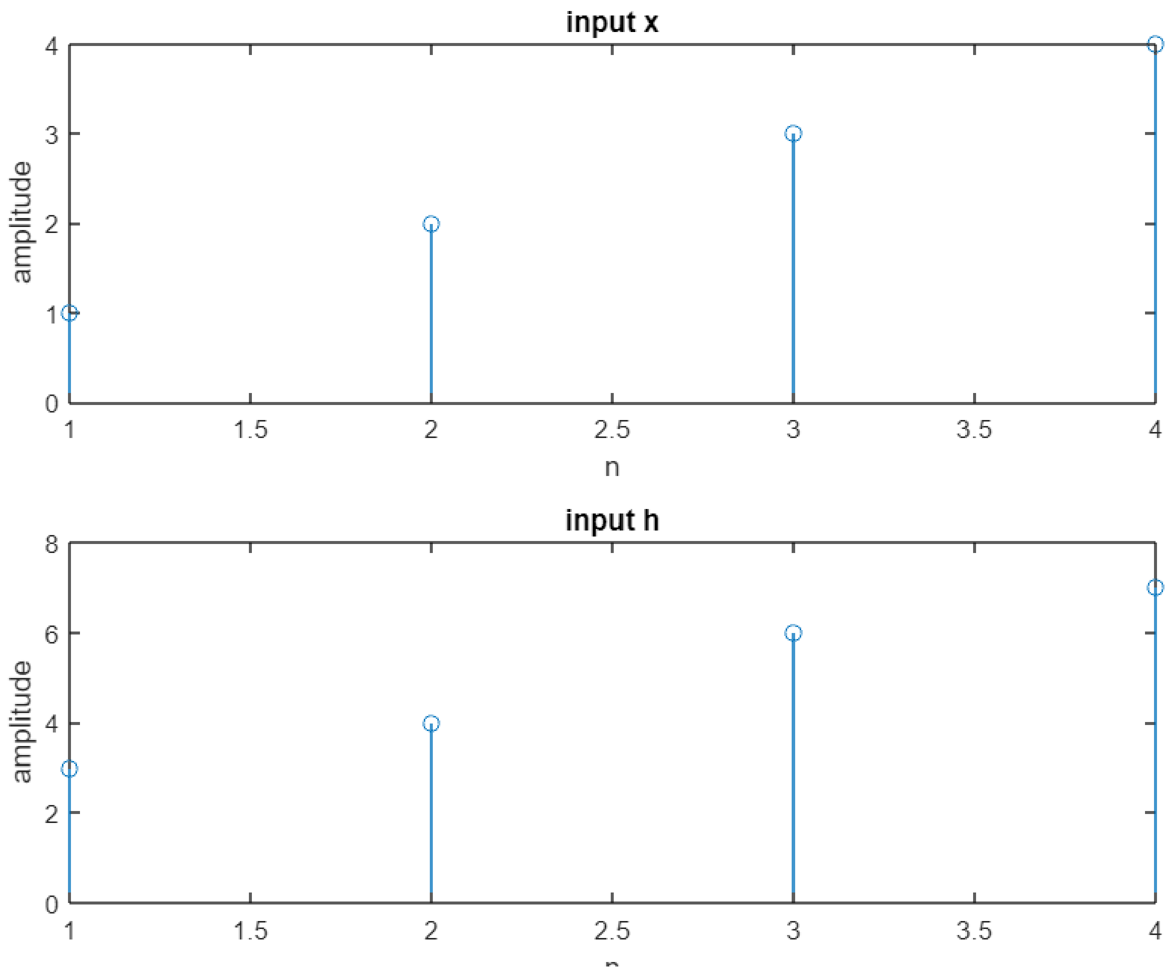
Hardware: IBM PC or compatible

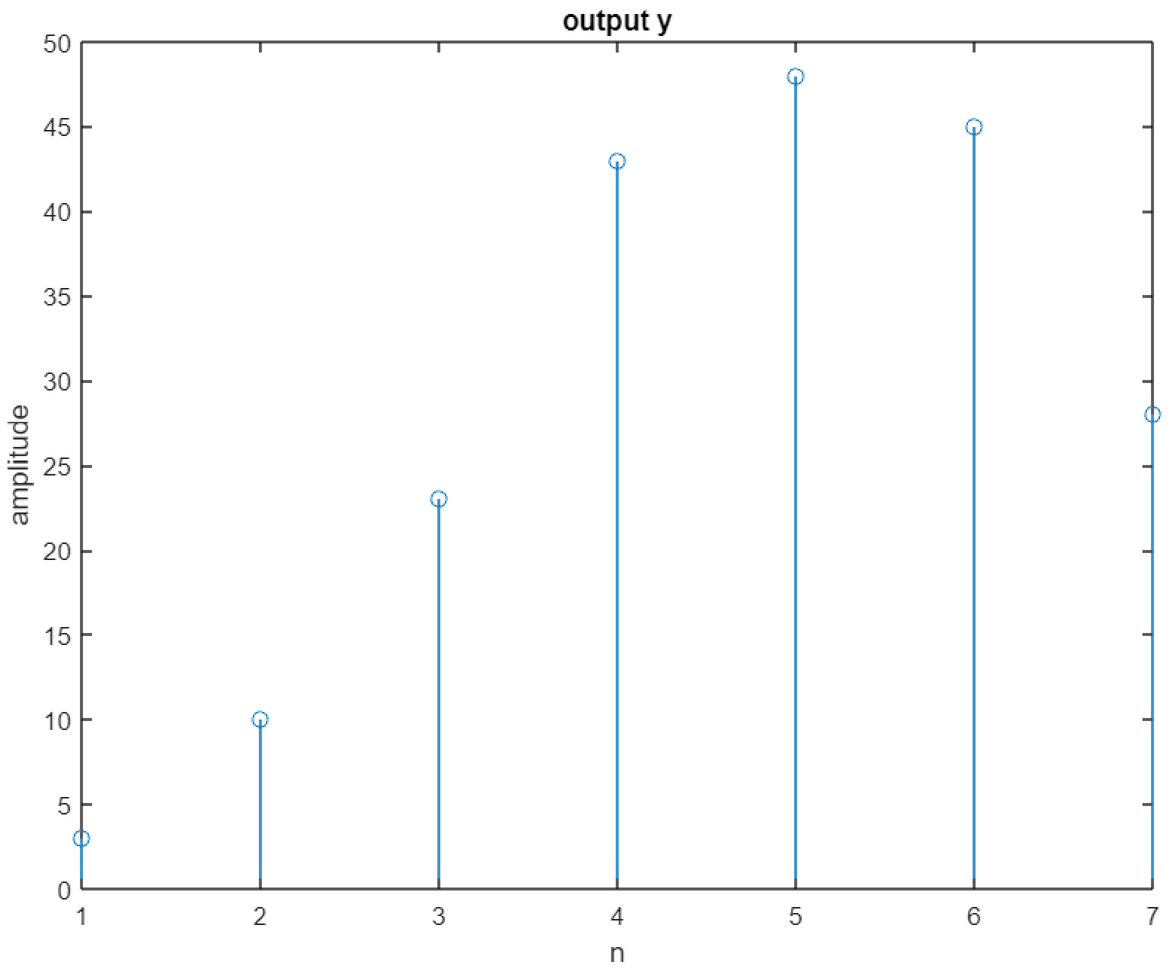
Software : MATLAB v5.1 or higher

Program:

```
clc;
close all;
clear all;
x=input('enter the 1st sequence');
h=input('enter the 2nd sequence');
y=conv(x,h);
subplot(2,1,1);
stem(x);
title('input x');
ylabel('amplitude');
xlabel('n');
subplot(2,1,2);
stem(h);
title('input h');
ylabel('amplitude');
xlabel('n');
figure(2);
stem(y);
title('output y');
```

```
ylabel('amplitude');
xlabel('n');
disp('the resultant signal');
```





Result:

The MATLAB program to implement linear convolution is written and the results are plotted.

Expt No: 3
Date:

CIRCULAR CONVOLUTION

Aim:

To write MATLAB programs for the Circular convolution.

Essentials required:

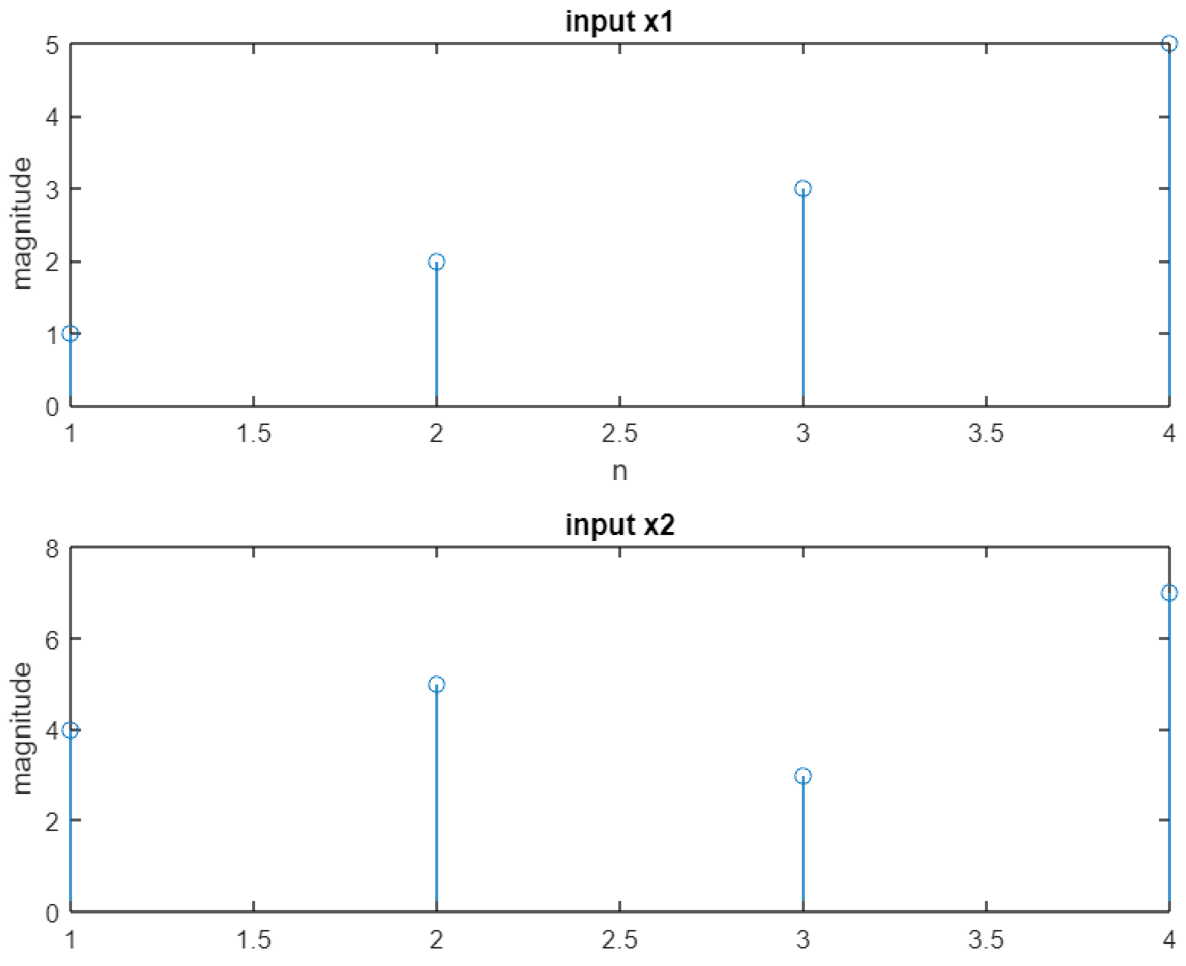
Hardware: IBM PC or compatible

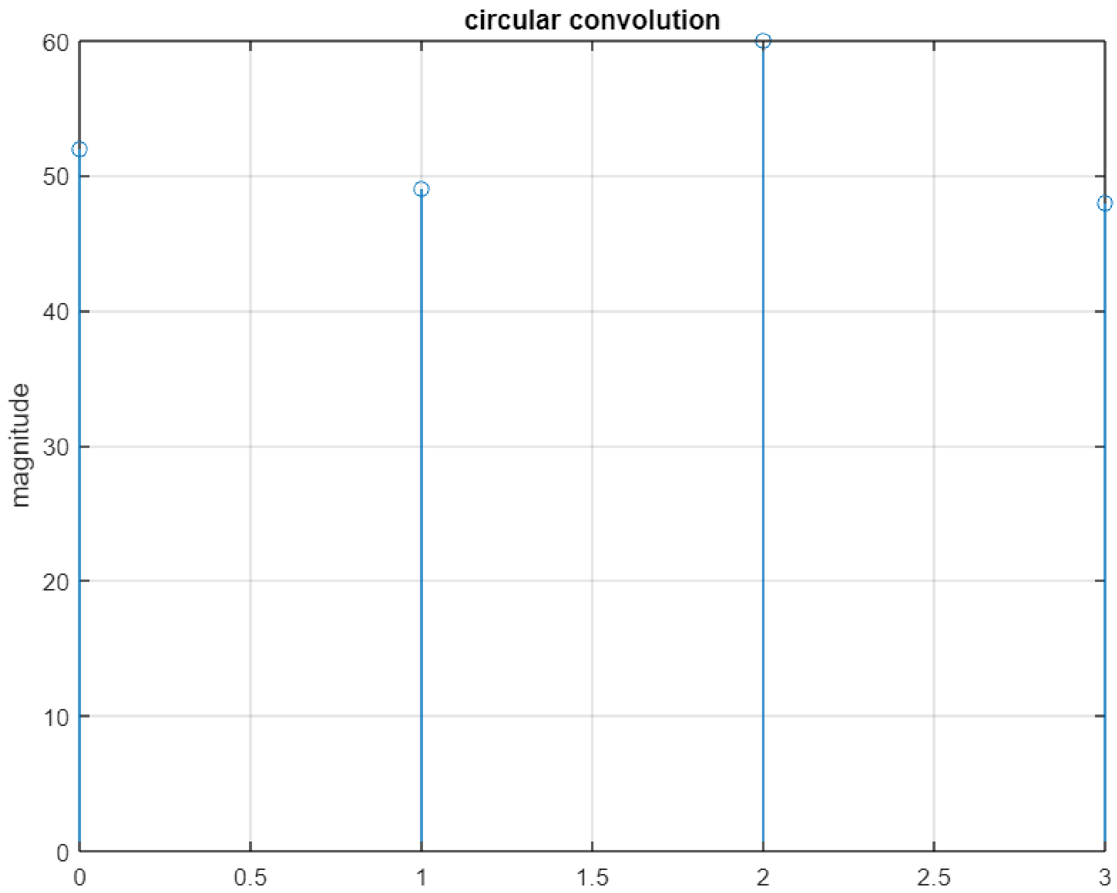
Software: MATLAB v5.1 or higher

Program:

```
clc;
x1=input('enter the first sequence');
x2=input('enter the second sequence');
subplot(2,1,1);
stem(x1);
title('input x1');
ylabel('magnitude');
xlabel('n');
subplot(2,1,2);
stem(x2);
title('input x2');
ylabel('magnitude');
xlabel('n');
n=max(length(x1),length(x2));
x1=fft(x1,n);
x2=fft(x2,n);
y=x1.*x2;
yc=ifft(y,n);
disp('circular convolution:');
```

```
disp(yc);
n=0:1:n-1;
figure(2);
stem(n,yc);grid;
xlabel('n');
ylabel('magnitude');
title('circular convolution');
```





Result:

The MATLAB programs to implement circular convolution is written and the results are plotted.

Expt No:4
Date:

ANALOG CHEBYSHEV FILTERS AND APPLY BILINEAR TRANSFORMATION

Aim:

To design analog Chebyshev filters and apply bilinear transformation

Essentials required:

Hardware: IBM PC or compatible

Software: MATLAB v5.1 or higher

Program:

```
clc; % clear screen
clear all; % clear work space
close all; % close all figure windows
fp = input('Enter the Pass band frequency in Hz = '); % input specifications
fs = input('Enter the Stop band frequency in Hz = ');
Fs = input('Enter the Sampling frequency in Hz = ');
Ap = input(' Enter the Pass band ripple in db:');
As = input('Enter the Stop band ripple in db:');
wp=2*pi*fp/Fs; % Analog frequency
ws=2*pi*fs/Fs;
Up = 2*tan(wp/2); % Prewrapped frequency
Us = 2*tan(ws/2);
[N,wn]= cheb1ord (Up,Us,Ap,As,'s'); %Calculate order and cutoff freq
disp('order of the filter N = ');
disp(N);
disp('Normalized cut off frequency = ');
disp(wn);
[num, den] = cheby1(N, Ap,wn,'s');
[b,a] = bilinear(num,den,1);
freqz(b,a,512,Fs);
printsys(b,a,'z');
```

OUTPUT:

enter the Pass band edge frequency in Hz = 100

enter the stop band frequency in Hz = 500

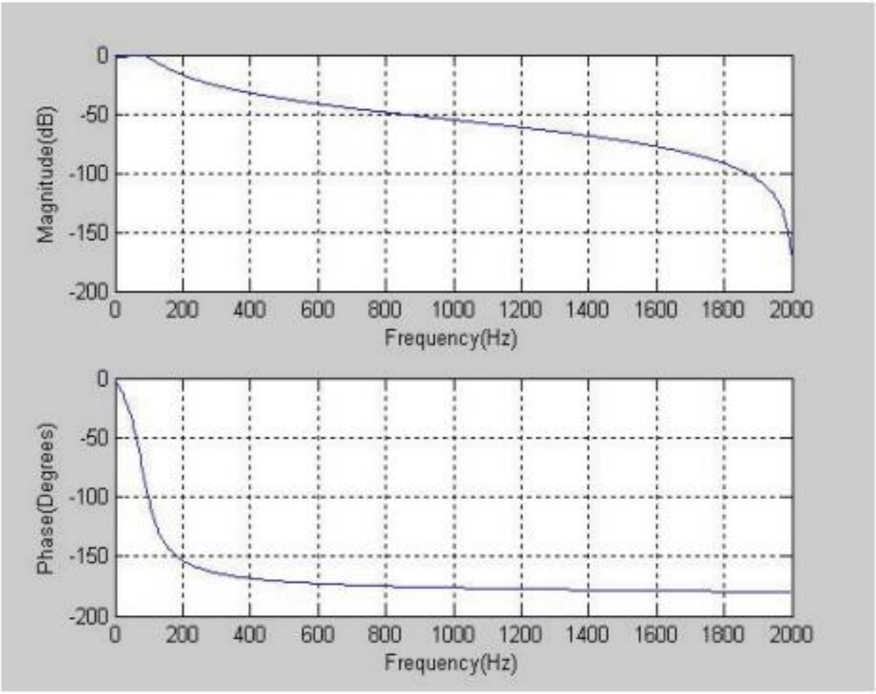
enter the sampling frequency in Hz = 4000

enter the pass band ripple n db = 2

enter the stop band attenuation in db =

20 order of the filter N = 2

Normalised cutoff frequency = 0.1574



Result:

The MATLAB programs to implement analog Chebyshev filters and apply bilinear transformation is written and the results are plotted.

Exp No: 5
Date:

BUTTERWORTH FILTERS AND APPLY BILINEAR TRANSFORMATION

Aim:

To design analog Butterworth filters and apply bilinear transformation

Essentials required:

Hardware: IBM PC or compatible

Software: MATLAB v5.1 or higher

Program:

```
clc; % clear screen

clear all; % clear screen

close all; % close all figure windows

fp = input('Enter the Pass band frequency in Hz = ' ); % input specifications

fs = input('Enter the Stop band frequency in Hz = ');

Fs = input('Enter the Sampling frequency in Hz =' );

Ap = input(' Enter the Pass band ripple in db:');

As = input('Enter theStop band ripple in db:');

wp=2*pi*fp/Fs; % Analog frequency

ws=2*pi*fs/Fs;

Up = 2*tan(wp/2);% Prewrapped frequency

Us = 2*tan(ws/2);

[n,wn]= buttord (Up,Us,Ap,As,'s'); %Calculate order and cutoff freq

disp('Order of the filter N =');

disp(n);

disp('Normalized cut off frequency = ');

disp(wn);

[num, den] = butter(n,wn,'s'); % analog filter transfer

[b,a] = bilinear(num, den,1); % conversion of analog filter to digital filter

freqz(b,a,512,Fs); % frequency response of the filter

printsys(b,a,'z'); % print the H(z) equation obtained on screen
```

OUTPUT:

enter the Pass band edge frequency in Hz = 500

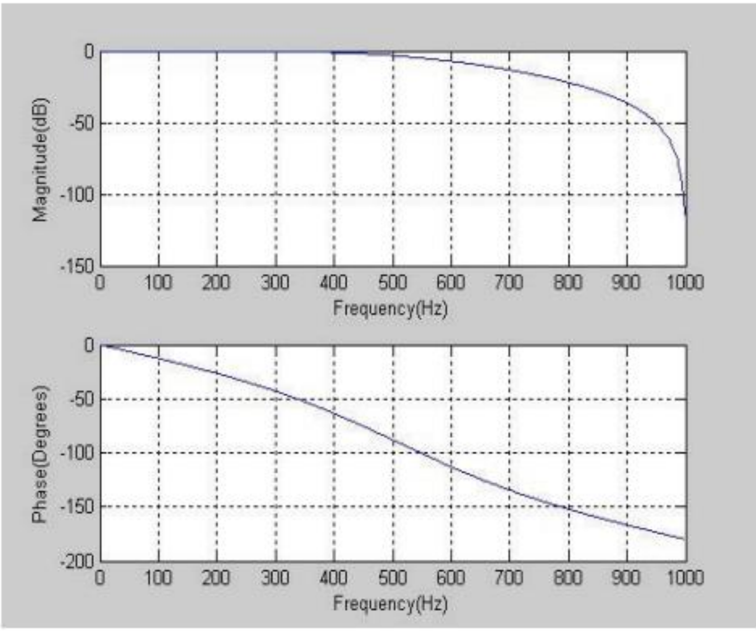
enter the stop band frequency in Hz = 750

enter the sampling frequency in Hz = 2000

enter the pass band ripple n db = 3.01

enter the stop band attenuation in db =
15 order of the filter N = 2

Normalised cutoff frequency = 2.052



Result:

The MATLAB program to implement analog Butterworth filters and apply bilinear transformation is written and the results are plotted.

Exp No: 6
Date:

ANALOG CHEBYSHEV FILTERS AND APPLY IMPULSE INVARIANCE TRANSFORMATION

Aim:

To design analog Chebyshev filters and apply impulse invariance transformation.

Consider Problem: Design a Chebyshev digital IIR using impulse invariant transformation by taking T=1 sec to satisfy the following specifications:

$$0.9 \le |H(e^{j\omega})| \le 1.0; \; for \; 0 \le \omega \le 0.28\pi$$
$$|H(e^{j\omega})| \le 0.24; \; for \; 0.5\pi \le \omega \le \pi$$

Essentials required:

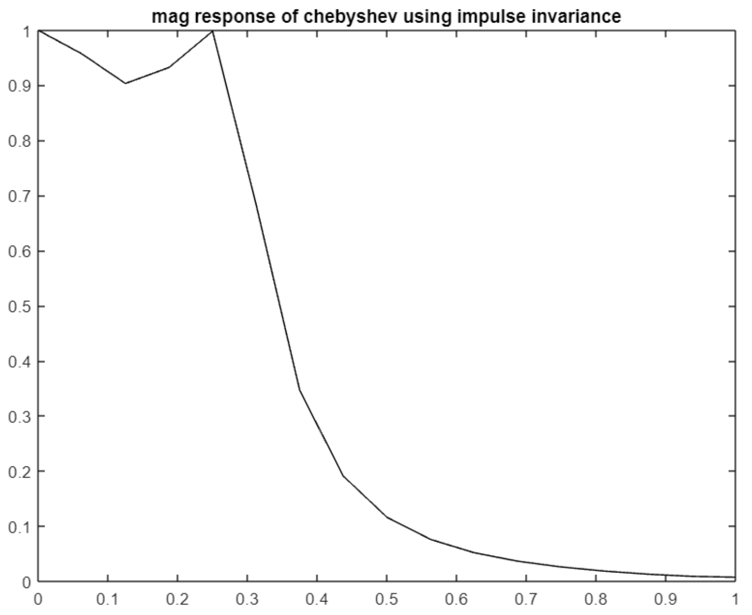
Hardware: IBM PC or compatible

Software: MATLAB v5.1 or higher

Program:

```
close all;
clear all;
clc
ap=0.9;
as=0.24;
p_d=0.28*pi;
s_d=0.5*pi;
t=1;
pass_attenuation=-20*log10(ap);
stop_attenuation=-20*log10(as);
p_a=p_d/t;
s_a=s_d/t;
[n,cf]=cheb1ord(p_a,s_a,pass_attenuation,stop_attenuation,'s');
[bn,an]=cheby1(n,pass_attenuation,1, 's');
hsn=tf(bn,an);
[b,a]=cheby1(n,pass_attenuation,cf,'s');
```

```
hs=tf(b,a);
[num,den]=impinvar(b,a,1/t);
hz=tf(num,den,t);
w=0:pi/16:pi;
hw=freqz(num,den,w);
hw_mag=abs(hw);
plot(w/pi,hw_mag,'k');
title('mag response of chebyshev using impulse invariance');
```



Result:

The MATLAB program to design analog Chebyshev filters and apply impulse invariance transformation is written and the results are plotted.

Exp No: 7
Date:

ANALOG BUTTERWORTH FILTERS AND APPLY IMPULSE INVARIANCE TRANSFORMATION

Aim:

To design analog butterworth filters and apply impulse invariance transformation

Consider a problem: Design a Butterworth IIR Filter using impulse invariance Transformation by taking T= 1 sec and

$$0.707 \le \left|H(e^{j\omega})\right| \le 1.0; for\ 0 \le \omega \le 0.3\pi$$
$$\left|H(e^{j\omega})\right| \le 0.2; for\ 0.75\pi \le \omega \le \pi$$

Essentials required:

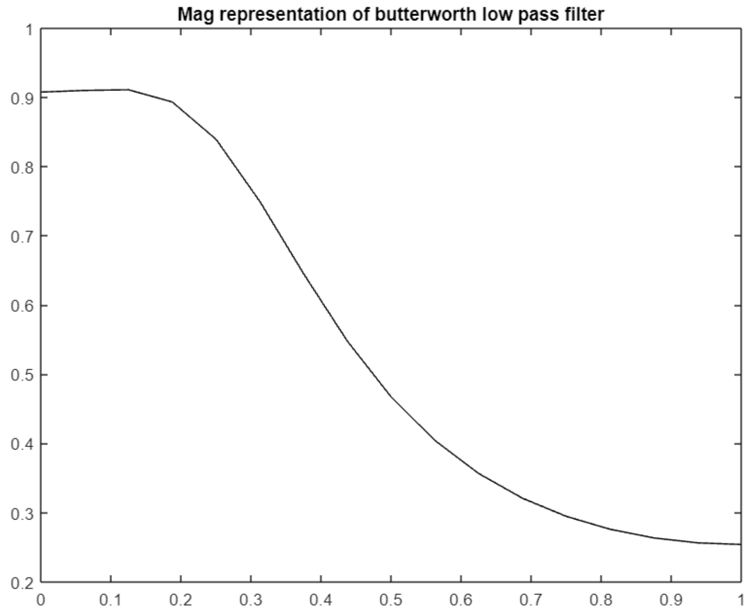
Hardware: IBM PC or compatible

Software: MATLAB v5.1 or higher

Program:

```
clear all;
clc;
Ap=0.707;
As=0.2;
omega_p=0.3*pi;
omega_s=0.75*pi;
T=1;
Pass_attenuation=-20*log10(Ap);
Stop_attenuation=-20*log10(As);
P_a=omega_p/T;
S_a=omega_s/T;
[N,CF]=buttord(P_a,S_a,Pass_attenuation,Stop_attenuation, 's');
[Bn,An]=butter(N,1,'s');
Hsn=tf(Bn,An);
[B,A]=butter(N,CF,'s');
HS=tf(B,A);
[num,den]= impinvar(B,A,1/T);
```

```
Hz=tf(num,den,T);
w=0:pi/16:pi;
Hw=freqz(num,den,w);
Hw_mag=abs(Hw);
plot(w/pi,Hw_mag,'k');
title('Mag representation of butterworth low pass filter');
```



Result:

The MATLAB program to design analog butterworth filters and apply impulse invariance transformation written and the results are plotted.

Exp No: 8
Date:

FIR FILTERS USING FOURIER SERIES METHOD AND FREQUENCY SAMPLING METHODS

Aim:

To design a FIR filter using Fourier series method and frequency sampling methods

Essentials required:

Hardware: IBM PC or compatible

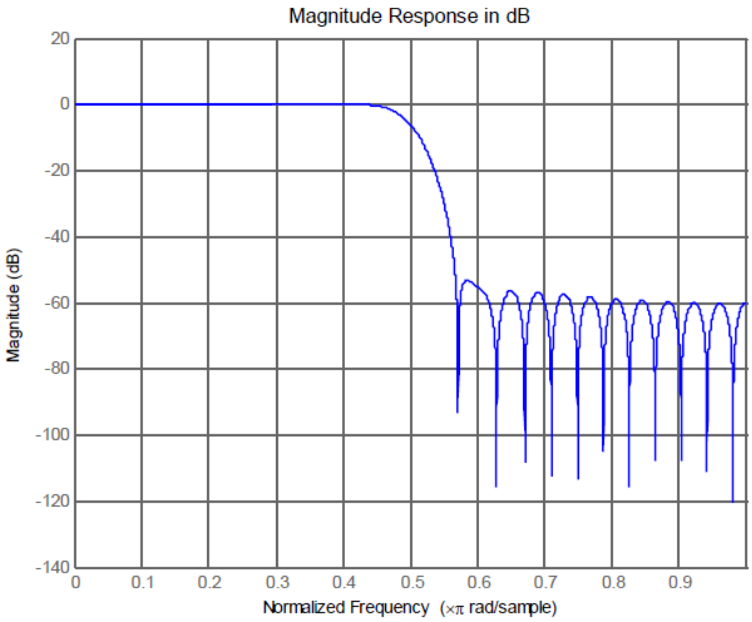
Software: MATLAB v5.1 or higher

Program:

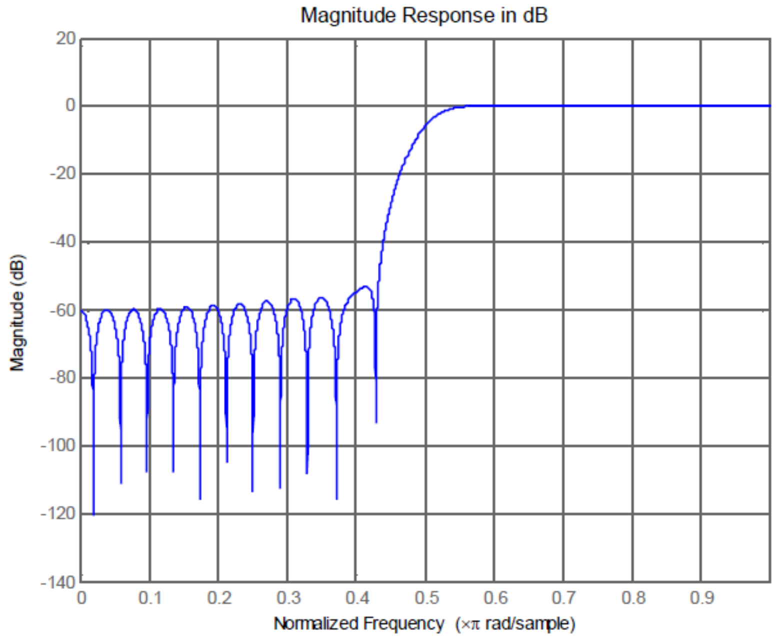
```
% fir filter
clc;
clear all;
close all;
%low pass filter
n=50;
wn=0.5;
b=fir1(n,wn);
fvtool(b,1);
%high pass filter
n=50;
wn=0.5;
b=fir1(n,wn,'high');
fvtool(b,1);
% band pass filter
n=50;
wn1=0.4;
wn2=0.8;
```

```
b=fir1(n,[wn1,wn2]);
fvtool(b,1);
% band stop filter
n=50;
wn1=0.4;
wn2=0.8;
b=fir1(n,[wn1,wn2],'stop');
fvtool(b,1);
```

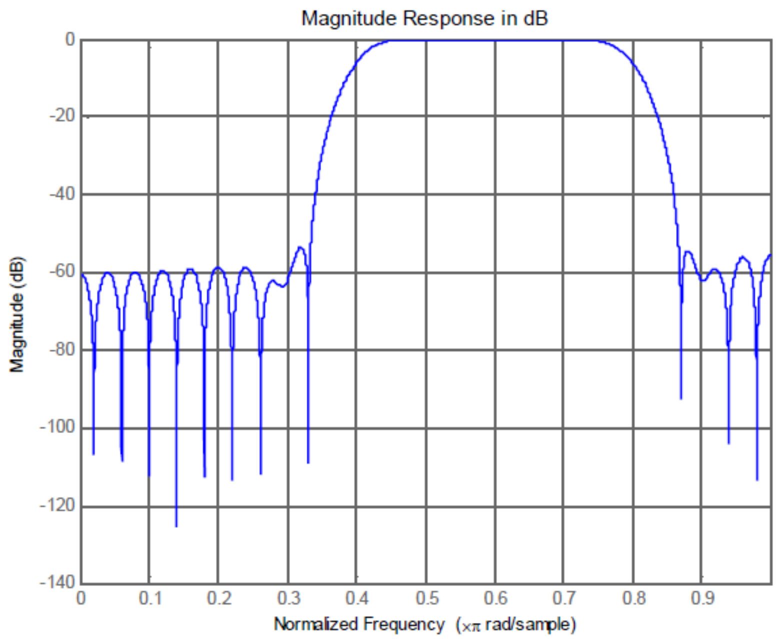
Low pass filter:



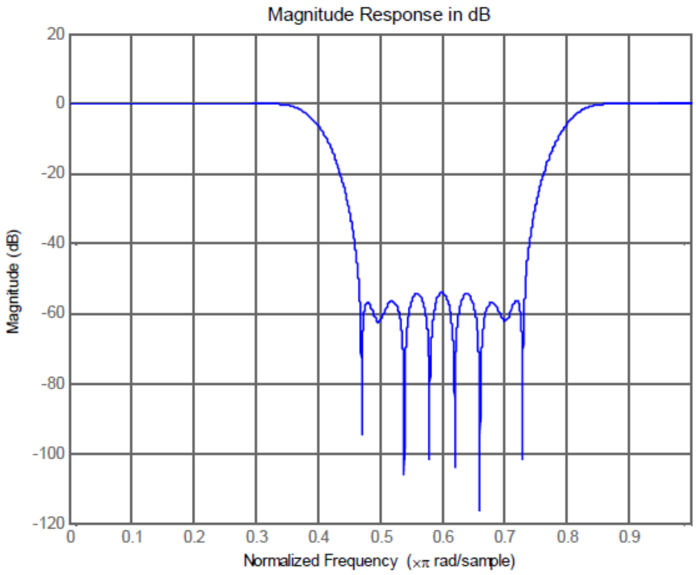
High Pass Filter:



Band Pass Filter:



Band Stop Filter:



Exp No: 9
Date:

FIR FILTERS USING DIFFERENT WINDOWING TECHNIQUES

Aim:

To design FIR filters using different windowing techniques

Essentials required:

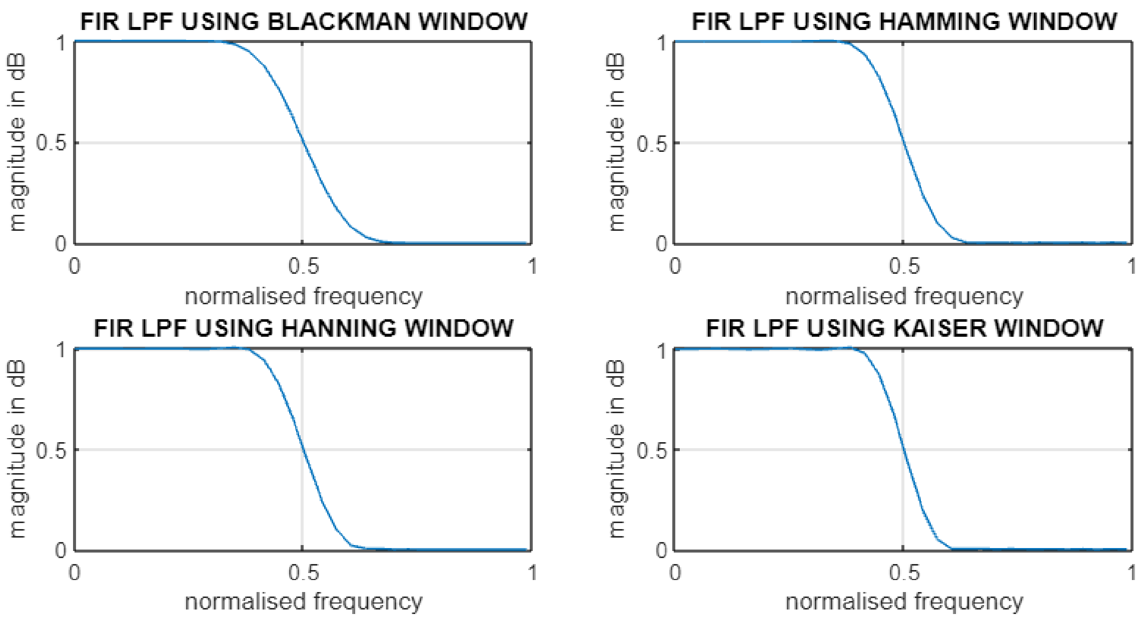
Hardware: IBM PC or compatible

Software: MATLAB v5.1 or higher

Program

```
%MATLAB program of FIR Low pass filter using Hanning %Hamming, Blackman and Kaiser window

clc;

wc=.5*pi;

N=25;

w=0:0.1:pi;

b=fir1(N,wc/pi,blackman(N+1));

h=freqz(b,1,w);

subplot(3,2,1)

plot(w/pi,abs(h))

grid;xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING BLACKMAN WINDOW')

b=fir1(N,wc/pi,hamming(N+1));

h=freqz(b,1,w);

subplot(3,2,2)

plot(w/pi,abs(h));

grid;

xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING HAMMING WINDOW')
```

```
b=fir1(N,wc/pi,hanning(N+1));
h=freqz(b,1,w);
subplot(3,2,3)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING HANNING WINDOW')
b=fir1(N,wc/pi,kaiser(N+1,3.5));
h=freqz(b,1,w);
subplot(3,2,4)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING KAISER WINDOW')
```



Result:

The MATLAB programs to design FIR filters using different windowing techniques are written and the results are plotted.

Exp No: 10
Date:

EFFECT OF QUANTIZATION

Aim:

To design Effect of Quantization

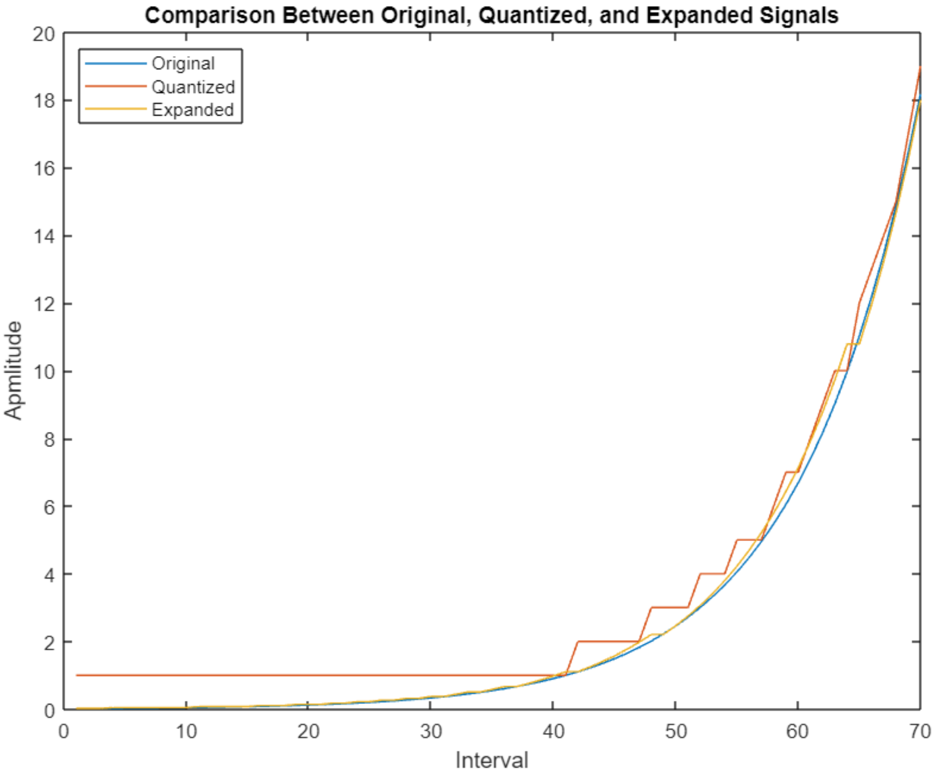
Essentials required:

Hardware: IBM PC or compatible

Software: MATLAB v5.1 or higher

Program:

```
sig = exp(-4:0.1:4);

V = max(sig);

partition = 0:2^6 - 1;

codebook = 0:2^6;

[~,qsig,distortion] = quantiz(sig,partition,codebook);

mu = 255; % mu-law parameter

csig_compressed = compand(sig,mu,V,'mu/compressor');

[~,quants] = quantiz(csig_compressed,partition,codebook);

csig_expanded = compand(quants,mu,max(quants),'mu/expander');

distortion2 = sum((csig_expanded - sig).^2)/length(sig);

[distortion, distortion2]

plot([sig' qsig' csig_expanded']);

title('Comparison Between Original, Quantized, and Expanded Signals');

xlabel('Interval');

ylabel('Apmlitude');

legend('Original','Quantized','Expanded','location','nw');

axis([0 70 0 20])
```



Result:

The MATLAB program to view the effects of Quantization is written and the results are plotted.