Unit I- DISCRETE FOURIER TRANSFORM

1.

Aim:

To generate common discrete-time signals.

Apparatus required:

Personal computer, SCILAB software.

PROGRAM:

UNIT IMPULSE SIGNAL

```
clc;
clear all;
close;
N=5; //SET LIMIT
t1=-5:5;
x1=[zeros(1,N),ones(1,1),zeros(1,N)];
subplot(2,4,1);
plot2d3(t1,x1)
xlabel('time');
ylabel('Amplitude');
title('Unit impulse signal');
```

UNIT STEP SIGNAL

```
t2=-5:5;
x2=[zeros(1,N),ones(1,N+1)];
 subplot(2,4,2);
 plot2d3(t2,x2)
 xlabel('time');
 ylabel('Amplitude');
 title('Unit step signal');
```

EXPONENTIAL SIGNAL

```
t3=0:1:20;
x3=exp(-t3);
    subplot(2,3,3);
    plot2d3(t3,x3);
    xlabel('time');
    ylabel('Amplitude');
    title('Exponential signal');
```

UNIT RAMP SIGNAL

```
    t4=0:20;
    x4=t4;
    subplot(2,3,4);
    plot2d3(t4,x4);
    xlabel('time');
    ylabel('Amplitude');
    title('Unit rampsignal');
```

SINUSOIDAL SIGNAL

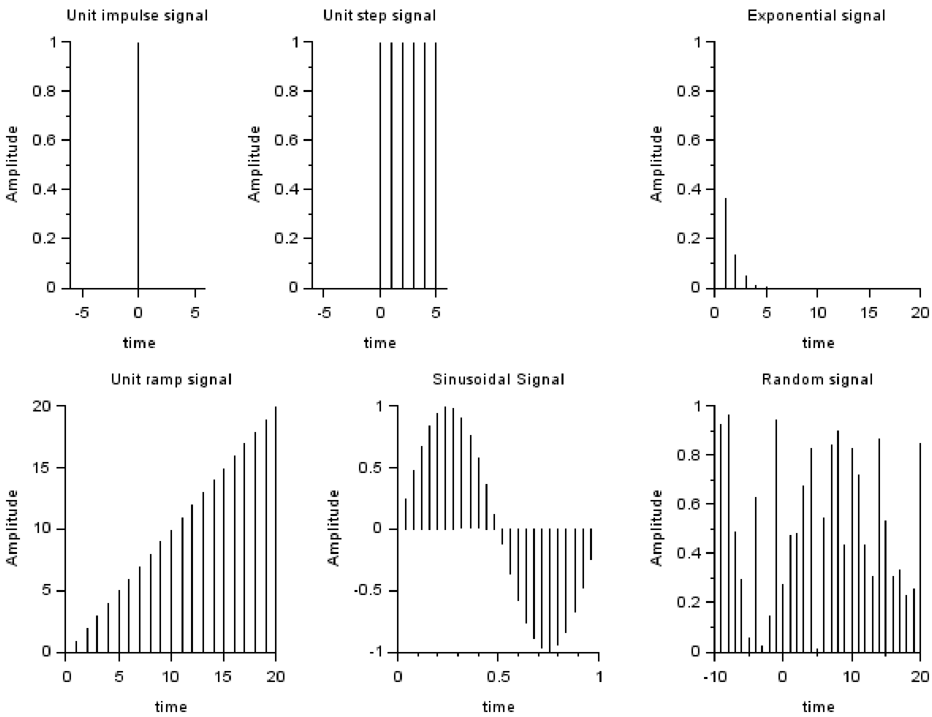
```
    t5=0:0.04:1;
    x5=sin(2*%pi*t5);
    subplot(2,3,5);
    plot2d3(t5,x5);
    title('Sinusoidal Signal')
    xlabel('time');
    ylabel('Amplitude');
```

RANDOM SIGNAL

```
    t6=-10:1:20;
    x6=rand(1,31);
```

```
subplot(2,3,6);
plot2d3(t6,x6);
xlabel('time');
ylabel('Amplitude');
title('Random signal');
```

Output:



2. //Program to Compute the 8-point DFT of the Sequence x[n]=[1,1,1,1,1,1,0,0]

```
clear;
clc ;
close ;
x = [1,1,1,1,1,1,0,0];
//DFT Computation
X = fft (x , -1);
//Display sequence X[k] in command window
disp(X,"X[k]=");
```

RESULT

X[k]=

column 1 to 5

6. - 0.7071068 - 1.7071068i 1. - i 0.7071068 + 0.2928932i 0

column 6 to 8

0.7071068 - 0.2928932i 1. + i - 0.7071068 + 1.7071068i

3. //Program to Compute Circular Convolution y[n] where Y[k]=X1[k].X2[k]

```
//x1[n]=[0,1,2,3,4]
//x2[n]=[0,1,0,0,0]
clear;
clc ;
close ;
x1=[0,1,2,3,4];
x2=[0,1,0,0,0];
//DFT Computation
X1=fft(x1,-1);
X2=fft(x2,-1);
Y=X1.*X2;
//IDFT Computation
y=round(fft(Y,1));
//Display sequence y[n] in command window
disp(y,"y[n]=");
```

RESULT:

y[n]= 4. 0. 1. 2. 3.

4. //Program to Compute the DFT of given Sequence

```
//x[n]=[1,-1,-1,-1,1,1,1,-1] using DIT-FFT Algorithm.
clear;
clc ;
close ;
x = [1,-1,-1,-1,1,1,1,-1];
//FFT Computation
X = fft (x , -1);
disp(X,'X(z) = ');
```

RESULT:

X(z) =

column 1 to 5

0 - 1.4142136 + 3.4142136i 2. - 2.i 1.4142136 - 0.5857864i 4.

```
      column 6 to 8
   1.4142136 + 0.5857864i    2. + 2.i  - 1.4142136 - 3.4142136i
```

5. //Program to find the DFT of a Sequence x[n]=[1,2,3,4,4,3,2,1]

```
//using DIF Algorithm.
clear;
clc ;
close ;
x = [1,2,3,4,4,3,2,1];
//FFT Computation
X = fft (x , -1);
disp(X,'X(z) = ');
```

RESULT:

```
X(z) =
      column 1 to 5
   20.  - 5.8284271 - 2.4142136i    0  - 0.1715729 - 0.4142136i    0
      column 6 to 8
  - 0.1715729 + 0.4142136i    0  - 5.8284271 + 2.4142136i
```

Unit II- INFINITE IMPULSE RESPONSE FILTERS

6. (i) //To Find out Bilinear Transformation of H(s)=2/((s+1)*(s+2))

```
clear;
clc ;
close ;
s=%s;
z=%z;
HS=2/((s+1)*(s+2));
T=1;
HZ=horner(HS,(2/T)*(z-1)/(z+1));
disp(HZ,'H(z) =');
```

RESULT:

```
H(z) =
             2
   2 + 4z + 2z
   -----------
           2
  - 4z + 12z
```

(ii) //To Find out Bilinear Transformation of H(s)=(s^2+4.525)/(s^2+0.692*s+0.504)

```
clear;
```

```
clc ;
close ;
s=%s;
z=%z;
HS=(s^2+4.525)/(s^2+0.692*s+0.504);
T=1;
HZ=horner(HS,(2/T)*(z-1)/(z+1));
disp(HZ,'H(z) =');
RESULT:
H(z) =
                                        2
    1.4478601 + 0.1783288z + 1.4478601z
    -----------------------------------
                                  2
          0.5298913 - 1.1875z + z
```

7. (i) //To Design the Filter using Impulse Invarient Method

```
clear;
clc ;
close ;
s=%s;
T=0.2;
HS=10/(s^2+7*s+10);
elts=pfss(HS);
disp(elts,'Factorized HS = ');
//The poles comes out to be at -5 and -2
p1=-5;
p2=-2;
z=%z;
HZ=T*((-3.33/(1-%e^(p1*T)*z^(-1)))+(3.33/(1-%e^(p2*T)*z^(-1))))
disp(HZ,'HZ = ');

RESULT:

Factorized HS =
        (1)
    3.3333333
    ---------
      2 + s
        (2)
  - 3.3333333
    ---------
      5 + s
 HZ  =
           0.2014254z
    --------------------------
```

```
   0.2465970 - 1.0381995z + Z2
 HZ =
         0.2014254z
   --------------------------

   0.2465970 - 1.0381995z + z2
```

(ii) //To Design the Filter using Impulse Invarient Method

```
clear;
clc ;
close ;
s=%s;
T=1;
HS=(2)/(s^2+3*s+2);
elts=pfss(HS);
disp(elts,'Factorized HS = ');
//The poles comes out to be at -2 and -1
p1=-2;
p2=-1;
z=%z;
HZ=(2/(1-%e^(p2*T)*z^(-1)))-(2/(1-%e^(p1*T)*z^(-1)))
disp(HZ,'HZ = ');
RESULT:

Factorized HS =
        (1)
      2
    -----
    1 + s
        (2)
    - 2
    -----
    2 + s
 HZ  =
         0.4650883z
   --------------------------

   0.0497871 - 0.5032147z + z2
 HZ =
         0.4650883z
   --------------------------

   0.0497871 - 0.5032147z + z2
```

8.

Design a H.P.F. with given specifications

```
clear;
clc ;
close ;
ap=3;//db
as=15;//db
op=500;//rad/sec
os=1000;//rad/sec
N=log(sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1)))/log(os/op);
disp(ceil(N),'Order of the filter, N =');
s=%s;
HS=1/((s+1)*(s^2+s+1));//Transfer Function for N=3
oc=1000//rad/sec
```

```
Order of the filter, N =
   3.
oc  =
   1000.
Normalized Transfer Function, H(s) =
                   3
                  s
   ------------------------------
                              2   3
   1.000D+09 + 2000000s + 2000s + s
```

9.

(a) Design of a lowpass type 1 Chebyshev filter with 2dB ripple in the passband and -30 dB attenuation at the stopband edge. The sampling frequency is assumed to be 3000Hz and the cutoff frequencies at 37.5Hz and 75Hz respectively.

```
>sf=3000;
-->f1=37.5;
-->f2=75;
 -->as=30;
-->ap=2;
-->om=[f1*(2*%pi)/sf,f2*(2*%pi)/sf];
 -->deltas=10.00**(-0.05*as);
-->deltap=(1.0-10.0**(-0.05*ap));
-->[cells,fact,zers,pols]=...
-->eqiir(’lp’,’ch1’,om,deltap,deltas);
```

```
-->cells
 cells  =

!                    2                       2 !
!   1 - 1.9711824z + z      1 - 1.8376851z + z   !
!   ------------------      ------------------   !
!                 2                       2      !
!     Nan +Nanz + z           Nan +Nanz + z      !
```

(b) Convert Analog LPF into [1].High Pass [2].Band Pass IIR Butterworth Filter //Using Analog Filter Transformations //For the given cutoff frequency Wc = 500 Hz

```
clear all;
clc;
close;
omegap =  500;
omegas =  1000;
delta1_in_dB = -3;
delta2_in_dB = -40;
delta1 = 10^(delta1_in_dB/20)
delta2 = 10^(delta2_in_dB/20)
//Calculation of Filter Order
N = log10((1/(delta2^2))-1)/(2*log10(omegas/omegap))
N = ceil(N)
omegac = omegap;
//Poles and Gain Calculation
[pols,gain]=zpbutt(N,omegac);
N =1;
//
omega_LPF = omegap;  //Analog LPF Cutoff frequency
omega_HPF = omega_LPF;  //Analog HPF Cutoff frequency
omega2 = 600;     //Upper Cutoff frequency
omega1 = 300;      //Lower Cutoff Frequency
omega0 = (omega2*omega1);
BW =  omega2 -  omega1;  //Bandwidth
disp('Analog LPF Transfer function')
[hs,pols,zers,gain] = analpf(N,'butt',[0,0],omega_LPF)
hs_LPF = hs;
hs_LPF(2) = hs_LPF(2)/500;
hs_LPF(3)= hs_LPF(3)/500;
s =poly(0,'s');
disp('Analog HPF Transfer function')
h_HPF = horner(hs_LPF,omega_LPF*omega_HPF/s)
disp('Analog BPF Transfer function')
num = (s^2)+omega0
den = BW*s
h_BPF = horner(hs_LPF,omega_LPF*(num/den))
//Plotting Low Pass Filter Frequency Response
```

```
figure
fr=0:.1:1000;
hf=freq(hs_LPF(2),hs_LPF(3),%i*fr);
hm=abs(hf);
plot(fr,hm)
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of LPF Filter Cutoff frequency = 500Hz','Analog
Frequency--->','Magnitude');
//Plotting High Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf_HPF=freq(h_HPF(2),h_HPF(3),%i*fr);
hm_HPF=abs(hf_HPF);
plot(fr,hm_HPF)
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1)
xtitle('Magnitude Response of HPF Filter Cutoff frequency = 500Hz','Analog
Frequency--->','Magnitude');
//Plotting Band Pass Filter Frequency Response
figure
fr=0:.1:1000;
hf_BPF=freq(h_BPF(2),h_BPF(3),%i*fr);
hm_BPF=abs(hf_BPF);
plot(fr,hm_BPF)
a=gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1)
    xtitle('Magnitude Response of BPF Filter Upper Cutoff frequency =
    600Hz & Lower Cutoff frequency = 300Hz','Analog  Frequency---
    >','Magnitude');
```

10. Design a Chebyshev Filter with Given Specifications

```
clear;
clc ;
close ;
ap=2.5;//db
as=30;//db
op=20;//rad/sec
```

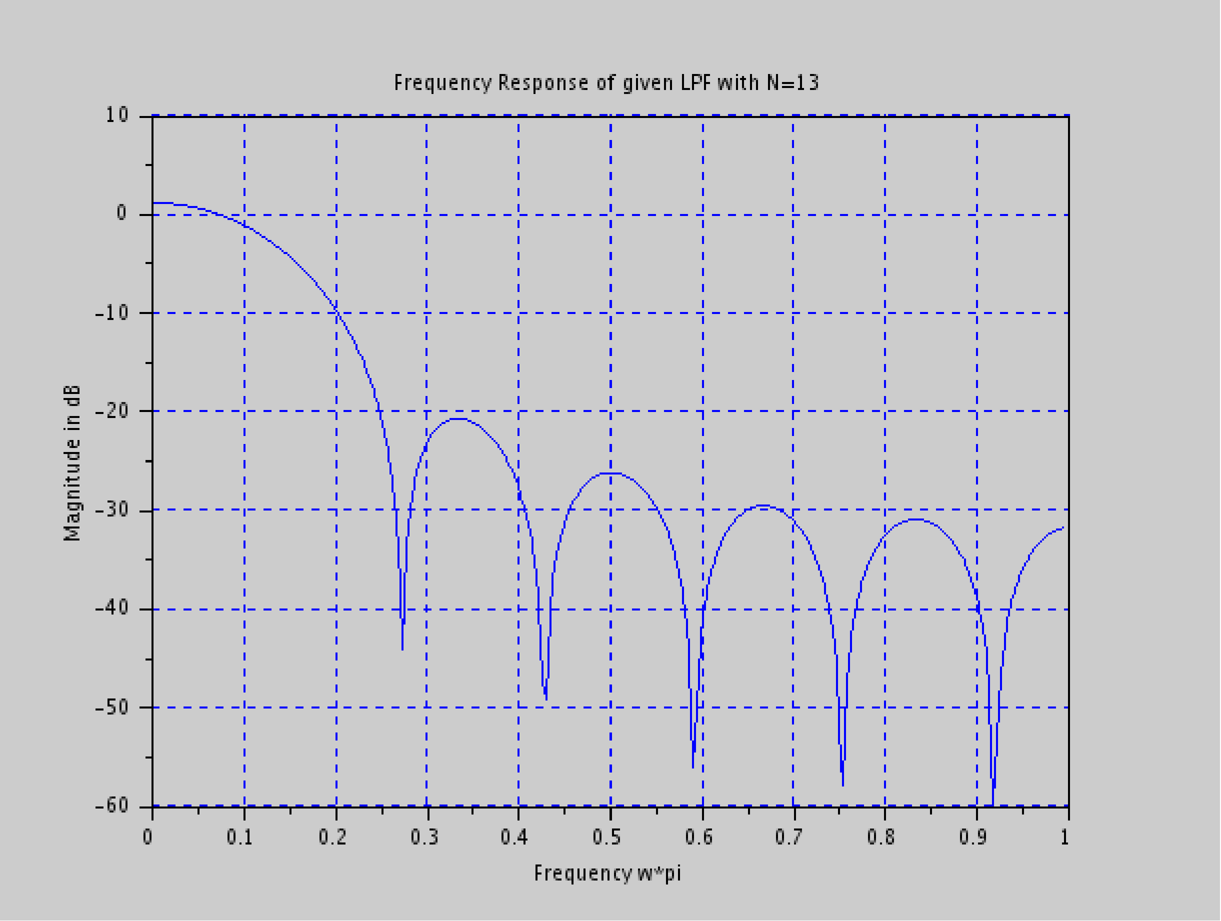
```
os=50;//rad/sec
N=acosh(sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1)))/acosh(os/op);
disp(ceil(N),'Order of the filter, N =');

 Order of the filter, N =
   3.
```

Unit III-FIR filter design

11.

Plot Magnitude Responce of given L.P.F. with specifications:

```
//N=13,w=pi/6
clear;
clc ;
close ;
alpha=6;
U=1;
for n=0+U:1:12+U
if n==7
hd(n)=0.167;
else
hd(n)=(sin(%pi*(n-U-alpha)/6))/(%pi*(n-U-alpha));
end
end
[hzm ,fr ]= frmag (hd ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
figure
plot (2*fr , hzm_dB )
a= gca ();
xlabel ('Frequency w*pi');
ylabel ('Magnitude in dB');
title ('Frequency Response of given LPF with N=13');
xgrid (2)
disp(hd,"Filter Coefficients,h(n)=");
```



Filter Coefficients,h(n)=

6.497D-18
0.0318310
0.0689161
0.1061033
0.1378322
0.1591549
0.167
0.1591549
0.1378322
0.1061033
0.0689161
0.0318310
6.497D-18

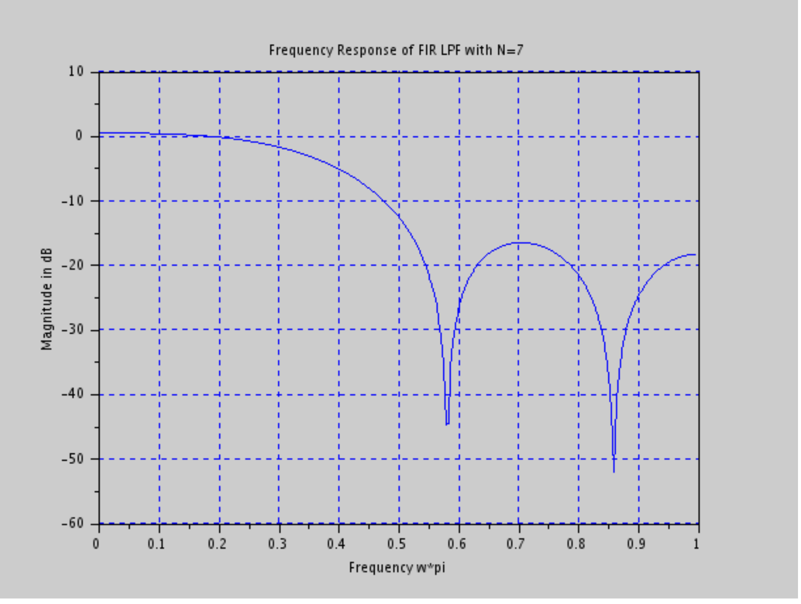
12. //Program to Plot Magnitude Responce of given L.P.F. Using Rectangular Window with specifications:

```
//N=7,fc=1000Hz,F=5000Hz
clear;
clc ;
close ;
N=7;
U=4;
```

```
h_Rect=window('re',N);
for n=-3+U:1:3+U
if n==4
hd(n)=0.4;
else
hd(n)=(sin(2*%pi*(n-U)/5))/(%pi*(n-U));
end
h(n)=hd(n)*h_Rect(n);
end
[hzm ,fr ]= frmag (h ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
figure
plot (2*fr , hzm_dB )
a= gca ();
xlabel ('Frequency w*pi');
ylabel ('Magnitude in dB');
title ('Frequency Response of FIR LPF with N=7');
xgrid (2)
disp(h,"Filter Coefficients,h(n)=");
```

RESULT:

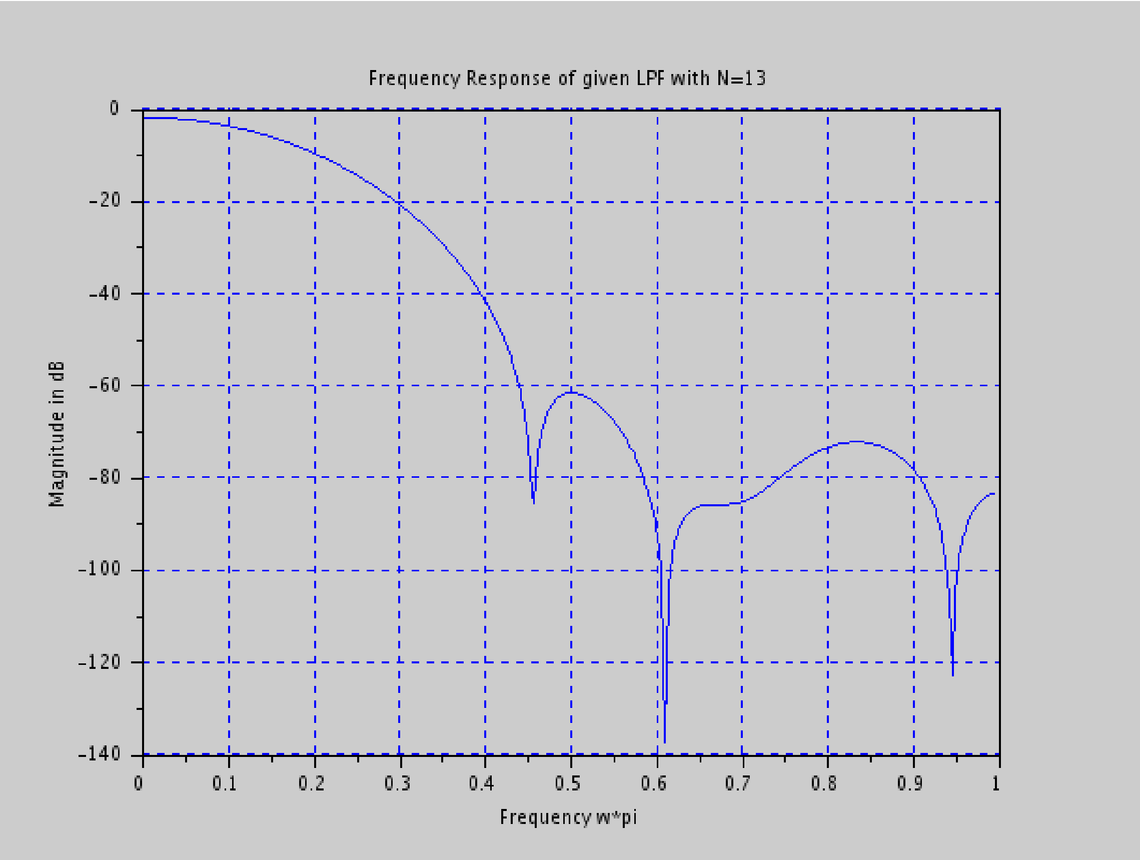
```
Filter Coefficients,h(n)=
  - 0.0623660
    0.0935489
    0.3027307
    0.4
    0.3027307
    0.0935489
  - 0.0623660
```



13.

Plot Magnitude Responce of given L.P.F. with specifications:

```
//N=13,w=pi/6
//Using Hamming Window
clear;
clc ;
close ;
N=13;
alpha=6;
U=1;
h_hamm=window('hm',N);
for n=0+U:1:12+U
if n==7
hd(n)=0.167;
else
hd(n)=(sin(%pi*(n-U-alpha)/6))/(%pi*(n-U-alpha));
end
h(n)=hd(n)*h_hamm(n);
end
[hzm ,fr ]= frmag (h ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
figure
plot (2*fr , hzm_dB )
a= gca ();
xlabel ('Frequency w*pi');
ylabel ('Magnitude in dB');
title ('Frequency Response of given LPF with N=13');
xgrid (2)
disp(h,"Filter Coefficients,h(n)=");
disp(h,"Filter Coefficients,h(n)=");
```



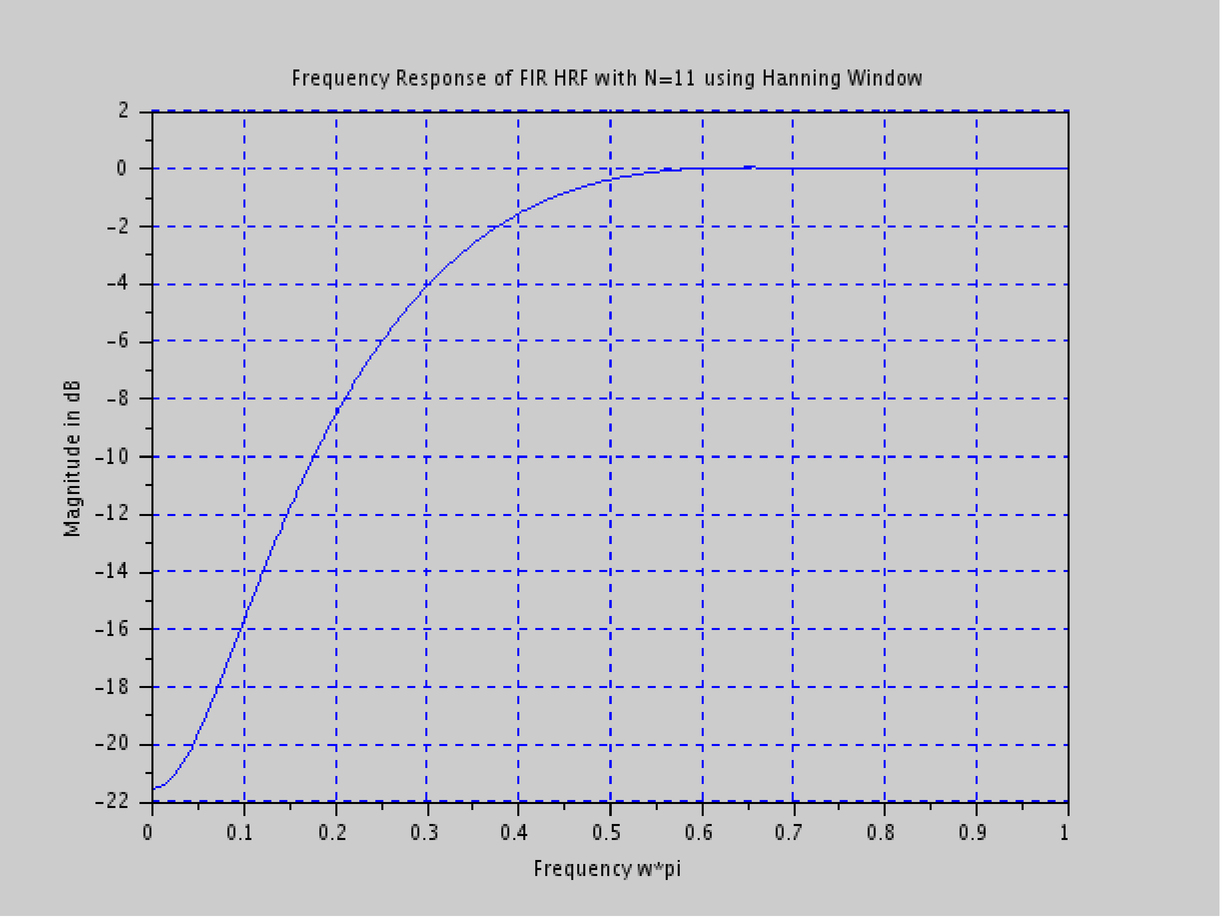
Filter Coefficients,h(n)=

```
5.198D-19
0.0045082
0.0213640
0.0572958
0.1061308
0.1493465
0.167
0.1493465
0.1061308
0.0572958
0.0213640
0.0045082
5.198D-19
```

14.

```
//Program to Plot Magnitude Responce of ideal H.P.F.
//using Hanning Window
//wc1=0.25*pi
//N=11
clear;
clc ;
close ;
N=11;
U=6;
```

```
h_hann=window('hn',N);
for n=-5+U:1:5+U
if n==6
hd(n)=0.75;
else
hd(n)=(sin(%pi*(n-U))-sin(%pi*(n-U)/4))/(%pi*(n-U));
end
h(n)=h_hann(n)*hd(n);
end
[hzm ,fr ]= frmag (h ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
figure
plot (2*fr , hzm_dB )
a= gca ();
xlabel ('Frequency w*pi');
ylabel ('Magnitude in dB');
title ('Frequency Response of FIR HRF with N=11 using Hanning Window');
xgrid (2);
```

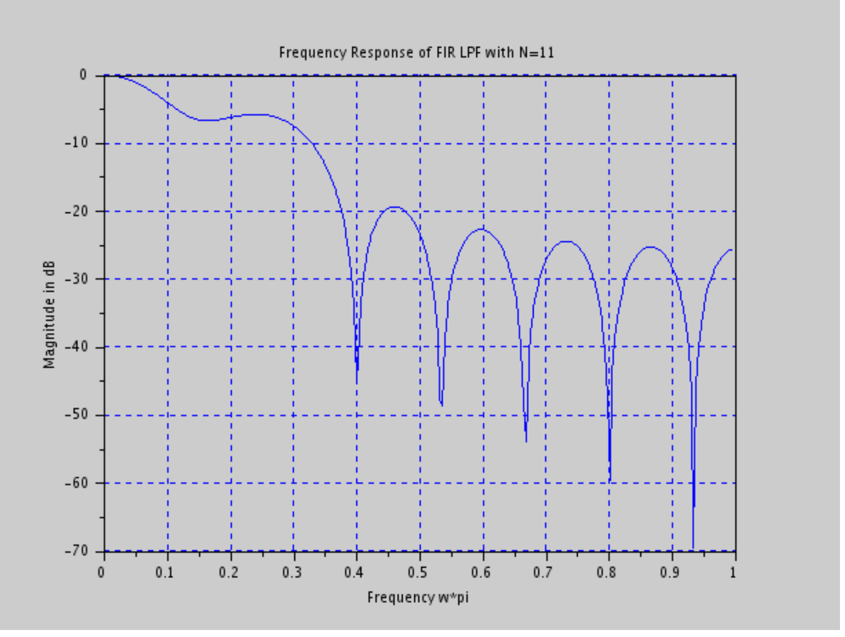


15. //Program to design Frequency Sampling Method FIR L.P.F. filter with following specifications:

//N=15, wc=pi/4

```
clear;
clc ;
close ;
N=15;
U=1;
for n=0+U:1:N-1+U
h(n)=(1+cos(2*%pi*(7-n)/N)+cos(4*%pi*(7-n)/N))/N;
end
[hzm ,fr ]= frmag (h ,256) ;
hzm_dB = 20* log10 (hzm)./ max ( hzm );
figure;
plot (2*fr , hzm_dB );
a= gca ();
xlabel ('Frequency w*pi');
ylabel ('Magnitude in dB');
title ('Frequency Response of FIR LPF with N=11');
xgrid (2)
```

RESULT:



Unit IV- FINITE WORDLENGTH EFFECTS AND MULTIRATE SIGNAL PROCESSING

```
16. //To Compare the Variance of Output due to A/D Conversion process
//y(n)=0.8y(n-1)+x(n)
clear;
clc ;
```

```
close ;
n=8;  //Bits
r=100;  //Range
Q=2*r/(2^n);  //Quantization Step Size
Ve=(Q^2)/12;
Vo=Ve*(1/(1-0.8^2));
disp(Q,'QUANTIZATION STEP SIZE =');
disp(Ve,'VARIANCE OF ERROR SIGNAL =');
disp(Vo,'VARIANCE OF OUTPUT =');
```

RESULT:

```
QUANTIZATION STEP SIZE =
   0.78125
 VARIANCE OF ERROR SIGNAL =
   0.0508626
 VARIANCE OF OUTPUT =
   0.1412851
```

17. Output noise power

```
/ Output noise power
clc;clear;close;
z=poly(0,'z');
H=0.5*z/(z-0.5);
B=8;
pn=2^(-2*B)/12;          //Noise power
X=H*horner(H,1/z)/z;
r=roots(denom(X));
rl=length(r);
rc=coeff(denom(X))
q1=[];q2=[];
for n=1:rl                //Loop to separate poles inside the unit circle
    if (abs(r(n))<1) then
        q1=[q1 r(n)];
    else
        q2=[q2 r(n)];
    end
end
P=numer(X)/rc(length(rc));
Q1=poly(q1,'z');
Q2=poly(q2,'z');
I=residu(P,Q1,Q2);        //Residue Calculation
po=pn*I;                  //Output Noise power
disp(pn,'Input Noise power');
disp(po,'Output Noise power');
```

```
rc  =
  - 0.5    1.25  - 0.5
 Input Noise power
    0.0000013
 Output Noise power
    0.0000004
```

18. (i) Deadband interval

```
  clc; clear;
//y(n)=0.9y(n-1)+x(n)
//Input x(n)=0
n=-1;y=12;        //Initial Condition y(-1)=12
flag=1;
while n<8
    n=n+1;
    y=[y 0.9*y(n+1)];
    yr=round(y);
end
disp(n,'n=');
disp(y,'y(n)-exact');
disp(yr,'y(n)-rounded');
disp([-yr(n+2) yr(n+2)],'Deadband interval ')

n=
    8.
 y(n)-exact
         column 1 to 8
    12.    10.8    9.72    8.748    7.8732    7.08588    6.377292    5.7395628
         column  9 to 10
    5.1656065    4.6490459
 y(n)-rounded
    12.    11.    10.    9.    8.    7.    6.    6.    5.    5.
 Deadband interval
  - 5.    5.
```

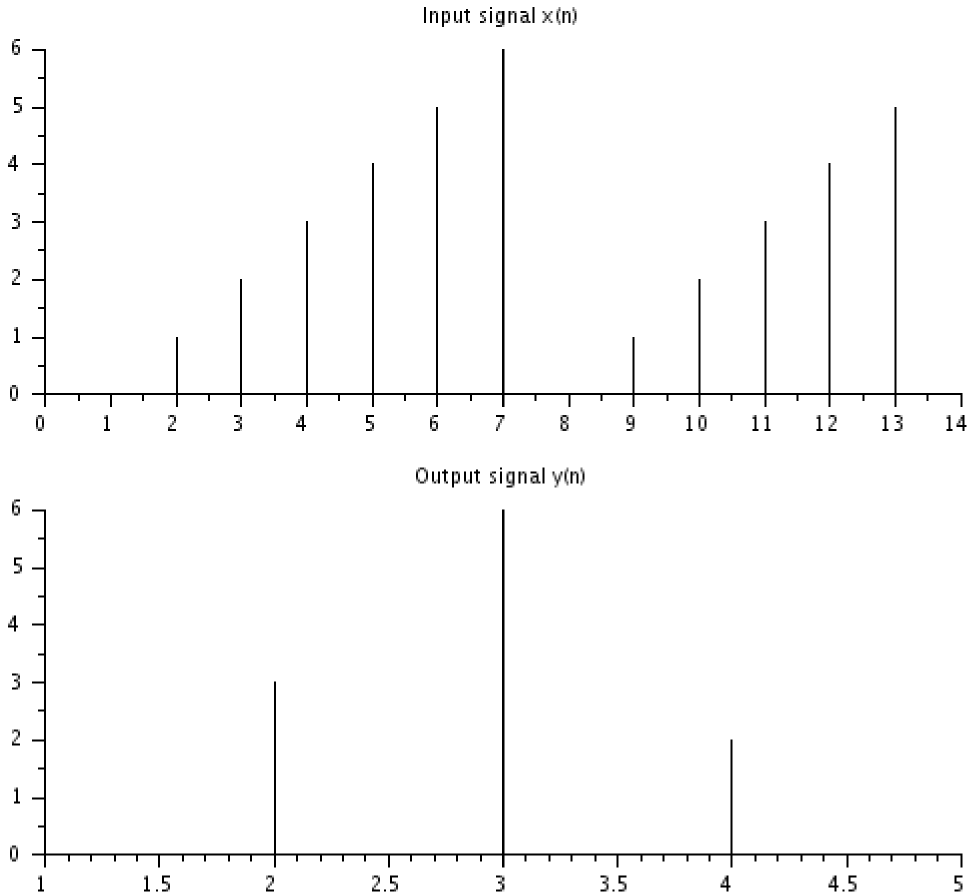
(or)

(ii) Deadband interval

```
clc; clear;
//y(n)=0.9y(n-1)+x(n)
a=0.9;
l=ceil(0.5/(1-abs(a)));
disp([-l l],'Deadband interval ')

Deadband interval
  - 6.    6.
```

19. Decimation

```
clc;clear;close;
x=[0:6 0:6];
y=x(1:3:length(x));
disp(x,'Input signal x(n)=');
disp(y,'Output signal of decimation process by factor three y(n)');
subplot(2,1,1);
plot2d3(x);title('Input signal x(n)');
subplot(2,1,2);
plot2d3(y);title('Output signal y(n)');
```

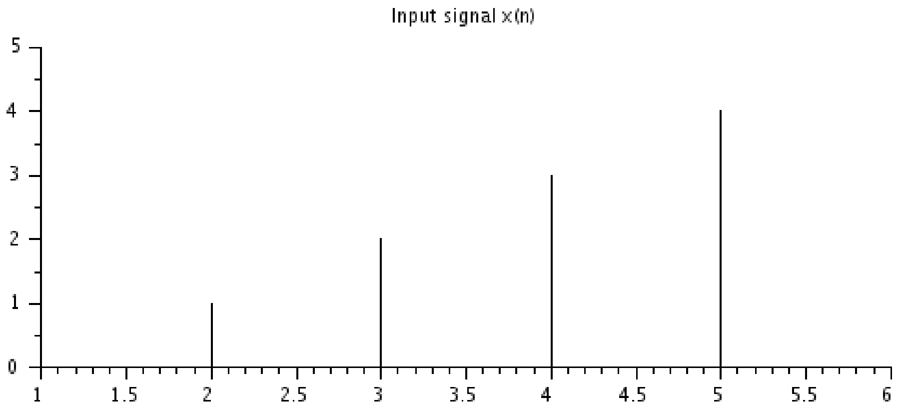


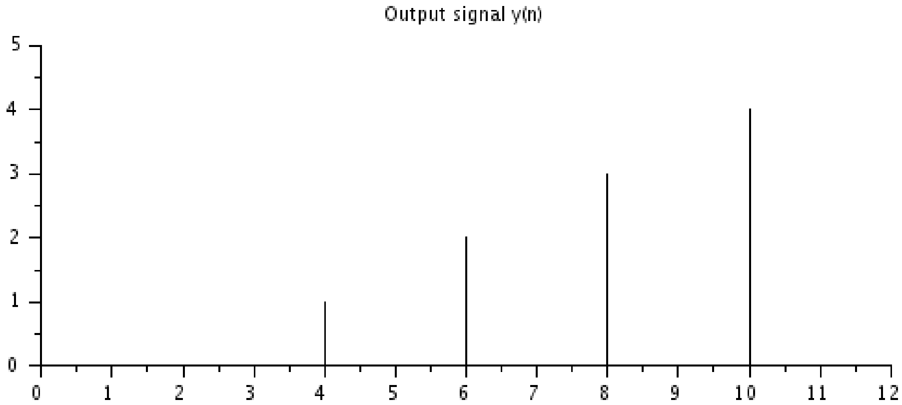

```
Input signal x(n)=
      column  1 to 13
   0.    1.    2.    3.    4.    5.    6.    0.    1.    2.    3.    4.    5.
      column 14
   6.
 Output signal of decimation process by factor three y(n)
   0.    3.    6.    2.    5.
```

20. Interpolation

```
clc;clear;close;

x=0:5;
```

```
y=[];
for i=1:length(x)
y(1,2*i)=x(i);
end
disp(x, 'Input signal x(n)=');
disp(y, 'Output signal of interpolation process with factor two y(n)');
subplot(2,1,1);
plot2d3(x);title('Input signal x(n)');
subplot(2,1,2);
plot2d3(y);title('Output signal y(n)');
```






```
Input signal x(n)=

   0.    1.    2.    3.    4.    5.

 Output signal of interpolation process with factor two y(n)

   0.    0.    0.    1.    0.    2.    0.    3.    0.    4.    0.    5.
```

21. (i) //To Design an Analog Butterworth Filter

```
clear;
clc ;
close ;
op=0.2*%pi;
os=0.4*%pi;
e1=0.9;
l1=0.2;
epsilon=sqrt(1/(e1^2)-1);
lambda=sqrt(1/(l1^2)-1);
N=log(lambda/epsilon)/log(os/op);
disp(ceil(N),'Order of the filter, N =');
s=%s;
HS=1/((s^2+0.76537*s+1)*(s^2+1.8477*s+1));//Transfer Function for N=4
oc=op/epsilon^(1/ceil(N));
HS1=horner(HS,s/oc);
disp(HS1,'Normalized Transfer Function, H(s) =');
```

RESULT:

```
Order of the filter, N =
    4.
 Normalized Transfer Function, H(s) =
                             1
    ---------------------------------------------------

    1    + 3.469403s + 6.0185667s2 + 6.1159251s3 + 3.1075263s4
```

9. (ii) //To Design an Analog Butterworth Filter

```
clear;
clc ;
close ;
ap=2;//db
as=10;//db
op=20;//rad/sec
os=30;//rad/sec
N=log(sqrt((10^(0.1*as)-1)/(10^(0.1*ap)-1)))/log(os/op);
disp(ceil(N),'Order of the filter, N =');
s=%s;
HS=1/((s^2+0.76537*s+1)*(s^2+1.8477*s+1));//Transfer Function for N=4
oc=op/(10^(0.1*ap)-1)^(1/(2*ceil(N)));
HS1=horner(HS,s/oc);
disp(HS1,'Normalized Transfer Function, H(s) =');
```

RESULT:

Order of the filter, N =

4.

Normalized Transfer Function, H(s) =

1

----

1 + 0.1221815s + 0.0074644s2 + 0.0002671s2 + 0.0000048s4

Unit V- DIGITAL SIGNAL PROCESSORS

Extra lab experiments

1. SOUND PLAY COMMAND IN SCILAB

```
Reading a Speech Signal &
//Play the signal
clear;
clc;
[y,Fs,bits_y]=wavread("E:\4.wav");
sound(y,Fs,16)
```

2. This function is used to design an equalizer which will be useful to compensate the aperture effect produced in the flat top sampling.

```
function [E]=EqualizerFor_ApertureEff(Ts)

//Equalizer to Compensate for aperture effect
T_Ts = 0.01:0.01:Ts;
E(1) =1;
for i = 2:length(T_Ts)
  E(i) = ((%pi/2)*T_Ts(i))/(sin((%pi/2)*T_Ts(i)));
end
a =gca();
a.data_bounds = [0,0.8;0.8,1.2];
plot2d(T_Ts,E,5)
xlabel('Duty cycle T/Ts')
ylabel('1/sinc(0.5(T/Ts))')
title('Normalized equalization (to compensate for aperture effect) plotted versus T/Ts')
endfunction
```

3. This function sholud be used along with funciton adapt_filt

```
function[y]= pow_1(x,N)
    xold = 0.0;
```

```
    for n =1:N
        sumx = xold+(x(n)^2);
        xold = sumx;
    end
    y = sumx/N;
endfunction
```

4. This program is used to perform speech noise cancellation using LMS adaptive filter in scilab

```
//Caption: Speech Noise Cancellation using LMS Adaptive Filter

clc;
//Reading a speech signal
[x,Fs,bits]=wavread("E:\4.wav");
order = 40;  // Adaptive filter order
x = x';
N = length(x);
t = 1:N;
//Plot the speech signal
figure(1)
subplot(2,1,1)
plot(t,x)
title('Noise free Speech Signal')
//Generation of noise signal
noise = 0.1*rand(1,length(x));
//Adding noise with speech signal
for i = 1:length(noise)
    primary(i)= x(i)+noise(i);
end
//Plot the noisy speech signal
subplot(2,1,2)
plot(t,primary)
title('primary = speech+noise (input 1)')
//Reference noise generation
for i = 1:length(noise)
    ref(i)= noise(i)+0.025*rand(10);
end
//Plot the reference noise
figure(2)
```

```
subplot(2,1,1)
plot(t,ref)
title('reference noise (input 2)')
//Adaptive filter coefficients initialized to zeros
w = zeros(order,1);
Average_Power = pow_1(x,N)
mu = 1/(10*order*Average_Power); //Adaptive filter step size
//Speech noise cancellation
for k = 1:110
    for i =1:N-order-1
        buffer = ref(i:i+order-1); //current order points of reference
        desired(i) = primary(i)-buffer'*w; // dot product the reference & filter
        w = w+(buffer.*mu*desired(i)); //update filter coefficients
    end
end
//Plot the Adaptive Filter output
subplot(2,1,2)
plot([1:length(desired)],desired)
title('Denoised Speech Signal at Adaptive Filter Output')
//Calculation of Mean Squarred Error between the original speech signal and
//Adaptive filter output
for i =1:N-order-1
    err(i) = x(i)-desired(i);
    square_error(i)= err(i)*err(i);
end
MSE = (sum(square_error))/(N-order-1);
MSE_dB = 20*log10(MSE);
//Playing the original speech signal
sound(x,Fs,16)
//Delay between playing sound signals
for i = 1:1000
    j = 1;
end
/////////////////////////////////
//Playing Noisy Speech Signal
sound(primary,Fs,16)
//Delay between playing sound signals
for i = 1:1000
    j = 1;
```

```
end
//////////////////////////////////
//Playing denoised speech signal (Adaptive Filter Output)
sound(desired,Fs,16)
```

5. This program is used to get speech or voice signal informations such as sampling rate, bit deoth and tine duration of speech signal etc

```
//Caption: Reading a Speech Signal &
//[1]. Displaying its sampling rate
//[2]. Number of bits used per speech sample
//[3]. Total Time duration of the speech signal in seconds
clear;
clc;
[y,Fs,bits]=wavread("E:\4.wav");
a = gca();
plot2d(y);
a.x_location = 'origin';
title('Speech signal with Sampling Rate = 8 KHz, No. of Samples = 8360')
disp(Fs,'Sampling Rate in Hz Fs = ');
disp(bits,'Number of bits used per speech sample b =');
N = length(y);
T = N/Fs;
disp(N,'Total Number of Samples N =')
disp(T,'Duration of speech signal in seconds T=')
//Result
//Sampling Rate in Hz Fs =
//     8000.
//Number of bits used per speech sample b =
//    16.
//Total Number of Samples N =
//    8360.
//Duration of speech signal in seconds T=
//    1.045
```