DSP important topics

Generate a functional sequence of a signal (Sine, Cosine, triangular, Square, Saw tooth and sinc) using MATLAB function.

```
%Program for sine wave
t=0:0.1:10;
y=sin(2*pi*t);
subplot(3,3,1);
plot(t,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Sine wave');

%Program for cosine wave
t=0:0.1:10;
y=cos(2*pi*t);
subplot(3,3,2);
plot(t,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Cosine wave');

%Program for square wave
t=0:0.001:10;
y=square(t);
subplot(3,3,3);
plot(t,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Square wave');

%Program for sawtooth wave
t=0:0.1:10;
y=sawtooth(t);
subplot(3,3,4);
plot(t,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Sawtooth wave');

%Program for Triangular wave
t=0:.0001:20;
y=sawtooth(t,.5); % sawtooth with 50% duty cycle (triangular)
subplot(3,3,5);
plot(t,y);
ylabel ('Amplitude');
xlabel ('Time Index');
title('Triangular waveform');

%Program for Sinc Pulse
t=-10:.01:10;
y=sinc(t);
axis([-10 10 -2 2]);
```

```
subplot(3,3,6)
plot(t,y);
ylabel ('Amplitude');
xlabel ('Time Index');
title('Sinc Pulse');

% Program for Exponential Growing signal
t=0:.1:8;
a=2;
y=exp(a*t);
subplot(3,3,7);
plot(t,y);
ylabel ('Amplitude');
xlabel ('Time Index');
title('Exponential growing Signal');

% Program for Exponential Growing signal
t=0:.1:8;
a=2;
y=exp(-a*t);
subplot(3,3,8);
plot(t,y);
ylabel ('Amplitude');
xlabel ('Time Index');
title('Exponential decaying Signal');
```

Generate a discrete time signal sequence (Unit step, Unit ramp, Sine, Cosine, Exponential, Unit impulse) using MATLAB function.

```
%Program for unit step sequence clc;
N=input('Enter the length of unit step sequence(N)= ');
n=0:1:N-1;
y=ones(1,N);
subplot(3,2,1);
stem(n,y,'k');
xlabel('Time')
ylabel('Amplitude')
title('Unit step sequence');
%Program for unit ramp sequence
N1=input('Enter the length of unit ramp sequence(N1)= ');
n1=0:1:N1-1;
y1=n1;
subplot(3,2,2);
stem(n1,y1,'k'); xlabel('Time');
ylabel('Amplitude');
title('Unit ramp sequence');
%Program for sinusoidal sequence
N2=input('Enter the length of sinusoidal sequence(N2)=');
n2=0:0.1:N2-1;
y2=sin(2*pi*n2);
subplot(3,2,3);
stem(n2,y2,'k');
xlabel('Time');
ylabel('Amplitude');
```

```
title('Sinusoidal sequence');
%Program for cosine sequence
N3=input('Enter the length of the cosine sequence(N3)=');
n3=0:0.1:N3-1;
y3=cos(2*pi*n3);
subplot(3,2,4);
stem(n3,y3,'k');
xlabel('Time');
ylabel('Amplitude');
title('Cosine sequence');
%Program for exponential sequence
N4=input('Enter the length of the exponential sequence(N4)= ');
n4=0:1:N4-1;
a=input('Enter the value of the exponential sequence(a)='); y4=exp(a*n4);
subplot(3,2,5);
stem(n4,y4,'k');
xlabel('Time');
ylabel('Amplitude');
title('Exponential sequence');
%Program for unit impulse
n=-3:1:3;
y=[zeros(1,3),ones(1,1),zeros(1,3)];
subplot(3,2,6);
stem(n,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Unit impulse');
```

Write MATLAB programs to find out the linear convolution and Circular convolution of two sequences

```
% linear convolution
x=input('Enter the first sequence - ');
h=input('Enter the second sequence - ');
m=length(x);
n=length(h);
y=zeros(1,m+n-1);
h=[h,zeros(1,m+n-1)];
x=[x,zeros(1,m+n-1)];
cnt=0;
for i=1:1:m+n-1;
    cnt=i;
    for j=1:1:n
        y(1,cnt)=y(1,cnt)+h(j)*x(i);
        if cnt<m+n-1
            cnt=cnt+1;
        end

    end
end
disp (y)
```

```
% Circular Convolution
clc;
close all;
clear all;
x1=input('Enter the first sequence :');
```

```
x2=input('Enter the second sequence: ');
N1=length(x1);
N2=length(x2);
N=max(N1,N2);

if(N2>N1)
x4=[x1,zeros(1,N-N1)];
x5=x2;
elseif(N2==N1)
        x4=x1;
        x5=x2;
else
x4=x1;
x5=[x2,zeros(1,N-N2)];
end

x3=zeros(1,N);
for m=0:N-1
x3(m+1)=0;
for n=0:N-1
        j=mod(m-n,N);
        x3(m+1)=x3(m+1)+x4(n+1).*x5(j+1);
end
end

subplot(4,1,1)
stem(x1);
title('First Input Sequence');
xlabel('Samples');
ylabel('Amplitude');
subplot(4,1,2)
stem(x2);
title('Second Input Sequence');
xlabel('Samples');
ylabel('Amplitude');
subplot(4,1,3)
stem(x3);
title('Circular Convolution Using Modulo Operator');
xlabel('Samples');
ylabel('Amplitude');

%In built function
y=cconv(x1,x2,N);
subplot(4,1,4)
stem(y);
title('Circular Convolution using Inbuilt Function');
xlabel('Samples');
ylabel('Amplitude');
```

linear convolution of two given sequences

(A) Using CONV command

```
clc;
x1=input('enter the first sequence');
subplot(3,1,1);
stem(x1);
ylabel('amplitude');
title('plot of the first sequence');
```

```
x2=input('enter 2nd sequence');
subplot(3,1,2);
stem(x2);
ylabel('amplitude');
title('plot of 2nd sequence');
f=conv(x1,x2);
disp('output of linear conv is');
disp(f);
xlabel('time index n');
ylabel('amplitude f');
subplot(3,1,3);
stem(f);
title('linear conv of sequence');
```

(B) Linear convolution Using DFT and IDFT

```
clc;
clear all;
x1=input('enter the first sequence');
x2=input('enter the second sequence');
n=input('enter the no of points of the dft');
subplot(3,1,1);
stem(x1,'filled');
title('plot of first sequence');
subplot(3,1,2);
stem(x2,'filled');
title('plot the second sequnce');
n1 =length(x1);
n2 = length(x2);
m = n1+n2-1; % Length of linear convolution
x = [x1 zeros(1,n2-1)]; % Padding of zeros to make it of
length m
y = [x2 zeros(1,n1-1)];
x_fft = fft(x,m);
y_fft = fft(y,m);
dft_xy = x_fft.*y_fft;
y=ifft(dft_xy,m);
disp('the circular convolution result is ......');
disp(y);
subplot(3,1,3);
stem(y,'filled');
title('plot of circularly convoluted sequence');
```

computation of N point DFT of a given sequence and to plot magnitude and phase spectrum.

```
N = input('Enter the the value of N(Value of N in N-Point DFT)');
x = input('Enter the sequence for which DFT is to be calculated');
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-1j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
Xk=x*WNnk;
MagX=abs(Xk) % Magnitude of calculated DFT
PhaseX=angle(Xk)*180/pi % Phase of the calculated DFT
figure(1);
subplot(2,1,1);
stem(k,MagX);
subplot(2,1,2);
plot(k,PhaseX);
```

Write MATLAB program for spectrum analysing signal using FFT.

```
N=input('type length of DFT= ');
T=input('type sampling period= ');
freq=input('type the sinusoidal freq= ');
k=0:N-1;
f=sin(2*pi*freq*1/T*k);
F=fft(f);
stem(k,abs(F));
grid on;
xlabel('k');
ylabel('X(k)');
```

FIND THE FFT OF A GIVEN SEQUENCE

```
clc;
clear all;
close all;
x=input('Enter the sequence : ')
N=length(x)
xK=fft(x,N)
xn=ifft(xK)
n=0:N-1;
subplot (2,2,1);
stem(n,x);
xlabel('n---->');
ylabel('amplitude');
```

```
title('input sequence');
subplot (2,2,2);
stem(n,abs(xK));
xlabel('n---->');
ylabel('magnitude');
title('magnitude response');
subplot (2,2,3);
stem(n,angle(xK));
xlabel('n---->');
ylabel('phase');
title('Phase responce');
subplot (2,2,4);
stem(n,xn);
xlabel('n---->');
ylabel('amplitude');
title('IFFT');
```

write a program to design the FIR low pass, High pass, Band pass and Band stop filters using RECTANGULAR window and find out the response of the filter by using MATLAB.

```
clear all;
rp=input('Enter the PB ripple rp =');
rs=input('Enter the SB ripple rs =');
fp=input('Enter the PB ripple fp =');
fs=input('Enter the SB ripple fs =');
f=input('Enter the sampling frequency f =');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
den=14.6*(fs-fp)/f;
n=ceil(num/den);
n1=n+1;
if(rem(n,2)~=0)
    n=n1;
    n=n-1;
end;
y=boxcar(n1); %using rectangular window
%LPF
b=fir1(n,wp,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);
plot(o/pi,m);
xlabel('Normalized frequency------>');
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF LPF');
%HPF
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);
plot(o/pi,m);
xlabel('Normalized frequency------>');
```

```
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF HPF');
%BPF
wn=[wp ws];
b=fir1(n,wn,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);
plot(o/pi,m);
xlabel('Normalized frequency------>');
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF BPF');
%BSF
b=fir1(n,wn,'stop',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);
plot(o/pi,m);
xlabel('Normalized frequency------>');
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF BSF');
```

output

```
Enter the PB ripple rp =.03
Enter the SB ripple rs =.05
Enter the PB ripple fp =2000
Enter the SB ripple fs =3000
Enter the sampling frequency f =9000
```

Write a program to design the FIR low pass, High pass, Band pass and Band stop filters using HANNING window and find out the response of the filter by using MATLAB.

```
clear all;
rp=input('Enter the PB ripple rp =');
rs=input('Enter the SB ripple rs =');
fp=input('Enter the PB ripple fp =');
fs=input('Enter the SB ripple fs =');
f=input('Enter the sampling frequency f =');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13; den=14.6*(fs-fp)/f; n=ceil(num/den);
n1=n+1;
if(rem(n,2)~=0)
n=n1;
n=n-1;
end;
y=hanning(n1);
%LPF
b=fir1(n,wp,y);
[h,O]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);
plot(O/pi,m);
```

```
xlabel('Normalized freqency------>');
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF LPF');
%HPF
b=fir1(n,wp,'high',y);
[h,O]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);
plot(O/pi,m);
xlabel('Normalized freqency------>');
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF HPF');
%BPF
wn=[wp ws];
b=fir1(n,wn,y);
[h,O]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);
plot(O/pi,m);
xlabel('Normalized freqency------>');
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF BPF');
%BSF
b=fir1(n,wn,'stop',y);
[h,O]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);
plot(O/pi,m);
xlabel('Normalized freqency------>');
ylabel('Gain in db--------.');
title('MAGNITUDE RESPONSE OF BSF');
```

Enter the PB ripple rp =.03
Enter the SB ripple rs =.02
Enter the PB ripple fp =1500
Enter the SB ripple fs =2000
Enter the sampling frequency f =9000

IMPLEMENTATION OF LP FIR FILTER

```
clc;
clear all;
close all;
n=20;
fp=200;
fq=300;
fs=1000;
fn=2*fp/fs;
window=boxcar(n+1); %rectangular window
% window=hanning(n+1); %Hanning window
% window=hamming(n+1); %Hanning window
b=fir1(n,fn,window);
[H W]=freqz(b,1,128);
subplot(2,1,1);
plot(W/pi,abs(H));
```

```
title('magnitude response of lpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
subplot(2,1,2);
plot(W/pi,angle(H));
title('phase response of lpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');
```

IMPLEMENTATION OF HP FIR FILTER

```
clc;
clear all;
close all;
n=20;
fp=200;
fq=300;
fs=1000;
fn=2*fp/fs;
window=boxcar(n+1); %rectangular window
% window=hanning(n+1); %Hanning window
% window=hamming(n+1); %Hanning window
b=fir1(n,fn,'high',window);
[H W]=freqz(b,1,128);
subplot(2,1,1);
plot(W/pi,abs(H));
title('magnitude response of lpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
subplot(2,1,2);
plot(W/pi,angle(H));
title('phase response of lpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');
```

design a Butterworth digital IIR filter using MATLAB

```
clear all;
clc; close all;
format long
rp=input('enter the pass band ripple');
rs=input('enter the stop band ripple');
wp=input('enter the pass band frequency');
ws=input('enter the stop band frequency');
fs=input('enter the sampling frequency');
w1=2*wp/fs;
w2=2*ws/fs;
%LOW PASS FILTER
[n,wn]=buttord(w1,w2,rp,rs);
%[b,a]=zp2tf(z,p,k);
```

```
[b,a]= butter(n,wn);
W=0:0.01:pi;
[h,om]=freqz(b,a,W);
m=20*log10(abs(h));
an=angle(h); %figure(1);
subplot(2,1,1);
plot(om/pi,m);
ylabel('gainin db…>');
xlabel('(a)normalized frequency…>');
%figure(1);
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b)normalized frequency…>');
ylabel('phase in radians…>');
%HIGH PASS FILTER
[n,wn]=buttord(w1,w2,rp,rs);
[b,a]=butter(n,wn,'high');
w=0:0.01:pi;
[h,om]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure(2);
subplot(2,1,1);
plot(om/pi,m);
ylabel('gainin db…>');
xlabel('(a)normalized frequency…>');
figure(2);
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b)normalized frequency…>');
ylabel('phase in radians…>');
%BAND PASS FILTER
[n]=buttord(w1,w2,rp,rs);
wn=[w1,w2];
[b,a]=butter(n,wn,'bandpass');
w=0:0.01:pi;
[h,om]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure(3);
subplot(2,1,1);
plot(om/pi,m);
ylabel('gainin db…>');
xlabel('(a)normalized frequency…>');
figure(3);
subplot(2,1,2);
plot(om/pi,an);
```

IMPLEMENTATION OF LP IIR FILTER

```
clc;
clear all;
close all;
disp('enter the IIR filter design specifications');
rp=input('enter the passband ripple:');
rs=input('enter the stopband ripple:');
wp=input('enter the passband freq:');
ws=input('enter the stopband freq:');
fs=input('enter the sampling freq:');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs,'s');
disp('Frequency response of IIR LPF is:');
[b,a]=butter(n,wn,'low','s');
w=0:.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure,subplot(2,1,1);plot(om/pi,m);
title('magnitude response of IIR filter is:');
xlabel('(a) Normalized freq. -->');
ylabel('Gain in dB-->');
subplot(2,1,2);plot(om/pi,an);
title('phase response of IIR filter is:');
xlabel('(b) Normalized freq. -->');
ylabel('Phase in radians-->');
```

enter the IIR filter design specifications

enter the passband ripple:15

enter the stopband ripple:60

enter the passband freq:1500

enter the stopband freq:3000

enter the sampling freq:7000

IMPLEMENTATION OF HP IIR FILTER

```
clc;
clear all;
close all;
disp('enter the IIR filter design specifications');
rp=input('enter the passband ripple:');
rs=input('enter the stopband ripple:');
wp=input('enter the passband freq:');
ws=input('enter the stopband freq:');
```

```
fs=input('enter the sampling freq:');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs,'s');
disp('Frequency response of IIR LPF is:');
[b,a]=butter(n,wn,'high','s');
w=0:.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure,subplot(2,1,1);plot(om/pi,m);
title('magnitude response of IIR filter is:');
xlabel('(a) Normalized freq. -->');
ylabel('Gain in dB-->');
subplot(2,1,2);plot(om/pi,an);
title('phase response of IIR filter is:');
xlabel('(b) Normalized freq. -->');
ylabel('Phase in radians-->');
```

Verify and plot the sampling theorem in MATLAB

```
clc;
clear all;
t=0:00.001:1;
fm=input('Enter the modulating signal frequency :');
x=sin(2*pi*fm*t);

fs1=input('Enter the sampling frequency < modulating signal :');
fs2=input('Enter the sampling frequency = modulating signal :');
fs3=input('Enter the sampling frequency > modulating signal :');
% sampling at fs < 2fm
n=0:(1/fs1):1;
x1=sin(2*pi*fm*n);

% sampling at fs = 2fm
n=0:(1/fs2):1;
x2=sin(2*pi*fm*n);

% sampling at fs > 2fm
n=0:(1/fs3):1;
x3=sin(2*pi*fm*n);

% Plotting signals
subplot(2,2,1);
    plot(t,x);
xlabel('time');
ylabel('amplitude');
title('MESSAGE SIGNAL');
subplot(2,2,2);
    stem(n,x1);
xlabel('time');
ylabel('amplitude');
```

```
title(' fs < 2fm ');
subplot(2,2,3);
    stem(n,x2);
xlabel('time');
ylabel('amplitude');
title(' fs = 2fm ');
subplot(2,2,4);
    stem(n,x3);
xlabel('time');
ylabel('amplitude');
title(' fs > 2fm ');
```

OUTPUT

Enter the modulating signal frequency :10

Enter the sampling frequency < modulating signal :3

Enter the sampling frequency = modulating signal :10

Enter the sampling frequency > modulating signal :100

Verify Paravel's theorem in MATLAB

```
%Program to verify the Parseval's theorem
clear all
close all
clc
N=5000;
x=randi(10,N,1)+1i.*randi(10,N,1);
y=randi(10,N,1)+1i.*randi(10,N,1);
%To find x(n).y*(n)
yc=conj(y);
disp('Left hand side of Parseval's theorem')
lhs=sum(x.*yc)
X=fft(x);
Y=fft(y);
Yc=conj(Y);
%To find (1/N).X(n).Y*(n)
disp('right hand side of Parseval's theorem')
rhs=(1/N).*sum(X.*Yc)
```

Implement the overlap add block convolution method in MATLAB for N=10

```
% Overlap Add method for Linear Convolution
close All
```

```
clear All
clc
N=input('Enter the length of x(n) : ');
x=rand(1,N);        % Random N Numbers
h=input('Enter the values of h(n)=');
L=length(h);
N1=length(x);
M=length(h);
lc=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 stage=1:S
    xm=[x(index) zeros(1,M-1)];     % Selecting sequence to process
    X1=fft(xm,L+M-1);
    Y=X1.*H;
    Y=ifft(Y);
    Z=X((length(X)-M+2):length(X))+Y(1:M-1);    %Samples Added in
every stage
    X=[X(1:(stage-1)*L) Z Y(M:M+L-1)];
    index=stage*L+1:(stage+1)*L;
end
i=1:N1+M-1;
X=X(i);
figure()
subplot(2,1,1)
stem(lc);
title('Convolution Using inbuilt function')
xlabel('n');
ylabel('y(n)');
subplot(2,1,2)
stem(X);
title('Convolution Using Overlap Add Method')
```

```
xlabel('n');
ylabel('y(n)');
```

Implement the overlap save block convolution method in MATLAB for N=10

```
% Overlap Save method for Linear Convolution
close All
close All
clear All
clc
N=input('Enter the length of x(n) : ');
x=rand(1,N);        % Random N Numbers
h=input('Enter the values of h(n)=');
L=length(h);
N1=length(x);
M=length(h);
lc=conv(x,h);
x=[x zeros(1,mod(-N1,L)) zeros(1,L)];
N2=length(x);
h=[h zeros(1,L-1)];
H=fft(h,L+M-1);
S=N2/L;
index=1:L;
xm=x(index);        % For first stage Special Case
x1=[zeros(1,M-1) xm];   %zeros appeded at Start point
X=[];
for stage=1:S
    X1=fft(x1,L+M-1);
    Y=X1.*H;
    Y=ifft(Y);
    index2=M:M+L-1;
    Y=Y(index2);        %Discarding Samples
    X=[X Y];
    index3=(((stage)*L)-M+2):((stage+1)*L);     % Selecting Sequence
to process
    if(index3(L+M-1)<=N2)
    x1=x(index3);
    end
end;
i=1:N1+M-1;
X=X(i);
figure()
subplot(2,1,1)
stem(lc);
title('Convolution Using inbuilt function')
xlabel('n');
ylabel('y(n)');
```

```
subplot(2,1,2)
stem(X);
title('Convolution Using Overlap Save Method')
xlabel('n');
ylabel('y(n)');
```