1. Generation of Sequence :

```
clc;
clear all;
close all;
disp('Program for Waveform Generation');
opt = 1; % Initialize the option for the while loop
while opt == 1
   % Display the menu options
   disp('Which waveform do you want to generate?');
   disp('1. Impulse, 2. Step, 3. Ramp, 4. Exponential, 5. Sine, 6. Cosine, 7.
Triangle, 8. Sawtooth, 9. Random Signal');

   % Take user input for the type of waveform
   k = input('Enter your choice: ');

   % Switch case for waveform selection
   switch k
       % Impulse Waveform
       case 1
           n = -20:20; % Range of discrete-time
           x = (n == 0); % Impulse is 1 only at n=0, else 0
           subplot(5, 2, 1);
           stem(n, x, 'k', 'LineWidth', 1.5);
           xlabel('n -->');
           ylabel('Amplitude');
           title('Unit Impulse Signal');

       % Step Waveform
       case 2
           n = -20:20; % Range of discrete-time
           x = (n >= 0); % Step is 1 for n >= 0, else 0
           subplot(5, 2, 2);
           stem(n, x, 'k', 'LineWidth', 1.5);
           xlabel('n -->');
           ylabel('Amplitude');
           title('Unit Step Signal');

       % Ramp Waveform
       case 3
           n = -20:20; % Range of discrete-time
           x = max(0, n); % Ramp is n for n >= 0, else 0
           subplot(5, 2, 3);
           stem(n, x, 'k', 'LineWidth', 1.5);
           xlabel('n -->');
           ylabel('Amplitude');
           title('Ramp Signal');
```

```matlab
    % Exponential Waveform
    case 4
        n = -20:20; % Range of discrete-time
        x = exp(n .* (n >= 0)); % Exponential growth for n >= 0, else 0
        subplot(5, 2, 4);
        stem(n, x, 'k', 'LineWidth', 1.5);
        xlabel('n -->');
        ylabel('Amplitude');
        title('Exponential Signal');

    % Sine Waveform
    case 5
        n = 0:(pi/32):(4*pi); % Range of continuous-time
        x = sin(n); % Sine signal
        subplot(5, 2, 5);
        plot(n, x, 'k', 'LineWidth', 1.5);
        xlabel('n -->');
        ylabel('Amplitude');
        title('Sine Signal');

    % Cosine Waveform
    case 6
        n = 0:(pi/32):(4*pi); % Range of continuous-time
        x = cos(n); % Cosine signal
        subplot(5, 2, 6);
        plot(n, x, 'k', 'LineWidth', 1.5);
        xlabel('n -->');
        ylabel('Amplitude');
        title('Cosine Signal');

    % Triangular Waveform
    case 7
        n = 0:0.2:20; % Range of continuous-time
        x = sawtooth(n, 0.5); % Triangular signal
        subplot(5, 2, 7);
        plot(n, x, 'k', 'LineWidth', 1.5);
        xlabel('n -->');
        ylabel('Amplitude');
        title('Triangular Signal');

    % Sawtooth Waveform
    case 8
        n = 0:0.2:20; % Range of continuous-time
        x = sawtooth(n); % Sawtooth signal
        subplot(5, 2, 8);
        plot(n, x, 'k', 'LineWidth', 1.5);
        xlabel('n -->');
        ylabel('Amplitude');
        title('Sawtooth Signal');
```

```
        % Random Signal
        case 9
            r = 1:10; % 10 random values
            x = rand(1, 10); % Generate random numbers
            subplot(5, 2, [9, 10]);
            stem(r, x, 'k', 'LineWidth', 1.5);
            xlabel('Index -->');
            ylabel('Amplitude');
            title('Random Signal');

        % Invalid Choice
        otherwise
            disp('Invalid choice. Please select a valid option.');
    end

    % Ask the user if they want to continue
    disp('Do you want to continue?');
    opt = input('If YES, press 1: ');
end
```

2. a) Auto Correlation

```
clc
clear all
close all
x=input('enter the input sequence x')
c=xcorr(x,x) %correlation using the function 'xcorr'
subplot(2,1,1)
stem(x)
xlabel('n')
ylabel('x(n)')
title('input x')
disp('auto correlated sequence')
disp(c)
subplot(2,1,2)
stem(c)
xlabel('n')
ylabel('c(n)')
title('auto correlated sequence')
```

b) Cross Correlation :

```
clc
clear all
close all
x=input('enter the input sequence x');
y=input('enter the input sequence y');
m=length(x); %length of x
n=length(y); %length of x
if (m-n) ~= 0
if m>n
y=[y zeros(1,(m-n))]; %append m-n number of zeros to the sequence 'y'
n=m;
else
x=[x zeros(1,(n-m))]; %append n-m number of zeros to the sequence 'x'
m=n;
end
end
c=xcorr(x,y); %correlation using the function 'xcorr'
subplot(3,1,1)
stem(x)
xlabel('n')
ylabel('x(n)')
title('input x')
subplot(3,1,2)
stem(y)
xlabel('n')
ylabel('y(n)')
title('input y')
disp('cross correlated sequence')
disp(c)
subplot(3,1,3)
stem(c)
xlabel('n')
ylabel('c(n)')
title('cross correlated sequence')
```

3. a) DFT & IDFT :

```
clc;
clear all;
close all;
x = input('Enter the sequence: ');
N = input('Enter the length: ');
% Zero-padding if N > length of x
if N > length(x)
   x = [x zeros(1, (N - length(x)))];
end
% Compute DFT
X = zeros(1, N); % Initialize DFT array
for k = 1:N
   for n = 1:N
       X(k) = X(k) + x(n) * exp(-j * (2 * pi / N) * (n - 1) * (k - 1));
   end
end
% Display DFT values
disp('DFT:');
disp(X);
% Plot input sequence
subplot(3, 1, 1);
stem(0:N-1, x, 'k');
xlabel('n ->');
ylabel('Amplitude');
title('Input Sequence');
% Magnitude response
mag_X = abs(X);
subplot(3, 1, 2);
stem(0:N-1, mag_X, 'k');
xlabel('n ->');
ylabel('Magnitude');
title('Magnitude Response');
% Phase response
phase_X = angle(X);
subplot(3, 1, 3);
stem(0:N-1, phase_X, 'k');
xlabel('n ->');
ylabel('Phase (radians)');
title('Phase Response');
% Compute IDFT
y = zeros(1, N); % Initialize IDFT array
for n = 1:N
   for k = 1:N
       y(n) = y(n) + (1 / N) * X(k) * exp(j * (2 * pi / N) * (n - 1) * (k -
1));
   end
end
```

```
% Display IDFT values
disp('IDFT:');
disp(y);
```

b) FFT & IFFT :

```
clc
clear all
close all
x=input('enter the sequence')
N=input('enter the length')
X=fft(x)
subplot(3,1,1) ;
stem(x,'k')
xlabel('time->')
ylabel('amp->')
title('input->')
mag_X=abs(X)
subplot(3,1,2)
stem(mag_X,'k')
xlabel('time->')
ylabel('amp->')
title('Magnitude response->')
phase_X=angle(X)
subplot(3,1,3)
stem(phase_X,'k')
xlabel('time->')
ylabel('amp->')
title('Phase response->')
y=ifft(X)
```

4. a) LINEAR CONVOLUTION :

```
clc
clear all
close all
x=input('enter the input sequence')
h=input('enter the impulse response')
l=length(x)+length(h)-1
y=conv(x,h)
subplot(3,1,1)
stem(x,'k')
xlabel('time->')
ylabel('amp->')
title('input->')
subplot(3,1,2)
stem(h,'k')
xlabel('time->')
ylabel('amp->')
title('impulse->')
subplot(3,1,3)
stem(y,'k')
xlabel('time->')
ylabel('amp->')
title('linear convolution->')
```

b) CIRCULAR CONVOLUTION USING FFT :

```
clc
clear all
close all
x=input('enter then input sequence');
h=input('enter the impulse response');
l1=length(x);
l2=length(h);
l3=max(l1,l2);
X=fft(x);
H=fft(h);
for i=1:1:l3
Y(i)=X(i)*H(i);
end
y=ifft(Y);
disp(y)
subplot(3,1,1)
stem(x,'k')
xlabel('n->')
ylabel('amp->')
title('input')
```

```
subplot(3,1,2)
stem(h,'k')
xlabel('n->')
ylabel('amp->')
title('impulse response')
subplot(3,1,3)
stem(y,'k')
xlabel('n->')
ylabel('amp->')
title('circular convolution using fft')
```

c) CIRCULAR CONVOLUTION USING BUILT IN FUNCTION :

```
clc
clear all
close all
x=input('enter then input sequence');
h=input('enter the impulse response');
l1=length(x);
l2=length(h);
l3=max(l1,l2);
X=fft(x);
H=fft(h);
for i=1:1:l3
Y(i)=X(i)*H(i);
end
y=ifft(Y);
disp(y)
subplot(3,1,1)
stem(x,'k')
xlabel('n->')
ylabel('amp->')
title('input')
subplot(3,1,2)
stem(h,'k')
xlabel('n->')
ylabel('amp->')
title('impulse response')
subplot(3,1,3)
stem(y,'k')
xlabel('n->')
ylabel('amp->')
title('circular convolution using fft')
```

5. a) DESIGN OF ANALOG IIR BUTTERWORTH LPF :

```
%Program for Butterworth IIR Lowpass analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Lowpass analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp=input('Enter the pass edge frequency: ');
fs=input('Enter the stop edge frequency: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs.
fs_min = 2*fs;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency; digital omega = analog omega* Ts = analog omega/fs_sf
wp=2*pi*fp/fs_sf;
ws=2*pi*fs/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nwp is %d\n',wp);
fprintf('ws is %d\n',ws);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using butter command with 's'
%option
[N wc]=buttord(wp,ws,rp1,rs1,'s');
[b a]=butter(N,wc,'s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till
pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
hf=tf(b,a)
disp(hf)
[z p]=tf2zp(b,a)
%Finding the magnitude response. Note: log10 should be used.
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
```

```
ylabel('Gain in dB-->');
title('Magnitude Response of LPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of LPF');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of LPF')
```

b) HPF :

```
%Program for Butterworth IIR Highpass analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Highpass analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fs=input('Enter the stop edge frequency: ');
fp=input('Enter the pass edge frequency: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fp.
fs_min = 2*fp;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
wp=2*pi*fp/fs_sf;
ws=2*pi*fs/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nws is %d\n',ws);
fprintf('wp is %d\n',wp);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
```

```
%Finding the coefficients of filter [b a] using butter command with 'high' and
's' option.
[N wc]=buttord(wp,ws,rp1,rs1,'s');
[b a]=butter(N,wc,'high','s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
hf=tf(b,a)
disp(hf)
[z p]=tf2zp(b,a)
%Finding the magnitude response. Note: log10 should be used.
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of HPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of HPF');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of HPF')
```

c) CHEBYSHEV BPF :

```
%Program for Chebtshev IIR Bandpass analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Bandpass analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fs1=input('Enter the stop edge frequency1: ');
fp1=input('Enter the pass edge frequency1: ');
fp2=input('Enter the pass edge frequency2: ');
fs2=input('Enter the stop edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs2.
fs_min = 2*fs2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
ws1=2*pi*fs1/fs_sf;
wp1=2*pi*fp1/fs_sf;
wp2=2*pi*fp2/fs_sf;
ws2=2*pi*fs2/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nws1 is %d\n',ws1);
fprintf('wp1 is %d\n',wp1);
fprintf('wp2 is %d\n',wp2);
fprintf('ws2 is %d\n',ws2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the cheb1ord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using cheby command with 's' option.
wp = [wp1 wp2];
ws = [ws1 ws2];
[N wc]=cheb1ord(wp,ws,rp1,rs1,'s');
[b a]=cheby1(N,rp,wc,'s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
hf=tf(b,a)
disp(hf)
[z p]=tf2zp(b,a)
%Finding the magnitude response. Note: log10 should be used.
```

```
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of BPF');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of BPF')
```

d) CHEBYSHEV BANDSTOP :

```
%Program for Butterworth IIR Bandstop analog filter
clc;
close all;
clear all;
fprintf('Program for Butterworth IIR Bandstop analog filter\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp1=input('Enter the pass edge frequency1: ');
fs1=input('Enter the stop edge frequency1: ');
fs2=input('Enter the stop edge frequency2: ');
fp2=input('Enter the pass edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fp2.
fs_min = 2*fp2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
wp1=2*pi*fp1/fs_sf;
ws1=2*pi*fs1/fs_sf;
ws2=2*pi*fs2/fs_sf;
```

```
wp2=2*pi*fp2/fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nwp1 is %d\n',wp1);
fprintf('ws1 is %d\n',ws1);
fprintf('ws2 is %d\n',ws2);
fprintf('wp2 is %d\n',wp2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the cheb1ord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using cheby command with 'stop' and
's' option.
wp = [wp1 wp2];
ws = [ws1 ws2];
[N wc]=cheb1ord(wp,ws,rp1,rs1,'s');
[b a]=cheby1(N,rp,wc,'stop','s');
%Computing the frequency response using freqs command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqs(b,a,w);
hf=tf(b,a)
disp(hf)
[z p]=tf2zp(b,a)
%Finding the magnitude response. Note: log10 should be used.
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BRF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of BRF ');
subplot(3,1,3)
zplane(z,p,'k')
title('Pole Zero Plot of BRF')
```

6. a) DIGITAL LPF :

```
clc;
close all;
clear all;
fprintf('Program for Digital IIR Butterworth Low Pass Filter using Impulse
Invariant Transformation\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp=input('Enter the passband edge frequency: ');
fs=input('Enter the stopband edge frequency: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs.
fs_min = 2*fs;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency; digital omega = analog omega* Ts = analog omega/fs_sf
wp=2*pi*fp/fs_sf;
ws=2*pi*fs/fs_sf;
analog_wp=wp*fs_sf;
analog_ws=ws*fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nwp is %d\n',wp);
fprintf('ws is %d\n',ws);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
%Finding the coefficients [b a] of filter using butter command with 's'
option.
[N wc]=buttord(analog_wp,analog_ws,rp1,rs1,'s')
[b a]=butter(N,wc,'s');
%Finding the digital filter coefficients [c d] using impinvar command.
[c d]=impinvar(b,a,fs_sf);
%Computing the frequency response using freqz command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqz(c,d,w);
disp('Analog Filter Transfer Function - Unnormlised: ')
hf=tf(b,a)
disp('Digital Filter Transfer Function: ')
hf1=tf(c,d,fs_sf)
disp(hf)
disp(hf1)
[z p]=tf2zp(c,d)
```

```
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of LPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of LPF');
%Plotting poles in the Z plane.
subplot(3,1,3)
zplane(z,p)
title('Pole Zero Plot of LPF')
```

b) HPF :

c) BPF USING IMPULSE INVARIANT INVARIANT TRANSFORMATION :

```
clc;
close all;
clear all;
fprintf('Program for Digital IIR Butterworth Band Pass Filter using Impulse
Invariant Transformation\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fs1=input('Enter the stop edge frequency1: ');
fp1=input('Enter the pass edge frequency1: ');
fp2=input('Enter the pass edge frequency2: ');
fs2=input('Enter the stop edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs2.
fs_min = 2*fs2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
```

```
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
ws1=2*pi*fs1/fs_sf;
wp1=2*pi*fp1/fs_sf;
wp2=2*pi*fp2/fs_sf;
ws2=2*pi*fs2/fs_sf;
analog_ws1=ws1*fs_sf;
analog_wp1=wp1*fs_sf;
analog_wp2=wp2*fs_sf;
analog_ws2=ws2*fs_sf;
%The normalised frequencies are wp and ws
fprintf('\nws1 is %d\n',ws1);
fprintf('wp1 is %d\n',wp1);
fprintf('wp2 is %d\n',wp2);
fprintf('ws2 is %d\n',ws2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
%the buttord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using butter command with 's'
option.
wp = [analog_wp1 analog_wp2];
ws = [analog_ws1 analog_ws2];
[N wc]=buttord(wp,ws,rp1,rs1,'s');
[b a]=butter(N,wc,'s');
%Finding the digital filter coefficients [c d] using impinvar command.
[c d]=impinvar(b,a,fs_sf);
%Computing the frequency response using freqz command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqz(c,d,w);
disp('Analog Filter Transfer Function - Unnormlised: ')
hf=tf(b,a)
disp('Digital Filter Transfer Function: ')
hf1=tf(c,d,fs_sf)
disp(hf)
disp(hf1)
[z p]=tf2zp(c,d)
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BPF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
```

```
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of BPF');
%Plotting poles in the Z plane.
subplot(3,1,3)
zplane(z,p)
title('Pole Zero Plot of BPF')
```

d) BSF USING BILINEAR TRANSFORMATION :

```
clc;
close all;
clear all;
fprintf('Program for Digital IIR Chebyshev Band Stop Filter using Bilinear
Transformation\n\n');
%We get the passedge, stopedge and sampling frequencies in Hz
fp1=input('Enter the pass edge frequency1: ');
fs1=input('Enter the stop edge frequency1: ');
fs2=input('Enter the stop edge frequency2: ');
fp2=input('Enter the pass edge frequency2: ');
%fs_min should be twice the maximum frequency. Here, fs_min = 2*fs2.
fs_min = 2*fp2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency: ');
%We get the attenuation in dB. rp will be around 0 to 3 dB
%rs will be around 30 to 50 dB
rp = input('\nEnter the passband ripple in dB: ');
rs = input('Enter the stopband attenuation in dB: ');
rp1=20*log10(rp)
rs1=20*log10(rs)
%We need to normalise wp,ws to pi. It is similar to finding the digital
%frequency digital omega = analog omega* Ts = analog omega/fs_sf
wp1=2*pi*fp1/fs_sf;
ws1=2*pi*fs1/fs_sf;
ws2=2*pi*fs2/fs_sf;
wp2=2*pi*fp2/fs_sf;
analog_wp1=2*fs_sf*(tan(wp1/2));
analog_ws1=2*fs_sf*(tan(ws1/2));
analog_ws2=2*fs_sf*(tan(ws2/2));
analog_wp2=2*fs_sf*(tan(wp2/2));
%The normalised frequencies are wp and ws
fprintf('\nws1 is %d\n',ws1);
fprintf('wp1 is %d\n',wp1);
fprintf('wp2 is %d\n',wp2);
fprintf('ws2 is %d\n',ws2);
%Computing the order(N) and cutoff frequency(wc) using wp,ws,rp,rs using
```

```
%the cheb1ord command with 's' option for analog filter.
%Finding the coefficients of filter [b a] using cheby command with 's' and
'stop' option.
wp = [analog_wp1 analog_wp2];
ws = [analog_ws1 analog_ws2];
[N wc]=cheb1ord(wp,ws,rp1,rs1,'s');
[b a]=cheby1(N,rp,wc,'stop','s');
%Finding the digital filter coefficients [c d] using bilinear command.
[c d]=bilinear(b,a,fs_sf);
%Computing the frequency response using freqz command
%Computing h for specific values of w i.e. 0, pi/100, 2*pi/100... till pi
%and storing the corresponding the w values
w=0:(pi/100):pi;
[h omega] = freqz(c,d,w);
disp('Analog Filter Transfer Function - Unnormlised: ')
hf=tf(b,a)
disp('Digital Filter Transfer Function: ')
hf1=tf(c,d,fs_sf)
disp(hf)
disp(hf1)
[z p]=tf2zp(c,d)
%Plotting magnitude versus omega.
mag_h=abs(h);
figure(1);
subplot(3,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BRF');
%Finding the phase response.
%Plotting phase versus omega.
angle_h = angle(h);
subplot(3,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase in radian-->');
title('Phase Response of BRF');
%Plotting poles in the Z plane.
subplot(3,1,3)
zplane(z,p)
title('Pole Zero Plot of BRF')
```

7. LPF USING HANNING WINDOW :

```
clc
close all
clear all
fprintf('Program for FIR Low Pass filter using windowing technique\n\n');
N =input('Enter the order of the filter: ');
fc=input('Enter the cut off frequency: ');
fs_min = 2 * fc;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency:');
wc=2*pi*fc / fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
if (n - 1) == alp
hd(n) = wc / pi
else
hd(n) = (sin((n-1-alp) * wc)) / (pi*(n-1-alp))
end
hannwin(n) = 0.5 - 0.5 * (cos(2*pi*n) / (N-1));
end
hw = hd .* hannwin;
[h omega] = freqz(hw,1,50);
mag_h = abs(h);
figure(1);
subplot(2,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of LPF');
angle_h = angle(h);
subplot(2,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase-->');
title('Phase Response of LPF');
```

b) FIR HPF USING RECTANGULAR TECHNIQUE :

```
clc
close all
clear all
fprintf('Program for FIR High pass filter using windowing technique\n\n');
N =input('Enter the order of the filter: ');
fc=input('Enter the cut off frequency: ');
fs_min = 2 * fc;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency:');
wc=2*pi*fc / fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
if (n - 1) == alp
hd(n) = (pi - wc) / pi
else
hd(n)=(sin((n-1-alp)*pi))-(sin(n-1-alp)*wc)/(pi*(n-1-alp))
end
rect_win(n)=1
end
hw=hd .* rect_win
[h omega] = freqz(hw,1,50);
mag_h = abs(h);
figure(1);
subplot(2,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of HPF');
angle_h = angle(h);
subplot(2,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase-->');
title('Phase Response of HPF');
```

c) FIR BPF USING HAMMING :

```
clc;
clear all;
close all;
N =input('Enter the order of the filter: ');
fs1=input('Enter the stop edge frequency1:');
fc1=input('Enter the pass edge frequency1:');
fc2=input('Enter the pass edge frequency2:');
fs2=input('Enter the stop edge frequency2:');
fs_min = 2*fs2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency:');
rp = input('\nEnter the passband ripple in dB:');
rs = input('Enter the stopband attenuation in dB:');
ws1=2*pi*fs1/fs_sf;
wc1=2*pi*fc1/fs_sf;
wc2=2*pi*fc2/fs_sf;
ws2=2*pi*fs2/fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
if (n - 1) == alp
hd(n) = (wc2-wc1) / pi
else
hd(n)=(sin((n-1-alp)*wc2))-(sin(n-1-alp)*wc1)/(pi*(n-1-alp))
hammwin(n) = 0.54 - 0.46 * (cos(2*pi*n) / (N-1));
end
end
hw = hd .* hammwin;
[h omega] = freqz(hw,1,50);
mag_h = abs(h);
figure(1);
subplot(2,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BPF');
angle_h = angle(h);
subplot(2,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase-->');
title('Phase Response of BPF');
```

d) BPF USING BLACKMAN WINDOW :

```
clc;
clear all;
close all;
N =input('Enter the order of the filter:');
fp1=input('Enter the pass edge frequency1:');
fc1=input('Enter the stop edge frequency1:');
fc2=input('Enter the stop edge frequency2:');
fp2=input('Enter the pass edge frequency2:');
fs_min = 2*fp2;
fprintf('\nEnter the sampling frequency greater than %d\n',fs_min');
fs_sf=input('Enter the sampling frequency:');
rp = input('\nEnter the passband ripple in dB:');
rs = input('Enter the stopband attenuation in dB:');
wp1=2*pi*fp1/fs_sf;
wc1=2*pi*fc1/fs_sf;
wc2=2*pi*fc2/fs_sf;
wp2=2*pi*fp2/fs_sf;
alp = (N-1)/2;
for n = 1 : 1 : N
if (n - 1) == alp
hd(n) =1 - ((wc2-wc1 )/ pi)
else
hd(n)=((sin((n-1-alp)*wc1))-(sin(n-1-alp)*wc2)+
sin((n-1-alp)*pi))/(pi*(n-1-alp))
blackwin(n) = 0.42 - 0.5 * (cos(2*pi*n) / (N-1)) + 0.08 * (cos(4*pi*n) /
(N-1));
end
end
hw = hd .* blackwin;
[h omega] = freqz(hw,1,50);
mag_h = abs(h);
figure(1);
subplot(2,1,1);
plot(omega/pi,mag_h);
xlabel('frequency normalised to 1 -->');
ylabel('Gain in dB-->');
title('Magnitude Response of BRF');
angle_h = angle(h);
subplot(2,1,2);
plot(omega/pi,angle_h)
xlabel('frequency normalised to 1 -->');
ylabel('Phase-->');
title('Phase Response of BRF');
```

8. a) LPF USING FOURIER SERIES :

```
clc
clear all
close all
wc=.5*pi;
N=11;
hd=zeros(1,N);
hd(1)=wc/pi;
n = 1:1:((N-1)/2)+1;
hd(n+1) = (sin(wc*n))  / (pi*n);
hn(n) = hd(n)
a=(N-1)/2;
w=0: pi/16:pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;
for m = 1:1:a;
Hw3 = hn(m+1) * ((exp(j*w*(m-a)) + exp(-j*w*(m+a)));
Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1
mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Magnitude -->');
title('Magnitude Response of LPF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of LPF');
```

b) HPF :

```
clc
clear all
close all
wc=.6*pi;
N=7;
hd=zeros(1,N);
hd(1)=1-(wc/pi);
n = 1:1:((N-1)/2)+1;
hd(n+1) = (-sin(wc*n))  / (pi*n);
hn(n) = hd(n)
```

```
a=(N-1)/2;
w=0: (pi/16):pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;
for m = 1:1:a;
Hw3 = hn(m+1) * ((exp(j*w*(m-a))) + exp(-j*w*(m+a)));
Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1
mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Magnitude -->');
title('Magnitude Response of HPF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of HPF');
```

c) BPF :

```
clc
clear all
close all
wc1=.375*pi;
wc2=.75*pi;
N=7;
hd=zeros(1,N);
hd(1)=(wc2-wc1)/pi;
n = 1:1:((N-1)/2)+1;
hd(n+1) = ((sin(wc2*n))-(sin(wc1*n)))  / (pi*n);
hn(n) = hd(n)
a=(N-1)/2;
w=0: (pi/16):pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;
for m = 1:1:a;
Hw3 = hn(m+1) * ((exp(j*w*(m-a))) + exp(-j*w*(m+a)));
Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1
mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
```

```
ylabel('Magnitude -->');
title('Magnitude Response of BPF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of BPF')
```

d) BSF :

```
clc
clear all
close all
wc1=.375*pi;
wc2=.75*pi;
N=7;
hd=zeros(1,N);
hd(1)=1-((wc2-wc1)/pi);
n = 1:1:((N-1)/2)+1;
hd(n+1) = (((sin(wc1*n))-(sin(wc2*n)))  / (pi*n));
hn(n) = hd(n)
a=(N-1)/2;
w=0: (pi/16):pi;
Hw1=hn(1)*exp(-j*w*a);
Hw2=0;
for m = 1:1:a;
Hw3 = hn(m+1) * ((exp(j*w*(m-a))) + exp(-j*w*(m+a)));
Hw2 = Hw2 + Hw3;
end
Hw = Hw2 + Hw1
mag_h=abs(Hw)
subplot(2,1,1);
plot(w/pi,mag_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Magnitude -->');
title('Magnitude Response of BRF');
angle_h=angle(Hw);
subplot(2,1,2);
plot(w/pi,angle_h);
xlabel('Normalised Frequency,w/pi -->');
ylabel('Phase -->');
title('Phase Response of BRF');
```

DIRECT ADD MODE :

ADDRESS LABEL OPCODE COMMENT
C000 LDP #100H Load data page pointer with #100H
C001 LACC 0 Load accumulator with content of 8000H
C002 ADD 1 Add accumulator content with content of 8001H
C003 SACL 2 Store lower content of accumulator to 8002H
C004 SACH 3 Store higher content of accumulator to 8003H
C005 R: B R Terminate the program

INDIRECT :

C000 LACC #2345H Load accumulator with immediate value #2345
C002 LAR AR1,#8000H Load auxiliary register AR1 with 8000H
C004 LAR AR2,#10H Load auxiliary register AR2 with 10H
C006 L1: MAR *,1 Modify auxiliary register pointer to point AR1
C007 SACL *+,0,2 Store accumulator content specified by AR1.Increment AR1 by 1. Modify ARP to point to AR2
C008 BANZ L1,*- Decrement AR2 and Branch to L1 if AR2 ≠ 0
C009 H: B H Stop the program

IMMEDIATE :

C000 LDP #100H Load Data Page pointer with data #100H
C001 LACC #037AH,0 Load Accumulator with the data 037AH
C002 SACL 0 Store lower accumulator content to location 8000H
C003 LACC #012EH,0 Load accumulator with data 012EH
C005 LT 0 Load Treg0 with content of 8000H [037AH]
C006 MPY 1 Multiply Treg0 content with content of 8001H and store result in P reg
C007 PAC Move the content of Product register to accumulator
C008 SACL 2 Store lower accumulator content to location 8002H
C009 SACH 3 Store higher accumulator content to location 8003H
C00A R: B R Terminate the program