DSP: Filter Design Assignment

# Kalpesh Patil (130040019) Filter Number: 13

# Filter Specification

## Filter 1

* Nature: Bandpass Filter
* Cutoff Frequencies (in kHz)

Ω*s1* = 8700 Ω*p1* = 10700

Ω*p2* = 20700 Ω*s2* = 22700

## Filter 2

* Nature: Bandstop Filter
* Cutoff Frequencies (in kHz)

Ω*p1* = 8900 Ω*s1* = 10900

Ω*s2* = 20900 Ω*p2* = 22900

# Filter Design

## Filter 1

### IIR Implementation

Monotonic in Passband as well as Stopband

#### Digital Frequencies

ω*s1* = 0.5466 ω*p1* = 0.6723

ω*p2* = 1.3006 ω*s2* = 1.4263

#### Equivalent Analog Frequencies

Ω*s1* = 0.2803 Ω*p1* = 0.3494

Ω*p2* = 0.7607 Ω*s2* = 0.8650

#### Low Pass Equivalent Filter

Where  
 and

Stringent of the two values obtained from bandpass specifications is chosen as stop band frequency for low pass filter. Thus specifications of low pass filter needed to be deigned are as follows  
Monotonic response is needed in both stopband as well as passband, hence Butterworth Filter is designed

Considering the minimum order required, we obtain

Poles of this Butterworth Filter are as follows

Transfer function is given by

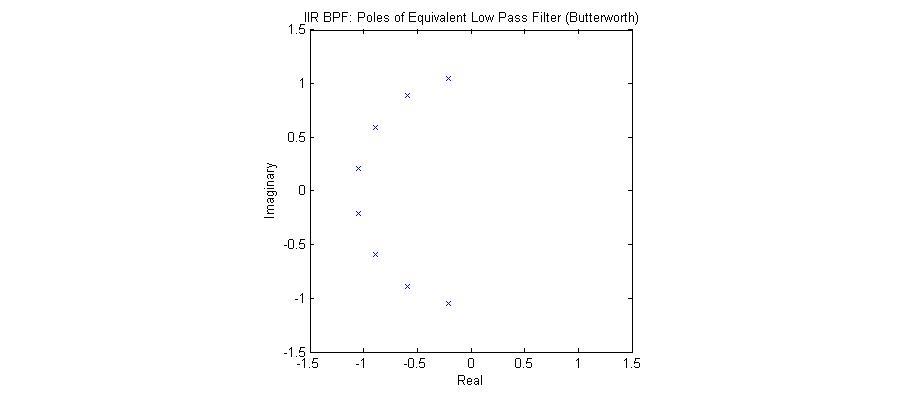


Fig1: Poles of Low Pass Equivalent (Butterworth) of IIR BPF

#### Analog Bandpass Filter

To obtain analog bandpass filter, following transformation was carried out in the transfer function

After transformation, following transfer function was obtained

#### Digital Filter

To obtain analog bandpass filter, following transformation was carried out in the transfer function

After transformation, following transfer function was obtained

Images for the pole-zero plot, magnitude response and phase response obtained using fvtool are shown below

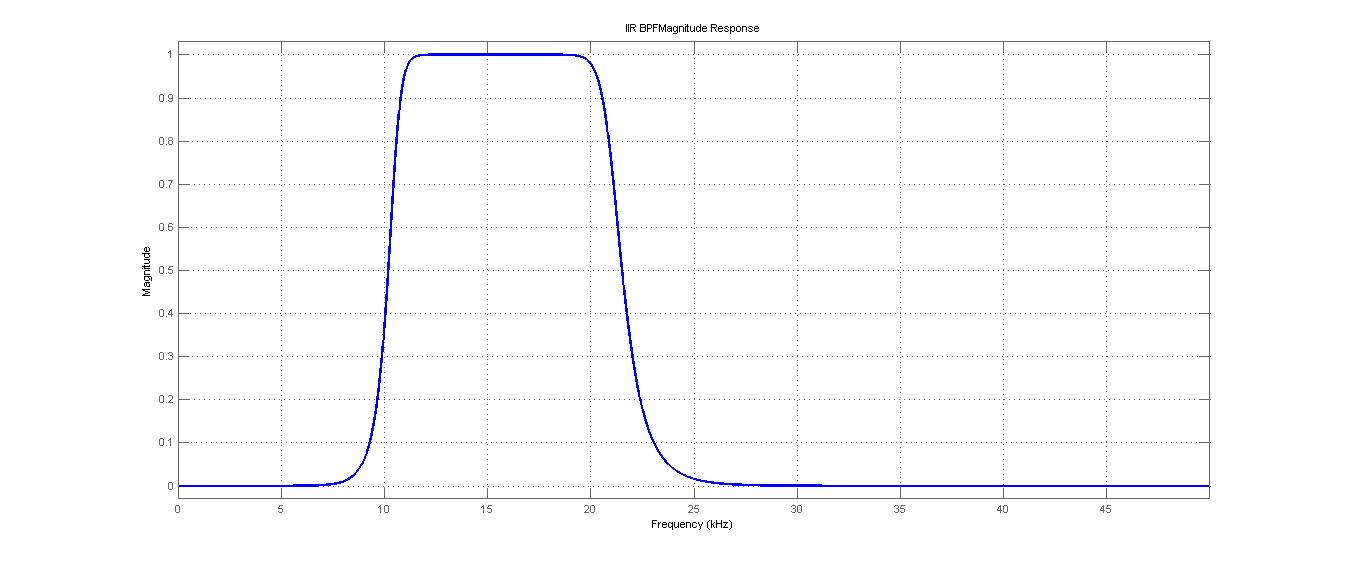
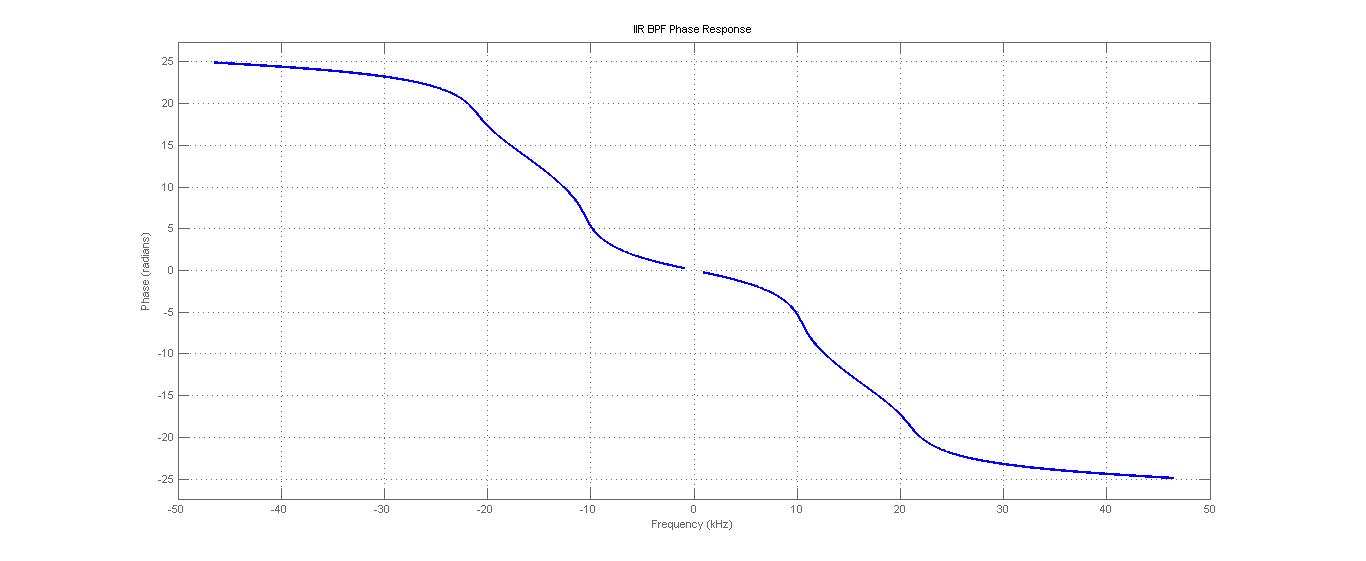
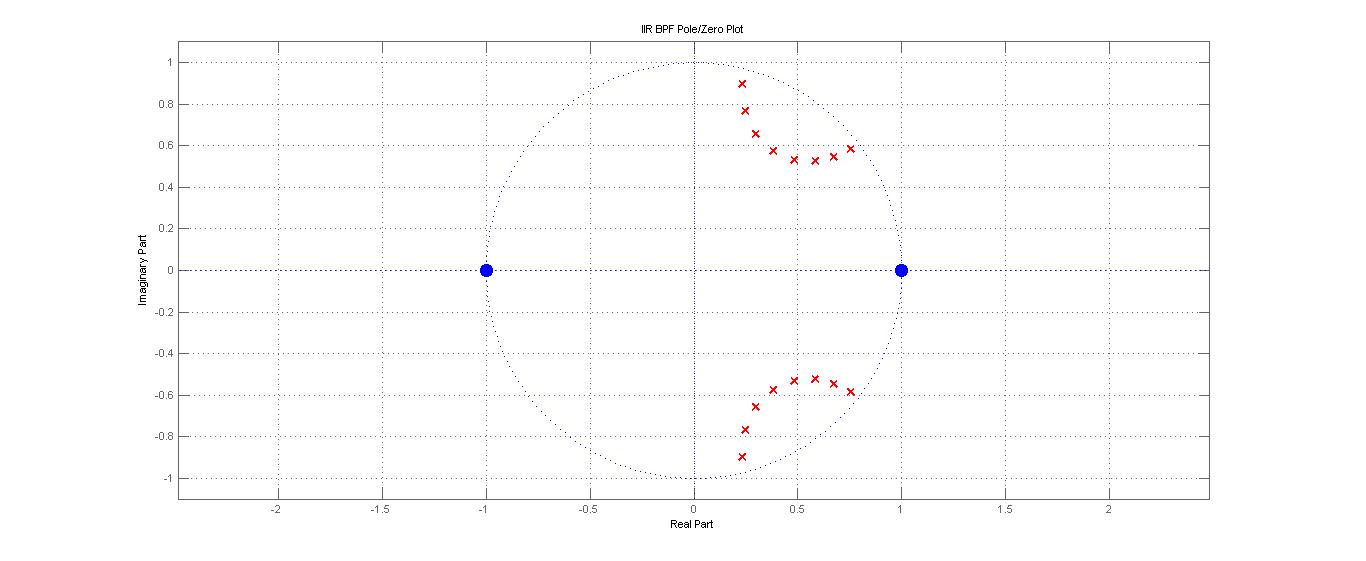


Fig2: Magnitude Response for IIR BPF

fig3: Phase Response for IIR BPF

fig4: Pole-Zero Plot for IIR BPF

#### Direct Form 2

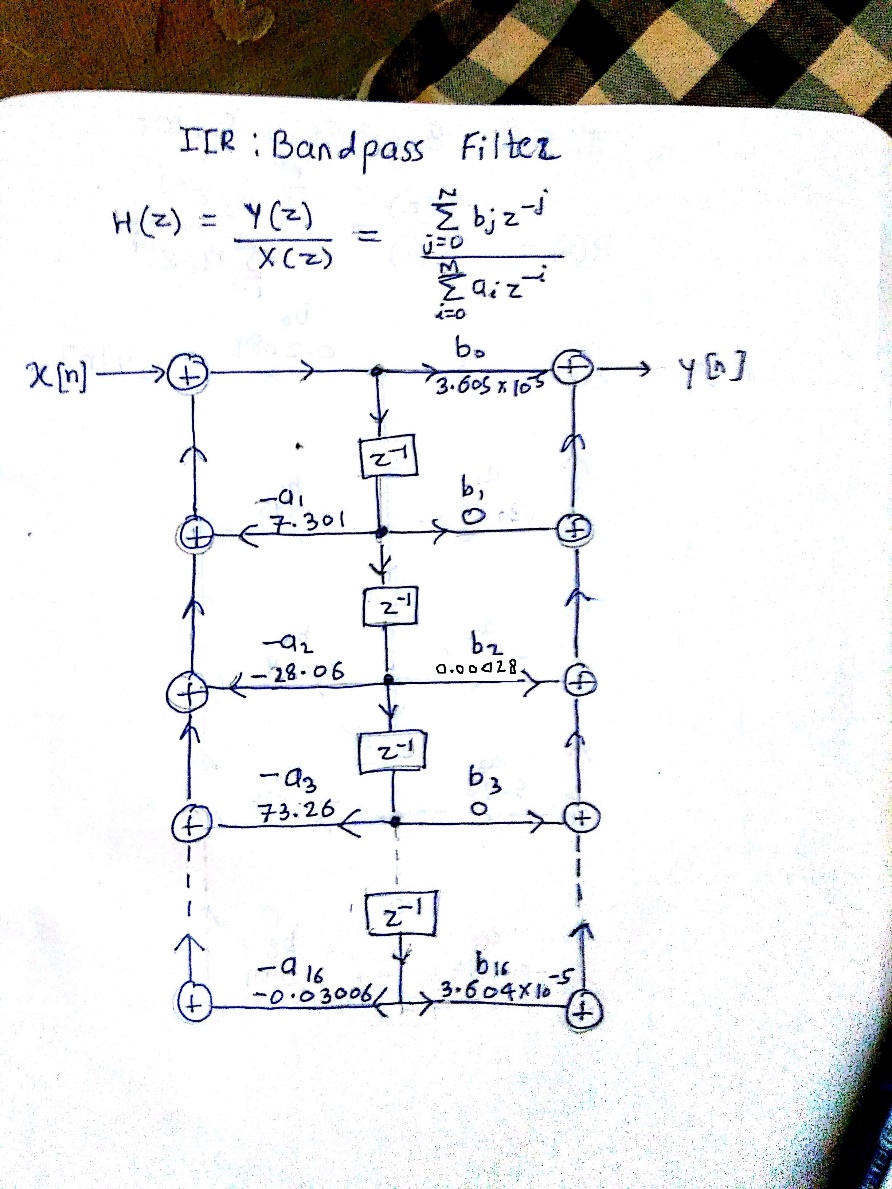


Fig5: Direct form 2 representation of IIR BPF

### FIR Implementation

#### Ideal Impulse Response

To truncate the ideal response we multiply it by Kaiser window. The length of this window and parameter is determined by using empirical formulae

#### Kaiser Window Parameters

where and By substituting values we get,   
We take to match specifications

#### Filter Response

Filter Coefficients [41] = [0.0193 0.0257 0.0078 -0.0023 0.0099 0.0153 -0.0123 -0.0465 - 0.0389 0.0070 0.0339 0.0150 -0.0004 0.0356 0.0764 0.0264 -0.1102 -0.1888 -0.0853 0.1293 0.2400 0.1293 -0.0853 -0.1888 -0.1102 0.0264 0.0764 0.0356 -0.0004 0.0150 0.0339 0.0070 -0.0389 -0.0465 -0.0123 0.0153 0.0099 -0.0023 0.0078 0.0257 0.0193]

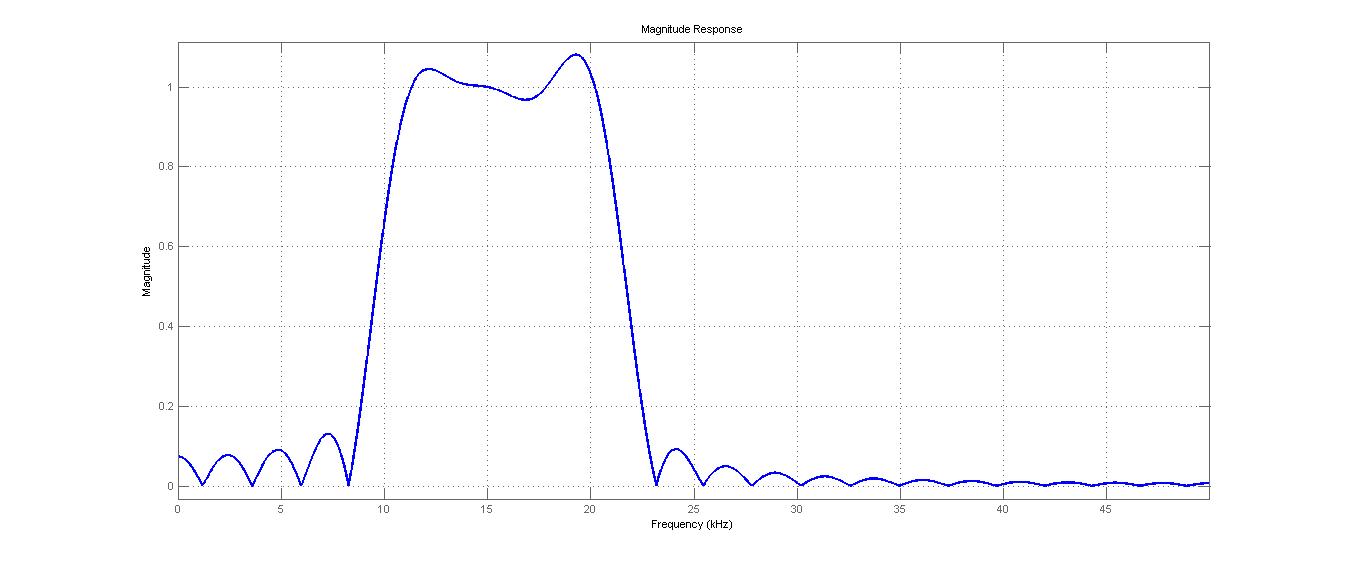


Fig6:Magnitude Response of FIR BPF

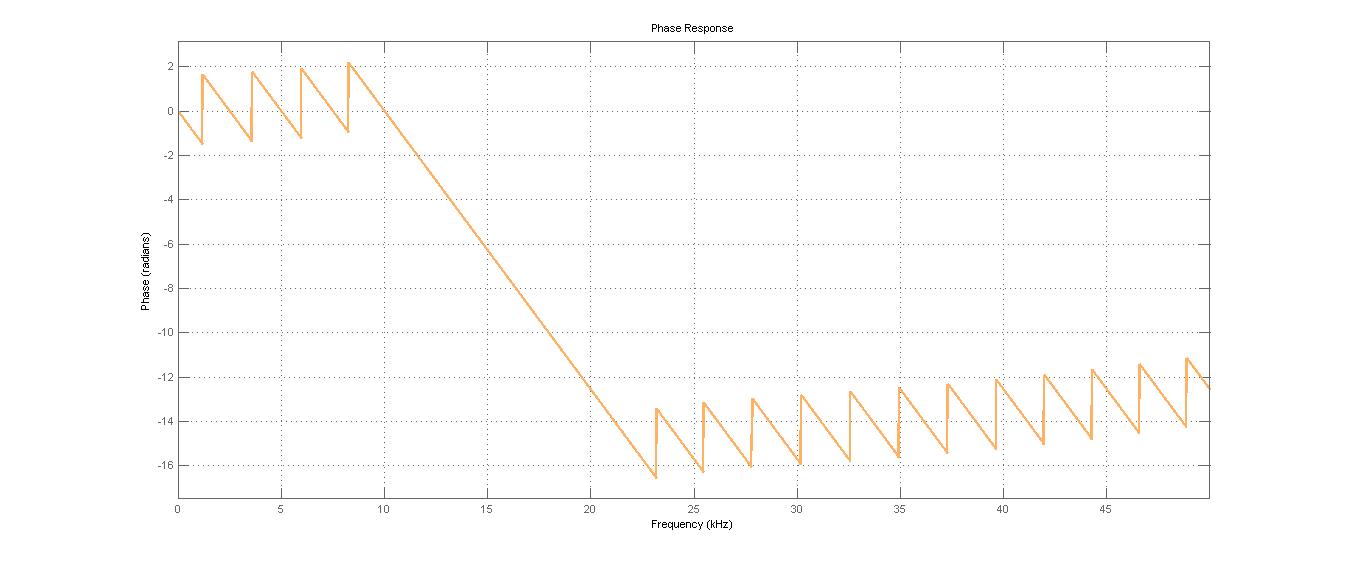


fig7:Phase Response for FIR BPF

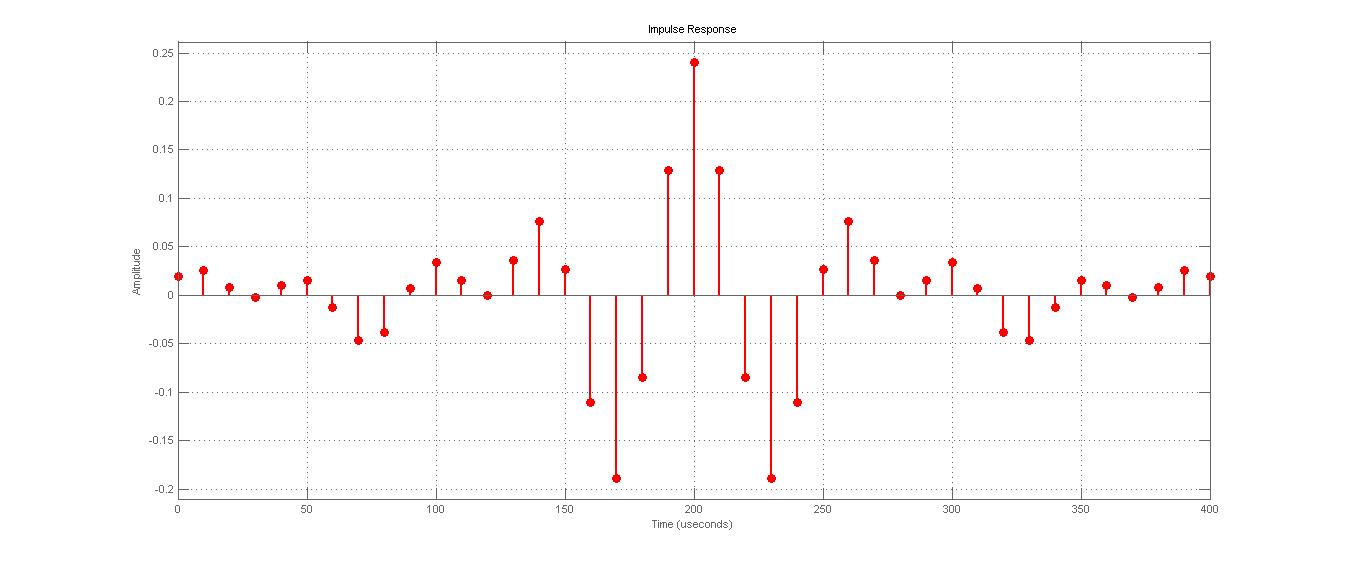


fig 8:Impulse Response for FIR BPF

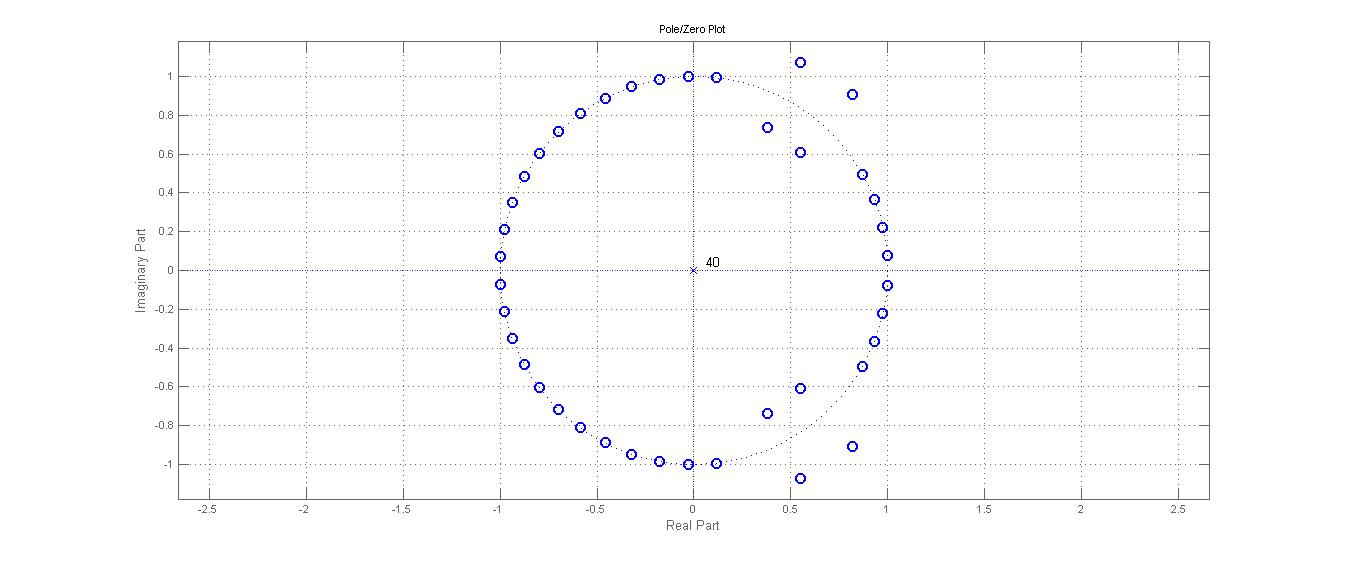


fig9:Zeros of FIR BPF

## Filter 2

### IIR Implementation

Equiripple in Passband and Monotonic Stopband

#### Digital Frequencies

ω*p1* = 0.5592 ω*s1* = 0.6849

ω*s2* = 1.3132 ω*p2* = 1.4388

#### Equivalent Analog Frequencies

Ω*p1* = 0.2871 Ω*s1* = 0.3565

Ω*s2* = 0.7707 Ω*p2* = 0.8761

#### Low Pass Equivalent Filter

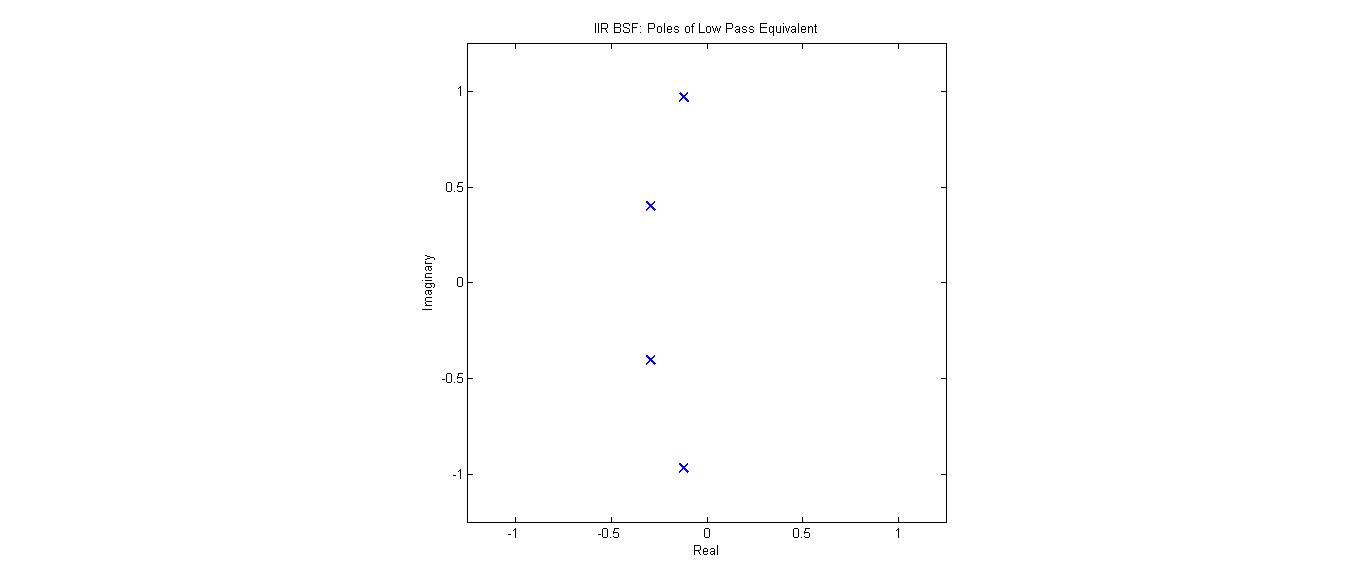
Where   
 and

Stringent of the two values obtained from bandpass specifications is chosen as stop band frequency for low pass filter. Thus specifications of low pass filter needed to be deigned are as follows  
Equiripple response is needed in passband and monotonic in stopband, hence Chebyshev Filter is designed

Considering the minimum order required, we obtain

Poles are given by

Transfer function is given by

Fig10: Poles of Low Pass Equivalent of IIR BSF (Chebyshev)

#### Analog Bandpass Filter

To obtain analog bandstop filter, following transformation was carried out in the transfer function

After transformation, following transfer function was obtained

#### Digital Filter

To obtain analog bandpass filter, following transformation was carried out in the transfer function

After transformation, following transfer function was obtained

Images for the pole-zero plot, magnitude response and phase response obtained using fvtool are shown below

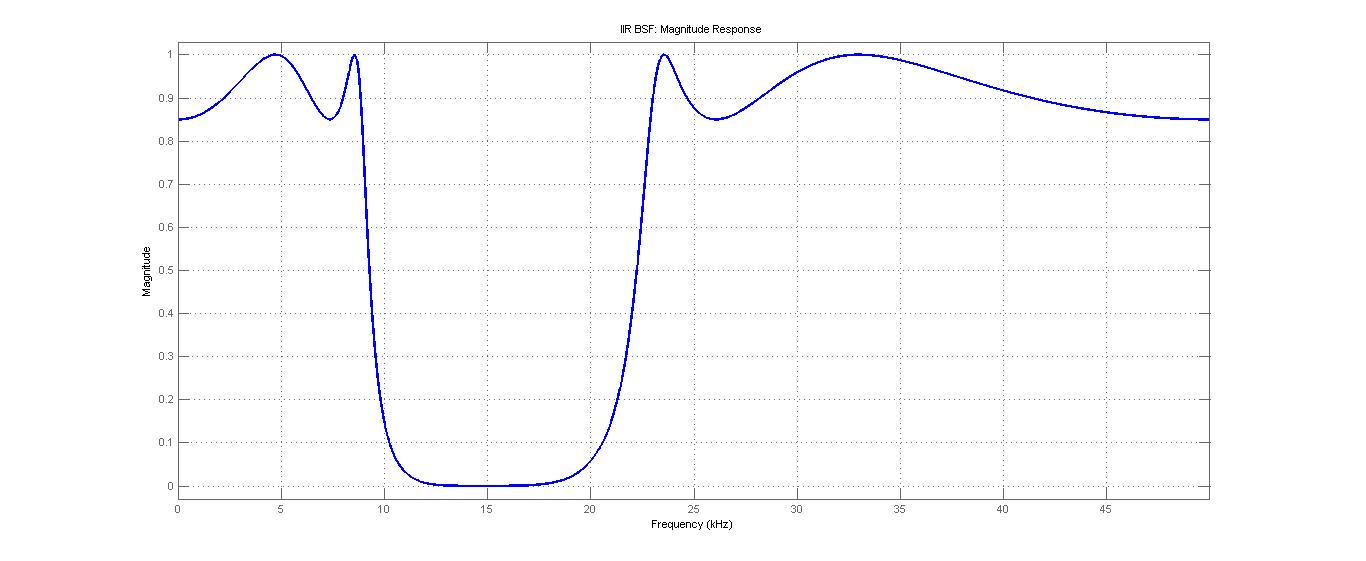
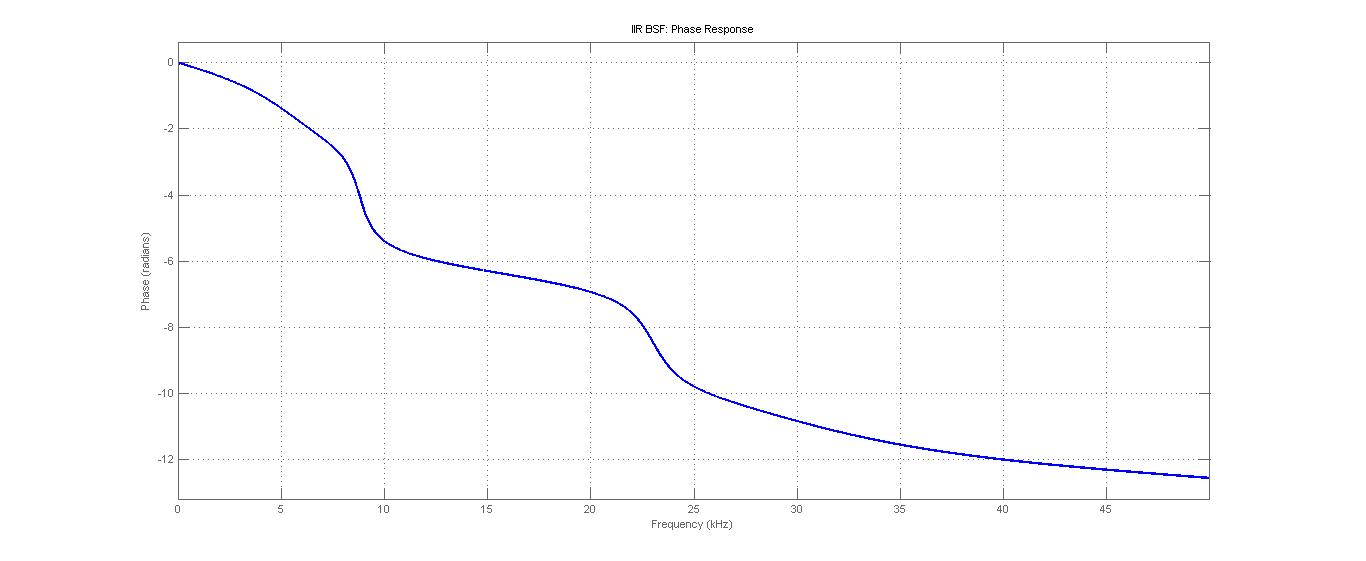
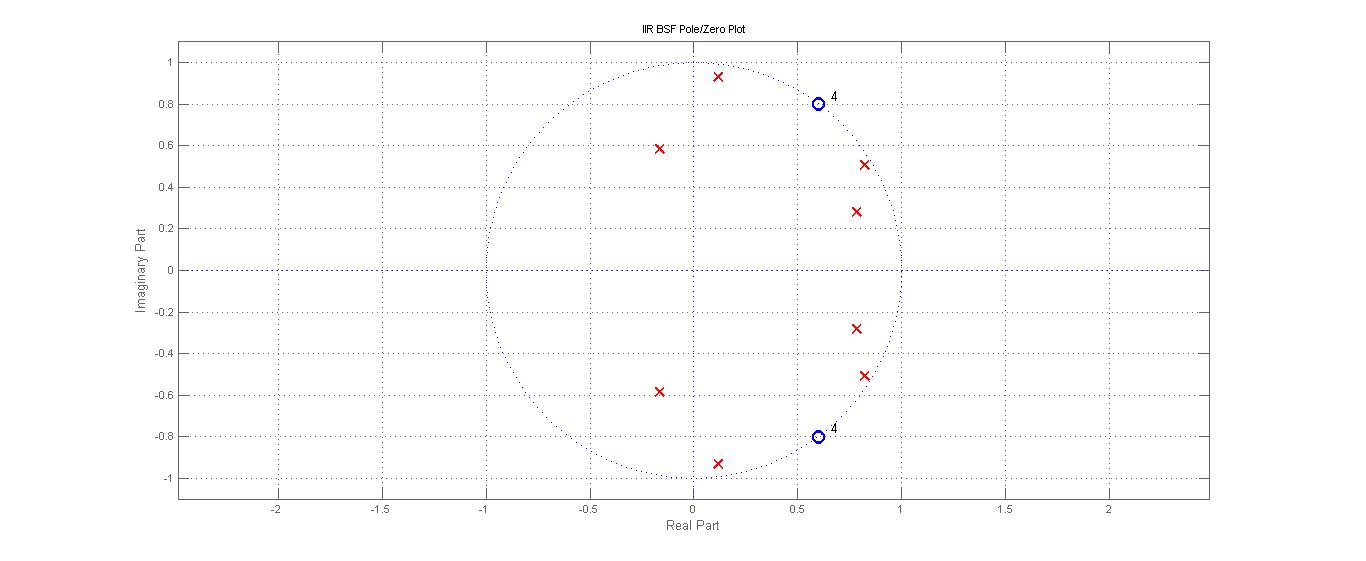
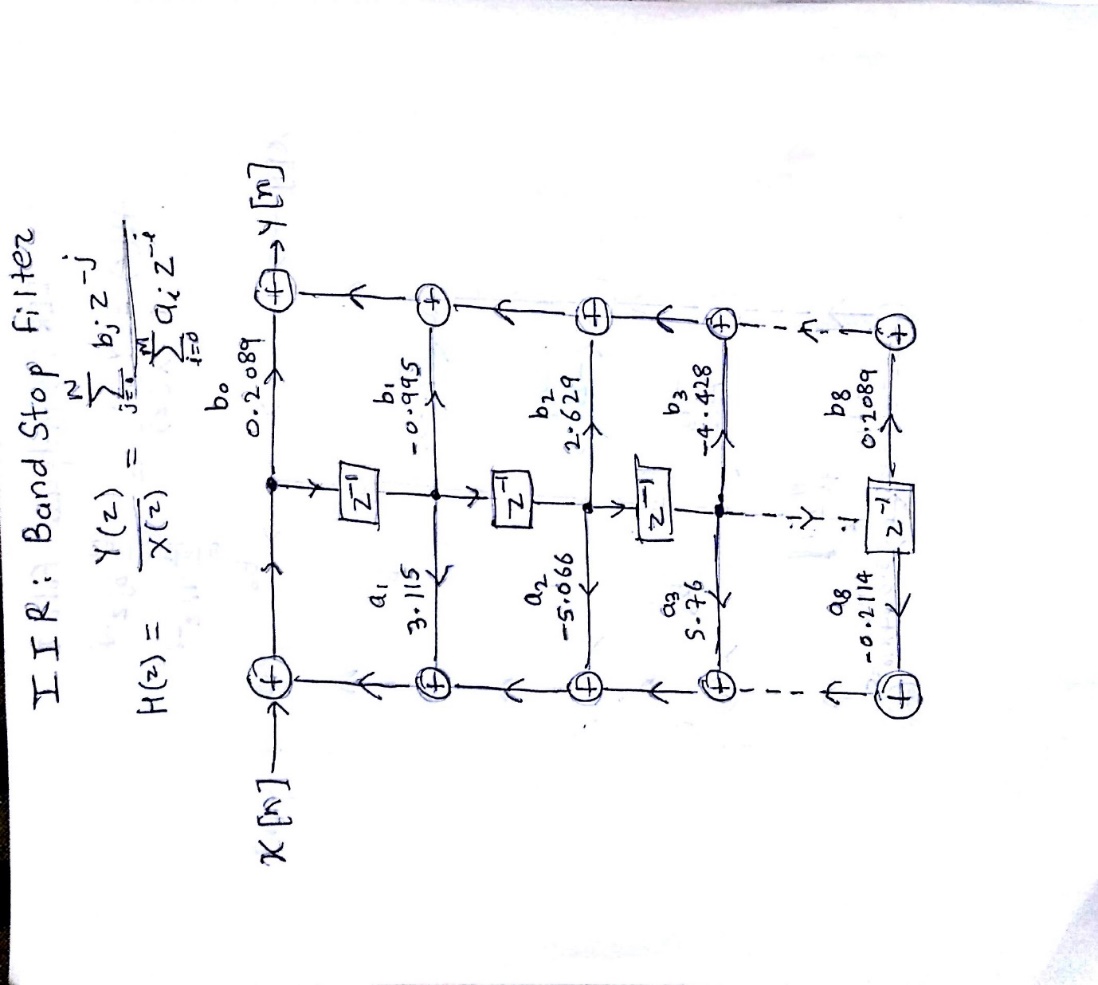


fig11: Magnitude Response of IIR BSF

fig12: Phase Response of IIR BSF

  
fig13: Pole-Zero Plot of IIR BSF

#### Direct Form 2

fig14: Direct Form 2   
 Representation of   
 IIR BSF

### FIR Implementation

#### Ideal Impulse Response

To truncate the ideal response we multiply it by Kaiser window. The length of this window and parameter is determined by using empirical formulae

#### Kaiser Window Parameters

where and

By substituting values we get,

#### Filter Response

Filter Coefficients[41] =

[-0.0129 -0.0256 -0.0110 0.0014 -0.0095 -0.0187 0.0058 0.0439 0.0436 -0.0003 -0.0316 -0.0160 0.0014 -0.0332 -0.0784 -0.0338 0.1043 0.1900 0.0903 -0.1268 0.7600 -0.1268 0.0903 0.1900 0.1043 -0.0338 -0.0784 -0.0332 0.0014 -0.0160 -0.0316 -0.0003 0.0436 0.0439 0.0058 -0.0187 -0.0095 0.0014 -0.0110 -0.0256 -0.0129]

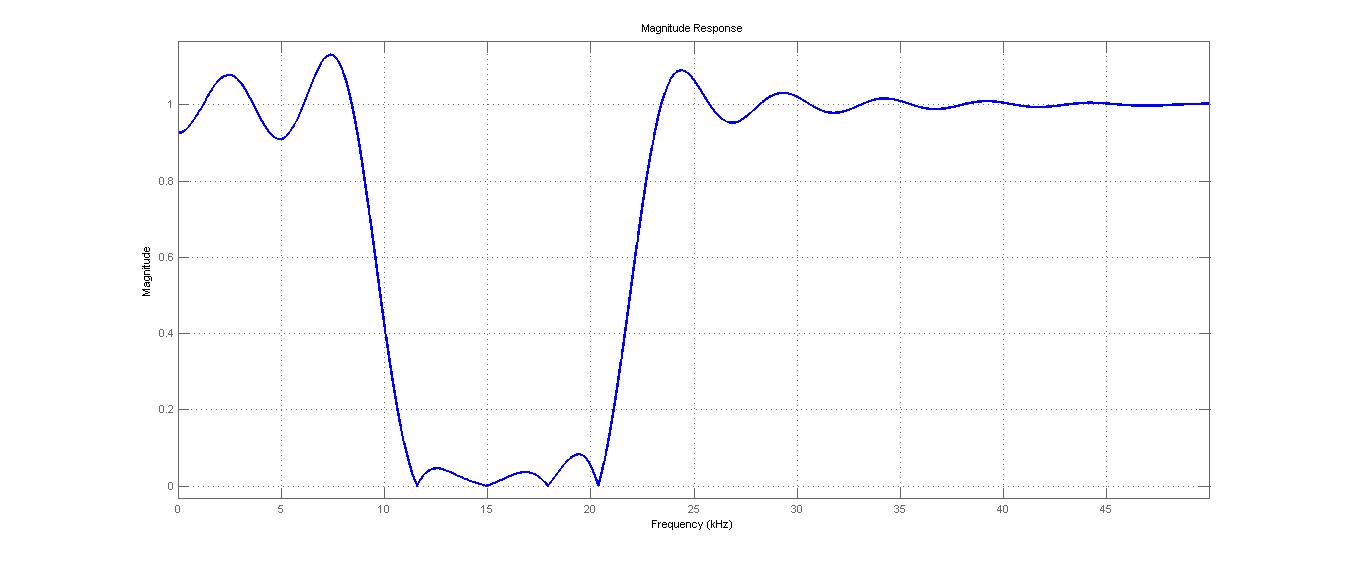


Fig15: Magnitude Response of FIR BSF

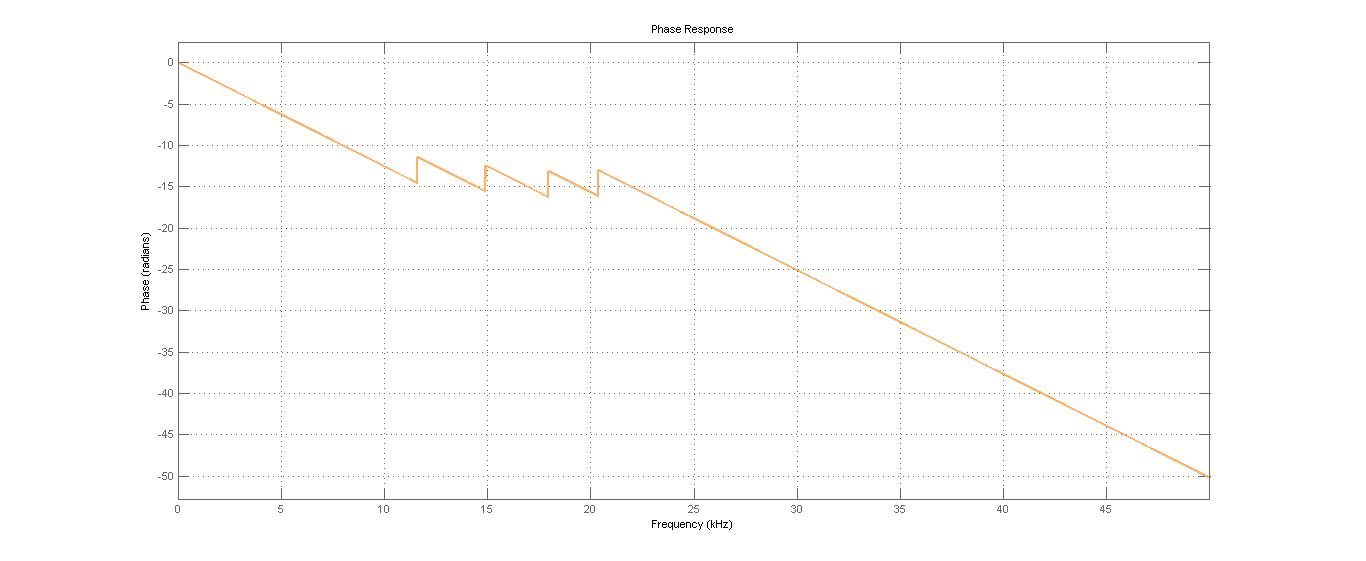
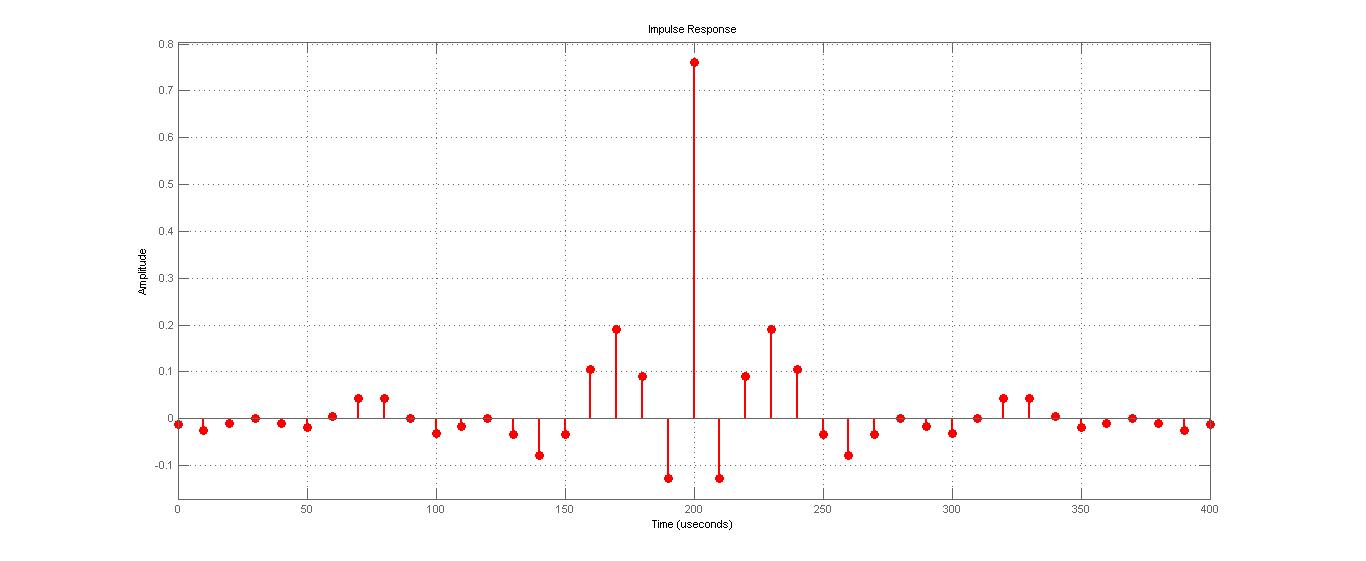


Fig16: Phase Response of FIR BSF



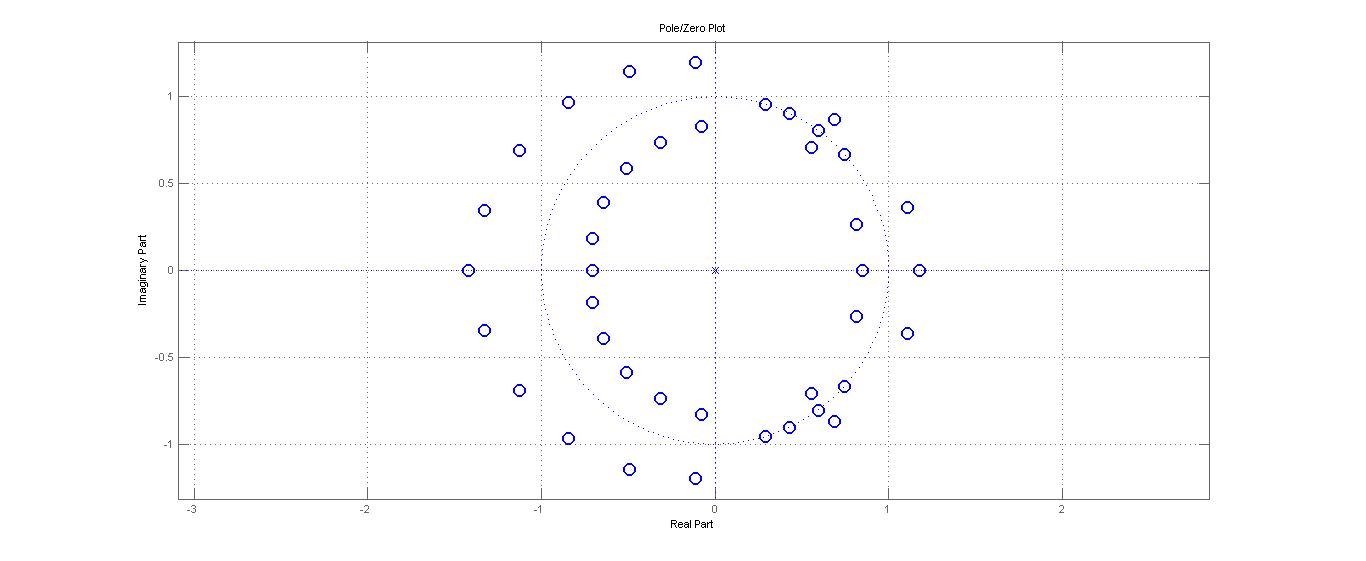
fig17: Impulse Response of FIR BSF

Fig18: Zeros of FIR BSF

# Codes

### IIR Bandpass Filter (Using Butterworth)

% % Kalpesh Patil - 130040019

% % filter number 13

% % BPF IIR

% % Butterworth

%finding cutoff frequencies

m = 13;

q\_m = floor(0.1\*m);

r\_m = m - 10\*q\_m;

Bl\_m = 4 + 0.7\*q\_m + 2\*r\_m;

Bh\_m = Bl\_m + 10;

delta\_1 = 0.15;

delta\_2 = 0.15;

f\_sample = 100\*1000;

omega\_s1 = (Bl\_m-2)\*1000;

omega\_p1 = (Bl\_m)\*1000;

omega\_p2 = (Bh\_m)\*1000;

omega\_s2 = (Bh\_m+2)\*1000;

f\_analog\_array = [omega\_s1 omega\_p1 omega\_p2 omega\_s2];

%normalized specs

f\_digital\_array = f\_analog\_array.\*2\*pi/f\_sample;

f\_eqv\_analog = tan(f\_digital\_array/2);

%analog LPF

omega\_not = sqrt(f\_eqv\_analog(2)\*f\_eqv\_analog(3));

B = f\_eqv\_analog(3) - f\_eqv\_analog(2);

%frequency transformation

f\_eqv\_analog\_LPF = (f\_eqv\_analog.^2 - omega\_not^2)./(B\*f\_eqv\_analog);

%designing this LPF

D1 = (1/(1-delta\_1)^2)-1;

D2 = (1/delta\_2^2)-1;

mod\_f\_eqv\_analog\_LPF = abs(f\_eqv\_analog\_LPF);

stringent\_omega\_s = min(mod\_f\_eqv\_analog\_LPF(1),mod\_f\_eqv\_analog\_LPF(4));

temp = log(sqrt(D2)/sqrt(D1))/log(stringent\_omega\_s/f\_eqv\_analog\_LPF(3));

N = ceil(temp);

%finding poles of equivalent low pass filter

omega\_p = f\_eqv\_analog\_LPF(3);

omega\_c = ((omega\_p/(D1^(1/(2\*N))))+(stringent\_omega\_s/(D2^(1/(2\*N)))))/2;

thetas = (2\*[1:2\*N] - 1)\*(pi)/(2\*N);

gain\_LF = omega\_c^N;

poles = 1i\*omega\_c\*exp(1i\*thetas);

poles\_LF = poles(1:N);

zeros\_LF = [];

[num\_LF,den\_LF] = zp2tf(zeros\_LF,poles\_LF,gain\_LF);

tf\_LF = tf(num\_LF,den\_LF);

%converting back to band pass filter

poles\_BF = zeros(1,2\*N);

poles\_BF(1:N) = (B/2).\*(poles\_LF-sqrt(poles\_LF.^2 - 4\*omega\_not^2/(B^2)));

poles\_BF(N+1:2\*N) = (B/2).\*(poles\_LF+ sqrt(poles\_LF.^2 - 4\*omega\_not^2/(B^2)));

zeros\_BF = zeros(1,N);

gain\_BF = gain\_LF\*B^N;

[num\_BF,den\_BF] = zp2tf(zeros\_BF',poles\_BF,gain\_BF);

tf\_BF = tf(num\_BF, den\_BF);

%converting to z domain

poles\_BF\_z = (1 + poles\_BF)./(1 - poles\_BF);

temp = prod(1-poles\_BF);

gain\_BF\_z = gain\_BF/temp;

zeros\_BF\_z = [ones(1,N),-ones(1,N)];

[num\_BF\_z, den\_BF\_z] = zp2tf(zeros\_BF\_z',poles\_BF\_z,gain\_BF\_z);

DigitalFilter = tf(num\_BF\_z, den\_BF\_z,1/f\_sample);

h = fvtool(num\_BF\_z, den\_BF\_z);

### FIR Bandpass Filter

% % Kalpesh Patil - 130040019

% % filter number 13

% % BPF FIR

m = 13;

q\_m = floor(0.1\*m);

r\_m = m - 10\*q\_m;

Bl\_m = 4 + 0.7\*q\_m + 2\*r\_m;

Bh\_m = Bl\_m + 10;

delta\_1 = 0.15;

delta\_2 = 0.15;

f\_sample = 100\*1000;

omega\_s1 = (Bl\_m-2)\*1000;

omega\_p1 = (Bl\_m)\*1000;

omega\_p2 = (Bh\_m)\*1000;

omega\_s2 = (Bh\_m+2)\*1000;

f\_analog\_array = [omega\_s1 omega\_p1 omega\_p2 omega\_s2];

%normalized specs

f\_digital\_array = f\_analog\_array.\*2\*pi/f\_sample;

% Kaiser window parameters

del\_omega1 = f\_digital\_array(4)-f\_digital\_array(3);

del\_omega2 = f\_digital\_array(2)-f\_digital\_array(1);

del\_omega = min(abs(del\_omega1),abs(del\_omega2));

A = -20\*log10(del\_omega);

% Order

N\_1 = (A-8)/(2\*2.285\*del\_omega);

N\_limit = ceil(N\_1);

N = N\_limit + 5;

if(A<21)

alpha=0;

else if(A<=50)

alpha = 0.5842\*(A-21)^0.4+0.07886\*(A-21);

else

alpha = 0.1102\*(A-8.7);

end

end

omega\_c1 = (f\_digital\_array(2)+f\_digital\_array(1))\*0.5;

omega\_c2 = (f\_digital\_array(4)+f\_digital\_array(3))\*0.5;

h\_ideal = [];

for k = -N:N

if(k~=0)

h\_ideal(k+N+1) = (sin(omega\_c2\*k)-sin(omega\_c1\*k))/(pi\*k);

else

h\_ideal(k+N+1) = 0;

end

end

h\_ideal(N+1) = ((omega\_c2-omega\_c1)/pi);

beta = alpha/N;

% Generating Kaiser window

h\_kaiser = kaiser(2\*N+1,beta);

h\_org = h\_ideal.\*h\_kaiser';

fvtool(h\_org);

### IIR Bandstop Filter (Using Chebyshev)

% % Kalpesh Patil - 130040019

% % filter number 13

% % BSF IIR

% % Chebyshev

%finding cutoff frequencies

m = 13;

q\_m = floor(0.1\*m);

r\_m = m - 10\*q\_m;

Bl\_m = 4 + 0.9\*q\_m + 2\*r\_m;

Bh\_m = Bl\_m + 10;

delta\_1 = 0.15;

delta\_2 = 0.15;

f\_sample = 100\*1000;

omega\_p1 = (Bl\_m-2)\*1000;

omega\_s1 = (Bl\_m)\*1000;

omega\_s2 = (Bh\_m)\*1000;

omega\_p2 = (Bh\_m+2)\*1000;

f\_analog\_array = [omega\_p1 omega\_s1 omega\_s2 omega\_p2];

%normalized specs

f\_digital\_array = f\_analog\_array.\*2\*pi/f\_sample;

f\_eqv\_analog = tan(f\_digital\_array/2);

%analog LPF

omega\_not = sqrt(f\_eqv\_analog(1)\*f\_eqv\_analog(4));

B = f\_eqv\_analog(4) - f\_eqv\_analog(1);

%frequency transformation

f\_eqv\_analog\_LPF = (B\*f\_eqv\_analog)./(omega\_not^2 - f\_eqv\_analog.^2);

%designing this LPF

D1 = (1/(1-delta\_1)^2)-1;

D2 = (1/delta\_2^2)-1;

epsilon = sqrt(D1);

mod\_f\_eqv\_analog\_LPF = abs(f\_eqv\_analog\_LPF);

stringent\_omega\_s = min(mod\_f\_eqv\_analog\_LPF(2),mod\_f\_eqv\_analog\_LPF(3));

temp = acosh(sqrt(D2)/sqrt(D1))/acosh(stringent\_omega\_s/f\_eqv\_analog\_LPF(1));

N = ceil(temp);

%finding poles of equivalent low pass filter

omega\_p = mod\_f\_eqv\_analog\_LPF(4);

Ak = ([1:2\*N]\*2 + 1)\*pi/(2\*N);

Bk = asinh(1/epsilon)/N;

temp\_real = omega\_p\*sin(Ak)\*sinh(Bk);

temp\_imag = omega\_p\*cos(Ak)\*cosh(Bk);

temp = temp\_real + 1i\*temp\_imag;

poles\_LF = zeros(1,N);

t = 1;

for k = 1:2\*N

if(temp\_real(k) < 0)

poles\_LF(t) = temp(k);

t = t+1;

end

end

if (mod(N,2) == 0)

d = 1/sqrt(1+epsilon^2);

else

d = 1;

end

g = ((-1)^N)\*prod(poles\_LF);

gain\_k = d\*g;

zeros\_LF = [];

[num\_LF,den\_LF] = zp2tf(zeros\_LF',poles\_LF,gain\_k);

tf\_LF = tf(num\_LF,den\_LF);

%converting back to band stop filter

poles\_BF = zeros(1,2\*N);

poles\_BF(1:N) = (B/2).\*(1./poles\_LF + sqrt(1./poles\_LF.^2 - 4\*omega\_not^2/(B^2)));

poles\_BF(N+1:2\*N) = (B/2).\*(1./poles\_LF - sqrt(1./poles\_LF.^2 - 4\*omega\_not^2/(B^2)));

zeros\_BF = [1i\*omega\_not\*ones(1,N)' ; -1i\*omega\_not\*ones(1,N)'];

gain\_BF = gain\_k/g;

[num\_BF,den\_BF] = zp2tf(zeros\_BF,poles\_BF,gain\_BF);

tf\_BF = tf(num\_BF, den\_BF);

%converting to z domain

poles\_BF\_z = (1 + poles\_BF)./(1 - poles\_BF);

temp = prod(1-poles\_BF);

gain\_BF\_z = gain\_BF\*((1+omega\_not^2)^N)/(prod(1-poles\_BF));

zeros\_BF\_z = (1 + zeros\_BF)./(1 - zeros\_BF);

[num\_BF\_z, den\_BF\_z] = zp2tf(zeros\_BF\_z,poles\_BF\_z,gain\_BF\_z);

DigitalFilter = tf(num\_BF\_z, den\_BF\_z,1/f\_sample);

h = fvtool(num\_BF\_z, den\_BF\_z);

### FIR Bandstop Filter

% % Kalpesh Patil - 130040019

% % filter number 13

% % BSF FIR

m = 13;

q\_m = floor(0.1\*m);

r\_m = m - 10\*q\_m;

Bl\_m = 4 + 0.9\*q\_m + 2\*r\_m;

Bh\_m = Bl\_m + 10;

delta\_1 = 0.15;

delta\_2 = 0.15;

f\_sample = 100\*1000;

omega\_p1 = (Bl\_m-2)\*1000;

omega\_s1 = (Bl\_m)\*1000;

omega\_s2 = (Bh\_m)\*1000;

omega\_p2 = (Bh\_m+2)\*1000;

f\_analog\_array = [omega\_p1 omega\_s1 omega\_s2 omega\_p2];

%normalized specs

f\_digital\_array = f\_analog\_array.\*2\*pi/f\_sample;

% Kaiser window parameters

del\_omega1 = f\_digital\_array(4)-f\_digital\_array(3);

del\_omega2 = f\_digital\_array(2)-f\_digital\_array(1);

del\_omega = min(abs(del\_omega1),abs(del\_omega2));

A = -20\*log10(del\_omega);

% Order

N\_1 = (A-8)/(2\*2.285\*del\_omega);

N\_limit = ceil(N\_1);

N = N\_limit + 5;

if(A<21)

alpha=0;

else if(A<=50)

alpha = 0.5842\*(A-21)^0.4+0.07886\*(A-21);

else

alpha = 0.1102\*(A-8.7);

end

end

omega\_c1 = (f\_digital\_array(2)+f\_digital\_array(1))\*0.5;

omega\_c2 = (f\_digital\_array(4)+f\_digital\_array(3))\*0.5;

h\_ideal = [];

for k = -N:N

if(k~=0)

h\_ideal(k+N+1) = - (sin(omega\_c2\*k)-sin(omega\_c1\*k))/(pi\*k);

else

h\_ideal(k+N+1) = 0;

end

end

h\_ideal(N+1) = 1 - ((omega\_c2-omega\_c1)/pi);

beta = alpha/N;

% Generating Kaiser window

h\_kaiser = kaiser(2\*N+1,beta);

h\_org = h\_ideal.\*h\_kaiser';

fvtool(h\_org);