3.1 Filter design

3.1.1 Filter design

The basic idea to design this system is to make the reconstruction as perfect as possible. In order to do this, I started from 3 crucial indexes of P.R. and try to solve them separately.

$$Perfect\ Reconstruction \begin{cases} (1)\ Aliasing\ Cancellation \\ (2)\ Phase\ Distortion \\ (3)\ Amplitude\ Distortion \end{cases}$$

(1) Aliasing Cancellation

As is mentioned in the slides, to cancel the aliasing caused by sampling, we need to meet the following requirements:

$$\begin{cases} F_0(z) = H_1(-z) \\ F_1(z) = -H_0(-z) \end{cases}$$

And I choose to use the simple form in the slides:

$$H_1(z) = H_0(-z) => \ h_1(n) = (-1)^n h_0(n)$$

And the cancellation requirements above become:

$$\begin{cases} F_0(z) = H_0(z) \\ F_1(z) = -H_0(-z) \end{cases}$$

If we translate these into time domain, we have

$$h_1(n) = (-1)^n h_0(n)$$

$$\begin{cases} f_0(n) = h_0(n) \\ f_1(n) = -(-1)^n h_0(n) \end{cases}$$

According to this relationship, we can generate $h_1(n), f_0(n), f_1(n)$ from $h_0(n)$.

(2) Phase Distortion

This is not really a problem. $h_0(n)$ is a FIR filter, so that it will not cause any phase distortion.

(3) Amplitude Distortion

According to the slides, we can eliminate amplitude distortion if and only if the polyphase component of $H_0(z)$, $E_0(z)$ and $E_1(z)$ are pure delays. That is to say, $H_0(z)$ has the form of:

$$H_0(z) = c_0 z^{-2n_0} + c_1 z^{-(2n_1+1)}$$

However, in this case, the order of $h_0(n)$ filter is 1 which means this filter has poor passband and stopband qualities and poor low pass response. That is not what we want. In other words, if $h_0(n)$ can offer good low pass response, amplitude distortion is inevitable. What we could do is to reduce it as much as possible.

I chose Kaiser window as a basic model to design $h_0(n)$ because of the flexibility. Comparing to other window filters, the ripple shape of Kaiser window is much easier to control and modify with filter coefficients, N and β.

The coefficients I start from are N = 29, Fc = 0.5, β = 9. Fig. 1 is its frequency response.

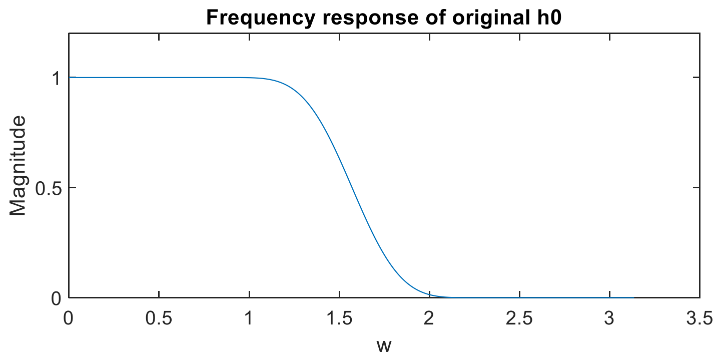

Fig. 1

3.1.2 Optimization

As is discussed above, the main conflict of this design is low pass response and amplitude distortion. I use the method in the given article to measure the amplitude distortion.

$$E_r = \sum_{\omega=0}^{\pi} (H^2(\omega) + H^2(\pi-\omega) - 1)^2, \quad this\ should\ be\ as\ small\ as\ possible$$

$$E_s = \sum_{\omega=stopband}^{\pi} H^2(\omega), \quad this\ refers\ to\ energy\ loss\ out\ of\ stop\ band$$

In addition, E_s is not important in my design because if I set the coefficient N and β large enough (e.g. N>19, β > 8), the ripples will not be obvious, and energy lost in side lobes will be very small.

My optimization process is listed as follows:

(1) Generate a FIR low pass filter (Kaiser window) which is $h_0(n)$
(2) Generate high pass filter $h_1(n)$ using the relationship: $h_1(n) = (-1)^n h_0(n)$
(3) Calculate their magnitudes, hmag0 which is $H(\omega)$, and hmag1 which is $H(\pi-\omega)$
(4) Calculate E_r to see if it is small enough. If not, choose another filter coefficient Fc and start from (1) again.

The final $h_0(n)$ filter I got is generated by the code as follows:

```
N    = 29;
Fc   = 0.5363563;
flag = 'scale';
Beta = 9;
win = kaiser(N+1, Beta);
h0  = fir1(N, Fc, 'low', win, flag);
```

With the frequency response (Fig. 2):

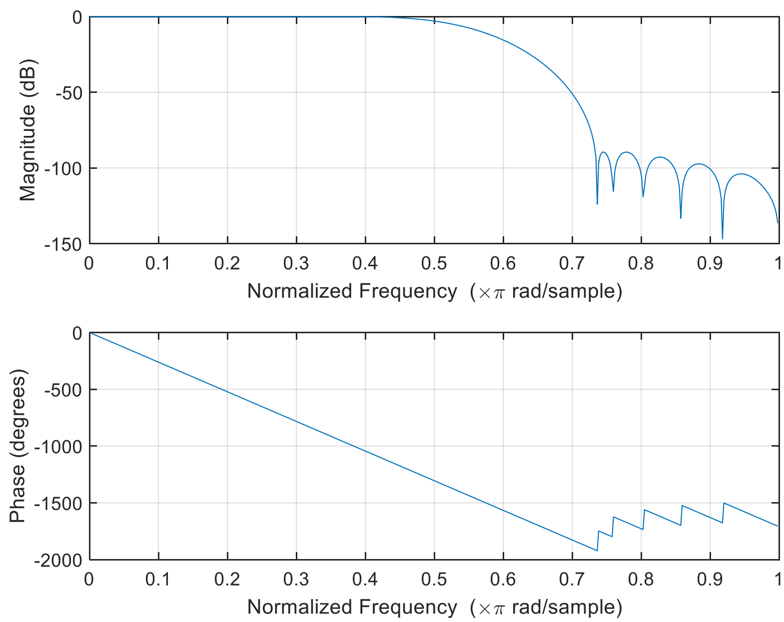

Fig. 2

Fig. 3 is an intuitive plot of transfer function of the system $|T(\omega)| = |H_0(\omega)|^2 + |H_1(\omega)|^2$

we can see that $|T(\omega)| = |H_0(\omega)|^2 + |H_1(\omega)|^2$ is basically 1 within $0 < \omega < \pi$

The optimization factor accordingly is $E_r = 1.0799 \times 10^{-4}$, almost zero.

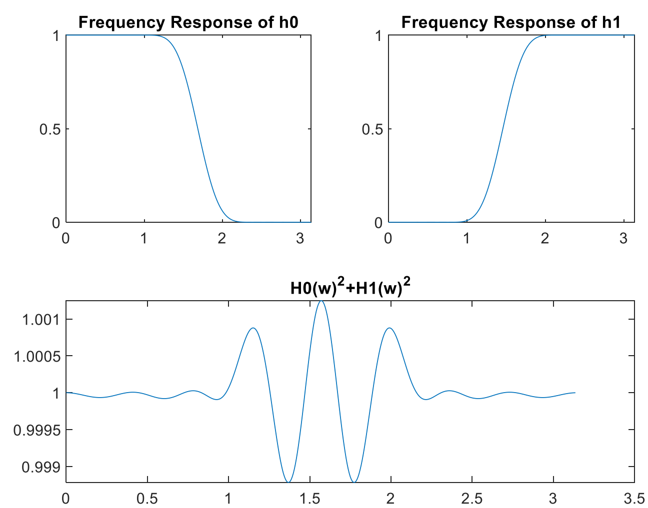

Fig. 3

3.2 Simulation

The basic idea of simulation is to choose an input signal x(n), make it go through the QFM system shown in Fig. 1, generate the output signal $\hat{x}(n)$. There are three things I need to point out.

(1) My input signal for simulation comes from the sampdata.m file provided in HW4 (I'm lazy :D).

(2) The method I use for upsampling by 2 is inserting zeros. This will result in a magnitude reduction in time domain by 2 after the second convolution in f0 and f1 filter. So, when I calculate the absolute error between input and output, $\hat{x}(n)$ must be multiplied by 2 before deducting $\mathrm{x(n)}$.

(3) The length of $\mathrm{x(n)}$ and $\hat{x}(n)$ are different because of convolution. $\hat{x}(n)$ is 2*length(h0) pads longer than $\mathrm{x(n)}$. As a result, I need to choose the specific part of $\hat{x}(n)$ to calculate the absolute error. In my case, the length of $\mathrm{x(n)}$ is 256, the length of $h_0(n)$ is 30, the length of $\hat{x}(n)$ is 316.

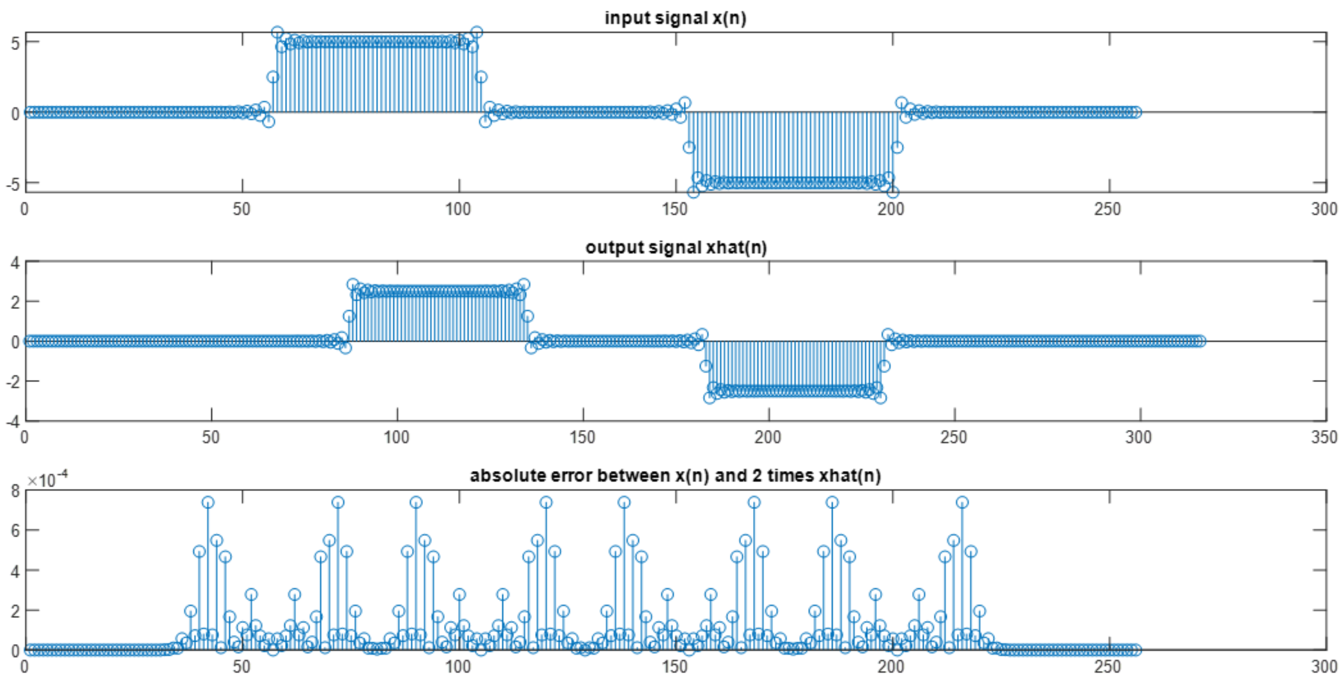

Fig. 4

As is shown in Fig. 5, the shape of input and output is basically alike. The absolute error is less than 0.0008.

3.3 Answers to the problem questions

I am able to achieve perfect reconstruction, but I choose not to. Because the filters will be of one-order and the low/high pass response of P.R. system is not good, it's a trade-off. During my optimization process, I tried to eliminate amplitude distortion and make the reconstruction as perfect as possible.

The main conflict of my design is low/high pass response and P.R. I tried my best to have it both ways.

PS: My answer to this problem is a bit of lengthy, sorry for the trouble and hair loss:)

3.4 MATLAB code

```
% generate low pass filter h0 based on Kaiser window
N    = 29;
Fc   = 0.5363563;
flag = 'scale';
Beta = 9;
win = kaiser(N+1, Beta);
h0  = fir1(N, Fc, 'low', win, flag);
figure(1);
freqz(h0);

% generate h1
n=rem(1:30,2);
m=1*(n>0.5)+(-1)*(n<=0.5); % times a 1,-1,1,-1... sequence
h1=h0.*m;

% intuitive plot of h0, h1, and amplitude distortion
[H0,w]=freqz(h0,1,512);
hmag0 = abs(H0);
figure(2);
subplot(2,2,1)
plot(w,hmag0);
title('Frequency Response of h0');

[H1,w]=freqz(h1,1,512);
hmag1 = abs(H1);
subplot(2,2,2)
plot(w,hmag1);
title('Frequency Response of h1');

subplot(2,2,[3 4])
plot(w,(hmag0.^2+hmag1.^2))
title('H0(w)^2+H1(w)^2');

% Optimization factor Er
one=ones(512,1);
Er=sum((hmag0.^2+hmag1.^2-one).^2);
Er;

% Simulation
% Generate input signal x(n)
ntaps = 65;
f = [0.0 0.9 0.95 1.0];
mag = [ 1.0 1.0 0.7071 0.0];
b = fir2(ntaps, f, mag);
```

```
n1 = length(b);
len1 = 256 - n1 + 1;
data = 5*[zeros(1, 24) ones(1, 48) zeros(1, 48) -1*ones(1, 48)
zeros(1,23)];
x = conv(b, data);

% x(n) go through QMF system
u0=conv(x,h0);
v0=dyaddown(u0,1);
w0=dyadup(v0,1);
u1=conv(x,h1);
v1=dyaddown(u1,1);
w1=dyadup(v1,1);
f0=h0;
f1=-h1;
xhat=conv(w0,f0)+conv(w1,f1);

% Simulation results
figure(3);
subplot(3,1,1)
stem(x);
title('input signal x(n)');
subplot(3,1,2)
stem(xhat);
title('output signal xhat(n)');
error=abs(xhat(31:286).*2-x);
subplot(3,1,3)
stem(error);
title('absolute error between x(n) and 2 times xhat(n)');
```