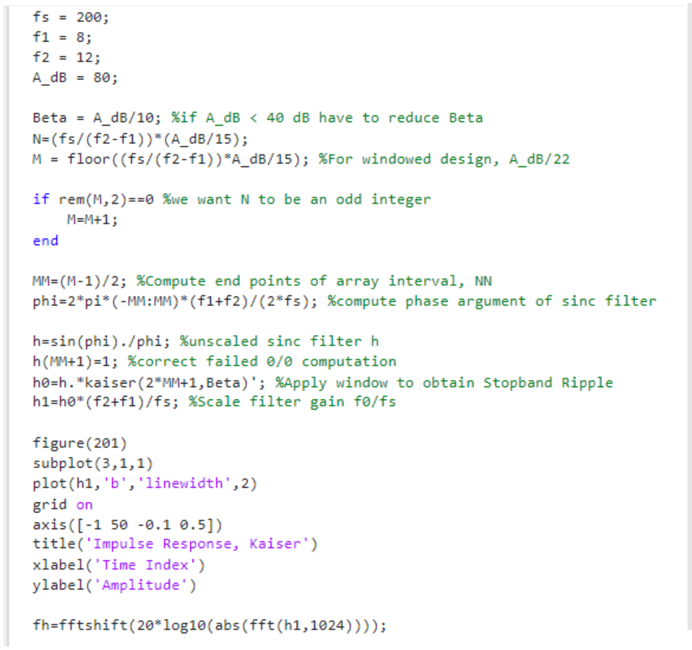
ECE_161B HW-4

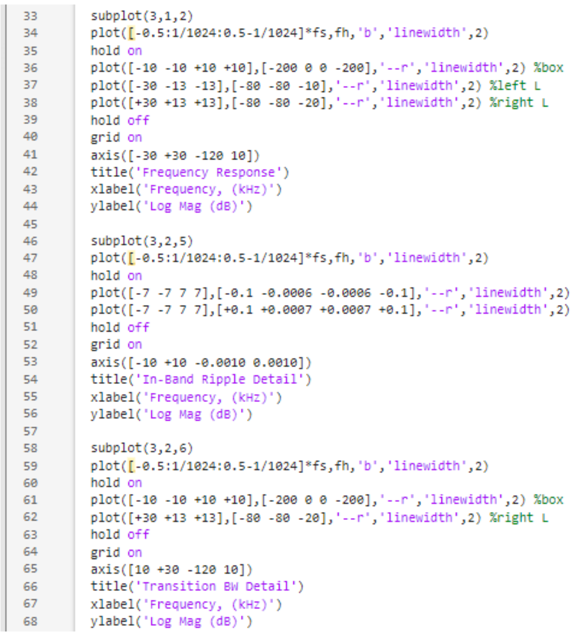
Iterated FIR Filter

In this HW problem we compare a direct implementation of a Narrow Bandwidth FIR Filter with a large Sample Rate to bandwidth ratio to an Iterated Design that implements the Narrowband Filter at a reduced sample rate and then interpolates up to the higher sample rate. We examine three options. The initial design is a FIR filter designed with a Kaiser Windowed Sinc filter. The Sample Rate is 200 kHz, the passband is 0 to 8 kHz, and the stopband is 12 kHz to 100 kHz. The 6 dB bandwidth is 10 kHz. The stopband attenuation is -80 dB (1 part in 10,000).

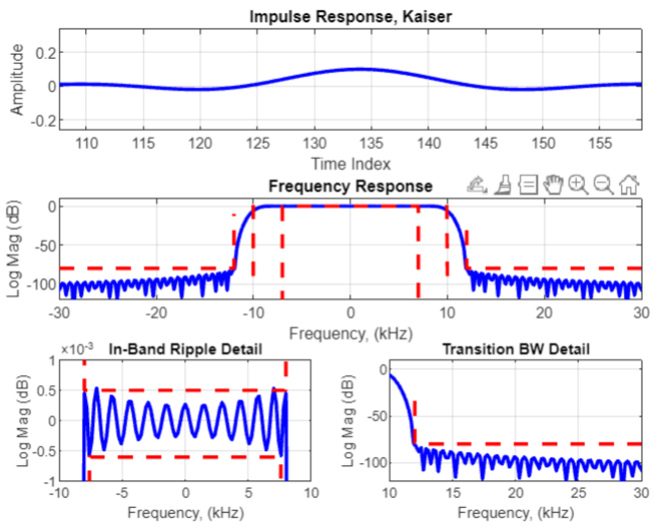
1. Estimate the filter length and design the single stage FIR Filter. On three subplots show the filter Impulse response h1, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple.



```
fs = 200;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
N=(fs/(f2-f1))*(A_dB/15);
M = floor((fs/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs); %compute phase argument of sinc filter

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h1=h0*(f2+f1)/fs; %Scale filter gain f0/fs

figure(201)
subplot(3,1,1)
plot(h1,'b','linewidth',2)
grid on
axis([-1 50 -0.1 0.5])
title('Impulse Response, Kaiser')
xlabel('Time Index')
ylabel('Amplitude')

fh=fftshift(20*log10(abs(fft(h1,1024))));
```

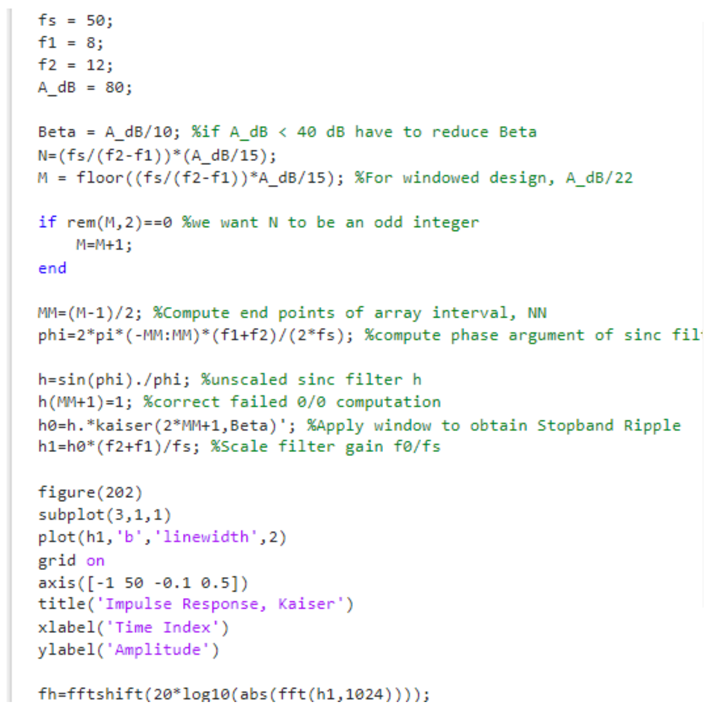


```
subplot(3,1,2)
plot([-0.5:1/1024:0.5-1/1024]*fs,fh,'b','linewidth',2)
hold on
plot([-10 -10 +10 +10],[-200 0 0 -200],'--r','linewidth',2) %box
plot([-30 -13 -13],[-80 -80 -10],'--r','linewidth',2) %left L
plot([+30 +13 +13],[-80 -80 -20],'--r','linewidth',2) %right L
hold off
grid on
axis([-30 +30 -120 10])
title('Frequency Response')
xlabel('Frequency, (kHz)')
ylabel('Log Mag (dB)')

subplot(3,2,5)
plot([-0.5:1/1024:0.5-1/1024]*fs,fh,'b','linewidth',2)
hold on
plot([-7 -7 7 7],[-0.1 -0.0006 -0.0006 -0.1],'--r','linewidth',2)
plot([-7 -7 7 7],[+0.1 +0.0007 +0.0007 +0.1],'--r','linewidth',2)
hold off
grid on
axis([-10 +10 -0.0010 0.0010])
title('In-Band Ripple Detail')
xlabel('Frequency, (kHz)')
ylabel('Log Mag (dB)')

subplot(3,2,6)
plot([-0.5:1/1024:0.5-1/1024]*fs,fh,'b','linewidth',2)
hold on
plot([-10 -10 +10 +10],[-200 0 0 -200],'--r','linewidth',2) %box
plot([+30 +13 +13],[-80 -80 -20],'--r','linewidth',2) %right L
hold off
grid on
axis([10 +30 -120 10])
title('Transition BW Detail')
xlabel('Frequency, (kHz)')
ylabel('Log Mag (dB)')
```

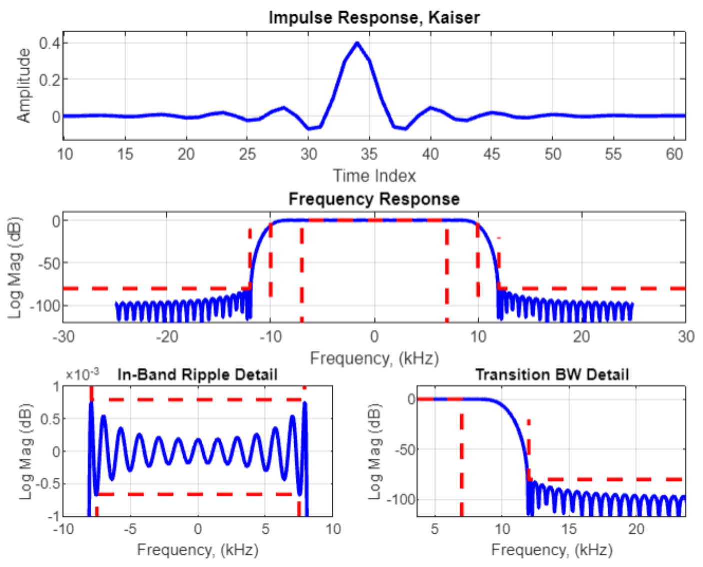


The filter length is 267 and the passband ripple is 0.0011

2. Now design the same filter response at the reduced sample rate of 50 kHz. The 4-to-1 sample rate reduction results in a filter of 1/4th the length. On three subplots show the filter Impulse response h2, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple.

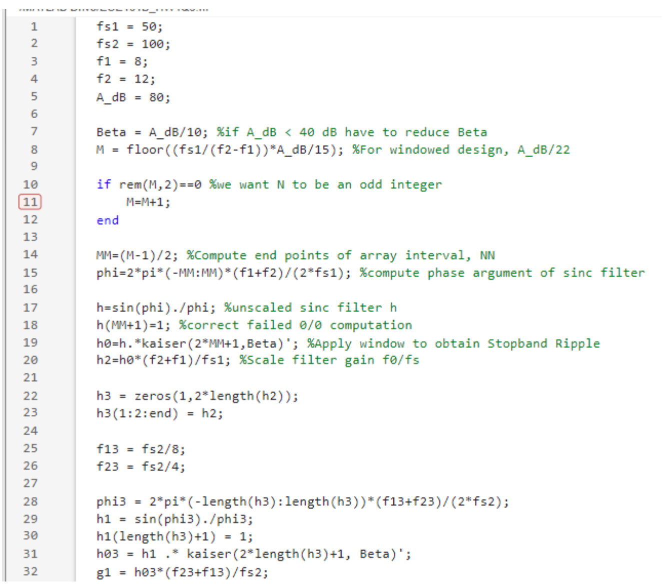


```
fs = 50;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
N=(fs/(f2-f1))*(A_dB/15);
M = floor((fs/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs); %compute phase argument of sinc fil

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h1=h0*(f2+f1)/fs; %Scale filter gain f0/fs

figure(202)
subplot(3,1,1)
plot(h1,'b','linewidth',2)
grid on
axis([-1 50 -0.1 0.5])
title('Impulse Response, Kaiser')
xlabel('Time Index')
ylabel('Amplitude')

fh=fftshift(20*log10(abs(fft(h1,1024))));
```

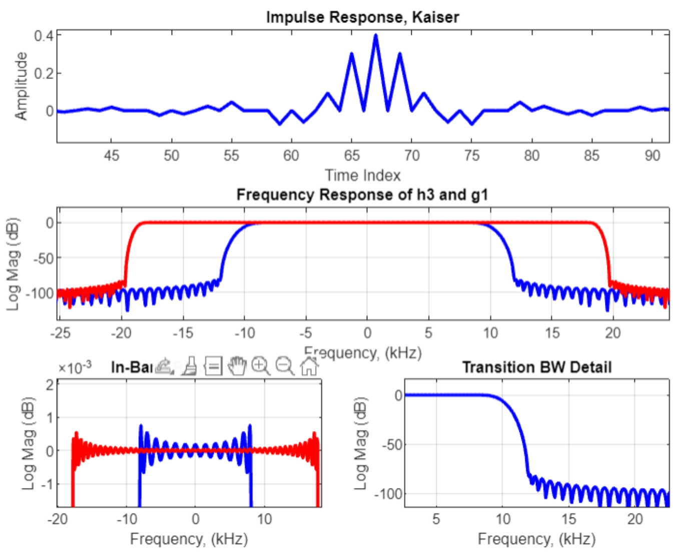


The length of the filter is now 67 and the passband ripple increased to 0.00145

3. Now zero-pack the h2 filter 1-to-2 to form a double length filter h3 (without additional filter coefficients). This filter has a sample rate of 100 kHz. On three subplots show the filter Impulse response h3, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple. Also design the first interpolating filter g1 that passes the h3 spectrum centered at DC but rejects the h3 spectrum centered at 50 kHz. Overlay the g1 frequency response (in red) on the h3 frequency response.

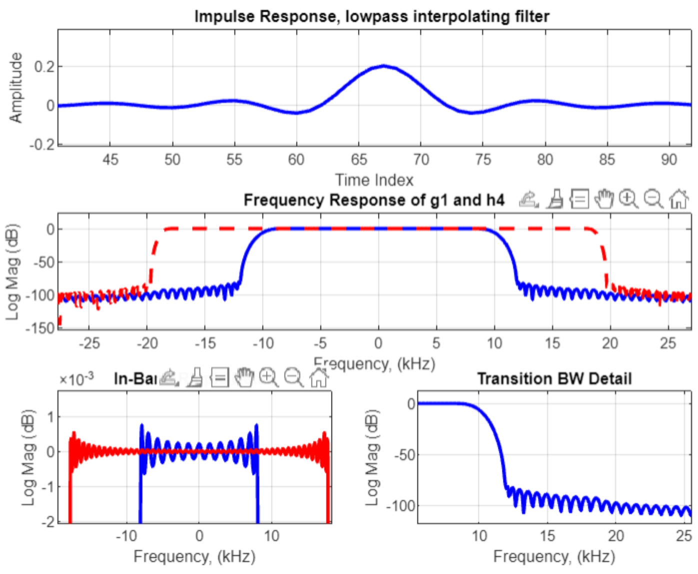


```
fs1 = 50;
fs2 = 100;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
M = floor((fs1/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs1); %compute phase argument of sinc filter

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h2=h0*(f2+f1)/fs1; %Scale filter gain f0/fs

h3 = zeros(1,2*length(h2));
h3(1:2:end) = h2;

f13 = fs2/8;
f23 = fs2/4;

phi3 = 2*pi*(-length(h3):length(h3))*(f13+f23)/(2*fs2);
h1 = sin(phi3)./phi3;
h1(length(h3)+1) = 1;
h03 = h1 .* kaiser(2*length(h3)+1, Beta)';
g1 = h03*(f23+f13)/fs2;
```



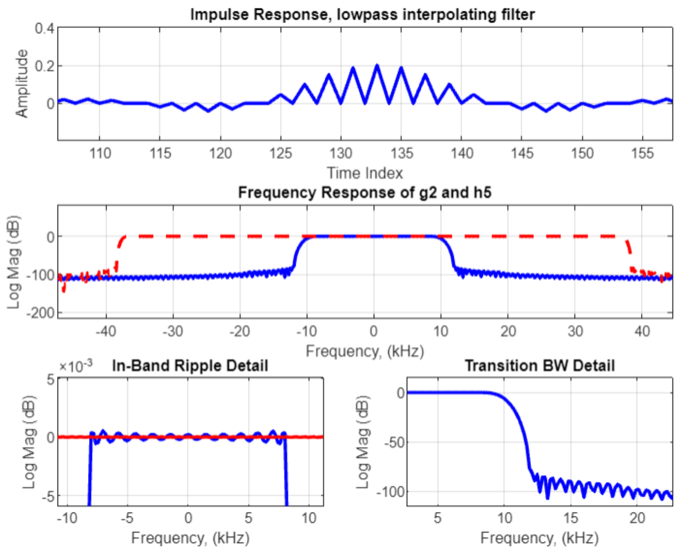
4. Convolve the zero packed h3 filter with the g1 interpolating filter to form the interpolated filter h4. On three subplots show the filter Impulse response h4, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple. Overlay the g1 frequency response (in dashed red) on the h4 frequency response.

```
fs1 = 50;
fs2 = 100;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
M = floor((fs1/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs1); %compute phase argument of sinc fil

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h2=h0*(f2+f1)/fs1; %Scale filter gain f0/fs

h3 = zeros(1,2*length(h2));
h3(1:2:end) = h2;

f13 = fs2/8;
f23 = fs2/4;

phi3 = 2*pi*(-length(h3):length(h3))*(f13+f23)/(2*fs2);
h1 = sin(phi3)./phi3;
h1(length(h3)+1) = 1;
h03 = h1 .* kaiser(2*length(h3)+1, Beta)';
g1 = h03*(f23+f13)/fs2;

h4 = conv(h3,g1, 'same');
```

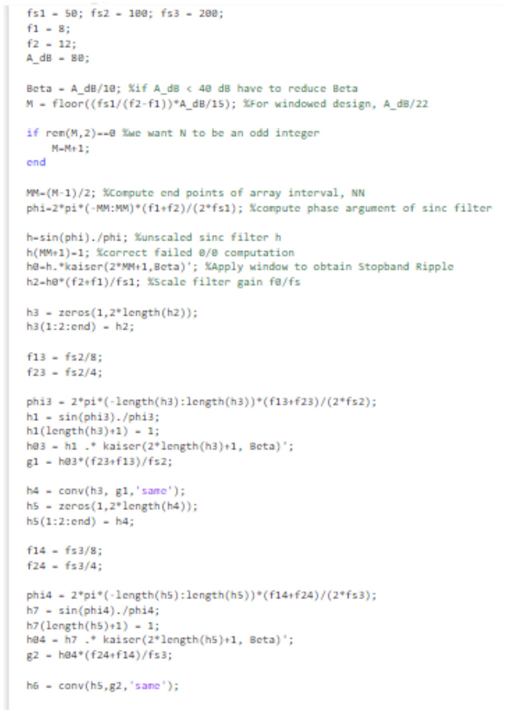


5. Now zero-pack the h4 filter 1-to-2 to form a double length filter h5 (without additional filter coefficients). This filter has a sample rate of 200 kHz. On three subplots show the filter Impulse response h5, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple. Also design the second interpolating filter g2 that passes the h5 spectrum centered at DC but rejects the h3 spectrum centered at 100 kHz. Overlay the g2 frequency response (in red) on the h5 frequency response.

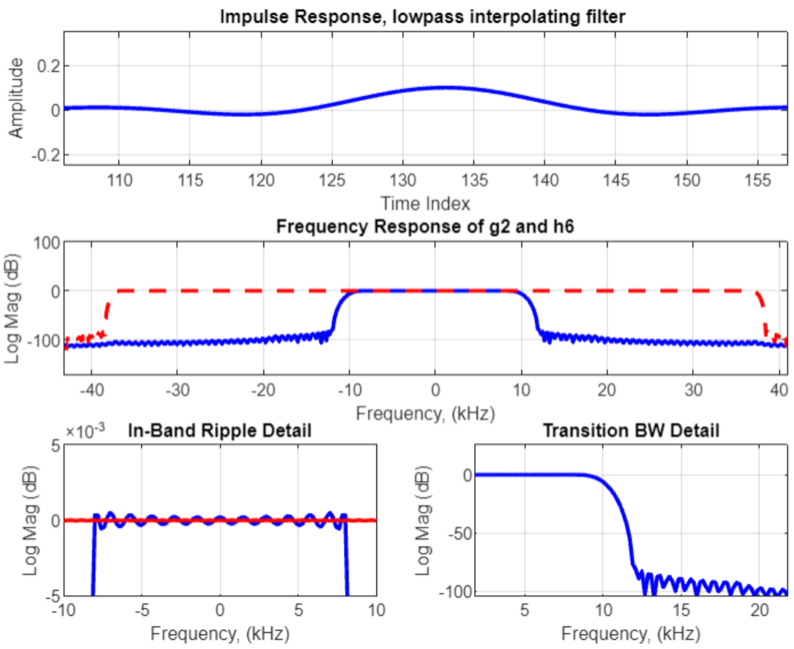
```
fs1 = 50;
fs2 = 100;
fs3 = 200;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
M = floor((fs1/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs1); %compute phase argument of sinc filter

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h2=h0*(f2+f1)/fs1; %Scale filter gain f0/fs

h3 = zeros(1,2*length(h2));
h3(1:2:end) = h2;


f13 = fs2/8;
f23 = fs2/4;

phi3 = 2*pi*(-length(h3):length(h3))*(f13+f23)/(2*fs2);
h1 = sin(phi3)./phi3;
h1(length(h3)+1) = 1;
h03 = h1 .* kaiser(2*length(h3)+1, Beta)';
g1 = h03*(f23+f13)/fs2;

h4 = conv(h3, g1, 'same');

%h5
h5 = zeros(1,2*length(h4));
h5(1:2:end) = h4;

f14 = fs3/8;
f24 = fs3/4;

phi4 = 2*pi*(-length(h5):length(h5))*(f14+f24)/(2*fs3);
h7 = sin(phi4)./phi4;
h7(length(h5)+1) = 1;
h04 = h7 .* kaiser(2*length(h5)+1, Beta)';
g2 = h04*(f24+f14)/fs3;
```



6. Convolve the zeros packed h5 filter with the g2 interpolating filter to form the interpolated filter h6. On three subplots show the filter Impulse response h6, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple. Overlay the g2 frequency response (in dashed red) on the h6 frequency response.



```
fs1 = 50; fs2 = 100; fs3 = 200;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
M = floor((fs1/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs1); %compute phase argument of sinc filter

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h2=h0*(f2+f1)/fs1; %Scale filter gain f0/fs

h3 = zeros(1,2*length(h2));
h3(1:2:end) = h2;

f13 = fs2/8;
f23 = fs2/4;

phi3 = 2*pi*(-length(h3):length(h3))*(f13+f23)/(2*fs2);
h1 = sin(phi3)./phi3;
h1(length(h3)+1) = 1;
h03 = h1 .* kaiser(2*length(h3)+1, Beta)';
g1 = h03*(f23+f13)/fs2;

h4 = conv(h3, g1,'same');
h5 = zeros(1,2*length(h4));
h5(1:2:end) = h4;

f14 = fs3/8;
f24 = fs3/4;

phi4 = 2*pi*(-length(h5):length(h5))*(f14+f24)/(2*fs3);
h7 = sin(phi4)./phi4;
h7(length(h5)+1) = 1;
h04 = h7 .* kaiser(2*length(h5)+1, Beta)';
g2 = h04*(f24+f14)/fs3;

h6 = conv(h5,g2,'same');
```

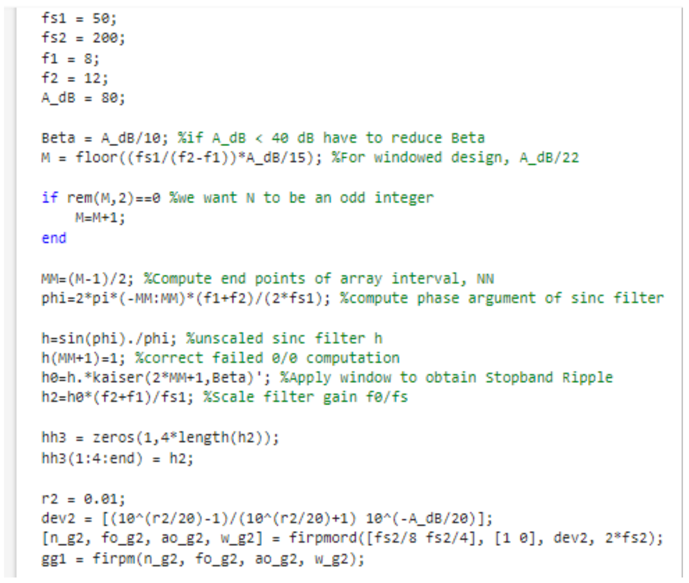


This graph looks just like h1, has the same length of 267 too.

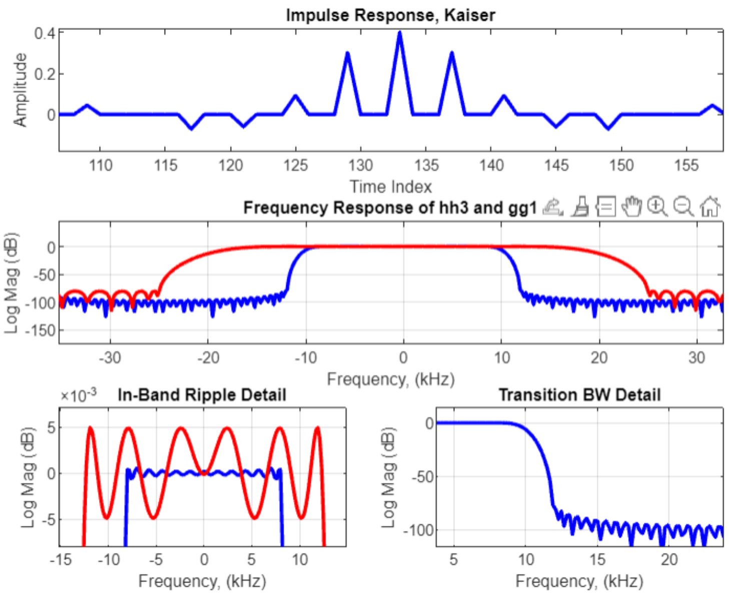
7. Comment on the workload of the direct filter implementation (h1) with that of the double interpolated and 1-to-2 resampled implementation (h6)

   The workload of h6 is a lot greater than the workload of h1 because h6 needed to use a lot of multiplication (multiple convolutions, zero-packings, etc.) which is very computationally demanding versus h1 which is very straightforward.

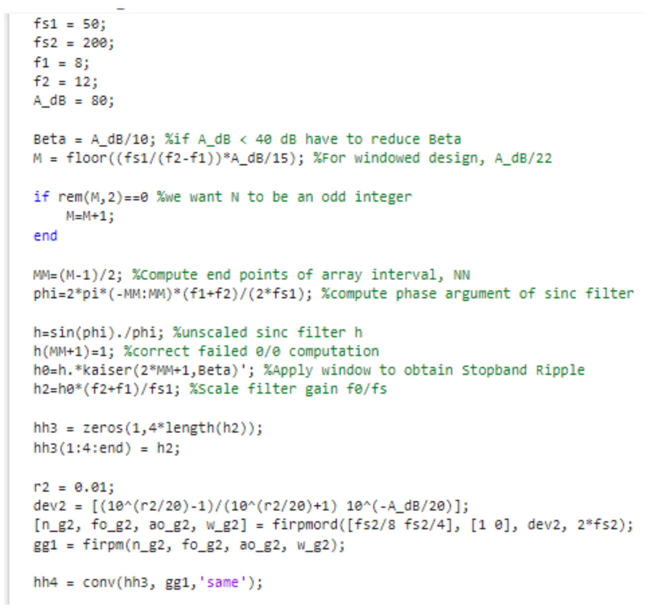
8. We now modify the problem to use the h2 67 tap filter to be zero packed 1-to-4 to form a quadruple length filter hh3 (without additional filter coefficients). This filter has a sample rate of 200 kHz. Three subplots show the filter Impulse response hh3, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple. Also design the first interpolating filter gg1 that passes the hh3 spectrum centered at DC but rejects the hh3 spectrum centered at 50 kHz and 100 kHz. Overlay the gg2 frequency response (in red) on the hh3 frequency response.



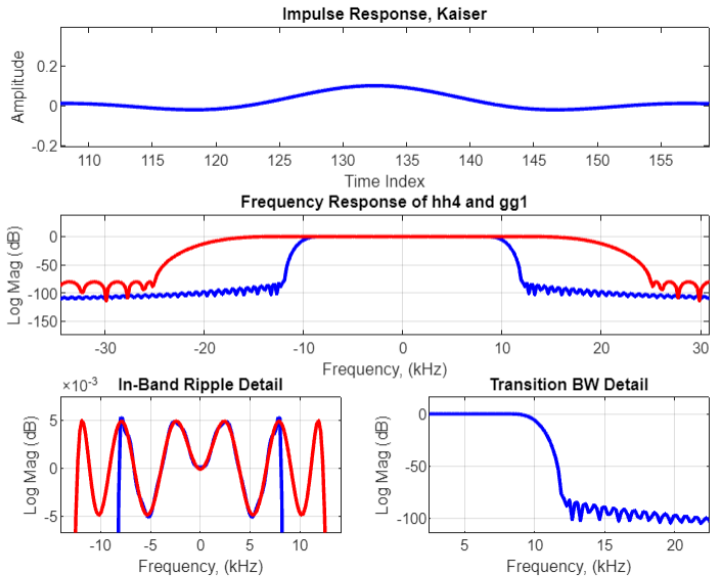
```
fs1 = 50;
fs2 = 200;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
M = floor((fs1/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs1); %compute phase argument of sinc filter

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h2=h0*(f2+f1)/fs1; %Scale filter gain f0/fs

hh3 = zeros(1,4*length(h2));
hh3(1:4:end) = h2;

r2 = 0.01;
dev2 = [(10^(r2/20)-1)/(10^(r2/20)+1) 10^(-A_dB/20)];
[n_g2, fo_g2, ao_g2, w_g2] = firpmord([fs2/8 fs2/4], [1 0], dev2, 2*fs2);
gg1 = firpm(n_g2, fo_g2, ao_g2, w_g2);
```



9. Convolve the zero packed hh3 filter with the gg1 interpolating filter to form the interpolated filter hh4. On three subplots show the filter Impulse response hh4, the Log Mag frequency response, and the zoom +/- 0.01 dB passband ripple. Overlay the gg1 frequency response (in dashed red) on



```
fs1 = 50;
fs2 = 200;
f1 = 8;
f2 = 12;
A_dB = 80;

Beta = A_dB/10; %if A_dB < 40 dB have to reduce Beta
M = floor((fs1/(f2-f1))*A_dB/15); %For windowed design, A_dB/22

if rem(M,2)==0 %we want N to be an odd integer
    M=M+1;
end

MM=(M-1)/2; %Compute end points of array interval, NN
phi=2*pi*(-MM:MM)*(f1+f2)/(2*fs1); %compute phase argument of sinc filter

h=sin(phi)./phi; %unscaled sinc filter h
h(MM+1)=1; %correct failed 0/0 computation
h0=h.*kaiser(2*MM+1,Beta)'; %Apply window to obtain Stopband Ripple
h2=h0*(f2+f1)/fs1; %Scale filter gain f0/fs

hh3 = zeros(1,4*length(h2));
hh3(1:4:end) = h2;

r2 = 0.01;
dev2 = [(10^(r2/20)-1)/(10^(r2/20)+1) 10^(-A_dB/20)];
[n_g2, fo_g2, ao_g2, w_g2] = firpmord([fs2/8 fs2/4], [1 0], dev2, 2*fs2);
gg1 = firpm(n_g2, fo_g2, ao_g2, w_g2);

hh4 = conv(hh3, gg1,'same');
```



This graph looks the same as h6 and h1, has the same length of 267.

NOTE: FOR QUESTION 8 AND 9, I USED THE FIRPM() BECAUSE IT GAVE A BETTER G1, NOTE THE DIFFERENT PASSBAND RIPPLE THAN THE PREVIOUS QUESTIONS BUT THEY'RE STILL CORRECT