EECE5666 (DSP) : Homework-8

Due on April 29, 2022 by 11:59 pm via submission portal.

NAME: McKean, Tyler

Table of Contents

Instructions

1. You are required to complete this assignment using Live Editor.
2. Enter your MATLAB script in the spaces provided. If it contains a plot, the plot will be displayed after the script.
3. All your plots must be properly labeled and should have appropriate titles to get full credit.
4. Use the equation editor to typeset mathematical material such as variables, equations, etc.
5. After completeing this assignment, export this Live script to PDF and submit the PDF file through the provided submission portal.
6. You will have only one attempt to submit your assignment. Make every effort to submit the correct and completed PDF file the first time.
7. Please submit your homework before the due date/time. A late submission after midnight of the due date will result in loss of points at a rate of 10% per hour until 8 am the following day, at which time the solutions will be published.

Default Plot Parameters

```
set(0,'defaultfigurepaperunits','points','defaultfigureunits','points');
set(0,'defaultaxesfontsize',10); set(0,'defaultaxeslinewidth',1.5);
set(0,'defaultaxestitlefontsize',1.4,'defaultaxeslabelfontsize',1.2);
```

Problem 8.1

Text Problem 15.25 (Page 960)

Determine the variance-gain for the following system when the input quantization noise is propagated through it:

$$H(z) = \frac{1 - 8z^{-1} + 19z^{-2} - 12z^{-3}}{1 - 0.4z^{-1} - 0.2375z^{-2} - 0.0188z^{-3}}.$$

MATLAB script:

```
clc, close all; clear;
b1 = [1 -8 19 -12]; a1 = [1 -0.4 -0.2375 -0.0188];
b2 = fliplr(b1)/b1(4); a2 = fliplr(a1)/a1(4);
K = b1(4)/a1(4);
[r p k] = residuez(conv(b1,b2),conv(a1,a2))
```

```
r = 6×1
    3.3253
   -2.9208
    0.0050
   -0.0050
    2.9208
   -3.3253
p = 6×1
   -9.9609
   -4.0054
    1.3332
    0.7501
   -0.2497
   -0.1004
k = 1
```

```
VG = K*(k+r(4)+r(5)+r(6))
```

```
VG = 376.8885
```

Thus, the variance-gain has been computed to be the value of 376.8885.

```
% Verify by taking large number of points
hn = filter(b1,a1,[1 zeros(1,9999)]);
sum(abs(hn).^2)
```

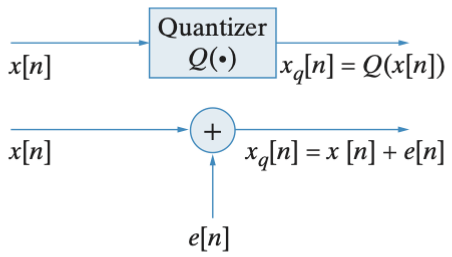
```
ans = 376.8885
```

```
% Verify using DSPUM Toolbox function
VG = VarGain(b1,a1)
```

```
VG = 376.8885
```

Problem 8.2

Consider the following statistical quantization error (noise) model (Figure 15.6 in the textbook).



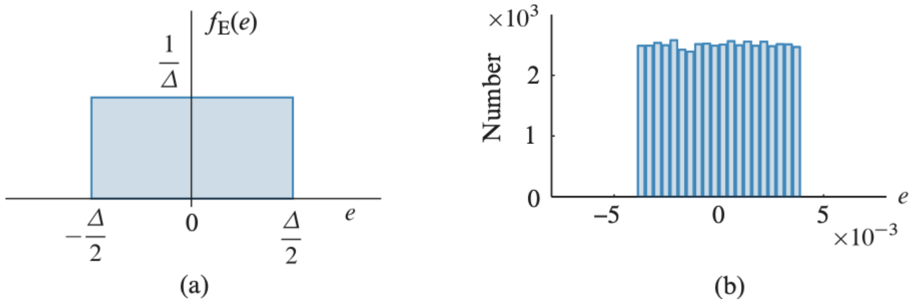
Under the conditions given in Section 15.2, the consecutive quantization noise $e[n]$ samples are uncorrelated. In this problem, we want to investigate under which conditions the consecutive noise samples, $e[n]$ and $e[n+1]$, are also statistically independent, that is, their joint probability density function (pdf) is a separable function of their marginal pdfs

$$f_{E_n,E_{n+1}}(x, y) = f_{E_n}(x) f_{E_{n+1}}(y). \qquad (8.2.1)$$

(a) Assume that $e[n]$ samples are statistically *independent and identically distributed* (IID) with uniform distribution over $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$, that is,

$$f_{E_n}(x) = f_{E_{n+1}}(e) = f_E(e) = \mathcal{U}\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right], \qquad (8.2.2)$$

where Δ is the quantization interval (Figure 15.8 in the textbook shown below).



Let $e_0[n] = \frac{1}{2\Delta} e[n]$ be the normalized quantization noise sequence and let

$$g[n] = e_0[n] + e_0[n+1] \qquad (8.2.3)$$

be the sum of the consecutive normalized quantization noise samples. Show that the pdf, $f_G(g)$, of $g[n]$ is given by the triangular distribution

$$f_G(g) = f_{E_0}(g) * f_{E_0}(g) = 2\Lambda(2g) \qquad (8.2.4)$$

where

$$\Lambda(g) \triangleq \begin{cases} 1 - |g|, & |g| \leq 1 \\ 0, & |g| > 1 \end{cases} \qquad (8.2.5)$$

is a triangular pulse between $-1 \leq g \leq 1$.

Solution:

Because we can assume that the samples of $e[n]$ are statistically independent and identically distributed, then the normalized quantization noise sequences

$e_0[n]$ and $e_0[n+1]$ carry this same property between them. Using the concept of independance of random variables,

if $Z = X + Y$ and both X and Y are independent from one eachother than the resulting pdf of Z, $f_Z = f_X * f_Y$ which is the convolution between the two probility density functions. Now since we can solve for $f_G(g)$ using only the pdf $f_{E_0}(g)$ this operation is performed using two uniformly distributed pdfs, which form the shape of rectangles, and the convolutional result of two rectangle is a triangle.

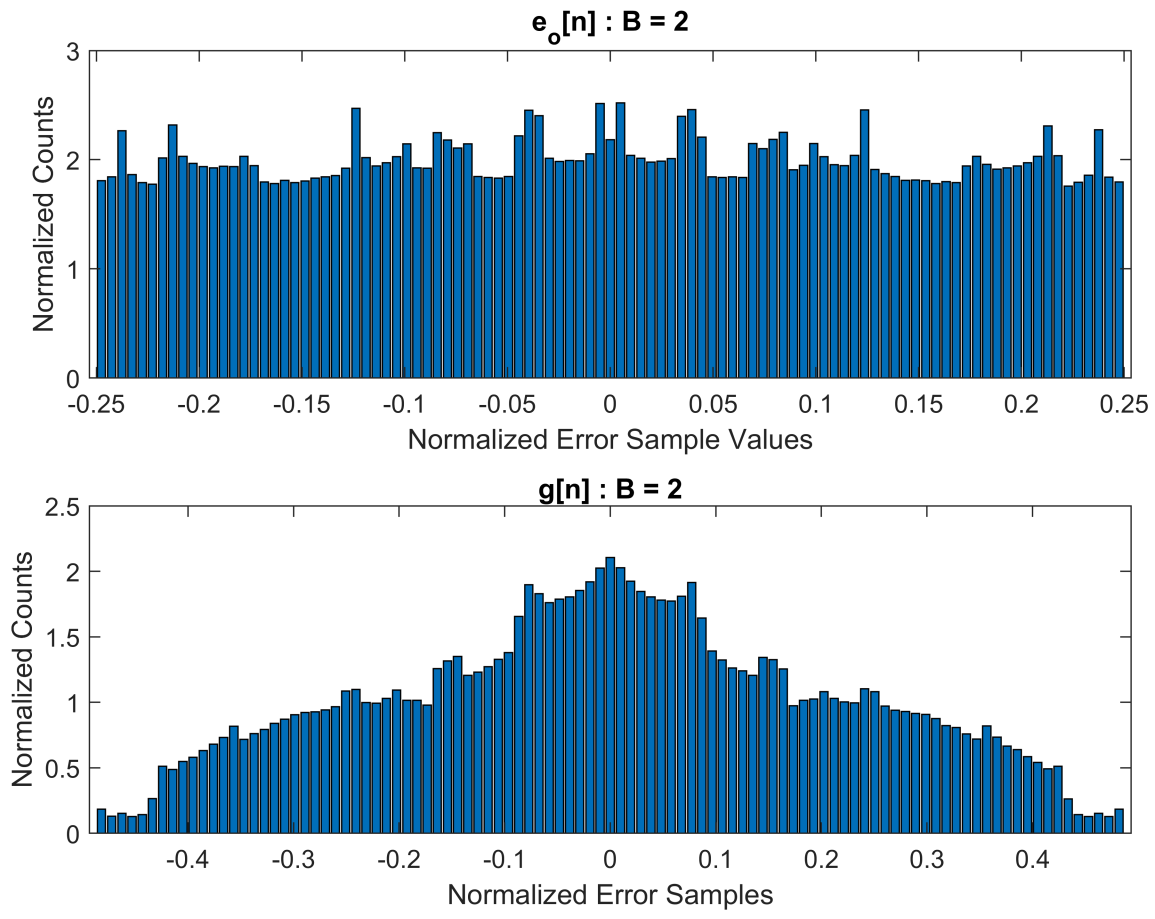
$$f_G(g) = f_{E_0}(g) * f_{E_0}(g)$$

$$f_G(g) = \sum_{k=0}^{N} f_{E_0}(k) f_{E_0}(g-k) = 2\Lambda(g)$$

(b) Let $x[n] = 0.5\left(\cos(n/7) + \sin(n/17)\right)$. Generate 500,000 samples of $x[n]$ and quantize $x[n]$ to $B+1=3$ bits where B is the number of fractional bits. Obtain the sequences $e_0[n]$ and $g[n]$. Compute and plot the histograms of both sequences in one figure using two rows and one column. Can you conclude that $e_0[n]$ is uniformly distributed and that its consecutive samples are independent? Explain.

Solution:

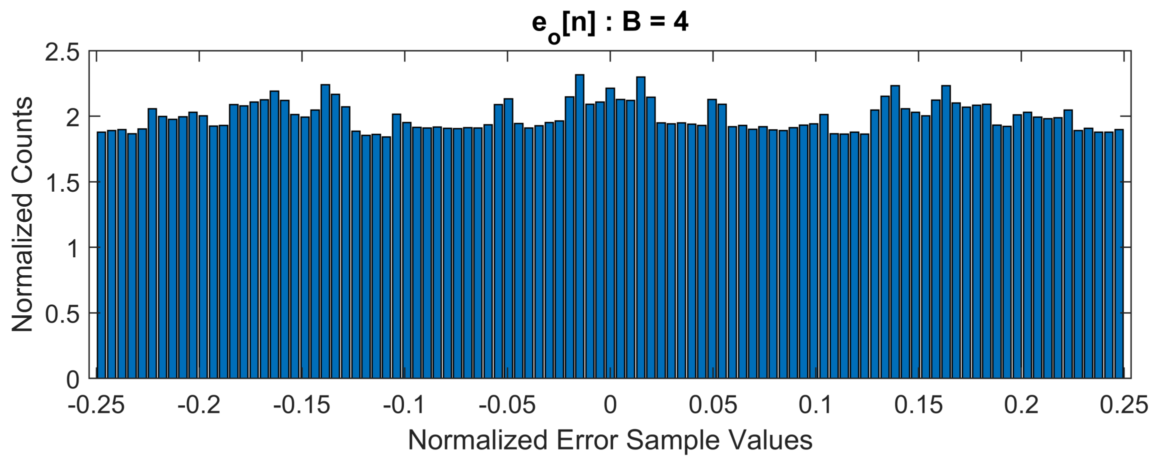
```
clc; clear; close all;
N = 500000; n = 1:N; xn = 0.5*(cos(n/7)+sin(n/17));
B = 2; L = B + 1; Nbins = 101;
[xqn,~,~] = dec2beqR(xn,L);
[xqbin, fx] = epdf(xn, Nbins);
Del = 2^-B; en = xqn - xn;
eo = en/(2*Del); eon = [0 eo];
eon1 = [eo 0]; gn = eon + eon1;
[eb, fe] = epdf(eo,Nbins);
[gb fG] = epdf(gn,Nbins);
figure('units','inches','position',[0,0,7,5])
subplot(2,1,1), bar(eb,fe), title('e_o[n] : B = 2')
xlabel('e_o[n]'), ylabel('Counts')
xlabel('Normalized Error Sample Values'), ylabel('Normalized Counts')
subplot(2,1,2), bar(gb,fG), title('g[n] : B = 2')
xlabel('Normalized Error Samples'), ylabel('Normalized Counts')
```

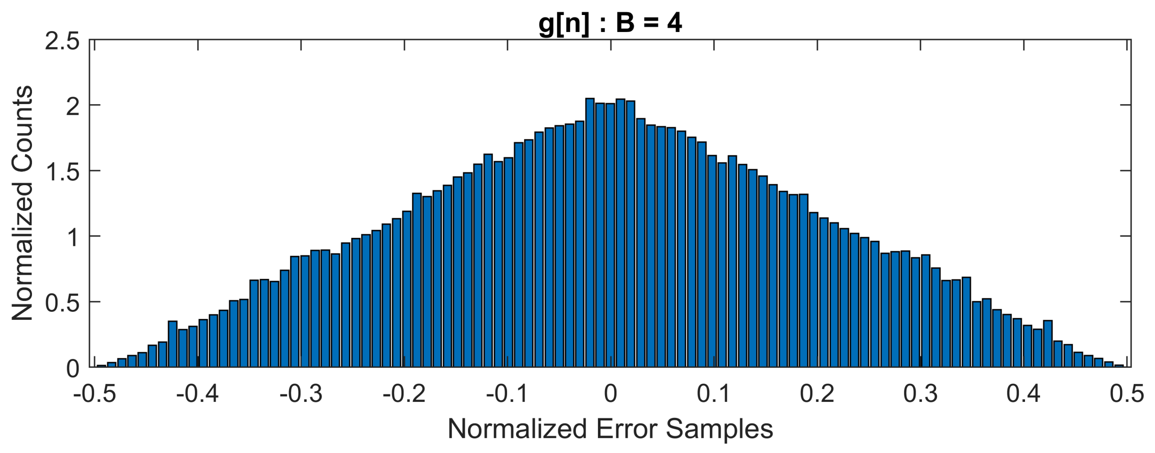


(c) Repeat part (b) using $B+1=5$ bits.

Solution:

```
B = 4; L = B + 1; Nbins = 101;
[xqn,~,~] = dec2beqR(xn,L);
[xqbin, fx] = epdf(xn, Nbins);
Del = 2^-B; en = xqn - xn;
eo = en/(2*Del); eon = [0 eo];
eon1 = [eo 0]; gn = eon + eon1;
[eb, fe] = epdf(eo,Nbins);
[gb fG] = epdf(gn,Nbins);
figure('units','inches','position',[0,0,7,5])
subplot(2,1,1), bar(eb,fe), title('e_o[n] : B = 4')
xlabel('e_o[n]'), ylabel('Counts')
xlabel('Normalized Error Sample Values'), ylabel('Normalized Counts')
subplot(2,1,2), bar(gb,fG), title('g[n] : B = 4')
xlabel('Normalized Error Samples'), ylabel('Normalized Counts')
```

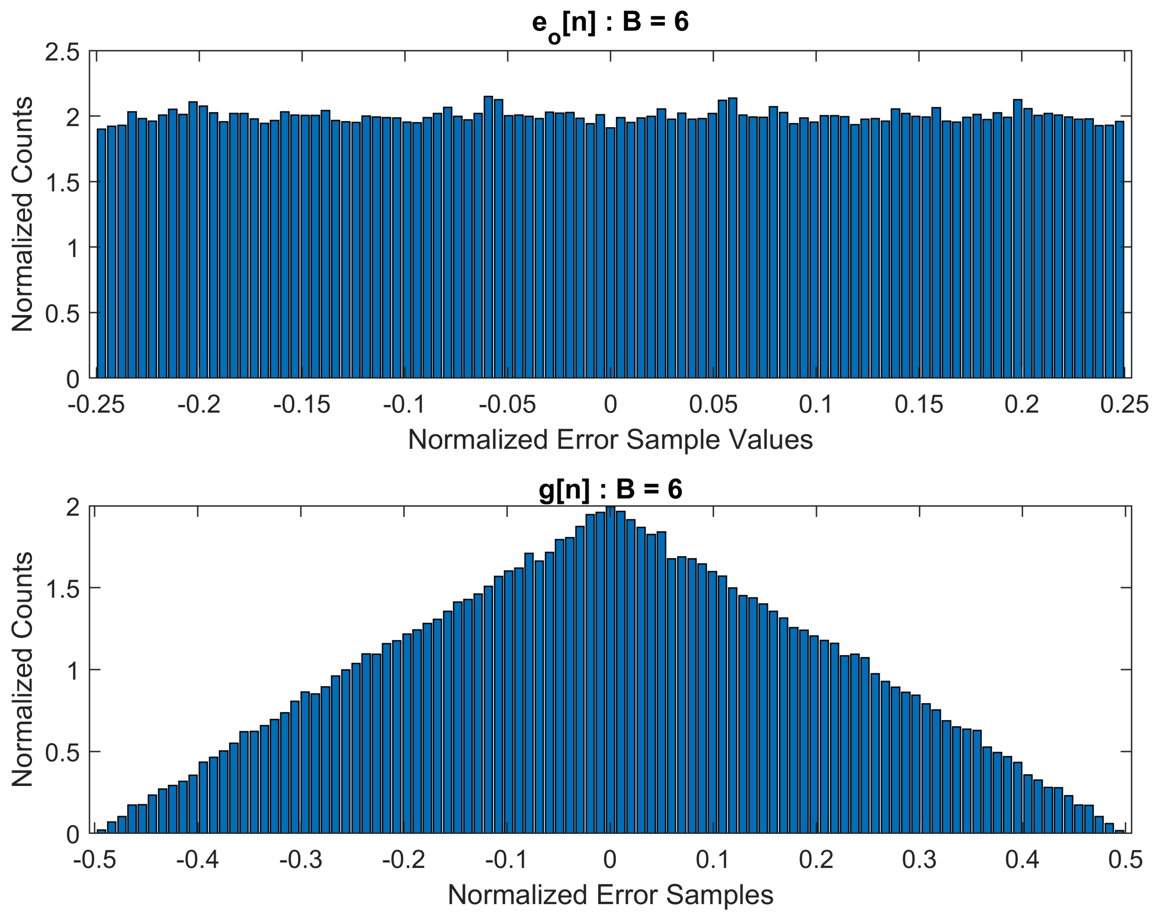




(**d**) Repeat part (b) using $B + 1 = 7$ bits.

Solution:

```
B = 6; L = B + 1; Nbins = 101;
[xqn,~,~] = dec2beqR(xn,L);
[xqbin, fx] = epdf(xn, Nbins);
Del = 2^-B; en = xqn - xn;
eo = en/(2*Del); eon = [0 eo];
eon1 = [eo 0]; gn = eon + eon1;
[eb, fe] = epdf(eo,Nbins);
[gb fG] = epdf(gn,Nbins);
figure('units','inches','position',[0,0,7,5])
subplot(2,1,1), bar(eb,fe), title('e_o[n] : B = 6')
xlabel('e_o[n]'), ylabel('Counts')
xlabel('Normalized Error Sample Values'), ylabel('Normalized Counts')
subplot(2,1,2), bar(gb,fG), title('g[n] : B = 6')
xlabel('Normalized Error Samples'), ylabel('Normalized Counts')
```



(e) What is the minimum value of B for which you can assume that consecutive samples are independent?

Solution:

After displaying the various quantization bits B, the minimum value for which can assume that the consecutive samples will be independent are for when $B = 6$, because the normalized error $e_0[n]$ is approximately uniformly distributed, which results in the appropriate triangle function when performing the convolution of the normalized error with itself.

Problem 8.3

Text Problem 15.30 (Page 961)

Consider the filter described by the system function

$$H[z] = \frac{1 - \sqrt{3}\,z^{-1}}{1 - z^{-1}/\sqrt{3}}, \quad |z| > 1/\sqrt{3}. \qquad (8.3.1)$$

```
clc; close all; clear;
```

(a) Show that this filter is an all-pass filter. Plot its magnitude response over $-\pi \le \omega \le \pi$ and verify.

Solution:

Taking the DT Fourier Transform of the $Z-\text{Transform}, \ H[z]$

$$H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}} = \frac{1-\sqrt{3}\,e^{-j\omega}}{1-\dfrac{1}{\sqrt{3}}e^{-j\omega}}$$

We notice that this filter has a pole at $p_1 = \dfrac{1}{\sqrt{3}}$ and a zero at $z_1 = \sqrt{3}$

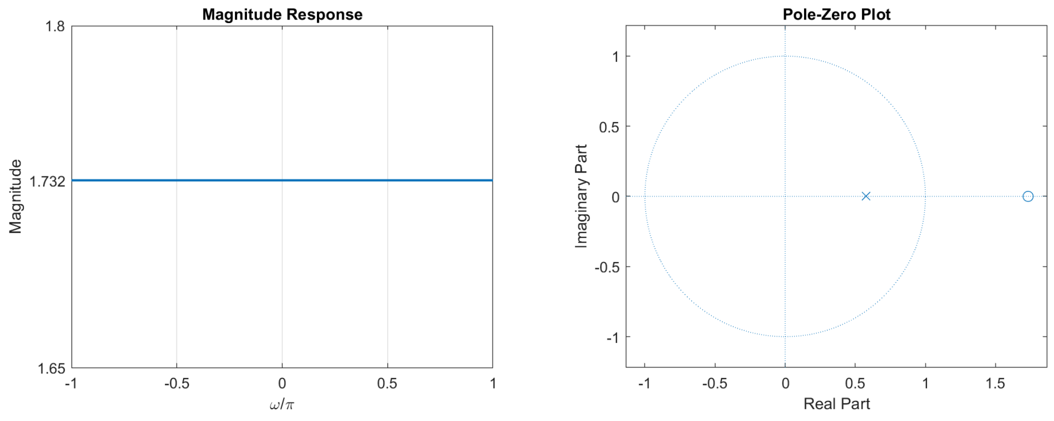
Thus, there is a complex reciprocal zero that is placed outside the unit circle for the pole that is within the unit circle, which is the definition of an allpass system.

MATLAB verification:

```
b = [1 -sqrt(3)]; a = [1 -1/sqrt(3)];
z = roots(b), p = roots(a)
```

```
z = 1.7321
p = 0.5774
```

```
omega = linspace(-pi,pi,1001);
H = freqz(b,a,omega);
Hmag = abs(H);
% Magnitude Response Plot
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1),plot(omega/pi,Hmag,'LineWidth',1.5),title('Magnitude Response'), grid on
xticks([-1 -0.5 0 0.5 1]), ylim([1.65 1.8]), yticks([1.65 1.732 1.8])
xlabel('\omega/\pi'),ylabel('Magnitude')
subplot(1,2,2), zplane(b,a), title('Pole-Zero Plot')
```



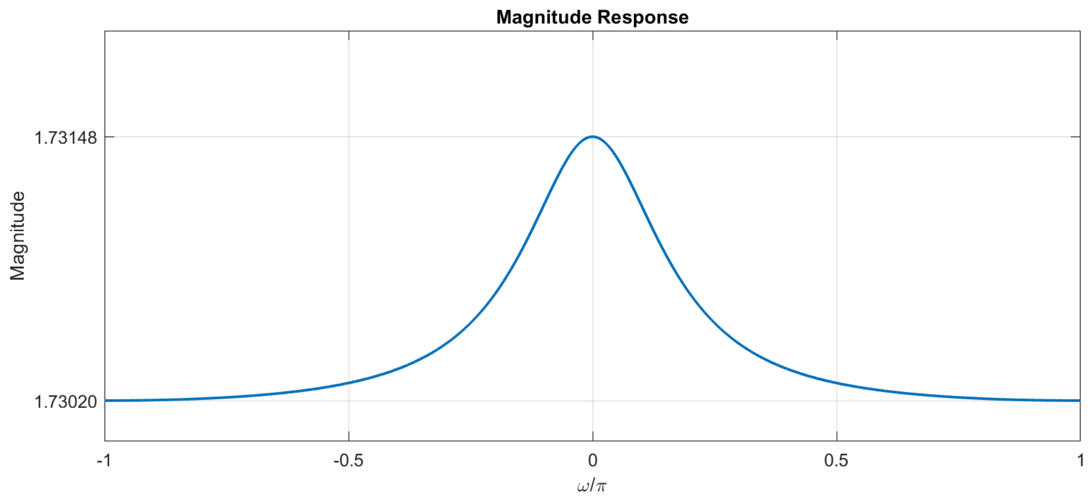
(b) Round the filter coefficients to $B = 8$ fraction bits using the **dec2beqR** function and then plot the magnitude response of the resulting filter. Is the filter still all-pass?

Solution:

```
B = 8; L1 = 2+B;
[BAhat,E1,B1] = dec2beqR([b;a],L1)
```

```
BAhat = 2×2
    1.0000   -1.7305
    1.0000   -0.5781
E1 = 1
B1 = 8
```

```
Bhat = BAhat(1,:); Ahat = BAhat(2,:);
BP = Bhat; AP = Ahat;
% Frequency Response
omega = linspace(-pi,pi,2001);
Hhat = freqz(BP,AP,omega);
HHmag = abs(Hhat);
HHdb = mag2db(Hmag);
% Magnitude Response Plot
figure('Units','inches','Position',[0,0,10,4]);
plot(omega/pi,HHmag,'LineWidth',1.5),title('Magnitude Response'), grid on
xticks([-1 -0.5 0 0.5 1]), ylim([1.73 1.732]), yticks([min(HHmag) max(HHmag)])
xlabel('\omega/\pi'),ylabel('Magnitude')
```



We observe from the Magnitude Response that the filter no longer has a constant G for all frequencies between the interval $[-\pi, \pi]$.

After rounding the filter coefficients to have $B = 8$ fractional bits, there is now a maximum boost for low frequencies of 1.73148 starting at $0 \, \frac{\text{rad}}{\text{sec}}$ and then attenuates to a value of $G = 1.7302$ at $\pm\pi$. Even though the gain term is subtle, this is no longer an Allpass system which should provide a constant G term for all frequencies.

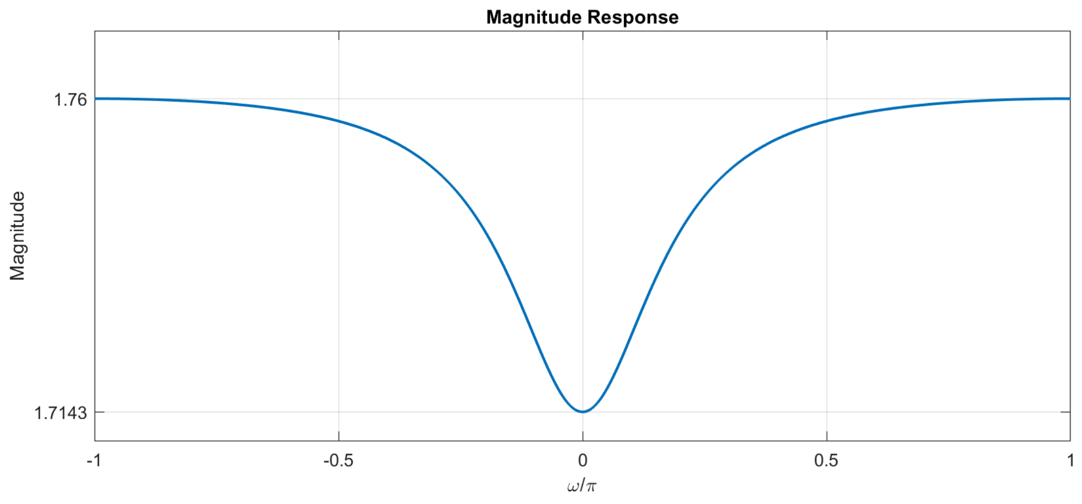
(c) Round the filter coefficients to $B = 4$ fractional bits using the **`dec2beqR`** function and then plot the magnitude response of the resulting filter. Is the filter still all-pass?

Solution:

```
B = 4; L2 = 2+B;
[BAhat,E2,B2] = dec2beqR([b;a],L2)
```

```
BAhat = 2×2
    1.0000   -1.7500
    1.0000   -0.5625
E2 = 1
B2 = 4
```

```
Bhat = BAhat(1,:); Ahat = BAhat(2,:);
BP = Bhat; AP = Ahat;
% Frequency Response
omega = linspace(-pi,pi,2001);
Hhat = freqz(BP,AP,omega);
HHmag = abs(Hhat);
HHdb = mag2db(Hmag);
% Magnitude Response Plot
figure('Units','inches','Position',[0,0,10,4]);
plot(omega/pi,HHmag,'LineWidth',1.5),title('Magnitude Response'), grid on
xticks([-1 -0.5 0 0.5 1]), ylim([1.71 1.77]), yticks([min(HHmag) 1.76])
xlabel('\omega/\pi'),ylabel('Magnitude')
```



After rounding the filter coefficients to have $B = 4$ fractional bits, we again observe the Magnitude response to determine if the filter still upholds the characteristic of an Allpass system. Here, we observe the opposite effect from the previous section, where now the filter has a minimum value of $G = 1.7143$ at $0\ \frac{\text{rad}}{\text{sec}}$ and then a higher value of $G = 1.76$ at the end points of $[-\pi, \pi]$. Since this filter does not consist of a constant G term for all frequencies, the resulting filter from rounding the filter coefficients is also not an Allpass system.

Problem 8.4

Text Problem 15.47 (Page 964)

Design a digital bandpass filter using the elliptic prototype to satisfy the requirements:

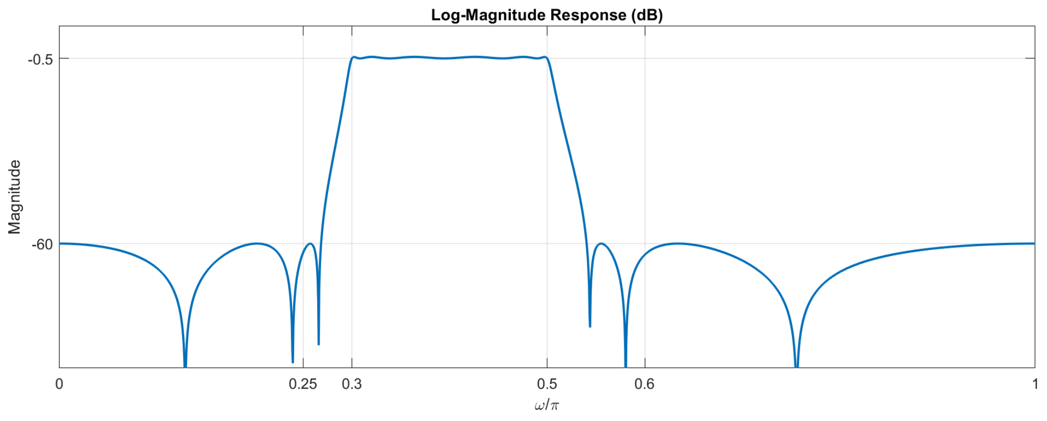
$$\omega_{s_1} = 0.25\pi,\ \omega_{p_1} = 0.3\pi,\ \omega_{p_2} = 0.5\pi,\ \omega_{s_2} = 0.6\pi,\ A_p = 0.5\text{ dB, and } A_s = 60\text{ dB.}$$

```
clc; close all; clear;
```

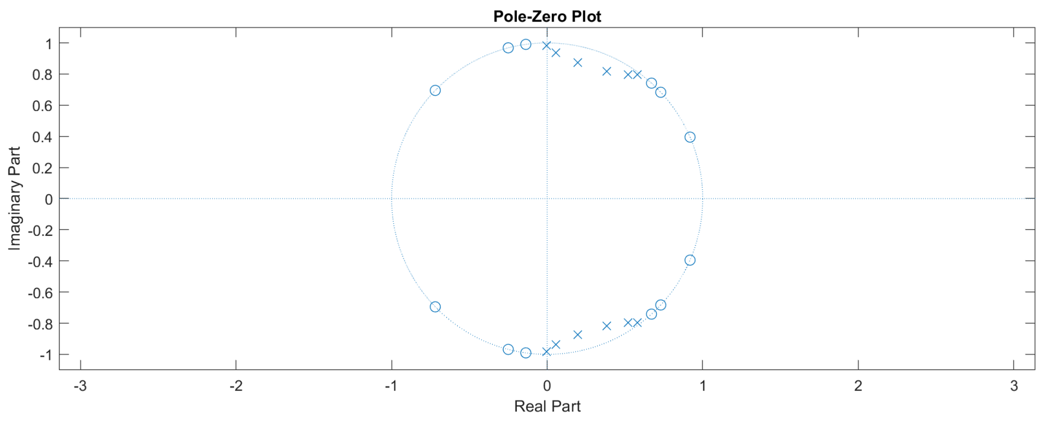
(a) Plot the magnitude response over $0 \le \omega \le \pi$ and pole-zero diagram of the filter.

MATLAB scipt:

```
omegas1 = 0.25*pi; omegap1 = 0.3*pi;
omegap2 = 0.5*pi; omegas2 = 0.6*pi;
Ap = 0.5; As = 60;
omegas = [omegas1 omegas2]/pi;
omegap = [omegap1 omegap2]/pi;
[N,omegac] = ellipord(omegap,omegas,Ap,As);
[b,a] = ellip(N,Ap,As,omegac);
% Frequency Response
omega = linspace(0,pi,2001);
H = freqz(b,a,omega);
Hmag = abs(H);
Hdb = mag2db(Hmag);
% Log-Magnitude Response Plot
figure('Units','inches','Position',[0,0,12,4]);
plot(omega/pi,Hdb,'LineWidth',1.5),title('Log-Magnitude Response (dB)'), grid on
xticks([0 0.25 0.3 0.5 0.6 1]), yticks([-60 -0.5]), ylim([-100 10]),
xlabel('\omega/\pi'),ylabel('Magnitude')
```



```
figure('Units','inches','Position',[0,0,12,4]); zplane(b,a), title('Pole-Zero Plot')
```



(b) Quantize the direct form coefficients to $L = 16$ bits using the **`dec2beqR`** function and plot the resulting magnitude response and pole-zero diagram.

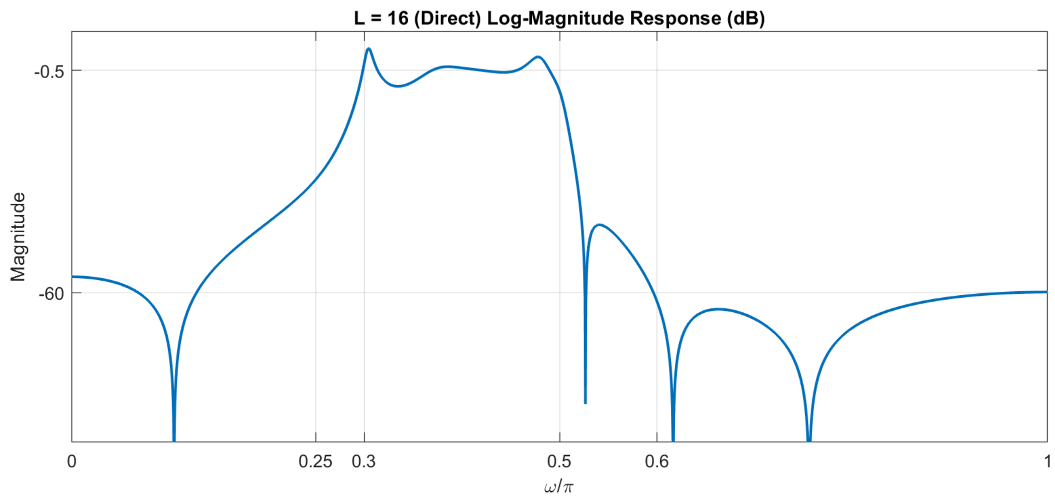
MATLAB scipt:

```
L1 = 16;
[BAhat,E1,B1] = dec2beqR([b;a],L1)
```

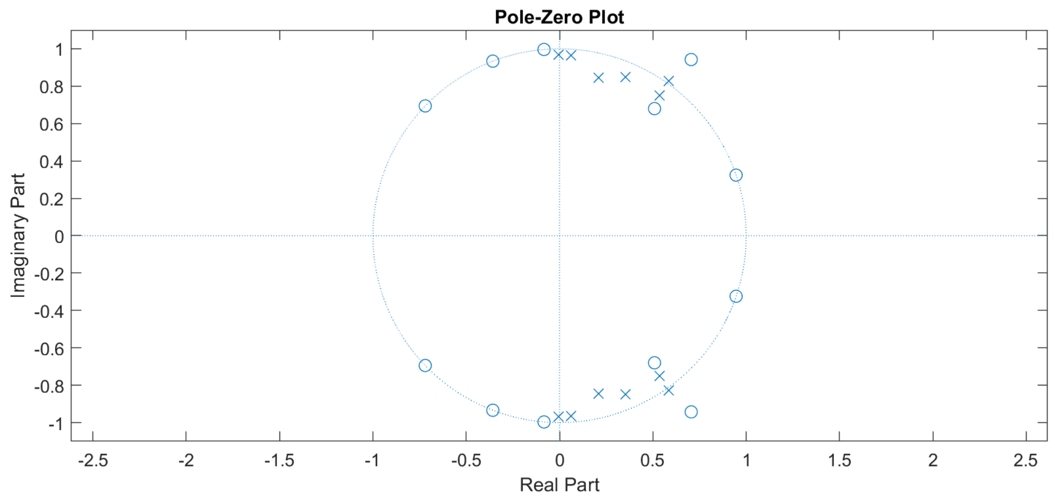
```
BAhat = 2×13
    0.0039   -0.0078    0.0137   -0.0176    0.0234   -0.0254    0.0254   -0.0254 ⋯
    1.0000   -3.4727    9.7734  -17.9453   28.2461  -34.2031   36.1855  -30.3906
E1 = 6
B1 = 9
```

Here when $L = 16$, the binary equivalent values of the direct form of the filter coefficients only have $B = 9$ fractional bits

```
Bhat = BAhat(1,:); Ahat = BAhat(2,:);
BP = Bhat; AP = Ahat;
% Frequency Response
omega = linspace(0,pi,2001);
H = freqz(BP,AP,omega);
Hmag = abs(H);
Hdb = mag2db(Hmag);
% Log-Magnitude Response Plot
figure('Units','inches','Position',[0,0,10,4]);
plot(omega/pi,Hdb,'LineWidth',1.5),title('L = 16 (Direct) Log-Magnitude Response (dB)'), grid o
xticks([0 0.25 0.3 0.5 0.6 1]), yticks([-60 -0.5]), ylim([-100 10]),
xlabel('\omega/\pi'),ylabel('Magnitude')
```



```
figure('Units','inches','Position',[0,0,10,4]); zplane(BP,AP), title('Pole-Zero Plot')
```



(c) Quantize the direct form coefficients to $L = 12$ bits using the **dec2beqR** function and plot the resulting magnitude response and pole-zero diagram.

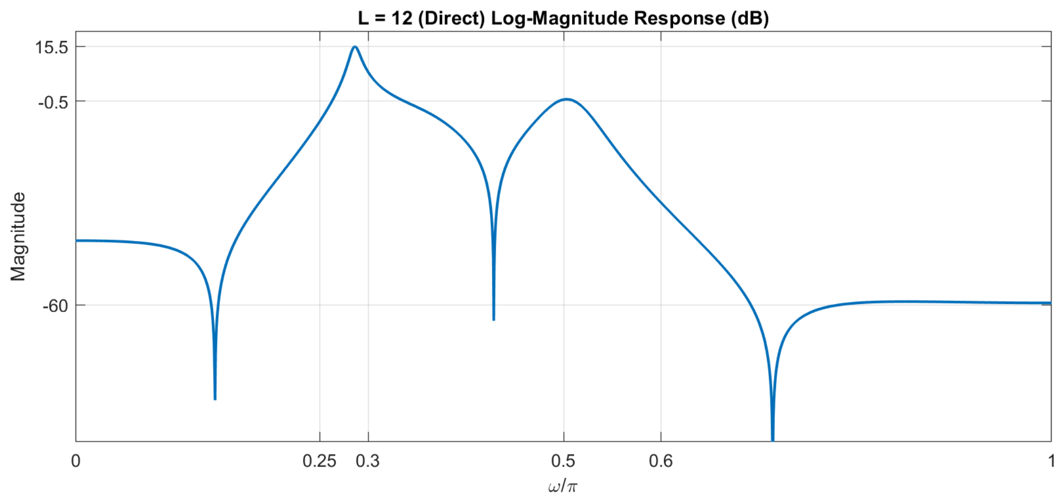
MATLAB scipt:

```
L2 = 12;
[BAhat,E2,B2] = dec2beqR([b;a],L2)
```

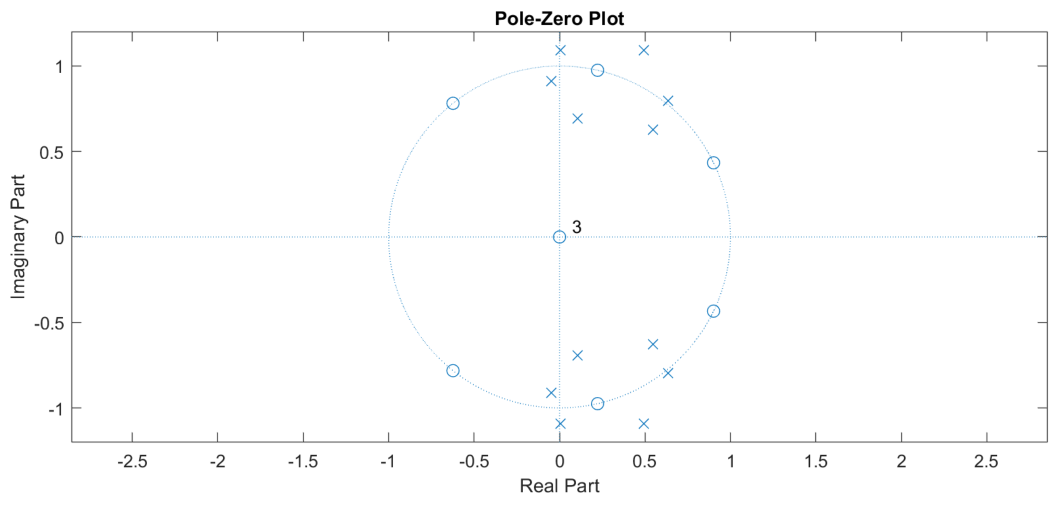
```
BAhat = 2×13
         0         0         0   -0.0313    0.0313   -0.0313    0.0313   -0.0313 ···
    1.0000   -3.4688    9.7813  -17.9375   28.2500  -34.2188   36.1875  -30.3750
E2 = 6
B2 = 5
```

Here when $L = 12$, the binary equivalent values of the direct form of the filter coefficients only have $B = 5$ fractional bits, which is even less fractional bits than the previous, so we should expect the results of the Frequency Response and Pole-Zero plot to be worse than the previous part.

```
Bhat = BAhat(1,:); Ahat = BAhat(2,:);
BP = Bhat; AP = Ahat;
% Frequency Response
omega = linspace(0,pi,2001);
H = freqz(BP,AP,omega);
Hmag = abs(H);
Hdb = mag2db(Hmag);
% Log-Magnitude Response Plot
figure('Units','inches','Position',[0,0,10,4]);
plot(omega/pi,Hdb,'LineWidth',1.5),title('L = 12 (Direct) Log-Magnitude Response (dB)'), grid o
xticks([0 0.25 0.3 0.5 0.6 1]), yticks([-60 -0.5 15.5]), ylim([-100 20]),
xlabel('\omega/\pi'),ylabel('Magnitude')
```



```
figure('Units','inches','Position',[0,0,10,4]); zplane(BP,AP), title('Pole-Zero Plot')
```



(d) Quantize the cascade form coefficients to $L = 12$ bits using the **`dec2beqR`** function and plot the resulting magnitude response and pole-zero diagram.

MATLAB scipt:

```
[sos G] = tf2sos(b,a), L3 = 12;
```

```
sos = 6×6
    1.0000    1.4384    1.0000    1.0000   -0.3948    0.7995
    1.0000   -1.8373    1.0000    1.0000   -0.7666    0.8125
    1.0000    0.5000    1.0000    1.0000   -0.1160    0.8837
    1.0000   -1.4609    1.0000    1.0000   -1.0381    0.9014
    1.0000    0.2743    1.0000    1.0000    0.0052    0.9661
    1.0000   -1.3417    1.0000    1.0000   -1.1629    0.9726
G = 0.0032
```

```
[SOShat,E3,B3] = dec2beqR(sos,L3)
```

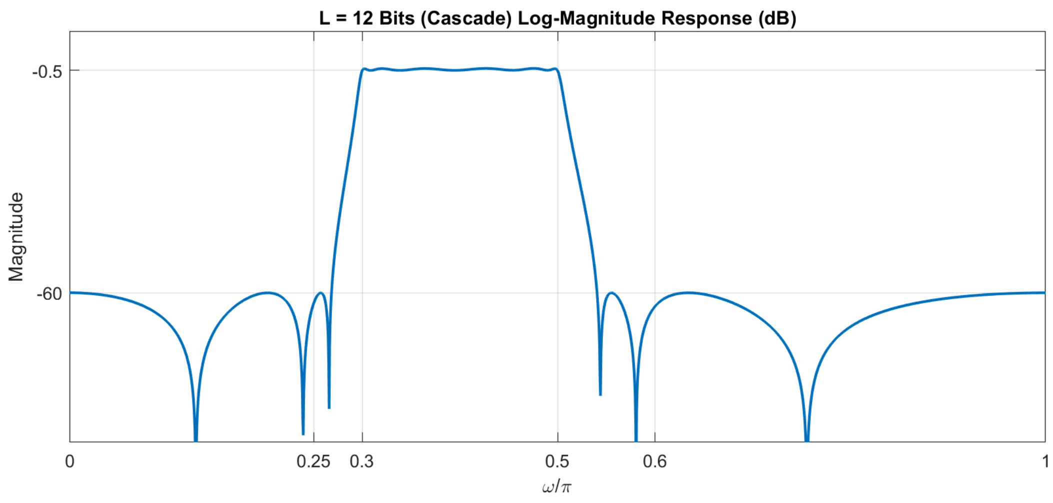
```
SOShat = 6×6
    1.0000    1.4385    1.0000    1.0000   -0.3945    0.7998
    1.0000   -1.8369    1.0000    1.0000   -0.7666    0.8125
    1.0000    0.5000    1.0000    1.0000   -0.1162    0.8838
    1.0000   -1.4609    1.0000    1.0000   -1.0381    0.9014
    1.0000    0.2744    1.0000    1.0000    0.0049    0.9658
    1.0000   -1.3418    1.0000    1.0000   -1.1631    0.9727
E3 = 1
B3 = 10
```

```
[Bhat,Ahat] = sos2tf(SOShat,G)
```

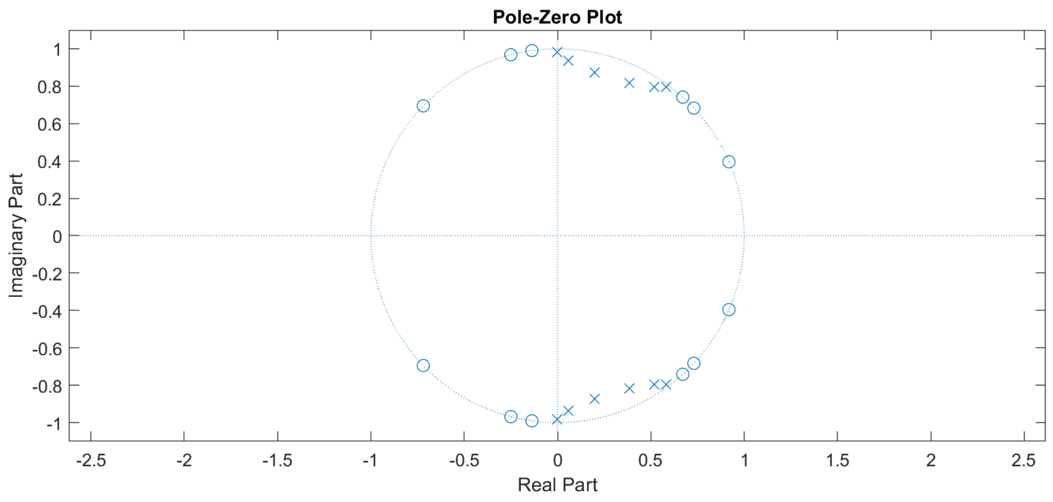
```
Bhat = 1×13
    0.0032   -0.0077    0.0130   -0.0178    0.0234   -0.0247    0.0249   -0.0247 ···
Ahat = 1×13
    1.0000   -3.4736    9.7754  -17.9501   28.2526  -34.2136   36.1955  -30.3998 ···
```

Here when $L = 12$, the binary equivalent values of the cascade form of the filter coefficients have $B = 10$ fractional bits, which provides the most precision out of all the cases thus far. We should now expect to see an appropriate Frequency Response of the filter and all of its Poles and Zeros contained within the unit circle.

```
BP = Bhat; AP = Ahat;
% Frequency Response
omega = linspace(0,pi,2001);
H = freqz(BP,AP,omega);
Hmag = abs(H);
Hdb = mag2db(Hmag);
% Log-Magnitude Response Plot
figure('Units','inches','Position',[0,0,10,4]);
plot(omega/pi,Hdb,'LineWidth',1.5),title('L = 12 Bits (Cascade) Log-Magnitude Response (dB)'),
xticks([0 0.25 0.3 0.5 0.6 1]), yticks([-60 -0.5]), ylim([-100 10]),
xlabel('\omega/\pi'),ylabel('Magnitude')
```



```
figure('Units','inches','Position',[0,0,10,4]); zplane(Bhat,Ahat), title('Pole-Zero Plot')
```



(e) Comment on your plots.

Comments:

Analyzing the direct form coefficients of the bandpass filter after rounding, we can see that when $L = 16$ the coefficients are represented using 6 integers bits and 9 fractional bits. The Magnitude Response displays a poorly designed filter that does not meet the design specfications and the Pole-Zero plots shows that one pair of complex poles are outside the unti circle. Thus, this filter is both unstable and unusable to satisfy the design requirements.

When observing the direct form filter coefficients rounded to $L = 12$ bits, the filter coefficients have been rounded to 6 integer and 5 fractional bits, which is even less precision than the previous part. The Magnitude Response displays even worse results than the previous and now three pairs of complex poles are outside the unit circle, which makes this filter even more unstable and, again, unusable to meet the design specifications.

Finally, when observing the cascade form filter coefficients when $L = 12$, the coefficients in cascade form are represented using 1 integer bit and 10 fractional bits, which is typical for cascade form coefficients since they are represented by smaller values. The resulting Magnitude Response satisfies the design requirements and the Pole-Zero plots displays all the appropriate poles within the unit circle verifying stability. These last plots demonstrate that cascade forms are preferred over direct forms for filter coefficient quantization.

Problem 8.5

Text Problem 15.50 (Page 965)

Consider the signal $x[n] = 0.49\left[\cos(n/13) + \sin(n/29)\right]$. It is multiplied by a constant $a = 0.3333$ and then the product is quantized to B fractional bits. Let $e[n] = Q(ax[n]) - ax[n]$. Generate $100,000$ samples of $x[n]$ and use the **`QNmodel`** function introduced in Problem 5 to answer the following parts.

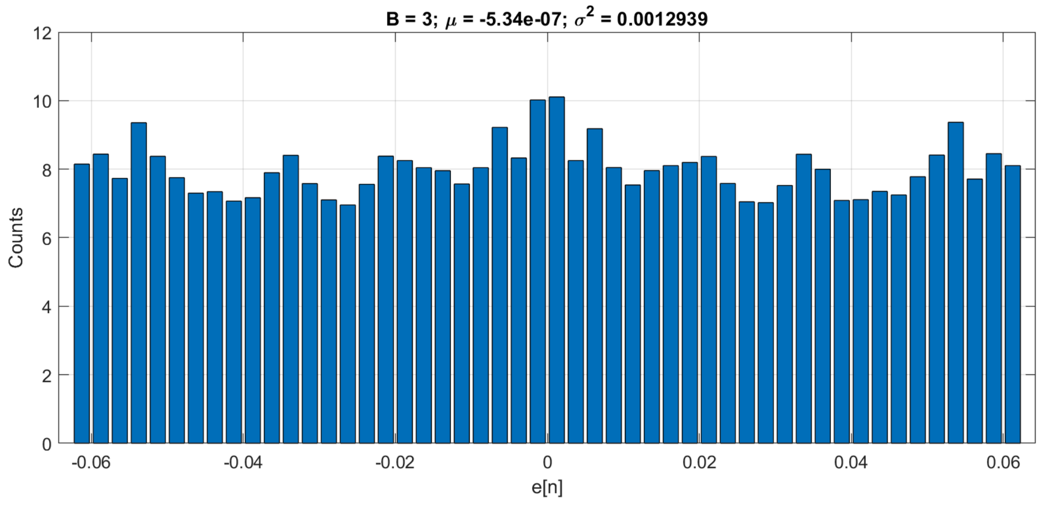
Note: The **`QNmodel`** function is developed in Chapter-15 Tutorial Problem 5. It is not a part of the book toolbox.

```
clc; close all; clear;
```

(a) Quantize $ax[n]$ to $B = 3$ bits and plot the histogram of the resulting error sequence.

MATLAB scipt:

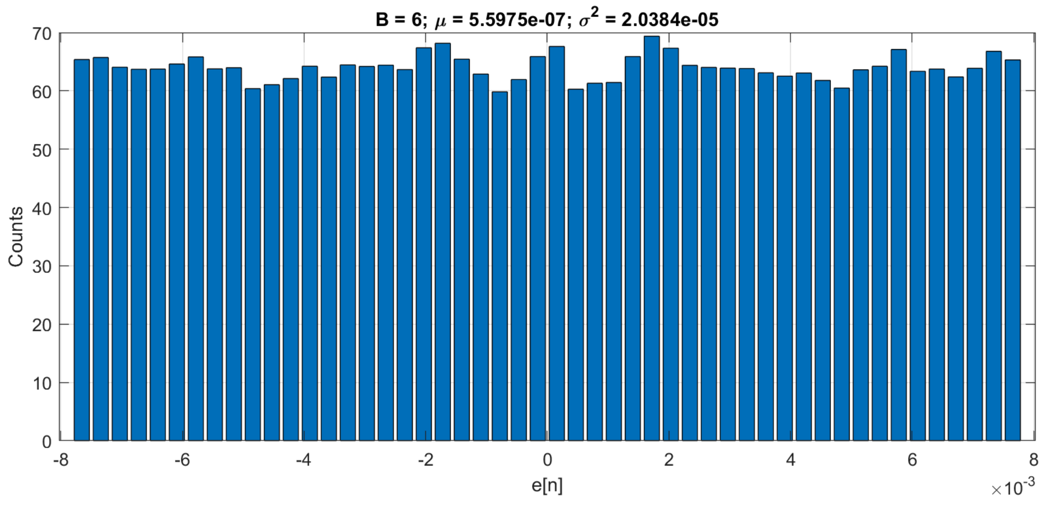
```
N = 100000; n = 1:N; a = 0.3333;
xn = 0.49*(cos(n/13)+sin(n/29));
B1 = 3; yn = a*xn;
[pe,eo,eavg,evar] = QNmodel(yn,B1);
figure('Units','inches','Position',[0,0,10,4]);
bar(eo,pe),grid on,xlabel('e[n]'),ylabel('Counts')
title(['B = ',num2str(B1),'; \mu = ',num2str(eavg),...
    '; \sigma^2 = ',num2str(evar)])
```



(b) Quantize $ax[n]$ to $B = 6$ bits and plot the histogram of the resulting error sequence.

MATLAB scipt:

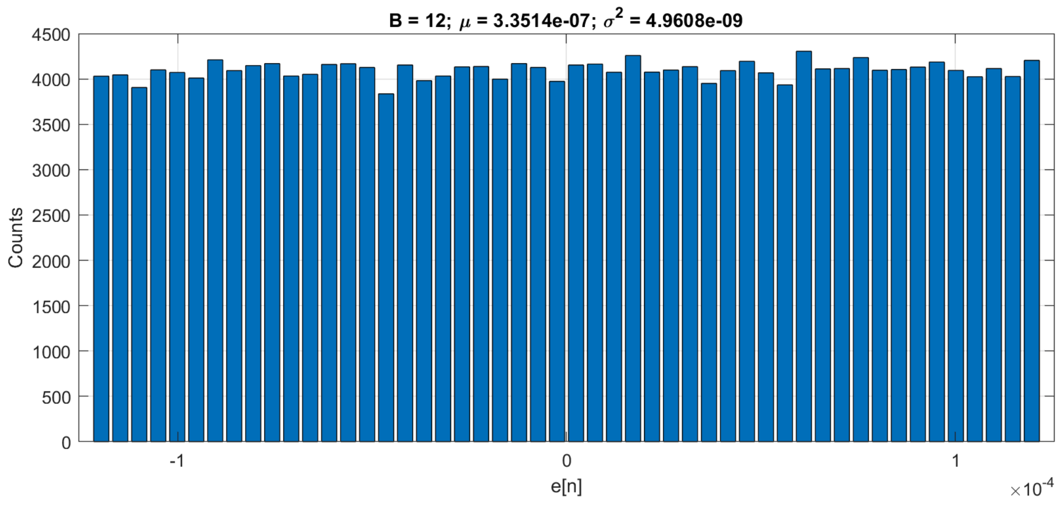
```
B2 = 6;
[pe,eo,eavg,evar] = QNmodel(yn,B2);
figure('Units','inches','Position',[0,0,10,4]);
bar(eo,pe),grid on,xlabel('e[n]'),ylabel('Counts')
title(['B = ',num2str(B2),'; \mu = ',num2str(eavg),...
   '; \sigma^2 = ',num2str(evar)])
```



(c) Quantize $ax[n]$ to $B = 12$ bits and plot the histogram of the resulting error sequence.

MATLAB scipt:

```
B3 = 12;
[pe,eo,eavg,evar] = QNmodel(yn,B3);
figure('Units','inches','Position',[0,0,10,4]);
bar(eo,pe),grid on,xlabel('e[n]'),ylabel('Counts')
title(['B = ',num2str(B3),'; \mu = ',num2str(eavg),...
   '; \sigma^2 = ',num2str(evar)])
```



(d) Comment on your histogram plots and on the validity of the multiplication quantization model.

Comments:

Multiplying $x[n]$ by the value of $a = 0.333$ will require a certain finite level of precision to properly represent the values after quantization. We notice as we start with $B = 3$ fractional bits, the error histogram is fairly random in height. As we increase the number of quantized bits to the final value of $B = 12$, we see that the error $e[n]$ becomes more uniformly distributed, which is verifies that the values of the sequence nonperiodic sequence have more precision at representing the result of the multiplication quantization model.

Problem 8.6

Consider the design of a 32-length linear-phase FIR bandpass filter that satisfies requirements of 60-dB stopband attenuation, lower stopband edge-frequency $\omega_{s_1} = 0.2\pi$, and upper stopband edge-frequency $\omega_{s_2} = 0.8\pi$.

```
clc; close all; clear;
```

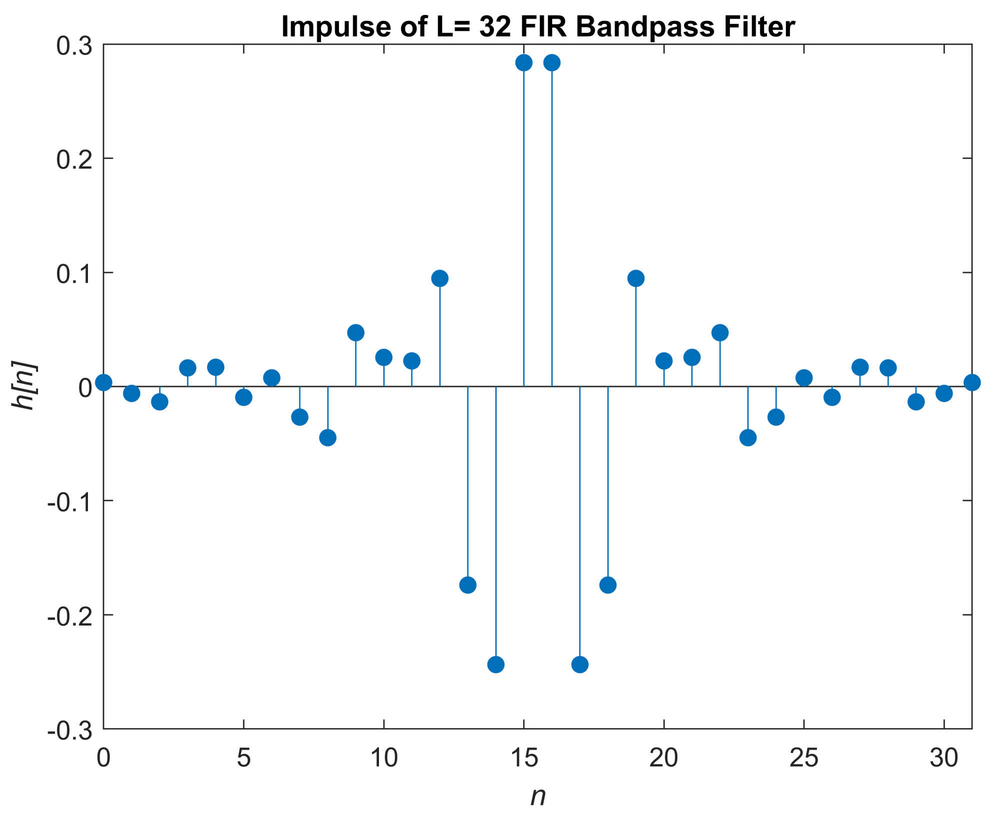
(a) Using the Parks-McClellan algorithm, design the above mentioned FIR bandpass filter. Since no other requirements are given, you are free to choose any additional needed parameters. The resulting filter impulse response $h[n]$ can be considered to have infinite precision.

Solution:

MATLAB scipt:

```
omegas1 = 0.2*pi; omegap1 = 0.35*pi;
omegap2 = 0.65*pi; omegas2 = 0.8*pi;
```

```
Ap = 1; As = 60;
[dp,ds1] = spec_convert(Ap,As,'rel','abs');
[dp,ds2] = spec_convert(Ap,As,'rel','abs');
% Estimate Filter using FIRPMORD function
f = [omegas1,omegap1,omegap2,omegas2]/pi; % Band-edge array
a = [0 1 0]; % Band-edge desired gain
dev = [ds1 dp ds2]; % Band tolerance
[M,fo,ao,W] = firpmord(f,a,dev); M = 31; L = M+1;
% Filter Design using FIRPM function
[h,delta] = firpm(M,fo,ao,W);
n = 0:L-1; omega = linspace(0,pi,1001);
figure
stem(n,h,"filled"), title('Impulse of L= 32 FIR Bandpass Filter')
xlabel('\it{n}'), ylabel('\it{h[n]}'), xlim([0 M])
```



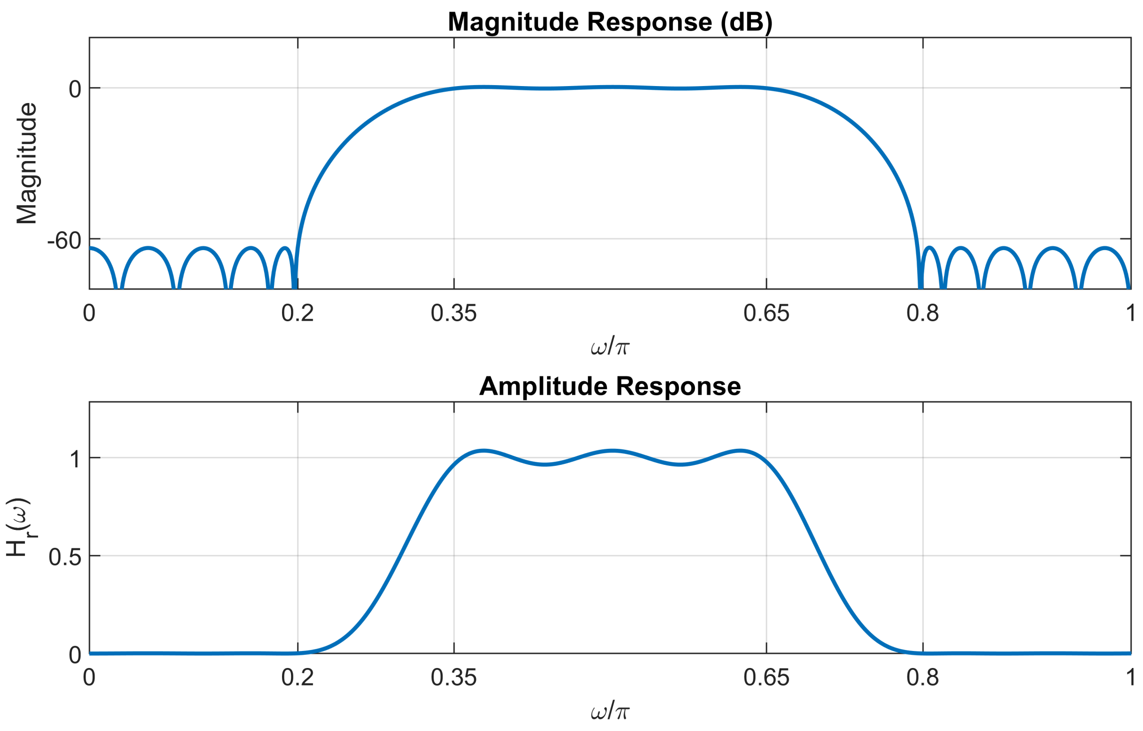
(b) Using infinite precision, plot the log-magnitude and amplitude responses of the designed filter. Use 2 rows and 1 column of subplots.

MATLAB scipt:

```
% Magnitude Response
H = freqz(h,1,omega); Hmag = abs(H);
Hdb = mag2db(Hmag);
% Amplitude Response
[Ha w2 P2 L2] = amplresp(h,omega);
```

```
*** Type-2 Linear-Phase Filter ***
```

```
% Magnitude Response Plot
figure('Units','inches','Position',[0,0,7,4]);
subplot(2,1,1), plot(omega/pi,Hdb,'LineWidth',1.5)
title('Magnitude Response (dB)'), grid on
xticks([0 0.2 0.35 0.65 0.8 1]), yticks([-60 0]), ylim([-80 20]),
xlabel('\omega/\pi'),ylabel('Magnitude')
% Amplitude Response Plot
subplot(2,1,2), plot(omega/pi,Ha,'LineWidth',1.5), title('Amplitude Response')
xlabel('\omega/\pi'), ylabel('H_r(\omega)'), xticks([0 0.2 0.35 0.65 0.8 1])
grid on, ylim([min(Ha) max(Ha)+.25])
```



(c) Quantize the direct-form coefficients to 4 decimals (by rounding). Now plot the log-magnitude and amplitude responses of the resulting filter. Use 2 rows and 1 column of subplots.

MATLAB scipt:

```
hhat = round(h,4)
```

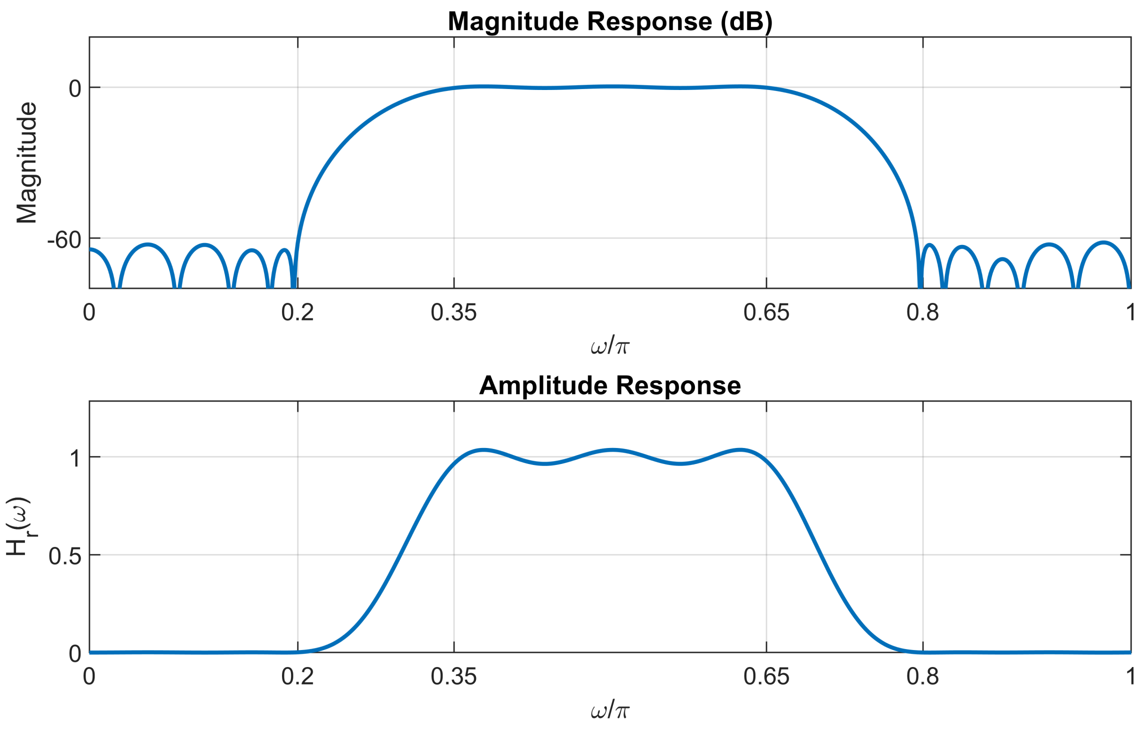
```
hhat = 1×32
    0.0035   -0.0061   -0.0134    0.0163    0.0169   -0.0095    0.0076   -0.0268 ⋯
```

```
% Magnitude Response
Hhat = freqz(hhat,1,omega); HHmag = abs(Hhat);
HHdb = mag2db(HHmag);
% Amplitude Response
[HHa w2 P2 L2] = amplresp(hhat,omega);
```

```
*** Type-2 Linear-Phase Filter ***
```

```
% Magnitude Response Plot
figure('Units','inches','Position',[0,0,7,4]);
```

```
subplot(2,1,1), plot(omega/pi,HHdb,'LineWidth',1.5)
title('Magnitude Response (dB)'), grid on
xticks([0 0.2 0.35 0.65 0.8 1]), yticks([-60 0]), ylim([-80 20]),
xlabel('\omega/\pi'),ylabel('Magnitude')
% Amplitude Response Plot
subplot(2,1,2), plot(omega/pi,HHa,'LineWidth',1.5), title('Amplitude Response')
xlabel('\omega/\pi'), ylabel('H_r(\omega)'), xticks([0 0.2 0.35 0.65 0.8 1])
grid on, ylim([min(HHa) max(HHa)+.25])
```



(d) Quantize the direct-form coefficients to 3 decimals (by rounding). Now plot the log-magnitude and amplitude responses of the resulting filter. Use 2 rows and 1 column of subplots.

MATLAB scipt:

```
hhat = round(h,3)
```

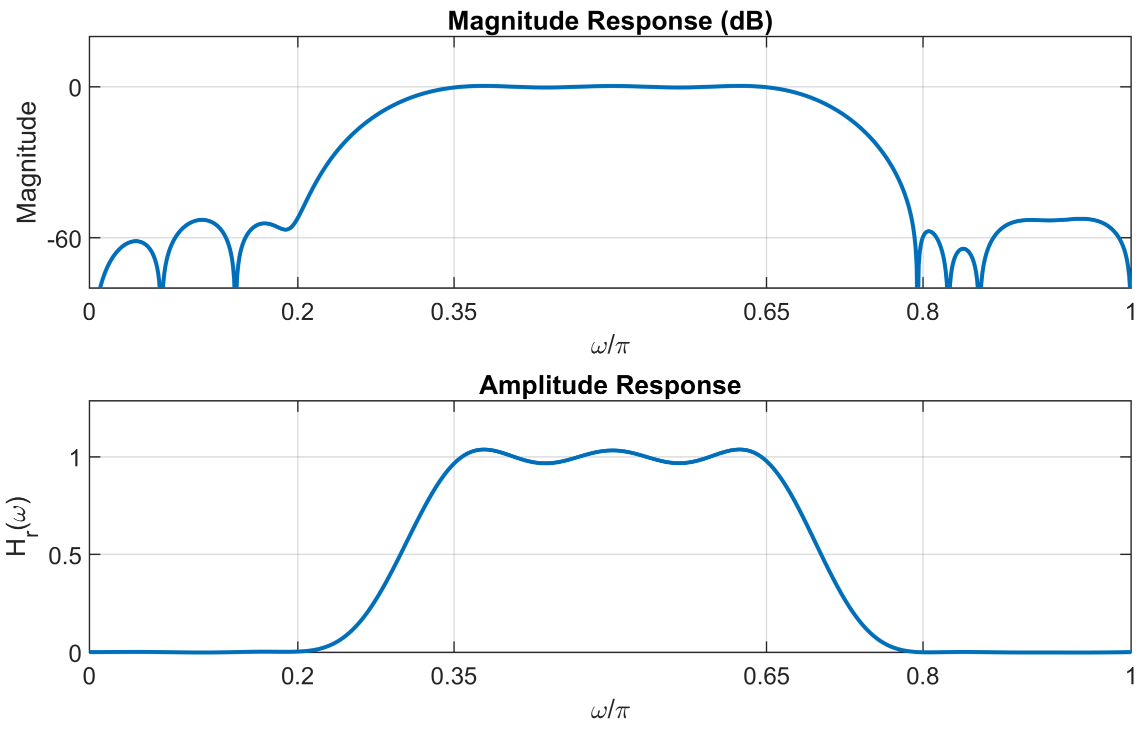
```
hhat = 1×32
    0.0030   -0.0060   -0.0130    0.0160    0.0170   -0.0090    0.0080   -0.0270 ···
```

```
% Magnitude Response
Hhat = freqz(hhat,1,omega); HHmag = abs(Hhat);
HHdb = mag2db(HHmag);
% Amplitude Response
[HHa w2 P2 L2] = amplresp(hhat,omega);
```

```
*** Type-2 Linear-Phase Filter ***
```

```
% Magnitude Response Plot
figure('Units','inches','Position',[0,0,7,4]);
subplot(2,1,1), plot(omega/pi,HHdb,'LineWidth',1.5)
```

```
title('Magnitude Response (dB)'), grid on
xticks([0 0.2 0.35 0.65 0.8 1]), yticks([-60 0]), ylim([-80 20]),
xlabel('\omega/\pi'),ylabel('Magnitude')
% Amplitude Response Plot
subplot(2,1,2), plot(omega/pi,HHa,'LineWidth',1.5), title('Amplitude Response')
xlabel('\omega/\pi'), ylabel('H_r(\omega)'), xticks([0 0.2 0.35 0.65 0.8 1])
grid on, ylim([min(HHa) max(HHa)+.25])
```



(e) Comment on the plots in parts (b), (c), and (d).

Comment:

With the infinite precision of the FIR coefficients, we can see in plots b.) that the design specifications are met. When we round the coefficients to have only 4 decimal places, the plots in part c.) still satisfy the design requirements. However, when we round to only 3 decimal places, the stopband attentuation is not satisfied in part d.), even though the amplitude response still looks reasonable. This shows that when rounding to a certain decimal place we are not performing the same equivalent rounding in binary fractional bits, as the round() is just simply rounding the binary bits to satisfy the closest decimal place specified.

(f) Based on the results of this problem, determine how many significant *bits* (and not decimals) are needed in practice to represent FIR direct form realizations.

Answer:

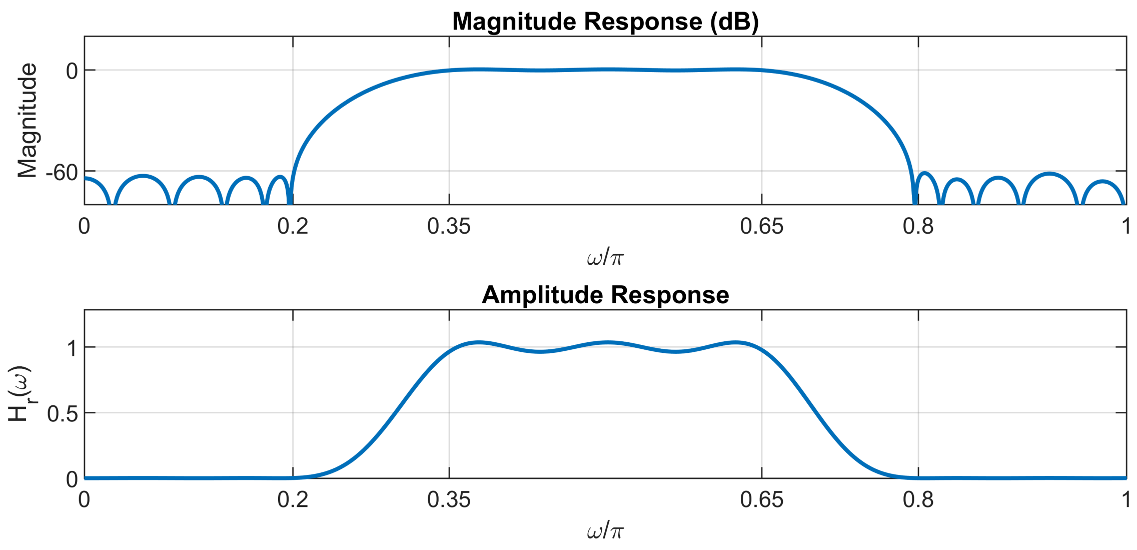
```
B = 14; L = 1+B;
[hhat,E,B] = dec2beqR(h,L)
```

```
hhat = 1×32
    0.0035   -0.0060   -0.0134    0.0163    0.0169   -0.0095    0.0076   -0.0268 ···
E = 0
B = 14
```

```
% Magnitude Response
Hhat = freqz(hhat,1,omega); HHmag = abs(Hhat);
HHdb = mag2db(HHmag);
% Amplitude Response
[HHa w2 P2 L2] = amplresp(hhat,omega);
```

```
*** Type-2 Linear-Phase Filter ***
```

```
% Magnitude Response Plot
figure('Units','inches','Position',[0,0,7,3]);
subplot(2,1,1), plot(omega/pi,HHdb,'LineWidth',1.5)
title('Magnitude Response (dB)'), grid on
xticks([0 0.2 0.35 0.65 0.8 1]), yticks([-60 0]), ylim([-80 20]),
xlabel('\omega/\pi'),ylabel('Magnitude')
% Amplitude Response Plot
subplot(2,1,2), plot(omega/pi,HHa,'LineWidth',1.5), title('Amplitude Response')
xlabel('\omega/\pi'), ylabel('H_r(\omega)'), xticks([0 0.2 0.35 0.65 0.8 1])
grid on, ylim([min(HHa) max(HHa)+.25])
```



After experimenting with number of bits B, I found that using $B = 14$ fractional bits was the minimum number of significant bits that satisfied my design specifications. Any value lower than this did not meet the design requirements for the stopband attenuation.

Problem 8.7

Consider the third-order elliptic lowpass filter given by

$$H(z) = \frac{0.1214\left(1 - 1.4211z^{-1} + z^{-2}\right)\left(1 + z^{-1}\right)}{\left(1 - 1.4928z^{-1} + 0.8612z^{-2}\right)\left(1 - 0.6183z^{-1}\right)}.$$

(a) If the filter is realized using a direct form structure, determine its pole sensitivity given by eq. (15.70), from the textbook, for $L = 16$ bit fixed-point number representation. Use MATLAB for these calculations.

MATLAB script:

```
clc; close all; clear;
% Determine Direct Form Coefficients
b_dir = 0.1214*conv([1 -1.4211 1],[1 1])
```

```
b_dir = 1×4
    0.1214   -0.0511   -0.0511    0.1214
```

```
a_dir = conv([1 -1.4928 0.8612],[1 -0.6183])
```

```
a_dir = 1×4
    1.0000   -2.1111    1.7842   -0.5325
```

```
% Quantize the poles to L = 16 bit fixed-point number representation
L = 16; B = L-1;
[ahat,~,~] = dec2beqR(a_dir(2:end),L)
```

```
ahat = 1×3
   -2.1111    1.7842   -0.5325
```

```
delta_a = abs(ahat - a_dir(2:end))
```

```
delta_a = 1×3
10^-4 ×
    0.1602    0.1855    0.0926
```

```
poles = roots(a_dir)
```

```
poles = 3×1 complex
   0.7464 + 0.5514i
   0.7464 - 0.5514i
   0.6183 + 0.0000i
```

```
N = 3; % # of poles
k = 1:N;
% Delta P1
p1 = poles(1); delta_a1 = delta_a(1);
denom1 = (poles(1)-poles(2))*(poles(1)-poles(3));
delta_p1 = max(sum(abs(delta_a1).*abs(p1.^(N-k)./denom1)))
```

```
delta_p1 = 7.1546e-05
```

```
% Delta P2
p2 = poles(2); delta_a2 = delta_a(2);
denom2 = (poles(2)-poles(1))*(poles(2)-poles(3));
delta_p2 = max(sum(abs(delta_a2).*abs(p2.^(N-k)./denom2)))
```

```
delta_p2 = 8.2879e-05
```

```
% Delta P3
p3 = poles(3); delta_a3 = delta_a(3);
denom3 = (poles(3)-poles(1))*(poles(3)-poles(2));
delta_p3 = max(sum(abs(delta_a3).*abs(p3.^(N-k)./denom3)))
```

```
delta_p3 = 5.7783e-05
```

```
directSensitivity = abs([delta_p1 delta_p2 delta_p3])
```

```
directSensitivity = 1×3
10^-4 ×
    0.7155    0.8288    0.5778
```

(b) Repeat part (a) above if the filter is realized using a cascade form structure containing second-order sections. Use MATLAB for these calculations.

MATLAB script:

```
sos = [1 -1.4211 1 1 -1.4928 0.8612; 1 1 0 1 -0.6183 0];
G = 0.1214;
[b,a] = sos2tf(sos,G)
```

```
b = 1×4
    0.1214   -0.0511   -0.0511    0.1214
a = 1×4
    1.0000   -2.1111    1.7842   -0.5325
```

```
[SOShat,~,~] = dec2beqR(sos,L)
```

```
SOShat = 2×6
    1.0000   -1.4211    1.0000    1.0000   -1.4928    0.8612
    1.0000    1.0000         0    1.0000   -0.6183         0
```

```
a_casc = conv(SOShat(1,4:6),SOShat(2,4:5))
```

```
a_casc = 1×4
    1.0000   -2.1111    1.7842   -0.5325
```

```
delta_a = abs(a_casc(2:end) - a(2:end))
```

```
delta_a = 1×3
10^-4 ×
    0.1602    0.1597    0.0820
```

```
polesCasc = roots(a)
```

```
polesCasc = 3×1 complex
   0.7464 + 0.5514i
   0.7464 - 0.5514i
   0.6183 + 0.0000i
```

```
N = 3; % # of poles
k = 1:N;
% Delta P1
```

```
p1casc = polesCasc(1); delta_a1 = delta_a(1);
denom1 = (polesCasc(1)-polesCasc(2))*(polesCasc(1)-polesCasc(3));
delta_p1_cascade = max(sum(abs(delta_a1).*abs(p1casc.^(N-k)./denom1)))
```

```
delta_p1_cascade = 7.1546e-05
```

```
% Delta P2
p2casc = polesCasc(2); delta_a2 = delta_a(2);
denom2 = (polesCasc(2)-polesCasc(1))*(polesCasc(2)-polesCasc(3));
delta_p2_cascade = max(sum(abs(delta_a2).*abs(p2casc.^(N-k)./denom2)))
```

```
delta_p2_cascade = 7.1363e-05
```

```
% Delta P3
p3casc = polesCasc(3); delta_a3 = delta_a(3);
denom3 = (polesCasc(3)-polesCasc(1))*(polesCasc(3)-polesCasc(2));
delta_p3_cascade = max(sum(abs(delta_a3).*abs(p3casc.^(N-k)./denom3)))
```

```
delta_p3_cascade = 5.1179e-05
```

```
cascadeSensitivity = abs([delta_p1_cascade delta_p2_cascade delta_p3_cascade])
```

```
cascadeSensitivity = 1×3
10^-4 ×
    0.7155    0.7136    0.5118
```

```
% Compare Both Sensitivities
directSensitivity, cascadeSensitivity
```

```
directSensitivity = 1×3
10^-4 ×
    0.7155    0.8288    0.5778
cascadeSensitivity = 1×3
10^-4 ×
    0.7155    0.7136    0.5118
```

(c) How do these sensitivity results compare for each realization?

Answer:

I'm seeing that both the sensitivity values for the direct and cascade form are the same order of magnitude and are rather close in value to each other. My intuition tells me that the cascade form should have a lower value for the sensitity since it provides that greatest distance arrangement in the pole-zero mapping. However, I'm assuming because we're using a value of $L = 16$ that the direct form coefficients are not being altered by the quantization enough to show any drastic differences between the two sensitivities.

Problem 8.8

Specifications of a digital lowpass filter are: $\omega_p = 0.25\pi$, $\omega_s = 0.35\pi$, $A_p = 0.5$ dB, and $A = 50$ dB.

```
clc; close all; clear;
```

(a) Design an IIR digital lowpass filter using the Chebyshev II prototype that satisfies the above reqirements. Express the system function $H(z)$ of the designed filter in the cascade form.

MATLAB script:

```
omegap = 0.25*pi; omegas = 0.35*pi; As = 50; Ap = 0.5;
[N,Omegac] = cheb2ord(omegap/pi,omegas/pi,Ap,As)
```

```
N = 8
Omegac = 0.3500
```

```
[b,a] = cheby2(N,As,Omegac,'low')
```

```
b = 1×9
    0.0125    0.0045    0.0270    0.0195    0.0310    0.0195    0.0270    0.0045 ⋯
a = 1×9
    1.0000   -3.1025    5.0643   -5.0546    3.3876   -1.5103    0.4429   -0.0753 ⋯
```

```
[b0,B,A] = dir2cas(b,a)
```

```
b0 = 0.0125
B = 4×3
    1.0000    1.6319    1.0000
    1.0000    0.1955    1.0000
    1.0000   -0.5920    1.0000
    1.0000   -0.8769    1.0000
A = 4×3
    1.0000   -0.4062    0.0608
    1.0000   -0.6136    0.2391
    1.0000   -0.9024    0.5079
    1.0000   -1.1803    0.8180
```

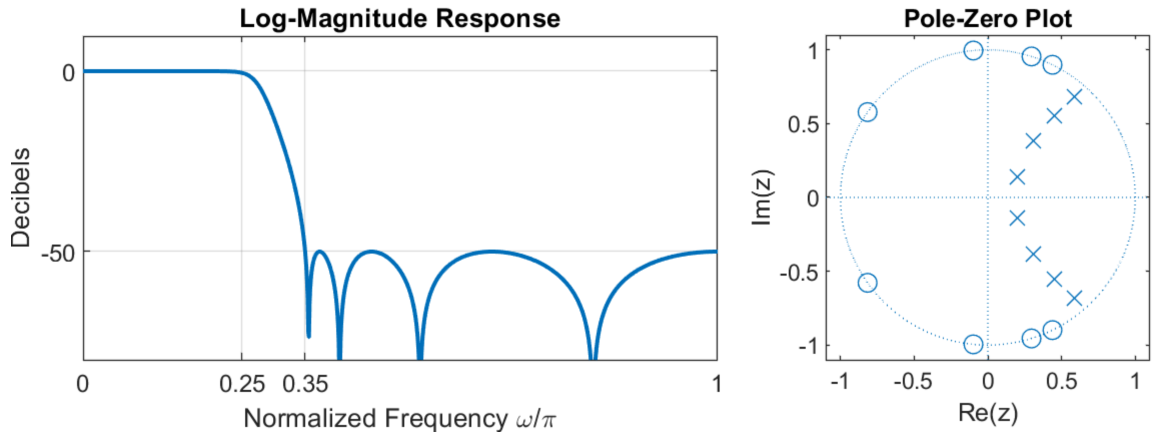
Cascade form expression for $H(z)$:

$$H(z) = \left(\frac{1+1.632z^{-1}+z^{-2}}{1-0.4062z^{-1}+0.0608z^{-2}}\right) \times \left(\frac{1+0.1955z^{-1}+z^{-2}}{1-0.6136z^{-1}+0.2391z^{-2}}\right)$$
$$\times \left(\frac{1-0.592z^{-1}+z^{-2}}{1-0.9024z^{-1}+0.5079z^{-2}}\right) \times \left(\frac{1-0.8769z^{-1}+z^{-2}}{1-1.1803z^{-1}+0.818z^{-2}}\right)$$

(b) Plot the log-magnitude (dB) response and pole-zero diagram of the filter over $0 \le \omega \le \pi$ in one figure. Use the following fragment for your figure subplots.

```
% Magnitude Response
omega = linspace(0,pi,1001);
H = freqz(b,a,omega); Hmag = abs(H);
Hdb = mag2db(Hmag);
figure('position',[0,0,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]); % Magnitude response plot
% Enter your plot code here
plot(omega/pi,Hdb,'LineWidth',1.5)
grid on, xticks([0 0.25 0.35 1])
yticks([-50 0]), ylabel('Decibels');
```

```
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response'); axis([0,1,-80,10]);
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]); % Pole-zero plot
% Enter your plot code here
zplane(b,a)
xlabel('Re(z)'); ylabel('Im(z)'); title('Pole-Zero Plot');
```



MATLAB script:

(c) Quantize the cascade form coefficients to $L = 16$ bits using the **`dec2beqR`** function and plot the resulting magnitude response and pole-zero diagram in one figure. Use the code fragment given above for your figure subplots.

MATLAB script:

```
% Quantize Cascade Form to L = 16
L = 16;
[sos,G] = tf2sos(b,a);
[SOShat,E,B] = dec2beqR(sos,L)
```

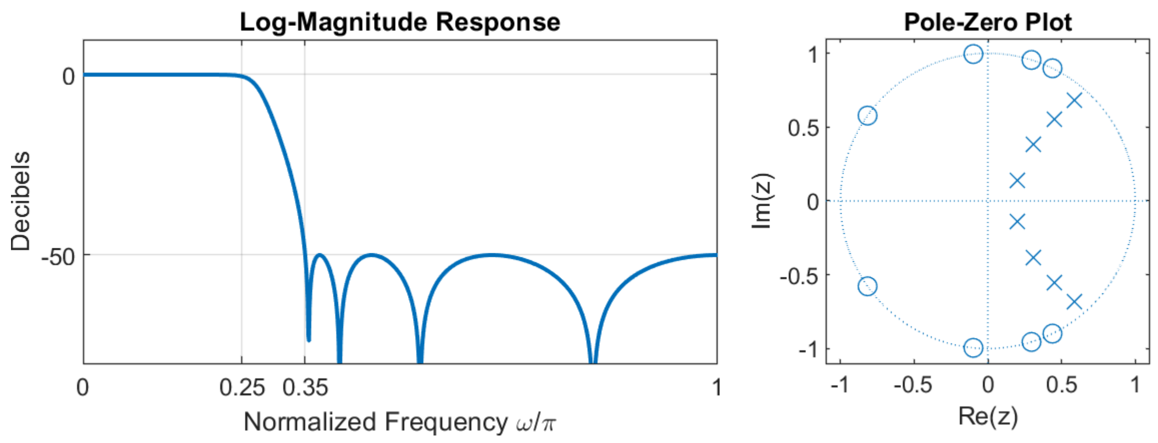
```
SOShat = 4×6
    1.0000    1.6319    1.0000    1.0000   -0.4063    0.0609
    1.0000    0.1954    1.0000    1.0000   -0.6136    0.2391
    1.0000   -0.5920    1.0000    1.0000   -0.9024    0.5079
    1.0000   -0.8769    1.0000    1.0000   -1.1802    0.8180
E = 1
B = 14
```

```
[bhat,ahat] = sos2tf(SOShat,G)
```

```
bhat = 1×9
    0.0125    0.0045    0.0270    0.0195    0.0310    0.0195    0.0270    0.0045 ···
ahat = 1×9
    1.0000   -3.1025    5.0642   -5.0544    3.3876   -1.5103    0.4429   -0.0753 ···
```

```
H = freqz(bhat,ahat,omega); Hmag = abs(H);
Hdb = mag2db(Hmag);
figure('position',[0,0,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]); % Magnitude response plot
% Enter your plot code here
plot(omega/pi,Hdb,'LineWidth',1.5)
```

```
grid on, xticks([0 0.25 0.35 1])
yticks([-50 0]), ylabel('Decibels');
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response'); axis([0,1,-80,10]);
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]); % Pole-zero plot
% Enter your plot code here
zplane(b,a)
xlabel('Re(z)'); ylabel('Im(z)'); title('Pole-Zero Plot');
```



(d) Quantize the cascade form coefficients to $L = 12$ bits using the **dec2beqR** function and plot the resulting magnitude response and pole-zero diagram in one figure. Use the code fragment given above for your figure subplots.

MATLAB script:

```
% Quantize Cascade Form to L = 12
L = 12;
[SOShat,E,B] = dec2beqR(sos,L)
```

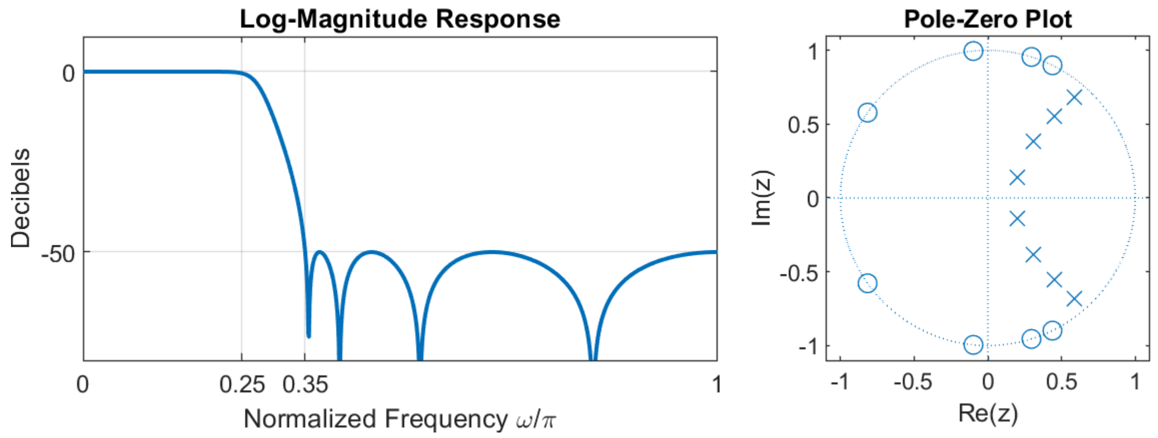
```
SOShat = 4×6
    1.0000    1.6318    1.0000    1.0000   -0.4063    0.0605
    1.0000    0.1953    1.0000    1.0000   -0.6133    0.2393
    1.0000   -0.5918    1.0000    1.0000   -0.9023    0.5078
    1.0000   -0.8770    1.0000    1.0000   -1.1807    0.8184
E = 1
B = 10
```

```
[bhat,ahat] = sos2tf(SOShat,G)
```

```
bhat = 1×9
    0.0125    0.0045    0.0270    0.0195    0.0310    0.0195    0.0270    0.0045 ⋯
ahat = 1×9
    1.0000   -3.1025    5.0642   -5.0540    3.3868   -1.5096    0.4425   -0.0752 ⋯
```

```
% Magnitude Response
omega = linspace(0,pi,1001);
H = freqz(bhat,ahat,omega); Hmag = abs(H);
Hdb = mag2db(Hmag);
figure('position',[0,0,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]); % Magnitude response plot
```

```
% Enter your plot code here
plot(omega/pi,Hdb,'LineWidth',1.5)
grid on, xticks([0 0.25 0.35 1])
yticks([-50 0]), ylabel('Decibels');
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response'); axis([0,1,-80,10]);
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]); % Pole-zero plot
% Enter your plot code here
zplane(b,a)
xlabel('Re(z)'); ylabel('Im(z)'); title('Pole-Zero Plot');
```



(e) Quantize the cascade form coefficients to $L = 8$ bits using the **`dec2beqR`** function and plot the resulting magnitude response and pole-zero diagram in one figure. Use the code fragment given above for your figure subplots.

MATLAB script:

```
% Quantize Cascade Form to L = 8
L = 8;
[SOShat,E,B] = dec2beqR(sos,L)
```

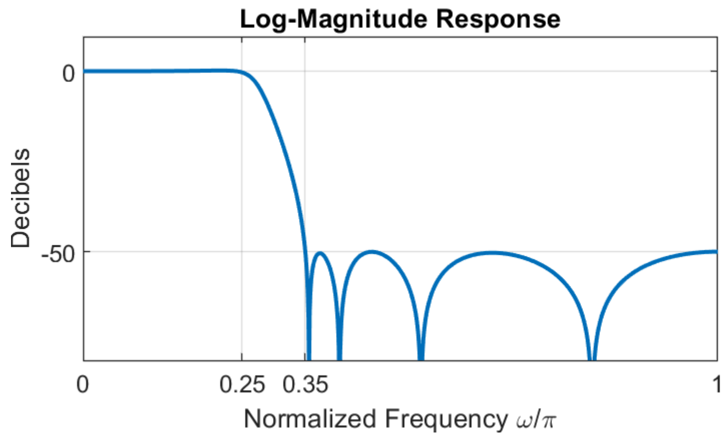
```
SOShat = 4×6
    1.0000    1.6250    1.0000    1.0000   -0.4063    0.0625
    1.0000    0.2031    1.0000    1.0000   -0.6094    0.2344
    1.0000   -0.5938    1.0000    1.0000   -0.9063    0.5156
    1.0000   -0.8750    1.0000    1.0000   -1.1875    0.8125
E = 1
B = 6
```

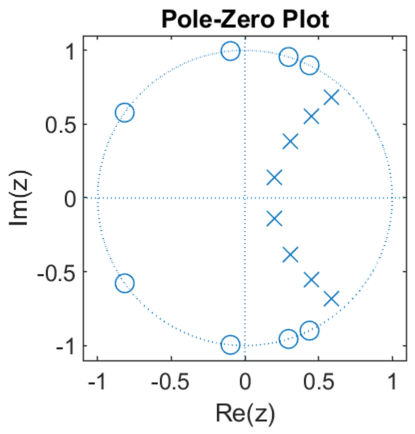
```
[bhat,ahat] = sos2tf(SOShat,G)
```

```
bhat = 1×9
    0.0125    0.0045    0.0271    0.0194    0.0314    0.0194    0.0271    0.0045 ⋯
ahat = 1×9
    1.0000   -3.1094    5.0752   -5.0637    3.3914   -1.5109    0.4431   -0.0756 ⋯
```

```
% Magnitude Response
omega = linspace(0,pi,1001);
H = freqz(bhat,ahat,omega); Hmag = abs(H);
Hdb = mag2db(Hmag);
figure('position',[0,0,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]); % Magnitude response plot
```

```
% Enter your plot code here
plot(omega/pi,Hdb,'LineWidth',1.5)
grid on, xticks([0 0.25 0.35 1])
yticks([-50 0]), ylabel('Decibels');
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response'); axis([0,1,-80,10]);
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]); % Pole-zero plot
% Enter your plot code here
zplane(b,a)
xlabel('Re(z)'); ylabel('Im(z)'); title('Pole-Zero Plot');
```





(f) Comment on your plots.

Answer:

We notice that even though we repeatedly quantize and reduce the length of significant bits for the cascade form coefficients, there is no change in the performance or the stability of the filters. The fractional bits decreased from $14 \text{ to } 10 \text{ to } 6$ for each quantizing iteration, though the resulting plots did not suffer in meeting the design specifications or have any discerning movement of poles or zeros. Again, this demonstrates how effective the cascade form coefficients are to quantizing its values and why it is preferred over the direct form representation.

```
function [H,bins,eavg,evar] = QNmodel(x,B)
% Compute quantization error statistics of the input sequence
xm = abs(x(:));
E = max(max(0,fix(log2(xm+eps)+1))); % Integer bits
L = B + E + 1;
[beq,~,~] = dec2beqR(x,L);
en = beq - x;
[bins H] = epdf(en,50);
eavg = mean(en);
evar = var(en);
end

function [A,B] = spec_convert(C,D,typein,typeout)
%  typein: 'abs' or 'rel' or 'ana'
% typeout: 'abs' or 'rel' or 'ana'
%     C,D: input specifications
%     A,B: output specifications

% Enter your function code below
if nargin > 4
    error('too many input arguments')
end

switch typein
    case 'abs'
        switch typeout
            case 'rel'
                Ap = 20*log10((1+C)/(1-C)); % Relative Output: Passband ripple
                As = floor(20*log10((1+C)/D)); % Relative Output: Stopband Attenuation
                A = Ap; B = As;
            case 'ana'
                Ap = 20*log10((1+C)/(1-C)); % Relative Output: Passband ripple
                As = floor(20*log10((1+C)/D)); % Relative Output: Stopband Attenuation
                A = sqrt(10^(Ap/10)-1); % Analog Output: Passband
                B = floor(10^(As/20)); % Analog Output: Stopband
        end
    case 'rel'
        switch typeout
            case 'abs'
                dp = (10^(C/20)-1)/(1+10^(C/20)); % Absolute Output: Passband Error
                ds = (1+dp)/(10^(D/20)); % Absolute Output: Stopband Error
                A = dp; B = ds;
            case 'ana'
                epsilon = sqrt(10^(C/10)-1); % Analog Output: Passband
                B = floor(10^(D/20)); % Analog Output: Stopband
                A = epsilon;
        end
   case 'ana'
        switch typeout
            case 'rel'
                Ap = round(10*log10(C^(2)+1),2); % Relative Output: Passband ripple (in dB)
                As = 20*log10(D); % Relative Output: Stopband Attenuation (in dB)
                A = Ap; B = As;
            case 'abs'
                Ap = round(10*log10(C^(2)+1),2); % Relative Output: Passband ripple (in dB)
                As = 20*log10(D); % Relative Output: Stopband Attenuation (in dB)
                dp = (10^(Ap/20)-1)/(1+10^(Ap/20)); % Absolute Output: Passband Error
                ds = (1+dp)/(10^(Ap/20)); % Absolute Output: Stopband Error
                A = dp; B = ds;
        end
end
end

function [H1,H2,Q, estat] = StatModelR(xn,B,N)
% Statistical Model (Rounding) for A/D Quantization error and its Distribution
% ------------- ------------------------------------------------------------
% [H1,H2,Q] = StatModelR(xn,B,N);
% OUT: H1 = normalized histogram of e1
% H2 = normalized histogram of e2
% Q = normalized histogram bins
% estat = row vector: [[e1avg,e1std,e2avg,e2std]
% IN: B = bits to quantize
% N = number of samples of x(n)
% xn = samples of the sequence
% Plot variables
bM = 7; DbM = 2^bM; % Bin parameter
M = round((DbM)/2); % Half number of bins
bins = [-M+0.5:1:M-0.5]; % Bin values from -M to M
Q = bins/(DbM); % Normalized bins
% Quantization error analysis
xq = (round(xn*(2^B)))/(2^B); % Quantized to B bits
e1 = xq-xn; clear xn xq; % Quantization error
e2 = 0.5*(e1(1:N-1)+e1(2:N)); % Average of two adj errors
e1avg = mean(e1); e1std = std(e1); % Mean & std dev of the error e1
e2avg = mean(e2); e2std = std(e2); % Mean & std dev of the error e2
estat = [e1avg,e1std,e2avg,e2std];
% Probability distribution of e1
e1 = floor(e1*(2^(B+bM))); % Normalized e1 (int between -M & M)
e1 = sort([e1,-M-1:1:M]); %
H1 = diff(find(diff(e1)))-1; clear e1; % Error histogram
if length(H1) == DbM+1
H1(DbM) = H1(DbM)+H1(DbM+1);
H1 = H1(1:DbM);
end
H1 = H1/N; % Normalized histogram
% Probability distribution of e2
e2 = floor(e2*(2^(B+bM))); % Normalized e2 (int between -M & M)
e2 = sort([e2,-M-1:1:M]); %
H2 = diff(find(diff(e2)))-1; clear e2; % Error histogram
if length(H2) == DbM+1
H2(DbM) = H2(DbM)+H2(DbM+1);
H2 = H2(1:DbM);
end
H2 = H2/N; % Normalized histogram
end
```