EECE5666 (DSP) : Homework-8 Solutions

Table of Contents

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:

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

Solution: The following script obtains the result.

```
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,~,k] = residuez(conv(b1,b2),conv(a1,a2));
fprintf('Variance Gain is %7.4f',K*(k+sum(r(4:6))))
```

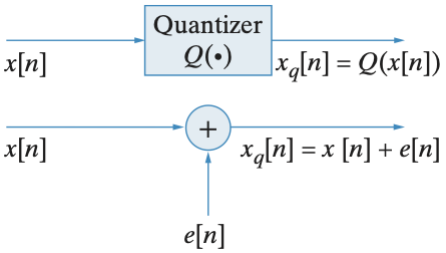
```
Variance Gain is 376.8885
```

```
%% Verification:
hn = filter(b1,a1,[1 zeros(1,999)]); sum(abs(hn).^2)
```

```
ans = 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

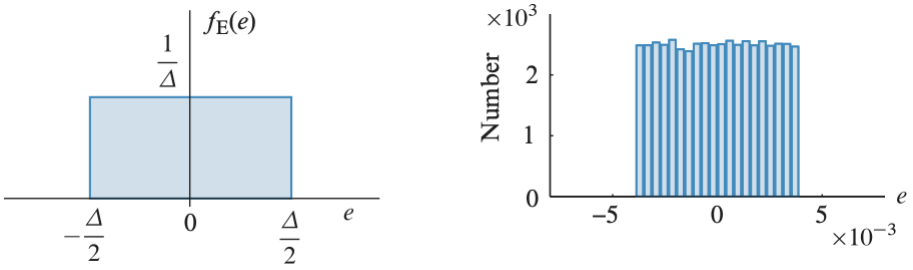
$$f_{E_n,E_{n+1}}(x,y) = f_{E_n}(x) f_{E_{n+1}}(y). \qquad (8.2.1)$$

```
clc; close all; clear;
newgr = [0.466,0.674,0.188]; % New green color
```

(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

$$f_G(g) = 2\Lambda(2g) \qquad (8.2.4)$$

where

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

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

Solution: Since $e[n]$ is IID with pdf $f_E(e) \sim \mathcal{U}\left[-\frac{\Delta}{2},\frac{\Delta}{2}\right]$, then the normalized sequence $e_0[n]$ is also IID with pdf, $f_{E_0}(e_0)$, which is obtained using the fundamental theorem of densities

$$f_{E_0}(e_0) = \left.\frac{f_E(e)}{\mathrm{d}e_0/\mathrm{d}e}\right|_{e=2\Delta e_0} = \frac{f_E(2\Delta e_0)}{1/(2\Delta)} = 2\Delta f_E(2\Delta e_0) \qquad (8.2.6)$$

But $f_E(e) = \mathcal{U}\left[-\frac{\Delta}{2},\frac{\Delta}{2}\right] = \begin{cases} 1/\Delta, & -\Delta/2 \le e \le \Delta/2 \\ 0, & \text{otherwise} \end{cases}$.

Substituting in (8.2.6), we obtain

$$f_{E_0}(e_0) = 2\Delta\left(\begin{cases} 1/\Delta, & -\Delta/2 \le 2\Delta e_0 \le \Delta/2 \\ 0, & \text{otherwise} \end{cases}\right)$$

$$= \begin{cases} 2, & -1/4 \le e_0 \le 1/4 \\ 0, & \text{otherwise} \end{cases} = \mathcal{U}\left[-\frac{1}{4},\frac{1}{4}\right]. \qquad (8.2.7)$$

Now, since $g[n] = e_0[n] + e_0[n+1]$, which is the sum of two IID random variables, the pdf of $g[n]$ is the convolution of the two identical uniform distributions. Thus,

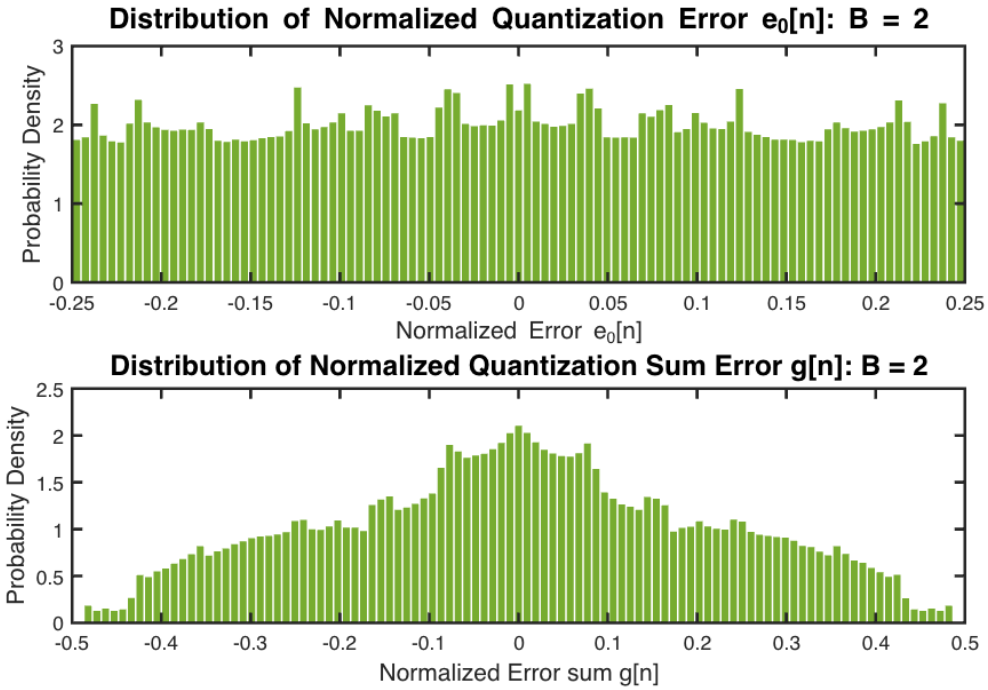
$$\begin{aligned} f_G(g) &= f_{E_0}(g) * f_{E_0}(g) = \mathcal{U}\left[-\tfrac{1}{4},\tfrac{1}{4}\right] * \mathcal{U}\left[-\tfrac{1}{4},\tfrac{1}{4}\right] \\ &= \begin{cases} 4\left(g+\frac{1}{4}+\frac{1}{4}\right), & -\frac{1}{2} \le g \le 0 \\ 4\left(\frac{1}{4}+\frac{1}{4}-g\right), & 0 \le g \le \frac{1}{2} \\ 0, & \text{else} \end{cases} = \begin{cases} 2+4g, & -\frac{1}{2} \le g \le 0 \\ 2-4g, & 0 \le g \le \frac{1}{2} \\ 0, & \text{else} \end{cases} \\ &= \begin{cases} 2-4|g|, & |g| \le 1/2 \\ 0, & \text{else} \end{cases} = 2\begin{cases} 1-|2g|, & |2g| \le 1 \\ 0, & \text{else} \end{cases} = 2\Lambda(2g) \end{aligned}$$

which proves $(8.2.4)$.

(b) Let $x[n] = 0.5\big(\cos(n/7) + \sin(n/17)\big)$. Generate 500,000 samples of $x[n]$. Using the **dec2beqR** function quantize $x[n]$ to $B+1=3$ bits where B is the number of fractional bits. Obtain the sequences $e_0[n]$ and $g[n]$ using $(8.2.3)$. Compute the histograms of both sequences using the **epdf** function and **bar** graph them in one figure with two rows and one column. Can you conclude that $e_0[n]$ is uniformly distributed and that its consecutive samples are statistically independent? Explain.

MATLAB script:

```
N = 500000; n = 1:N; xn = 0.5*(cos(n/7)+sin(n/17)); clear n;
B = 2; L = B+1;
[xq,~,~] = dec2beqR(xn,L); % Quantize the sequence
en = xq-xn; % Quantization noise;
Delta = 2^(-B); e0n = en/(2*Delta); % normalized quantization noise
gn = e0n(1:end-1)+e0n(2:end);
clear xn xq en; % Clear no longer needed large size vaiable to conserve memory
[e0,pe0]=epdf(e0n,101); % Histogram of e0n
[g,pg]=epdf(gn,101); % Histogram of gn
figure('Position',[0,0,8,5]*72);
subplot(2,1,1); % Normalized histogram of e0n
bar(e0,pe0,'FaceColor',newgr,'LineStyle','none'); axis([-0.25,0.25,0,3]);
xlabel('Normalized Error e_0[n]'); ylabel('Probability Density');
title('Distribution of Normalized Quantization Error e_0[n]: B = 2');
subplot(2,1,2); % Normalized histogram of gn
bar(g,pg,'FaceColor',newgr,'LineStyle','none'); axis([-0.5,0.5,0,2.5]);
xlabel('Normalized Error sum g[n]'); ylabel('Probability Density');
title('Distribution of Normalized Quantization Sum Error g[n]: B = 2');
```

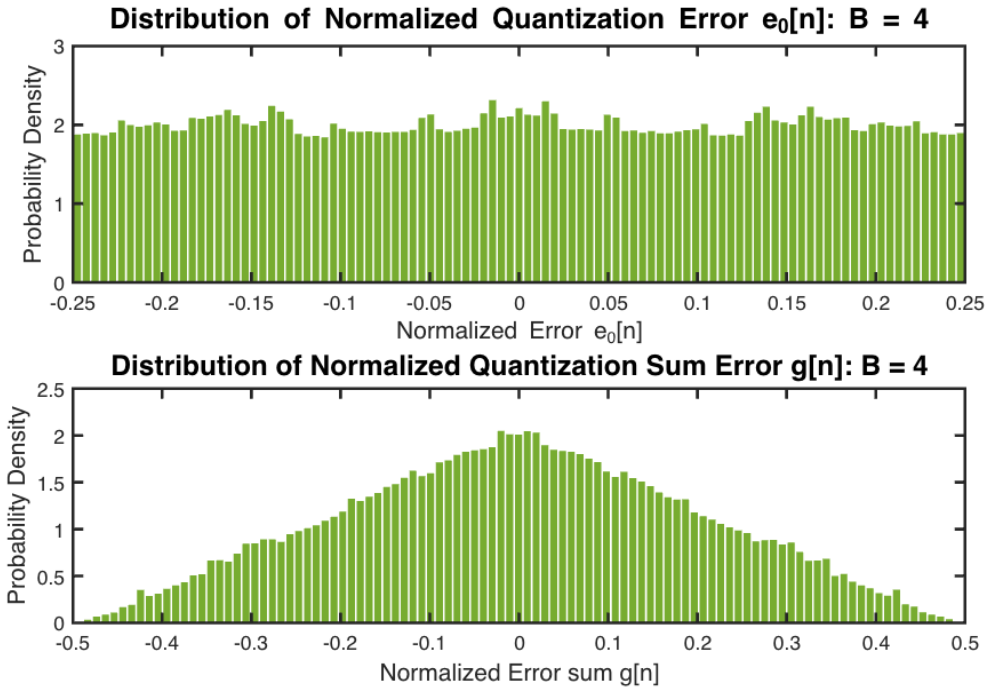


Explanation: The pdf of the normalized quantization error is not quite a uniform distribution while that of the sum of the consecutive samples is not triangular. This indicates that the error samples are not uniform and that the consecutive samples are not independent.

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

MATLAB script::

```
N = 500000; n = 1:N; xn = 0.5*(cos(n/7)+sin(n/17)); clear n;
B = 4; L = B+1;
[xq,~,~] = dec2beqR(xn,L); % Quantize the sequence
en = xq-xn; % Quantization noise;
Delta = 2^(-B); e0n = en/(2*Delta); % normalized quantization noise
gn = e0n(1:end-1)+e0n(2:end);
clear xn xq en; % Clear no longer needed large size vaiable to conserve memory
[e0,pe0]=epdf(e0n,101); % Histogram of e0n
[g,pg]=epdf(gn,101); % Histogram of gn
figure('Position',[0,0,8,5]*72);
subplot(2,1,1); % Normalized histogram of e0n
bar(e0,pe0,'FaceColor',newgr,'LineStyle','none'); axis([-0.25,0.25,0,3]);
xlabel('Normalized Error e_0[n]'); ylabel('Probability Density');
title('Distribution of Normalized Quantization Error e_0[n]: B = 4');
subplot(2,1,2); % Normalized histogram of gn
bar(g,pg,'FaceColor',newgr,'LineStyle','none'); axis([-0.5,0.5,0,2.5]);
xlabel('Normalized Error sum g[n]'); ylabel('Probability Density');
title('Distribution of Normalized Quantization Sum Error g[n]: B = 4');
```

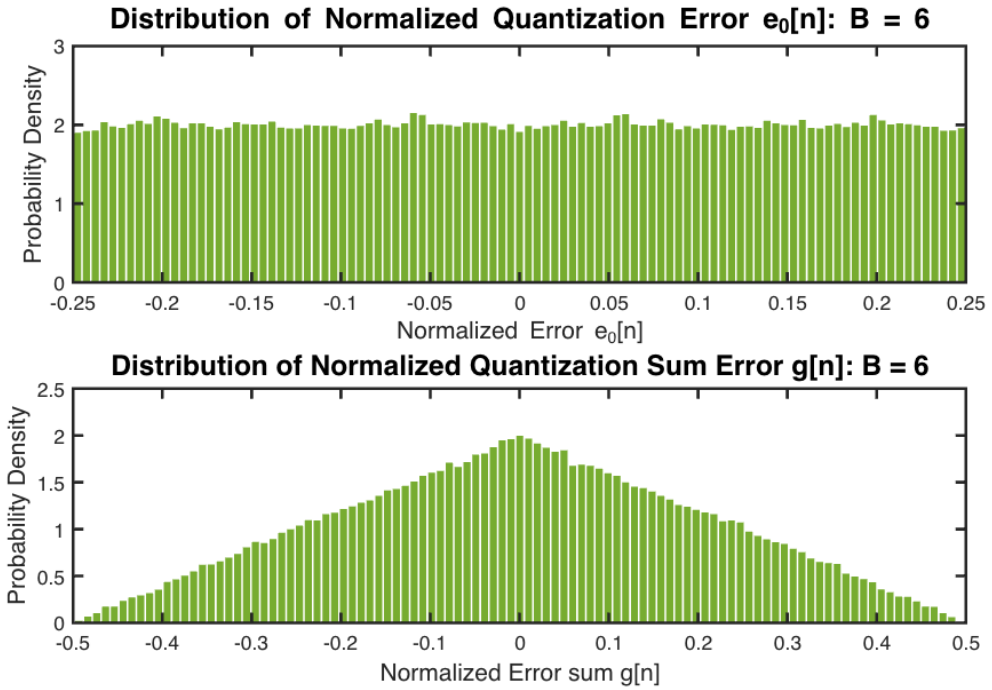


Explanation: The pdf of the normalized quantization error is very close (but not exact) to the uniform distribution while that of the sum of the consecutive samples is very close to a triangular distribution. This indicates that the consecutive samples are almost independent.

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

MATLAB script::

```
N = 500000; n = 1:N; xn = 0.5*(cos(n/7)+sin(n/17)); clear n;
B = 6; L = B+1;
[xq,~,~] = dec2beqR(xn,L); % Quantize the sequence
en = xq-xn; % Quantization noise;
Delta = 2^(-B); e0n = en/(2*Delta); % normalized quantization noise
gn = e0n(1:end-1)+e0n(2:end);
clear xn xq en; % Clear no longer needed large size vaiable to conserve memory
[e0,pe0]=epdf(e0n,101); % Histogram of e0n
[g,pg]=epdf(gn,101); % Histogram of gn
figure('Position',[0,0,8,5]*72);
subplot(2,1,1); % Normalized histogram of e0n
bar(e0,pe0,'FaceColor',newgr,'LineStyle','none'); axis([-0.25,0.25,0,3]);
xlabel('Normalized Error e_0[n]'); ylabel('Probability Density');
title('Distribution of Normalized Quantization Error e_0[n]: B = 6');
subplot(2,1,2); % Normalized histogram of gn
bar(g,pg,'FaceColor',newgr,'LineStyle','none'); axis([-0.5,0.5,0,2.5]);
xlabel('Normalized Error sum g[n]'); ylabel('Probability Density');
title('Distribution of Normalized Quantization Sum Error g[n]: B = 6');
```



Explanation: The pdf of the normalized quantization error is uniformly distributed while that of the sum of the consecutive samples is almost triangular. This indicates that the consecutive samples are independent.

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

Answer: Based on the above results, we can assume that when $B \geq 6$ fractional bits, the quantization noise is uniformly distributed and are independent of each other.

Problem 8.3

Text Problem 15.30 (Page 961)

Consider the filter described by the system function

$$y[n] = \frac{1 - \sqrt{3}\,z^{-1}}{1 - z^{-1}/\sqrt{3}}, \quad |z| > 1/\sqrt{3}. \qquad (8.2.1)$$

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

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

Solution: The frequency response from $(8.2.1)$ is given by

$$H(e^{j\omega}) = \frac{1 - \sqrt{3}\,e^{-j\omega}}{1 - \frac{1}{\sqrt{3}}e^{-j\omega}} = \frac{e^{j\omega} - \sqrt{3}}{e^{j\omega} - \frac{1}{\sqrt{3}}}$$

Hence

$$|H(e^{j\omega})| = \sqrt{\frac{(\cos\omega - \sqrt{3})^2 + (\sin\omega)^2}{(\cos\omega - 1/\sqrt{3})^2 + (\sin\omega)^2}} = \sqrt{3}$$

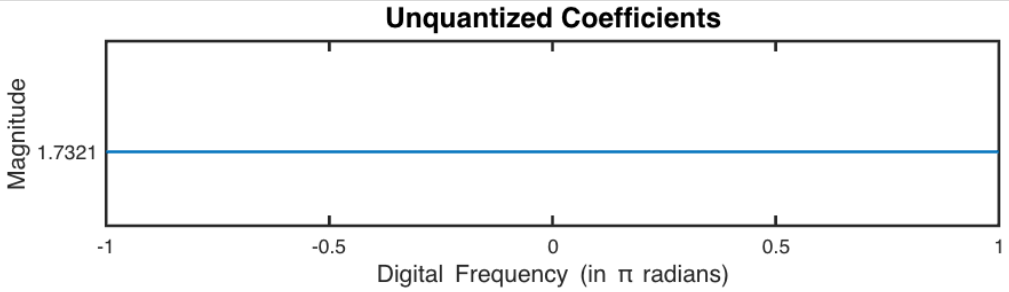
which is a constant for all ω. Hence the filter is an all-pass filter. Also note that the pole location $1/\sqrt{3}$ is the reciprocal of the zero location $\sqrt{3}$.

MATLAB verification:

```
b = [1,-sqrt(3)], a = [1,-1/sqrt(3)],
```

```
b = 1×2
    1.0000   -1.7321
a = 1×2
    1.0000   -0.5774
```

```
f = linspace(-1,1,1001); H = freqz(b,a,pi*f); Hmag = abs(H);
figure('position',[0,0,8,2]*72); plot(f,Hmag,'linewidth',1.5);
axis([-1,1,1.71,1.765]); ylabel('Magnitude');
title('Unquantized Coefficients');
xlabel('Digital Frequency (in \pi radians)');
set(gca,'ytick',sqrt(3),'xtick',(-1:0.5:1));
```



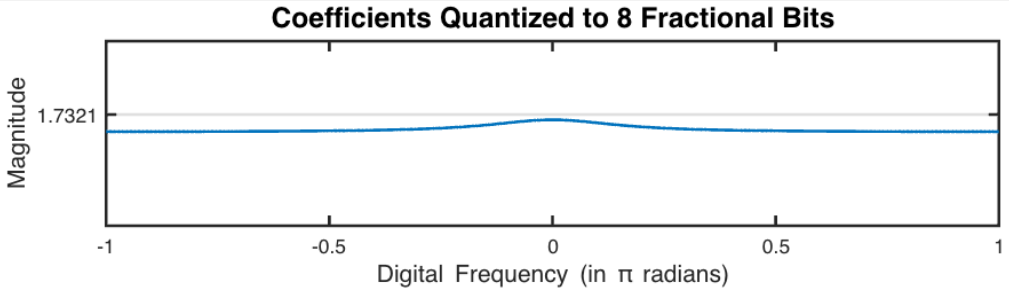
(b) Round the filter coefficients to $B = 8$ fraction bits using the **dec2beqR** function and then plot the magnitude response of the resulting filter. Is the filter still all-pass?

Solution: Note from the above filter coefficients that we will need one integer bit. Since we want 8 fractional bits, the total number of bits is $L = 1 + 1 + 8 = 10$.

```
L = 10; [bahat,~,~] = dec2beqR([b;a],L)
```

```
bahat = 2×2
    1.0000   -1.7305
    1.0000   -0.5781
```

```
bhat = bahat(1,:); ahat = bahat(2,:);
H = freqz(bhat,ahat,pi*f); Hmag = abs(H);
figure('position',[0,0,8,2]*72); plot(f,Hmag,'linewidth',1.5);
axis([-1,1,1.72,1.74]); ylabel('Magnitude');
title('Coefficients Quantized to 8 Fractional Bits');
xlabel('Digital Frequency (in \pi radians)');
set(gca,'ytick',sqrt(3),'xtick',(-1:0.5:1),'ygrid','on');
```



Observe that the magnitude response is not constant and hence the resulting filter is not an allpass system. Also pole location 0.5781 is not reciprocal of the zero location 1.7305.

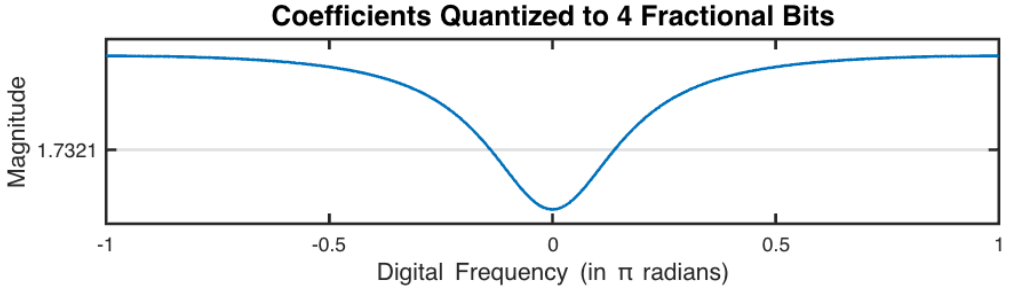
(c) Round the filter coefficients to $B = 4$ fractional bits using the **dec2beqR** function and then plot the magnitude response of the resulting filter. Is the filter still all-pass?

Solution: Note from the above filter coefficients that we will need one integer bit. Since we want 4 fractional bits, the total number of bits is $L = 1 + 1 + 4 = 6$.

```
L = 6; [bahat,~,~] = dec2beqR([b;a],L)
```

```
bahat = 2×2
    1.0000   -1.7500
    1.0000   -0.5625
```

```
bhat = bahat(1,:); ahat = bahat(2,:);
H = freqz(bhat,ahat,pi*f); Hmag = abs(H);
figure('position',[0,0,8,2]*72); plot(f,Hmag,'linewidth',1.5);
axis([-1,1,1.71,1.765]); ylabel('Magnitude');
title('Coefficients Quantized to 4 Fractional Bits');
xlabel('Digital Frequency (in \pi radians)');
set(gca,'ytick',sqrt(3),'xtick',(-1:0.5:1),'ygrid','on');
```



Observe that the magnitude response is certainly not constant and hence the resulting filter is not an allpass system. Also pole location 0.5625 is not reciprocal of the zero location 1.75.

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.

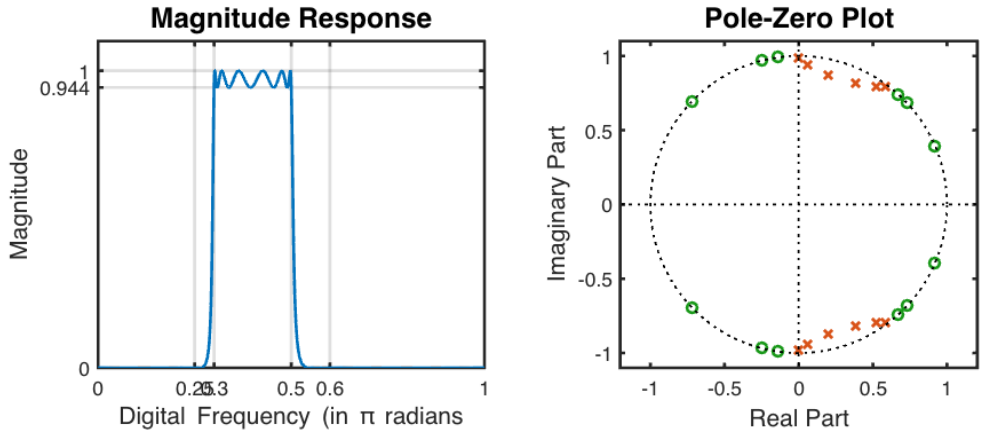
Solution: The required filter designed and its responses are plotted using the following script.

```
% Given Specifications
fs1 = 0.25; fp1 = 0.3; fp2 = 0.5; fs2 = 0.6; Ap = 0.5; As = 60;
fp = [fp1,fp2]; fs = [fs1,fs2];
[N,fc] = ellipord(fp,fs,Ap,As); N
```

```
N = 6
```

```
[b,a] = ellip(N,Ap,As,fc); % Direct form coefficients
f = linspace(0,1,1001); H = freqz(b,a,f*pi); Hmag = abs(H);
fcutoff = [0,fs1,fp1,fp2,fs2,1];
[epsi,~] = spec_convert(Ap,As,'rel','ana');
PBripple = round(sqrt(1/(1+epsi^2)),3);
% Filter Response Plots
figure('position',[0,0,8,3]*72);
subplot(1,2,1); % Magnitude response
```

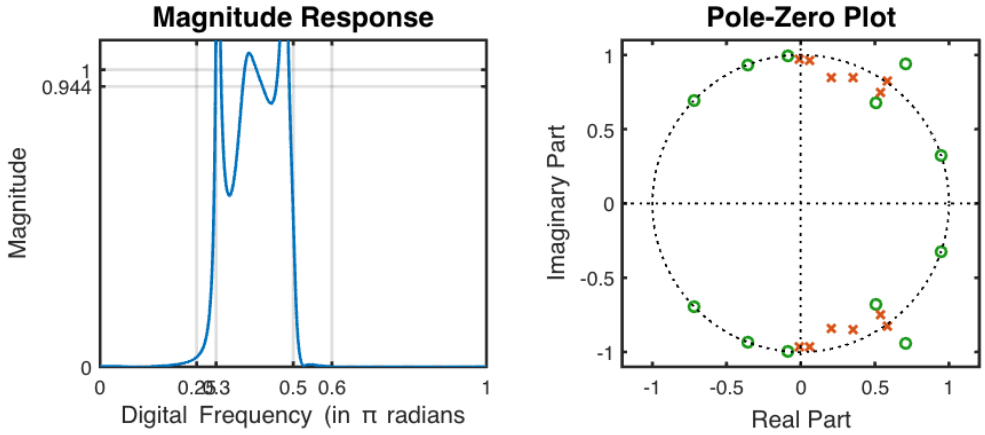
```
plot(f,Hmag,'linewidth',1.5); axis([0,1,0,1.1]);
xlabel('Digital Frequency (in \pi radians'); ylabel('Magnitude');
title('Magnitude Response'); set(gca,'xtick',fcutoff);
set(gca,'ytick',[0,PBripple,1]); grid;
subplot(1,2,2); % Pole-zero plot
[Hz,Hp,Hl] = zplane(b,a); title('Pole-Zero Plot');
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
```



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

Solution: We will quantize both **b** and **a** array together using matrix representation **`[b;a]`**, extract quantized arrays, and then obtain new plots. The following script performs all steps.

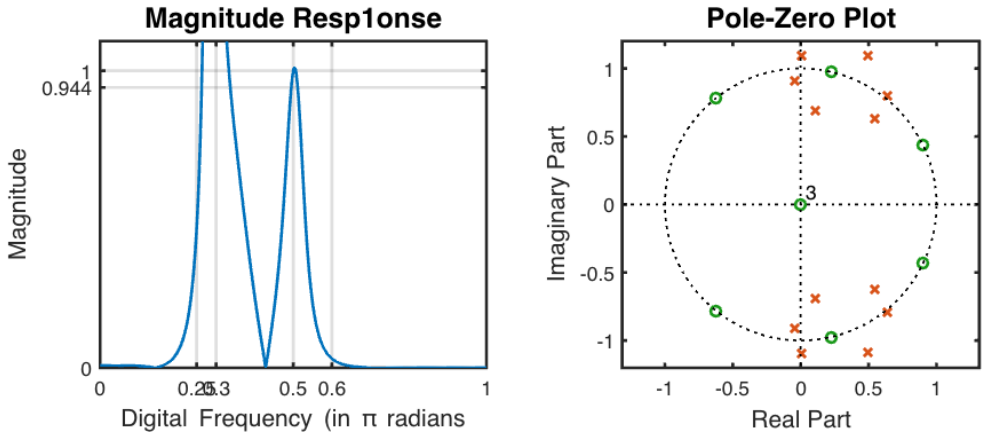
```
L = 16; [bahat,~,~] = dec2beqR([b;a],L);
bhat = bahat(1,:); ahat = bahat(2,:);
Hhat = freqz(bhat,ahat,f*pi); Hhatmag = abs(Hhat);
% Filter Response Plots
figure('position',[0,0,8,3]*72);
subplot(1,2,1); % Magnitude response
plot(f,Hhatmag,'linewidth',1.5); axis([0,1,0,1.1]);
xlabel('Digital Frequency (in \pi radians'); ylabel('Magnitude');
title('Magnitude Response'); set(gca,'xtick',fcutoff);
set(gca,'ytick',[0,PBripple,1]); grid;
subplot(1,2,2); % Pole-zero plot
[Hz,Hp,Hl] = zplane(bhat,ahat); title('Pole-Zero Plot');
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
```



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

Solution: We repeat the above procedure using $L = 12$ bits.

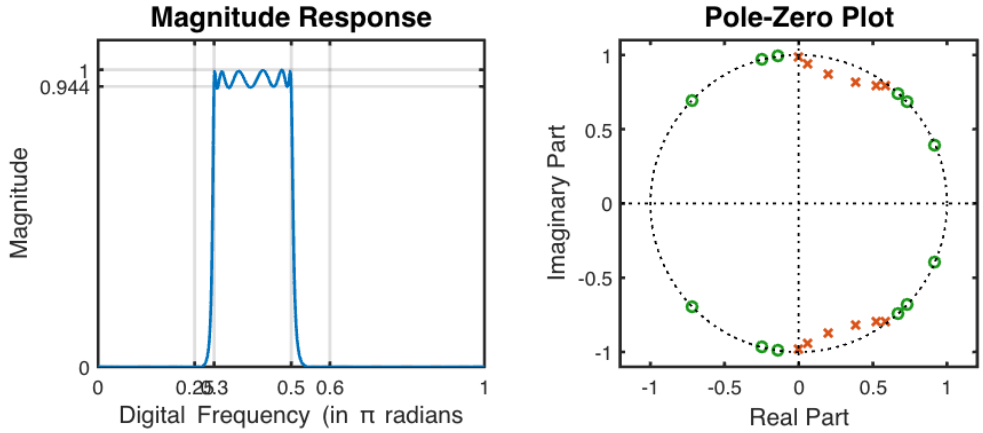
```
L = 12; [bahat,~,~] = dec2beqR([b;a],L);
bhat = bahat(1,:); ahat = bahat(2,:);
Hhat = freqz(bhat,ahat,f*pi); Hhatmag = abs(Hhat);
% Filter Response Plots
figure('position',[0,0,8,3]*72);
subplot(1,2,1); % Magnitude response
plot(f,Hhatmag,'linewidth',1.5); axis([0,1,0,1.1]);
xlabel('Digital Frequency (in \pi radians'); ylabel('Magnitude');
title('Magnitude Resp1onse'); set(gca,'xtick',fcutoff);
set(gca,'ytick',[0,PBripple,1]); grid;
subplot(1,2,2); % Pole-zero plot
[Hz,Hp,Hl] = zplane(bhat,ahat); title('Pole-Zero Plot');
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
```



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

Solution: We first convert direct form coefficients into cascade form coefficients, quantize them to $L = 12$ bits, convert quantized cascade coefficients into direct form (because we do not have a function equivalent to **freqz** using cascade coefficients), and then repeat the procedure of part (c). The following script performs all these steps.

```
[sos,G] = tf2sos(b,a); % cascade form coefficients
L = 12; [soshat,E3,B3] = dec2beqR([sos;[G,zeros(1,5)]],L);
Ghat = soshat(end,1); soshat = soshat(1:end-1,:);
[bhat,ahat] = sos2tf(soshat,Ghat);
Hhat = freqz(bhat,ahat,f*pi); Hhatmag = abs(Hhat/max(Hhat));
% Filter Response Plots
figure('position',[0,0,8,3]*72);
subplot(1,2,1); % Magnitude response
plot(f,Hhatmag,'linewidth',1.5); axis([0,1,0,1.1]);
xlabel('Digital Frequency (in \pi radians'); ylabel('Magnitude');
title('Magnitude Response'); set(gca,'xtick',fcutoff);
set(gca,'ytick',[0,PBripple,1]); grid;
subplot(1,2,2); % Pole-zero plot
[Hz,Hp,Hl] = zplane(bhat,ahat); title('Pole-Zero Plot');
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
```



(e) Comment on your plots.

Comments: The magnitude plots from parts (b) and (c) indicate that direct form implementations using even 16 bits give filters that fail miserably in satisfying specifications. The pole-zero plots indicate that we certainly get an unstable filter in (c) and a possibly unstable filter in (b). The cascade form implementation, on the other hand, give us a desirable and stable filter even for 12 bits.

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 to answer the following parts.

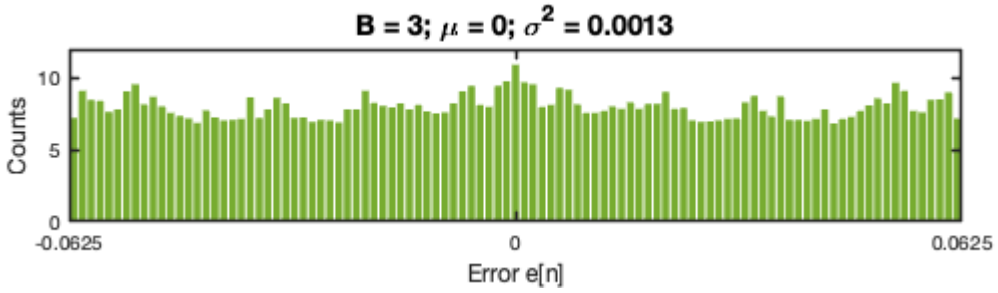
Note: The **`QNmodel`** function is developed in Chapter-15 Tutorial Problem 5. It is not a part of the book toolbox.

```
clc; close all; clear;
newgr = [0.466,0.674,0.188]; % New green color
```

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

Solution: The following script uses the **`QNmodel`** function to obtain error histogram.

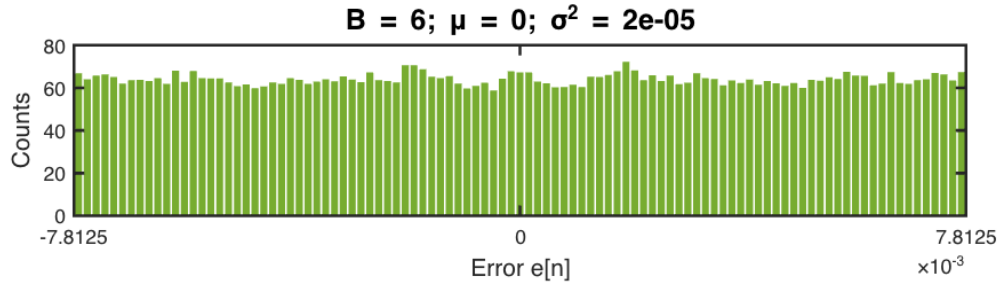
```
N = 100000; n = 1:N; xn = 0.49*(cos(n/13)+sin(n/29));
a = 0.3333; yn = a*xn; B = 3;
[pe,eo,eavg,evar] = QNmodel(yn,B);
eavg = round(eavg,2); evar = round(evar,4);
figure('position',[0,0,8,2]*72);
bar(eo,pe,'facecolor',newgr,'LineStyle','none');
axis([-2^(-(B+1)),2^(-(B+1)),0,12]);
xlabel('Error e[n]'); ylabel('Counts');
title(['B = ',num2str(B),'; \mu = ',num2str(eavg),'; \sigma^2 = ',...
    num2str(evar)]);
set(gca,'xtick',[-2^(-(B+1)),0,2^(-(B+1))]);
```



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

Solution: The following script uses the **`QNmodel`** function to obtain error histogram.

```
B = 6; [pe,eo,eavg,evar] = QNmodel(yn,B); eavg = round(eavg,2); evar = round(evar,6);
figure('position',[0,0,8,2]*72);
bar(eo,pe,'facecolor',newgr,'LineStyle','none');
axis([-2^(-(B+1)),2^(-(B+1)),0,80]);
xlabel('Error e[n]'); ylabel('Counts');
title(['B = ',num2str(B),'; \mu = ',num2str(eavg),'; \sigma^2 = ',num2str(evar)]);
set(gca,'xtick',[-2^(-(B+1)),0,2^(-(B+1))]);
```

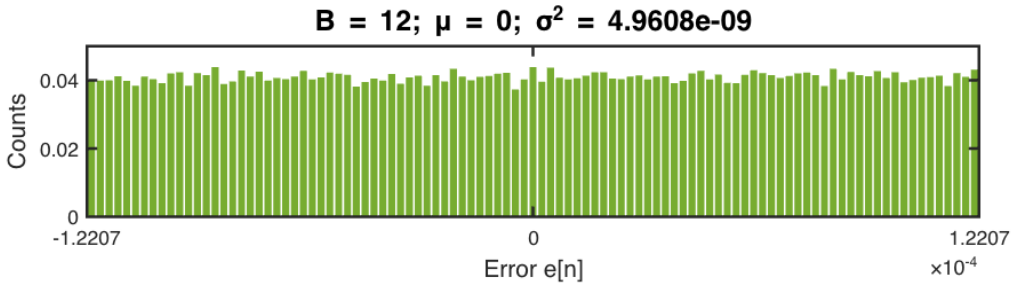


(c) Quantize $ax[n]$ to $B = 12$ bits and plot the histogram of the resulting error sequence.

Solution: The following script uses the **`QNmodel`** function to obtain error histogram.

```
B = 12;
[pe,eo,eavg,evar] = QNmodel(yn,B); eavg = round(eavg,2); evar = round(evar,16);
figure('position',[0,0,8,2]*72);
bar(eo,pe/N,'facecolor',newgr,'LineStyle','none');
axis([-2^(-(B+1)),2^(-(B+1)),0,5000/N]);
xlabel('Error e[n]'); ylabel('Counts');
title(['B = ',num2str(B),'; \mu = ',num2str(eavg),'; \sigma^2 = ',num2str(evar)]);
```

```
set(gca,'xtick',[-2^(-(B+1)),0,2^(-(B+1))]);
```



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

Comments: For $B = 3$ bits, the error histogram has 'boxy' shape but with non-uniform top. As B increases to 6 and 12, the histogram becomes more and more uniform. Thus our model of multiplication quantization error that assumes that the quantization error after multiplication is uniformly distributed between $-\Delta/2$ to $\Delta/2$ is accurate.

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: Note that the passband edge frequencies are not given. However, order of the filter is known. We also know that there is a relationship between transition bandwidth and order of the filter. Hence it is possible to determine passband edge frequencies iteratively. Since no other information is available, we are free to make few choices. We will assume that two transition bandwidths (upper and lower) are equal and that the ripples in each band are also equal. Then, starting with an assumed transition bandwidth $\Delta f = \Delta\omega/\pi$, we will compute $f_{p_1} = f_{s_1} + \delta f$ and $f_{p_2} = f_{s_2} - \delta f$, execute the **firpm** function, and note the resulting δ. If $\delta > \delta_s$ we will increase Δf, otherwise decrease Δf. We will iterate until $\delta \le \delta_s$ to obtain the desired filter impulse response. Since we assumed that $\delta_s = \delta_p$ we do not need a weighting function. Also note that since $\delta_s = \delta_p$, we have

$$A_s = -20\log_{10}\left(\frac{\delta_s}{1+\delta_s}\right) \quad \Rightarrow \quad \delta_s = \frac{10^{-A_s/20}}{1-10^{-A_s/20}}.$$

```
fs1 = 0.2; fs2 = 0.8; As = 60; % Given specifications
L = 32; M = L-1; % Given Order of the filter
deltas = 10^(-As/20)/(1-10^(-As/20)); % Stopband ripple
deltaf = 0.2; % Assumed value
fp1 = fs1+deltaf; fp2 = fs2-deltaf;
fo = [0,fs1,fp1,fp2,fs2,1]; ao = [0,0,1,1,0,0];
[~,delta] = firpm(M,fo,ao);
fprintf('Required ripple: %g, Obtained ripple: %g',deltas,delta);
```

```
Required ripple: 0.001001, Obtained ripple: 0.0017409
```

```
% Hence we need to increase deltaf. After few iterations deltaf = 0.2115
deltaf = 0.2115; % Value after iteration
```

```
fp1 = fs1+deltaf; fp2 = fs2-deltaf;
fo = [0,fs1,fp1,fp2,fs2,1]; a0 = [0,0,1,1,0,0];
[h,delta] = firpm(M,fo,ao);
fprintf('Required ripple: %g, Obtained ripple: %g',deltas,delta);
```

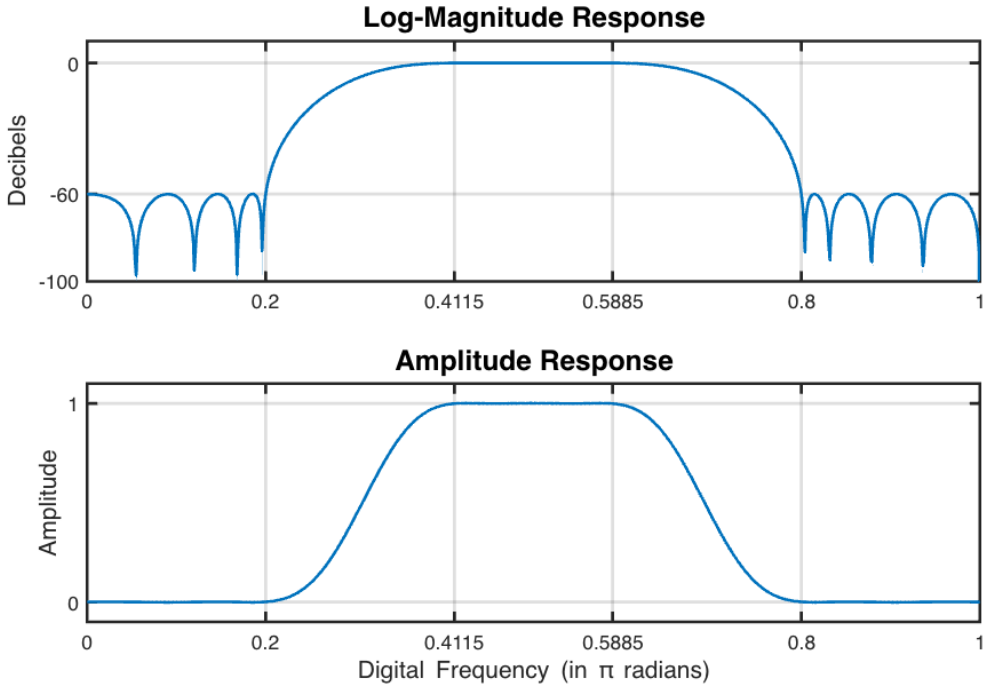
```
Required ripple: 0.001001, Obtained ripple: 0.000991197
```

Now we have the required filter impulse response.

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

Solution: The following script plots the responses

```
f = linspace(0,1,1001); H = zerophase(h,1,f*pi); Hmag = abs(H);
Hdb = 20*log10(Hmag/max(Hmag));
figure('position',[0,0,8,5]*72);
subplot(2,1,1); % Log-magnitude plot
plot(f,Hdb,'linewidth',1.5); axis([0,1,-100,10]);
ylabel('Decibels'); title('Log-Magnitude Response');
set(gca,'xtick',fo,'ytick',[-100,-As,0]); grid;
subplot(2,1,2); % Amplitude plot
plot(f,H,'linewidth',1.5); axis([0,1,-0.1,1.1]);
ylabel('Amplitude'); title('Amplitude Response');
xlabel('Digital Frequency (in \pi radians)');
set(gca,'xtick',fo,'ytick',[0,1]); grid;
```

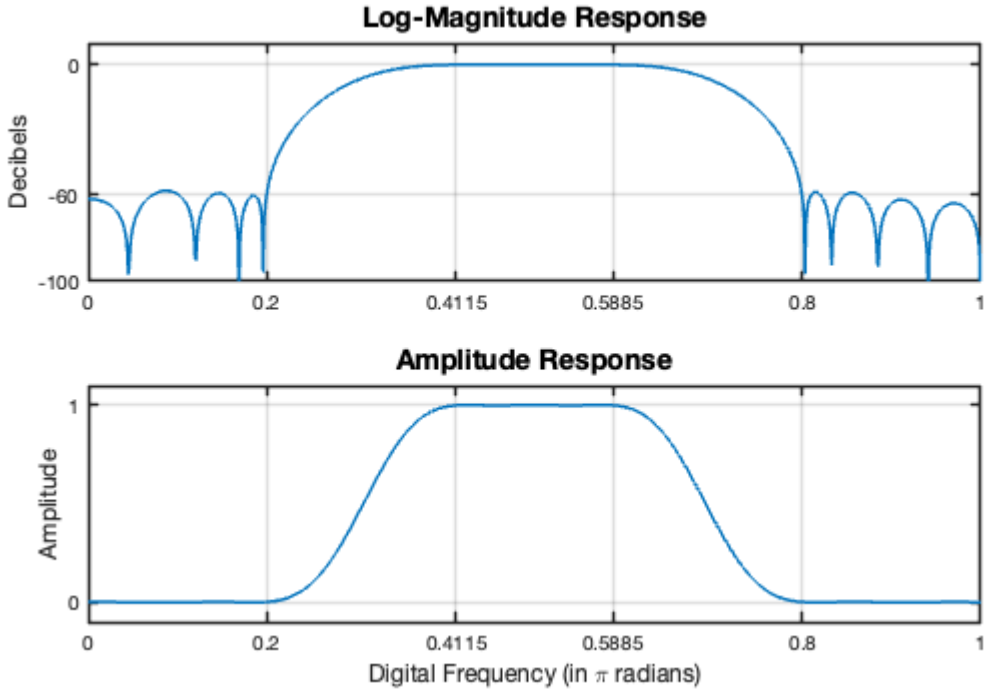


(c) Quantize the direct-form coefficients to 4 decimals (by rounding). Now plot the log-magnitude and amplitude responses of the resulting filter. Use 2 rows and 1 column of subplots.

Solution: The following script plots the responses

```
D = 4; h1 = round(h*10^D)/10^D; %Quantized to 4 decimals
H = zerophase(h1,1,f*pi); Hmag = abs(H);
```

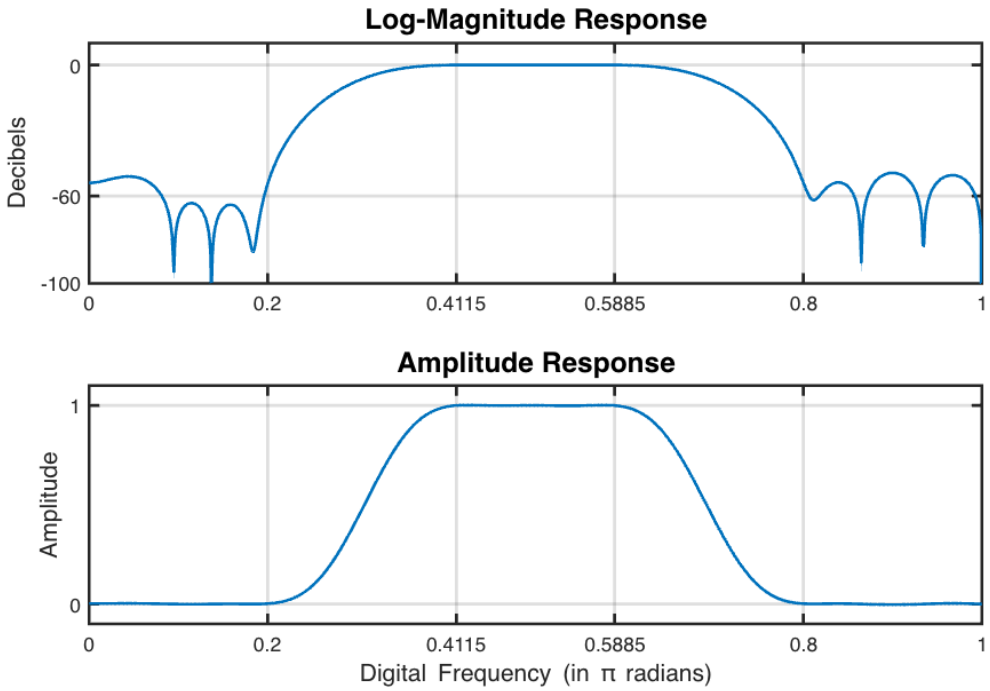
```
Hdb = 20*log10(Hmag/max(Hmag));
figure('position',[0,0,8,5]*72);
subplot(2,1,1); % Log-magnitude plot
plot(f,Hdb,'linewidth',1.5); axis([0,1,-100,10]);
ylabel('Decibels'); title('Log-Magnitude Response');
set(gca,'xtick',fo,'ytick',[-100,-As,0]); grid;
subplot(2,1,2); % Amplitude plot
plot(f,H,'linewidth',1.5); axis([0,1,-0.1,1.1]);
ylabel('Amplitude'); title('Amplitude Response');
xlabel('Digital Frequency (in \pi radians)');
set(gca,'xtick',fo,'ytick',[0,1]); grid;
```



(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.

Solution: The following script plots the responses.

```
D = 3; h2 = round(h*10^D)/10^D; %Quantized to 3 decimals
H = zerophase(h2,1,f*pi); Hmag = abs(H);
Hdb = 20*log10(Hmag/max(Hmag));
figure('position',[0,0,8,5]*72);
subplot(2,1,1); % Log-magnitude plot
plot(f,Hdb,'linewidth',1.5); axis([0,1,-100,10]);
ylabel('Decibels'); title('Log-Magnitude Response');
set(gca,'xtick',fo,'ytick',[-100,-As,0]); grid;
subplot(2,1,2); % Amplitude plot
plot(f,H,'linewidth',1.5); axis([0,1,-0.1,1.1]);
ylabel('Amplitude'); title('Amplitude Response');
xlabel('Digital Frequency (in \pi radians)');
set(gca,'xtick',fo,'ytick',[0,1]); grid;
```



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

Comment: Design specifications in part (c) clearly are not satisfied (although the amplitude doesn't reveal it) while those in part (b) are barely satisfied. This is because some zeros of the filter are closely clustered.

(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.

Solution: We need more than 4 decimal place accuracy. Since each digit is equal to $\log_2(10) = 3.3219$ bits we need $4(\log_2(10)) = 13.2877$ or 14 bits of binary accuracy.

Problem 8.7

Consider the third-order elliptic lowpass filter given by

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

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

(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;
%% Direct form realization of the given elliptic filter system function
b0 = 0.1214; B1 = [1,-1.4211,1]; B2 = [1,1]; b = b0*conv(B1,B2);
A1 = [1,-1.4928,0.8612]; A2 = [1,-0.6183]; a = conv(A1,A2); b,a
```

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

```
a1 = a(2); a2 = a(3); a3 = a(4); % Needed later
% Compute Poles
p = roots(a); p1 = p(1); p2 = p(2); p3 = p(3);
% Compute Derivatives
Del_p1a1 = -p1^(3-1)/((p1-p2)*(p1-p3));
Del_p1a2 = -p1^(3-2)/((p1-p2)*(p1-p3));
Del_p1a3 = -p1^(3-3)/((p1-p2)*(p1-p3));
Del_p2a1 = -p2^(3-1)/((p2-p3)*(p2-p1));
Del_p2a2 = -p2^(3-2)/((p2-p3)*(p2-p1));
Del_p2a3 = -p2^(3-3)/((p2-p3)*(p2-p1));
Del_p3a1 = -p3^(3-1)/((p3-p1)*(p3-p2));
Del_p3a2 = -p3^(3-2)/((p3-p1)*(p3-p2));
Del_p3a3 = -p3^(3-3)/((p3-p1)*(p3-p2));
% Compute Change in Coefficient values for N-bit Representation
L = 16; [ahat,E,B] = dec2beqR(a,L);
Del_a1 = abs(a1-ahat(2)); Del_a2 = abs(a2-ahat(3)); Del_a3 = abs(a3-ahat(4));
% Compute pole sensitity for N-bit Representation
Sp1 = abs(Del_p1a1*Del_a1 + Del_p1a2*Del_a2 + Del_p1a3*Del_a3);
Sp2 = abs(Del_p2a1*Del_a1 + Del_p2a2*Del_a2 + Del_p2a3*Del_a3);
Sp3 = abs(Del_p3a1*Del_a1 + Del_p3a2*Del_a2 + Del_p3a3*Del_a3);
% Print Results
fprintf('Direct-Form Pole Sensitivity for L=(1+%i+%i)-bit Representation:\n',E,B);
```

```
Direct-Form Pole Sensitivity for L=(1+2+13)-bit Representation:
```

```
fprintf('  Pole-1: %9.8f,   Pole-2: %9.8f,   Pole-3: %9.8f',Sp1,Sp2,Sp3);
```

```
   Pole-1: 0.00005743,   Pole-2: 0.00005743,   Pole-3: 0.00008378
```

(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:

```
%% (b) Cascade form
% Second-Order Section
    a1 = A1(2); a2 = A1(3);
    pso = roots(A1); p1 = pso(1); p2 = pso(2);
    % Compute Derivatives
    Del_p1a1 = -p1^(2-1)/(p1-p2); Del_p1a2 = -p1^(2-2)/(p1-p2);
    Del_p2a1 = -p2^(2-1)/(p2-p1); Del_p2a2 = -p2^(2-2)/(p2-p1);
    % Compute Change in Coefficient values for N-bit Representation
    L = L-1; [ahat,E1,B1] = dec2beqR(A1,L);
    Del_a1 = abs(a1-ahat(2)); Del_a2 = abs(a2-ahat(3));
    % Compute pole sensitity for N-bit Representation
    Sp1 = abs(Del_p1a1*Del_a1 + Del_p1a2*Del_a2);
    Sp2 = abs(Del_p2a1*Del_a1 + Del_p2a2*Del_a2);
% First-Order Section
    a1 = A2(2);
    pfo = roots(A2); p1 = pfo(1);
    % Compute Change in Coefficient value for N-bit Representation
    [ahat,E2,B2] = dec2beqR(A2,L);     Del_a1 = abs(a1-ahat(2));
```

```
    % Compute pole sensitity for N-bit Representation
    Sp3 = abs(Del_a1);
% Print Results
fprintf('Cascade-Form Pole Sensitivity for L=(1+%i+%i)-bit Representation:',E2,B2);
```

```
 Cascade-Form Pole Sensitivity for L=(1+1+13)-bit Representation:
```

```
fprintf('  Pole-1: %9.8f,  Pole-2: %9.8f,  Pole-3: %9.8f',Sp1,Sp2,Sp3);
```

```
   Pole-1: 0.00000703,  Pole-2: 0.00000703,  Pole-3: 0.00001387
```

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

Answer: The pole sensitivity results for the cascade structures are almost ten times (one order of magnitude) better than those for the direct form structures.

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;
om = linspace(0,1,501)*pi;
```

(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; % Specifications
[N,omegac] = cheb2ord(omegap/pi,omegas/pi,Ap,As); % Order calculation
[b,a] = cheby2(N,As,omegac); % Direct form coefficients
[sos,G] = tf2sos(b,a) % cascade form coefficients
```

```
 sos = 4×6
     1.0000    1.6319    1.0000    1.0000   -0.4062    0.0608
     1.0000    0.1955    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   -1.1803    0.8180
 G = 0.0125
```

Cascade form expression for $H(z)$:

$$H(z) = \left(\frac{1+1.6319z^{-1}+z^{-2}}{1-0.4062z^{-1}+0.0608z^{-2}}\right)\left(\frac{1+1.1955z^{-1}+z^{-2}}{1-0.6136z^{-1}+0.2391z^{-2}}\right)\times \left(\frac{1-0.5929z^{-1}+z^{-2}}{1-0.9024z^{-1}+0.5079z^{-2}}\right)\left(\frac{1-0.8769z^{-1}+z^{-2}}{1-1.1803z^{-1}+0.818z^{-2}}\right)$$

(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.

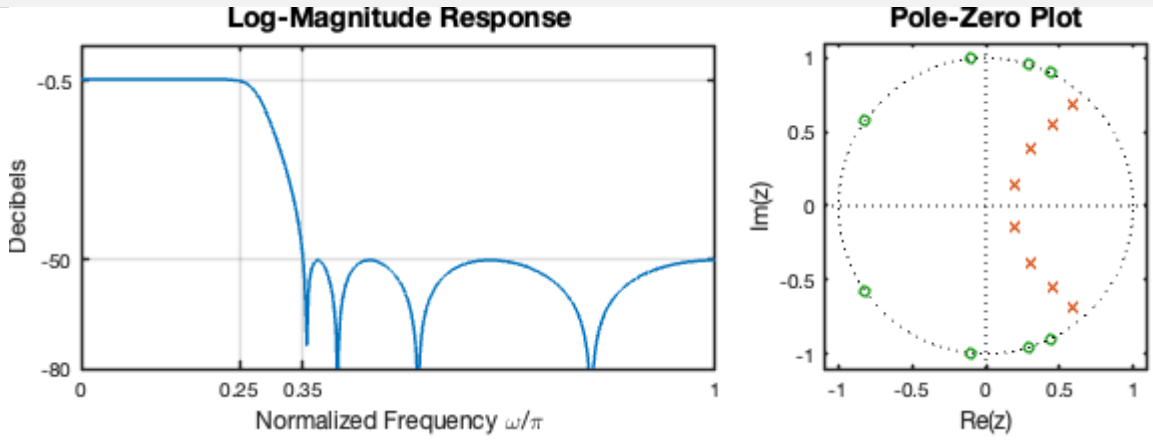
```
figure('position',[0,0,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]); % Magnitude response plot
% Enter your plot code here

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
```

```
xlabel('Re(z)'); ylabel('Im(z)'); title('Pole-Zero Plot');
```

MATLAB script:

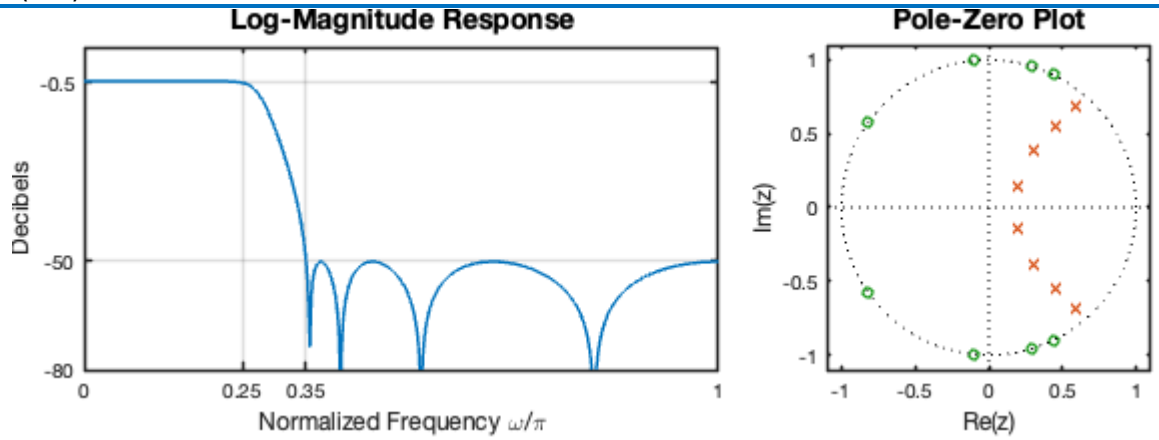
```
H = freqz(b,a,om); Hmag = abs(H); Hdb = 20*log10(Hmag);
figure('position',[1,1,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]);
plot(om/pi,Hdb,'linewidth',1.5); axis([0,1,-80,10]);
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response');
set(gca,'xtick',[0,omegap,omegas,pi]/pi,'ytick',[-80,-As,-Ap]); grid;
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]);
[Hz,Hp,Hl] = zplane(b,a);
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
xlabel('Re(z)'); ylabel('Im(z)'); title('Pole-Zero Plot');
```



(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:

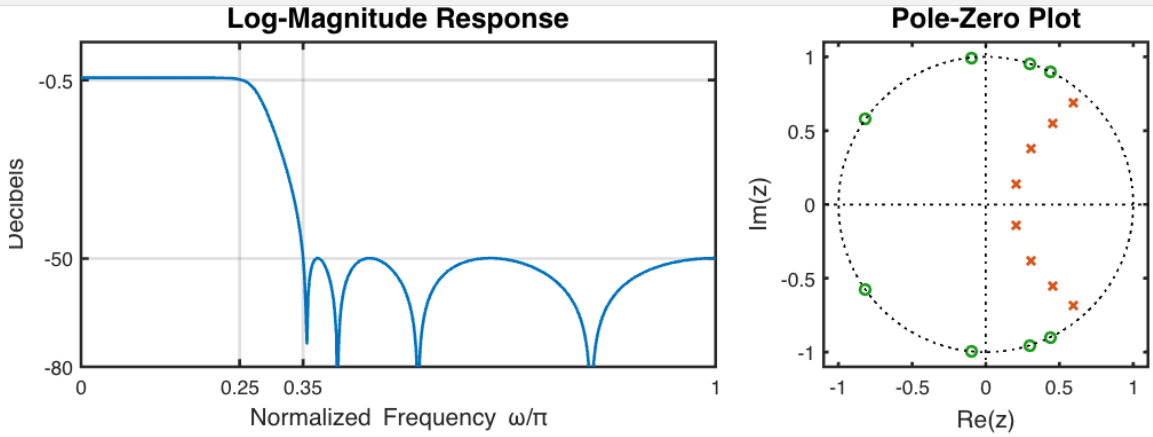
```
L = 16; [soshat,~,~] = dec2beqR([sos;G,zeros(1,5)],L);
[bhat,ahat] = sos2tf(soshat(1:end-1,:),soshat(end,1));
H = freqz(bhat,ahat,om); Hmag = abs(H); Hdb = 20*log10(Hmag);
figure('position',[1,1,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]);
plot(om/pi,Hdb,'linewidth',1.5); axis([0,1,-80,10]);
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response');
set(gca,'xtick',[0,omegap,omegas,pi]/pi,'ytick',[-80,-As,-Ap]); grid;
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]);
[Hz,Hp,Hl] = zplane(bhat,ahat);
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
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:

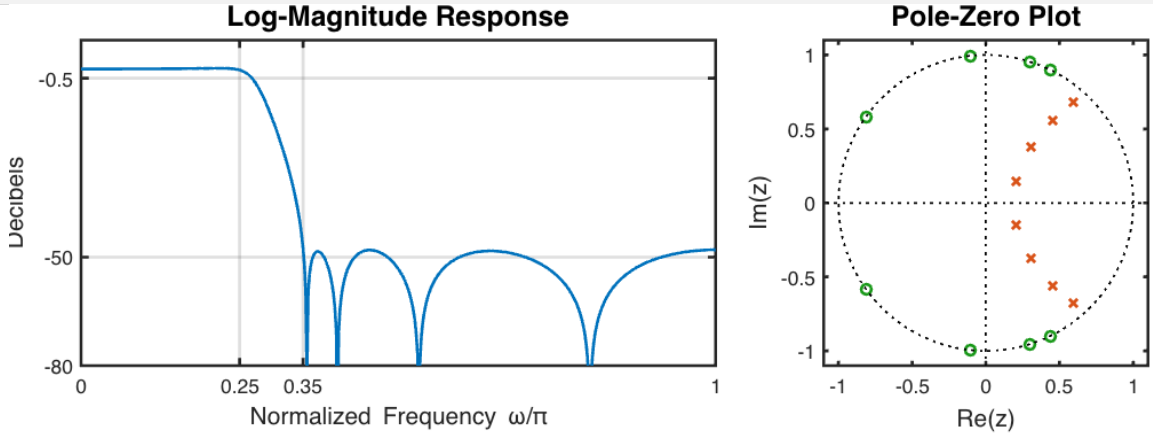
```
L = 12; [soshat,~,~] = dec2beqR([sos;G,zeros(1,5)],L);
[bhat,ahat] = sos2tf(soshat(1:end-1,:),soshat(end,1));
H = freqz(bhat,ahat,om); Hmag = abs(H); Hdb = 20*log10(Hmag);
figure('position',[1,1,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]);
plot(om/pi,Hdb,'linewidth',1.5); axis([0,1,-80,10]);
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response');
set(gca,'xtick',[0,omegap,omegas,pi]/pi,'ytick',[-80,-As,-Ap]); grid;
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]);
[Hz,Hp,Hl] = zplane(bhat,ahat);
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
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:

```
L = 8; [soshat,E,B] = dec2beqR([sos;G,zeros(1,5)],L);
[bhat,ahat] = sos2tf(soshat(1:end-1,:),soshat(end,1));
H = freqz(bhat,ahat,om); Hmag = abs(H); Hdb = 20*log10(Hmag);
figure('position',[1,1,8,3]*72);
subplot('position',[0.5/8,0.5/3,4.4/8,2.25/3]);
plot(om/pi,Hdb,'linewidth',1.5); axis([0,1,-80,10]);
xlabel('Normalized Frequency \omega/\pi'); ylabel('Decibels');
title('Log-Magnitude Response');
set(gca,'xtick',[0,omegap,omegas,pi]/pi,'ytick',[-80,-As,-Ap]); grid;
subplot('position',[5.65/8,0.5/3,2.25/8,2.25/3]);
[Hz,Hp,Hl] = zplane(bhat,ahat);
set(Hz,'markersize',5,'linewidth',1.5,'color',[0.1,0.6,0.1]);
set(Hp,'markersize',6,'linewidth',1.5,'color',[0.85,0.325,0.098]);
set(Hl,'linewidth',1,'color','k');
xlabel('Re(z)'); ylabel('Im(z)'); title('Pole-Zero Plot');
```



(f) Comment on your plots.

Answer: From the four sets of plots above, we note that the cascade form realization is acceptable for $L = 16$ and $L = 12$ bits in coefficient representation but breaks down when cascade coefficients are quantized to $L = 8$ bits.