"Well, it makes a difference to me!"

EE 311 **April 22, 2019**
Lecture 25

8.4 Coefficient Quantization Error

There are two ways to reduce the effects of quantization error.
The first of these is obvious – increase the number of bits.

The second method uses a realization structure that is less sensitive to quantization error. As we shall see, this is a viable option so that choosing the right realization for a given system is often a critical part of the design process.

To get a better idea of what quantization does consider the poles of a second order system with a denominator given by

$$D(z) = z \ + a_1 z + a_2$$

There is a complex pole set given by

$p_{1,2} = r\angle\theta$ where $r = \sqrt{a_2}$, and $\theta = \cos^{-1}(a_1/(-2\sqrt{a_2}))$

When a_1 and a_2 are quantized, the values of r and θ are restricted to a set of discrete points in the z-plane. Figure 8.24 shows this set of discrete locations for the case where the quantization level is set to 6-bits.

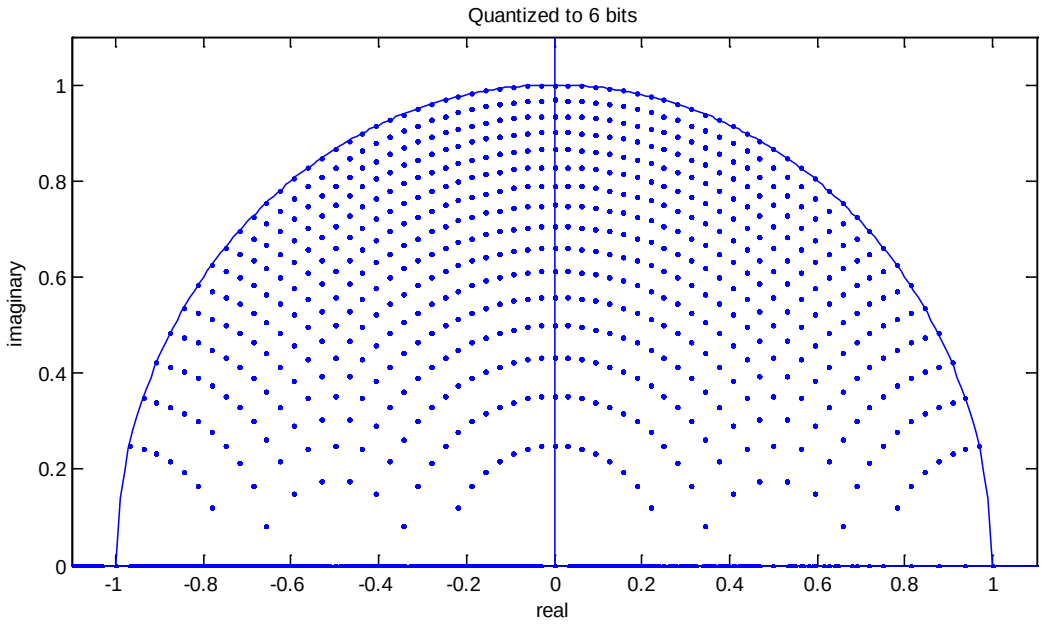

Figure 8.24

Points in this figure indicate discretized locations of roots of $z^2 + a_1 z + a_2 = 0$ when a1 and a2 are quantized to 6-bits.

Since there is more space between allowable pole locations for very low frequencies and very high frequencies, Figure 8.24 implies that poles (or zeros) with either a small angle or a large angle will be more sensitive to quantization error.

To generalize this a bit more, consider a system that has three real poles where the denominator is given by

$$D(z) = (z - p_1)(z - p_2)(z - p_3) = z^3 + a_1 z^2 + a_2 z + a_3$$

We want to determine how sensitive each pole, p_i is to quantization of each coefficient a_i. Taking the partial derivatives gives

$$-\frac{\partial p_1}{\partial a_1}(z - p_2)(z - p_3) = z^2 \text{ or}$$

$$\left.\frac{\partial p_1}{\partial a_1}\right|_{z=p_1} = \frac{-p_1^2}{(p_1 - p_2)(p_1 - p_3)}$$

Likewise,

$$\left.\frac{\partial p_2}{\partial a_1}\right|_{z=p2} = \frac{-p_2^2}{(p_2 - p_1)(p_2 - p_3)}$$

$$\left.\frac{\partial p_3}{\partial a_1}\right|_{z=p_3} = \frac{-p_3^2}{(p_3 - p_1)(p_3 - p_2)}$$

The total movement of a pole, say p_1 is given by

$$\Delta p_1 = \frac{\partial p_1}{\partial a_1}\Delta a_1 + \frac{\partial p_1}{\partial a_2}\Delta a_2 + \frac{\partial p_1}{\partial a_3}\Delta a_3$$

In general, for the case where we have N real poles we can write

$$\Delta p_k = \frac{-p_k^{N-j}}{\prod_{\substack{i=1 \\ i \neq k}}^{N}(p_k - p_i)}\Delta a_j \qquad (8.29)$$

From this equation we see that to minimize the movement in a pole due to the quantization of a coefficient we need to make the denominator larger. The denominator is the product of terms which can be represented with vectors whose magnitude is the distance between p_k and p_i. This would imply that we need to make the poles further apart. With several more pages of algebra a similar result can be attained for complex poles [2].

Figure 8.25 shows two 10th order Butterworth low pass filters which meet identical specifications except that one of them has twice the sample frequency of the other. The result of doubling the sample frequency is to move the cutoff frequency to a lower angle resulting in a more clustered set of poles which are, in turn, more sensitive to coefficient quantization error. Contrary to what many believe to be true, raising the sample frequency can increase the noise for a given filter.

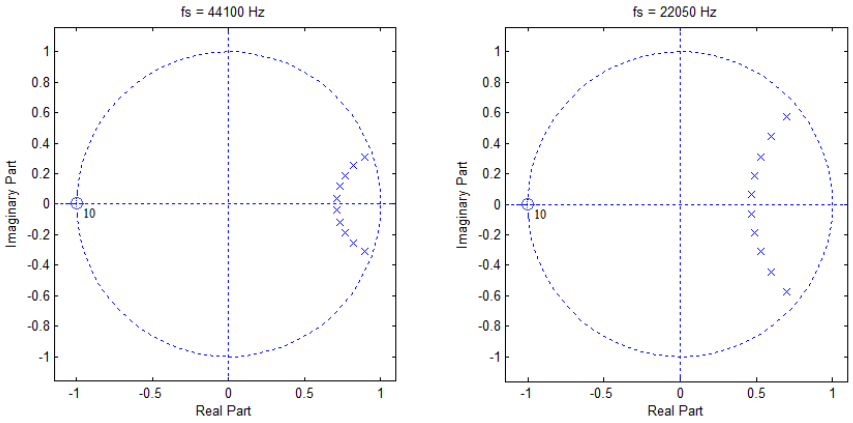

Figure 8.25
Pole/zero plots for two 10th order Butterworth low pass filters which meet the same specifications except the plot on the left has a sample frequency of 44,100 Hz and the plot on the right has a sample frequency of 22,050 Hz.

Forming a cascade or parallel realization can decrease the coefficient quantization noise. Consider the denominator of a second order section with complex poles given by

$$D(z) = z^2 + a_1 z + a_2$$

If the poles are located at $r\angle \pm\theta$, we have

$$a_1 = -(r/2)\cos(\theta) \text{ and } a_2 = r^2$$

If the quantized error is small we can use the chain rule to write

$$\Delta r = \frac{\partial r}{\partial a_1}\Delta a_1 + \frac{\partial r}{\partial a_2}\Delta a_2 = \frac{\Delta a_2}{2\sqrt{a_2}} = \frac{\Delta a_2}{2r}$$

Since $\cos(\theta) = a_1/(2\sqrt{a_2})$ we can take the derivative of both sides to get

$$\Delta\theta = \frac{\partial\theta}{\partial a_1}\Delta a_1 + \frac{\partial\theta}{\partial a_2}\Delta a_2 = \frac{-\Delta a_1}{2r\sin(\theta)} + \frac{\Delta a_2}{2r^2\tan(\theta)}$$

We see that a second order section has the greatest coefficient quantization error when the poles (or zeros) are close to the origin making r small, or when the pole frequency is low making θ small. This result is in accord with equation (8.29) since making θ larger makes the complex poles further apart.

The following example illustrates how cascaded second order sections can reduce the effects of quantization error.

Example 8.9
Create a Butterworth band stop filter to meet the specifications below. Quantize the coefficients to 12-bits and use the MATLAB® function `freqz` to plot the frequency response of the resulting filter. Compare this frequency response plot to the same frequency response plot created when the original filter is broken into second order sections and the coefficients of the second order sections are quantized to 12-bits.

Specifications:

Sample frequency: 11025 Hz
Pass and stop band ripple: 0.02
Pass bands: 0 to 600 Hz and 2500 Hz to fs/2
Stop band: 1100 Hz to 2000 Hz.

Solution

The following lines in MATLAB® create the band stop filter.

```
fs = 11025;
Rp = .02;RpDB = -20*log10(1-Rp);
Rs = .02;RsDB = -20*log10(Rs);
flp = 600;fls = 1100;fus = 2000;fup = 2500;
Wp = [flp fup]/(fs/2);
Ws = [fls fus]/(fs/2);
[N Wc] = buttord(Wp,Ws,RpDB,RsDB);
[num den] = butter(N, Wc, 'stop');
```

To quantize the coefficients in num and den to 12-bits we use a function called "quantize" which normalizes the maximum coefficient to 1, multiplies by 2^{12}, truncates the result by rounding, divides by 2^{12}, and denormalizes the result. For example, if we want to quantize 1.234567 to 12 bits when the maximum coefficient is 2.0 we do the following:

Normalize: 1.234567/2 = 0.6172835
Multiply by 4096: 2528.393216
Round to an integer: 2528.0
Divide by 4096: 2528/4096 = 0.6171875
Denormalize: 2 x 0.6171875 = 1.234375

After quantizing each coefficient of the numerator and denominator we use the `freqz` function to get the frequency response and plot the result.

To get the result for second order sections we use the function `tf2sos` to let MATLAB® factor num and den into quadratic equations and pair the pole and zero terms.

```
sos = tf2sos(num, den);
```

After this operation `sos` will be a matrix of the second order sections which we can "pull apart", find the magnitude response for each section, and multiply them to get a final result. The MATLAB® code looks like this.

```
num2 = sos(1:1,1:3); den2 = sos(1:1,4:6);
[H2 fd] = freqz(num2,den2,1024,fs);
H = abs(H2);
for i = 2:N
  num2 = sos(i:i,1:3); den2 = sos(i:i,4:6);
  [H2 fd] = freqz(num2,den2,1024,fs);
  H = H.*abs(H2);
end
plot(fd,abs(H2),'k');
```

Figure 8.26 shows the results for both methods on the same plot.

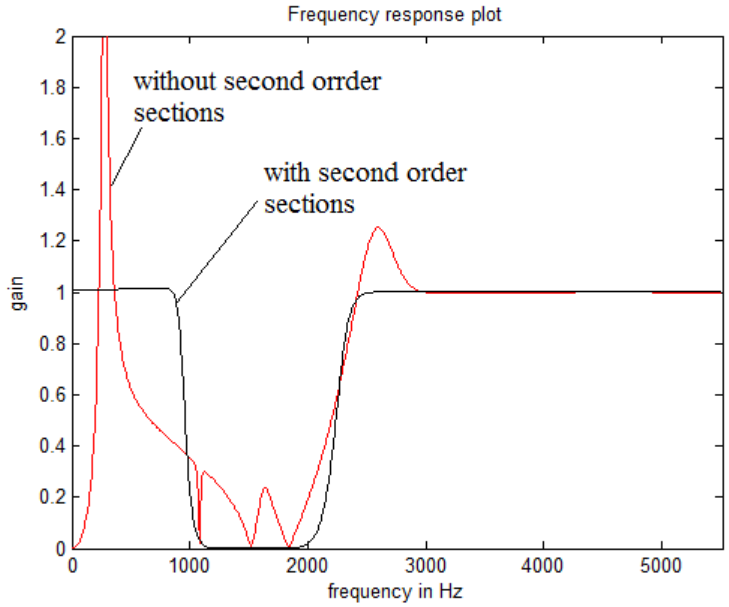

Figure 8.26
Frequency plot of an 18th order Butterworth band stop filter. All coefficients were truncated to 12-bits. In one case, the filter was evaluated as a sequence of second order sections.

We see that truncating the second order coefficients to 12-bits did not significantly alter the frequency response of the filter whereas, when we try to evaluate the 18th order filter as two 18th order polynomials with 12-bit coefficients the numerical error accumulates to make the filter nearly useless.

It is possible in most cases to get a very good idea of the level of quantization that can be tolerated by a given realization by finding the frequency response in MATLAB® using successively larger quantization levels for the coefficients. Figure 8.27 shows a 5th order low pass elliptic filter pass band when the quantization level varies from 18 bits to 10 bits.

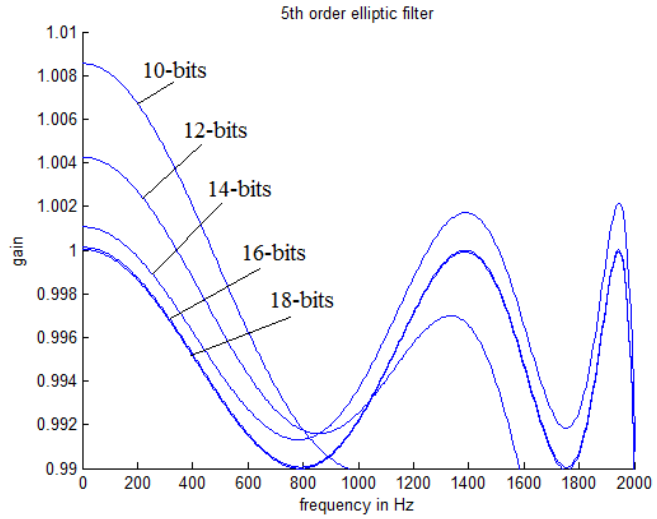

Figure 8.27
The pass band of a 5th order low pass elliptic filter for various quantization levels. The two smallest bit sizes produce filters which do not meet specifications.