Design IIR Filters Using Cascaded Biquads

This article shows how to implement a Butterworth IIR lowpass filter as a cascade of second-order IIR filters, or biquads. We’ll derive how to calculate the coefficients of the biquads and do some examples using a Matlab function `biquad_synth` provided in the Appendix. Although we’ll be designing Butterworth filters, the approach applies to any all-pole lowpass filter (Chebyshev, Bessel, etc). As we’ll see, the cascaded-biquad design is less sensitive to coefficient quantization than a single high-order IIR, particularly for lower cut-off frequencies [1, 2].

In an earlier post on IIR Butterworth lowpass filters [3], I presented the pole-zero form of the lowpass response $H(z)$ as follows:

$$H(z) = K \frac{(z + 1)^N}{(z - p_1)(z - p_2) \cdots (z - p_N)} \quad (1)$$

The N zeros at $z = 1$ ($\omega = \pi$ or $f = f_s/2$) occur when we transform the lowpass analog zeros from the $s$-domain to $z$-domain using the bilinear transform. Our goal is to convert $H(z)$ into a cascade of second-order sections. If we stipulate that N is even, then we can write $H(z)$ as:

$$H(z) = K_1 \frac{(z + 1)^2}{(z - p_1)(z - p_2)} \cdot K_2 \frac{(z + 1)^2}{(z - p_3)(z - p_4)} \cdots K_N \frac{(z + 1)^2}{(z - p_{N-1})(z - p_N)} \quad (2)$$

Each term in equation 2 is biquadratic – it has quadratic numerator and denominator. It is not necessary to use a separate gain $K$ for each term; we could also use just a single gain for the whole cascade.

The filter is even order, so all poles occur in complex-conjugate pairs. We’ll assign a complex-conjugate pole pair to the denominator of each term of equation 2. We can then write each term as:

$$H_k(z) = K_k \frac{(z + 1)^2}{(z - p_k)(z - p^*_k)}, \quad k = 1:N/2$$

where $p^*_k$ is the complex conjugate of $p_k$. Expanding the numerator and denominator, we get:

$$H_k(z) = K_k \frac{z^2 + 2z + 1}{z^2 + a_1z + a_2}$$

where $a_1 = -2*\text{real}(p_k)$ and $a_2 = |p_k|^2$. Dividing numerator and denominator by $z^2$, we get:

$$H_k(z) = K_k \frac{1 + 2z^{-1} + z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}} \quad (3)$$

We want the gain of each biquad section to equal 1 at $\omega = 0$. Letting $z = e^{j\omega}$, we have $z = 1$. Then:
\[ H_k(z) = 1 = K_k \sum b \frac{\sum a}{4} \]

so

\[ K_k = \frac{\sum a}{4} \quad (4) \]

where \( a = [a_1 a_2] \) are the denominator coefficients of the biquad section. Summarizing the coefficient values, we have:

\[
\begin{align*}
  b &= [1 \ 2 \ 1] \\
  a &= [1 \ -2 \ \text{real}(p_k) \ |p_k|^2] \\
  K &= \sum a/4
\end{align*}
\]

A biquad lowpass block diagram using the Direct form II structure [4,5] is shown in Figure 1. We will cascade \( N/2 \) biquads to implement a \( N \)th order filter (\( N \) even). Note that the feed-forward coefficients \( b \) have the same value for all \( N/2 \) biquads in a filter. This is evident from Equation 3.

![Figure 1 Biquad (second-order) lowpass all-pole filter](image_url)

Direct form II
Example

In this example, we’ll use `biquad_synth` to design a 6th order Butterworth lowpass filter with -3 dB frequency of 15 Hz and $f_s=100$ Hz. Note `biquad_synth` uses the bilinear transform with prewarping [3] to transform $H(s)$ to $H(z)$. The filter will consist of three biquads, as shown in Figure 2.

`biquad_synth` computes the denominator (feedback) coefficients $a$ of each biquad. The gains $K$ are computed separately. Note `biquad_synth` contains code developed in an earlier post on IIR Butterworth filter synthesis [3]. Here is the function call and the function output:

```matlab
N= 6; % filter order
fc= 15; % Hz -3 dB frequency
fs= 100; % Hz sample frequency
a= biquad_synth(N,fc,fs)
```

```
a =
  1.0000  -0.6599  0.1227
  1.0000  -0.7478  0.2722
  1.0000  -0.9720  0.6537
```

Each row of the matrix $a$ contains the denominator coefficients of a biquad. As we already determined, the numerator coefficients $b$ are the same for all three biquads:

```matlab
b= [1 2 1];
```

The gains for each biquad are, from equation 4:

```matlab
K1= sum(a(1,:)/4;
K2= sum(a(2,:)/4;
K3= sum(a(3,:)/4;
```

Now we can compute the frequency response of each biquad. The overall response is their product.

```matlab
[h1,f] = freqz(K1*b,a(1,:),512,fs);
h2 = freqz(K2*b,a(2,:),512,fs);
h3 = freqz(K3*b,a(3,:),512,fs);
h= abs(h1.*h2.*h3);
H= 20*log10(abs(h));
```

The magnitude response of each biquad and the overall response are plotted in Figure 3. The sequence of the biquads doesn’t matter in theory; however, placing the biquad with the peaking response ($h3$) last minimizes the chance of clipping.
Figure 2. 6th order lowpass filter using three biquads

Figure 3. 6th order lowpass Butterworth cascaded-biquad response. $f_c = 15$ Hz, $f_s = 100$ Hz.
Top: response of each biquad section (blue= h1, green= h2, red= h3).
Bottom: overall response
Coefficient Quantization

As I stated at the beginning, the cascaded-biquad design is less sensitive to coefficient quantization than a single high-order IIR, particularly for lower cut-off frequencies. To illustrate this, we’ll first look at how quantizing coefficients effects z-plane pole locations of a 6th order IIR filter. The following code finds the unquantized poles of the 6th order Butterworth filter with -3 dB frequency \( f_c = 5 \) Hz. (Note `butter [6]` is a function in the Matlab signal processing toolbox that synthesizes IIR Butterworth filters).

\[
\begin{align*}
f_c &= 5; \\
fs &= 100; \\
[b,a] &= \text{butter}(6,2*fc/fs); \quad \% \text{Matlab function for Butterworth LP IIR} \\
p &= \text{roots}(a); \quad \% \text{poles in z-plane}
\end{align*}
\]

The poles are plotted as the red x’s on the left side of Figure 4. We have also plotted the poles for \( f_c = 12 \) Hz (blue-ish x’s). Each set contains 6 poles. If we plot the poles of filters having \( f_c \) from 1 Hz to 25 Hz in 1 Hz increments, we get the plot on the right, where only the right side of the unit circle is shown. The lower values of \( f_c \) are on the right, near \( z = 1 \).

![Figure 4. Unquantized poles of 6th-order Butterworth IIR filter. Left: \( f_c = 5 \) Hz (red) and 12 Hz (blue). Right: \( f_c = 1 \) Hz to 25 Hz.](image-url)
Now let’s quantize the denominator coefficients and see how this effects the pole locations of Figure 4. Let nbits = the number of bits per unit of coefficient amplitude:

nbits = 16;

Here is the code to find the quantized poles for a single value of fc:

```matlab
fs = 100;
[b,a] = butter(6,2*fc/fs);
a_quant = round(a*2^nbits)/2^nbits;
p_quant = roots(a_quant);
```

Letting fc vary in 0.5 Hz increments from 0.5 to 25 Hz, we get the poles shown on the left of Figure 5. As you can see, as fc decreases, quantization causes the poles to depart from the desired locations. The right side of Figure 5 shows the effect of 10-bit quantization.

Figure 5. Effect of Quantization on poles of 6th-order Butterworth IIR filter.
Left: nbits = 16 Right: nbits = 10
We can do the same calculation for the biquads that make up the 6th order cascaded implementation. For example, here is the code to find the quantized poles of the second biquad for a single value of \( f_c \)
(recall that the matrix \( a \) has three rows containing the coefficients of 3 biquads).

```bash
nbits= 10;
a= biquad_synth(6,fc,fs);
a2= a(2,:); % 2nd biquad
a_quant= round(a2*2^nbits)/2^nbits;
p_quant = roots(a_quant);
```

This time, letting \( fc \) vary in 0.25 Hz increments from 0.25 to 25 Hz, we get the poles shown in Figure 6, which includes only quadrant 1 of the unit circle. The biquad performs much better than the 6th order filter, only departing dramatically from the unquantized curve for \( fc = 0.25 \) Hz. So we expect better performance from cascading three biquads vs. using a single 6th-order filter.

![Figure 6. Effect of Quantization on poles of one biquad, nbits = 10.](image)

Now we’re finally ready to compare frequency response of a biquad-cascade filter vs. a conventional IIR filter when the denominator coefficients are quantized. The cutoff frequency and quantization level are chosen to stress the conventional filter. We’ll leave the numerator coefficients of the conventional filter
as floating-point. Interestingly, when implementing the biquad filter we get exact numerator coefficient values “for free”: since \( b = [1 \ 2 \ 1] \), we can implement \( b_0 \) and \( b_2 \) as no-ops and \( b_1 \) as a bit shift.

For the biquad filter, we use biquad_synth to find the coefficients for \( f_c = 6.7 \) Hz:

```matlab
fc=6.7; % Hz -3 dB frequency
fs= 100; % Hz sample frequency
a = biquad_synth(6,fc,fs) % a has 3 rows, one for each biquad
a =
1.0000 -1.3088  0.4340
1.0000 -1.4162  0.5516
1.0000 -1.6508  0.8087
```

For the conventional filter, we again use the Matlab function butter:

```matlab
[b,a]= butter(6,2*fc/fs);
a =
1.0000 -4.3757  8.1461 -8.2269  4.7417 -1.4761  0.1936
```

In each case, we quantize coefficients to 10 bits per unit of coefficient amplitude:

```matlab
nbits= 10;
a_quant= round(a*2^nbits)/2^nbits; % quantize denom coeffs
```

First, we’ll look at the quantized pole locations. For the conventional filter, the quantized poles are:

```matlab
p_quant= roots(a_quant);
```

For the biquad implementation, the quantized poles are:

```matlab
p1= roots(a_quant(1,:))';
p2= roots(a_quant(2,:))';
p3= roots(a_quant(3,:))';
p_quant= [p1 p2 p3];
```

Figure 7 shows the z-plane poles for the floating-point and quantized coefficients. Quantization has little effect on the biquad version, but has a large effect on the conventional filter.

Now let’s compare the magnitude responses for quantized coefficients. We compute the response of the biquad version in the same way used to obtain Figure 3. Figure 8 shows the magnitude responses. As you would expect from the pole plots, the conventional implementation has poor performance, while the biquad implementation shows no noticeable effect due to quantization.

So how low can we go with this \( N= 6 \) Butterworth cascaded-biquad filter? As we reduce \( f_c \), the quantization of 1024 steps per unit of coefficient amplitude eventually takes a toll. Figure 9 shows the z-plane poles and magnitude response for \( f_c = 1.6 \) Hz. As you can see, the magnitude response is sagging. If we stay above \( f_c \) of 2.5 Hz = \( f_s/40 \), the response error is less than 0.1 dB.

Besides adding coefficient bits, there are other ways to improve performance of narrow-band IIR filters. See for example the post in reference [7].
Figure 7. Z-plane pole locations with quantized denominator coefficients. $N=6$, $f_c=6.7$ Hz, $f_s=100$ Hz.  
Blue = floating point, Red = 10 bit quantization. 
Left: biquad implementation  Right: conventional implementation.

Figure 8. Magnitude response with quantized denom coeffs. $N=6$, $f_c=6.7$ Hz, $f_s=100$ Hz, nbits = 10.  
Blue = biquad implementation, green= conventional implementation.
Figure 9. Z-plane poles and magnitude response of biquad-cascade filter with quantized denominator coefficients. Blue x’s = floating-point and red x’s = quantized. N= 6, $f_c = 1.6$ Hz, $f_s = 100$ Hz, nbits= 10.
References


Appendix  

Matlab Function biquad_synth

This program is provided as-is without any guarantees or warranty. The author is not responsible for any damage or losses of any kind caused by the use or misuse of the program.

% biquad_synth.m    2/10/18 Neil Robertson
% Synthesize even-order IIR Butterworth lowpass filter as cascaded biquads.
% This function computes the denominator coefficients a of the biquads.

% N= filter order (must be even)
% fc= -3 dB frequency in Hz
% fs= sample frequency in Hz
% a = matrix of denominator coefficients of biquads. Size = (N/2,3)
% each row of a contains the denominator coeffs of a biquad.
% There are N/2 rows.
% Note numerator coeffs of each biquad= K*[1 2 1], where K = (1 + a1 + a2)/4.
% function a = biquad_synth(N,fc,fs);

if fc>=fs/2;
    error('fc must be less than fs/2')
end

if mod(N,2)~=0
    error('N must be even')
end

% I. Find analog filter poles above the real axis (half of total poles)
k= 1:N/2;
theta= (2*k -1)*pi/(2*N);
pa= -sin(theta) + j*cos(theta);  % poles of filter with cutoff = 1 rad/s
pa= fliplr(pa);  %reverse sequence of poles - put high Q last

% II.  scale poles in frequency
Fc= fs/pi * tan(pi*fc/fs);  % continuous pre-warped frequency
pa= pa*2*pi*Fc;  % scale poles by 2*pi*Fc

% III. Find coeffs of biquads

% poles in the z plane
p= (1 + pa/(2*fs))./(1 - pa/(2*fs));  % poles by bilinear transform

% denominator coeffs
for k= 1:N/2;
    a1= -2*real(p(k));
    a2= abs(p(k))^2;
    a(k,:)= [1 a1 a2];  %coeffs of biquad k
end