Discretization of a Fourth-Order Butterworth Filter

Pieter P

This is an example on how to design a filter in the analog domain, and then use the bilinear transform to transform it to the digital domain, while preserving the cut-off frequency.

We'll be using formulas derived on the Bilinear Transform and Butterworth Filters pages.

Design criteria

In this example, we'll design a digital fourth order Butterworth low-pass filter, with a sample frequency of 360 Hz and a cut-off frequency of 45 Hz.

Frequency Pre-Warping

As discussed in the page on the Bilinear Transform, we have to apply pre-warping to the cut-off frequency before designing a filter. If we don't the cut-off frequency will shift to an incorrect frequency when we discretize the filter.

First, let's calculate the normalized digital frequency $\omega_{c,d}$, using the cut-off frequency f_c and the sample frequency f_s:

$$
\begin{aligned}
f_c &= 45 \text{ Hz} \\
f_s &= 360 \text{ Hz} \\
T_s &= \frac{1}{f_s} \\
\omega_c &= 2\pi f_c \\
&\approx 282.7 \text{ rad } s^{-1} \\
\omega_{c,d} &= \frac{\omega_c}{f_s} \\
&= \frac{\pi}{4} \frac{\text{rad}}{\text{sample}} \approx 0.7854 \frac{\text{rad}}{\text{sample}}
\end{aligned}
$$

The Nyquist-Shannon sampling theorem tells us that we can never sample frequencies higher than $f_s/2$ without losing information. This also means that the cut-off frequency can never be higher than half of the sample frequency. Or in other words, all normalized frequencies will be in the interval $[0, \pi]$.

Next, we'll use the pre-warping formula we derived in the page on the Bilinear Transform, in order to calculate the analog design frequency $\omega_{c,a}$:

$$
\begin{aligned}
\omega_{c,a} &= \frac{2}{T_s} \tan\left(\frac{\omega_{c,d}}{2}\right) \\
&= 720 \tan\left(\frac{\pi}{8}\right) \approx 298.2 \text{ rad } s^{-1}
\end{aligned}
$$

Note that this frequency is relatively close to ω_c, but it is *not* the same. The higher the cut-off frequency (relative to the sample frequency), the larger the error between ω_c and $\omega_{c,a}$.

Designing the Butterworth filter in the Analog Domain

Now that we know the pre-warped analog cut-off frequency, we can start designing the analog filter.
We'll use the formula for the Butterworth low-pass filter derived in the page on Butterworth Filters:

$$
H_4(s') = \frac{1}{B_4(s')} \qquad \text{where } s' \triangleq \frac{s}{\omega_{c,a}} \tag{1}
$$

$$
B_4(s') = \left(s'^2 - 2\cos\left(2\pi\frac{4+1}{4\cdot4}\right)s' + 1\right)\left(s'^2 - 2\cos\left(2\pi\frac{2+4+1}{4\cdot4}\right)s' + 1\right)
$$

Defining these constants will make the calculations much easier:

$$
\begin{aligned}
\alpha &\triangleq -2\cos\left(\frac{5\pi}{8}\right) \\
&= \sqrt{2 - \sqrt{2}}
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
\beta &\triangleq -2\cos\left(\frac{7\pi}{8}\right) \\
&= \sqrt{2 + \sqrt{2}}
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
B_4(s') &= \left(s'^2 + \alpha s' + 1\right)\left(s'^2 + \beta s' + 1\right) \\
&= s'^4 + s'^3(\alpha + \beta) + s'^2(\alpha\beta + 2) + s'(\alpha + \beta) + 1
\end{aligned} \tag{4}
$$

Discretizing the Analog Filter

We can now just apply the Bilinear Transform to the analog transfer function, by substituting $s = \frac{2}{T_s}\frac{z-1}{z+1}$. Therefore:

$$s' = \frac{2f_s}{\omega_{a,c}}\frac{z-1}{z+1}$$

Again, we'll introduce a constant to simplify the expression:

$$\gamma \triangleq \frac{2f_s}{\omega_{a,c}} = \frac{2f_s}{2f_s \tan\left(\frac{\omega_{c,d}}{2}\right)} = \cot\left(\pi\frac{f_c}{f_s}\right)$$

$$= \cot\left(\frac{\pi}{8}\right) = 1 + \sqrt{2} \tag{5}$$

$$s' = \gamma\frac{z-1}{z+1} \tag{6}$$

What follows is just rearranging the expression of $B_4(s')$ from Equation 4, using the substitution of Equation 6.
Finally, we end up with an expression for the transfer function, using Equation 1, and we can determine the coefficients using the constants defined in Equations 2, 3 & 5.

$$B_4(s') = s'^4 + s'^3(\alpha+\beta) + s'^2(\alpha\beta+2) + s'(\alpha+\beta) + 1$$

$$B_4(z) = \gamma^4\frac{(z-1)^4}{(z+1)^4}$$
$$+ \gamma^3\frac{(z-1)^3(z+1)}{(z+1)^4}(\alpha+\beta)$$
$$+ \gamma^2\frac{(z-1)^2(z+1)^2}{(z+1)^4}(\alpha\beta+2)$$
$$+ \gamma\frac{(z-1)(z+1)^3}{(z+1)^4}(\alpha+\beta)$$
$$+ \frac{(z+1)^4}{(z+1)^4}$$

$$= \frac{1}{(z+1)^4}\begin{bmatrix} & \gamma^4(z-1)^4 \\ + & \gamma^3(z-1)^3(z+1)(\alpha+\beta) \\ + & \gamma^2(z-1)^2(z+1)^2(\alpha\beta+2) \\ + & \gamma(z-1)(z+1)^3(\alpha+\beta) \\ + & (z+1)^4 \end{bmatrix}$$

$$= \frac{1}{(z+1)^4}\begin{bmatrix} \gamma^4 & (& z^4 & - & 4z^3 & + & 6z^2 & - & 4z & + & 1 &) & \\ + \gamma^3 & (& z^4 & - & 2z^3 & & & + & 2z & - & 1 &) & (\alpha+\beta) \\ + \gamma^2 & (& z^4 & & & - & 2z^2 & & & + & 1 &) & (\alpha\beta+2) \\ + \gamma & (& z^4 & + & 2z^3 & & & - & 2z & - & 1 &) & (\alpha+\beta) \\ + & (& z^4 & + & 4z^3 & + & 6z^2 & + & 4z & + & 1 &) & \end{bmatrix}$$

$$= \frac{1}{(z+1)^4}\begin{bmatrix} \left(\gamma^4 + \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) + \gamma(\alpha+\beta) + 1\right) & z^4 \\ + & \left(-4\gamma^4 - 2\gamma^3(\alpha+\beta) + 2\gamma(\alpha+\beta) + 4\right) & z^3 \\ + & \left(6\gamma^4 - 2\gamma^2(\alpha\beta+2) + 6\right) & z^2 \\ + & \left(-4\gamma^4 + 2\gamma^3(\alpha+\beta) - 2\gamma(\alpha+\beta) + 4\right) & z \\ + & \left(\gamma^4 - \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) - \gamma(\alpha+\beta) + 1\right) & \end{bmatrix}$$

$$H_4(z) = \frac{1}{B_4(z)}$$

$$= \frac{(z+1)^4}{\begin{bmatrix} \left(\gamma^4 + \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) + \gamma(\alpha+\beta) + 1\right) & z^4 \\ + & \left(-4\gamma^4 - 2\gamma^3(\alpha+\beta) + 2\gamma(\alpha+\beta) + 4\right) & z^3 \\ + & \left(6\gamma^4 - 2\gamma^2(\alpha\beta+2) + 6\right) & z^2 \\ + & \left(-4\gamma^4 + 2\gamma^3(\alpha+\beta) - 2\gamma(\alpha+\beta) + 4\right) & z \\ + & \left(\gamma^4 - \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) - \gamma(\alpha+\beta) + 1\right) & \end{bmatrix}}$$

$$\triangleq \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}}$$

$$
\begin{aligned}
b_0 &= 1 \\
b_1 &= 4 \\
b_2 &= 6 \\
b_3 &= 4 \\
b_4 &= 1 \\[6pt]
a_0 &= \gamma^4 + \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) + \gamma(\alpha+\beta) + 1 \quad \approx 97.95 \\
a_1 &= -4\gamma^4 - 2\gamma^3(\alpha+\beta) + 2\gamma(\alpha+\beta) + 4 \quad \approx -192.8 \\
a_2 &= 6\gamma^4 - 2\gamma^2(\alpha\beta+2) + 6 \quad \approx 170.0 \\
a_3 &= -4\gamma^4 + 2\gamma^3(\alpha+\beta) - 2\gamma(\alpha+\beta) + 4 \quad \approx -70.96 \\
a_4 &= \gamma^4 - \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) - \gamma(\alpha+\beta) + 1 \quad \approx 11.79
\end{aligned}
$$

Frequency Response & Pole-Zero Map

We can check the filter's frequency response to make sure that we didn't make any mistakes. As mentioned in other pages, the frequency response of a digital system can be a obtained by evaluating the transfer function $H(z)$ along the unit circle ($z = e^{j\omega}$). We'll plot the magnitude in decibels.

$$
A(\omega) = 20 \log_{10} \left| H\left(e^{j\omega}\right) \right|
$$

We can also plot the phase angle of the response:

$$
\phi(\omega) = \angle H\left(e^{j\omega}\right)
$$

Bode Diagram

Pole-Zero Map
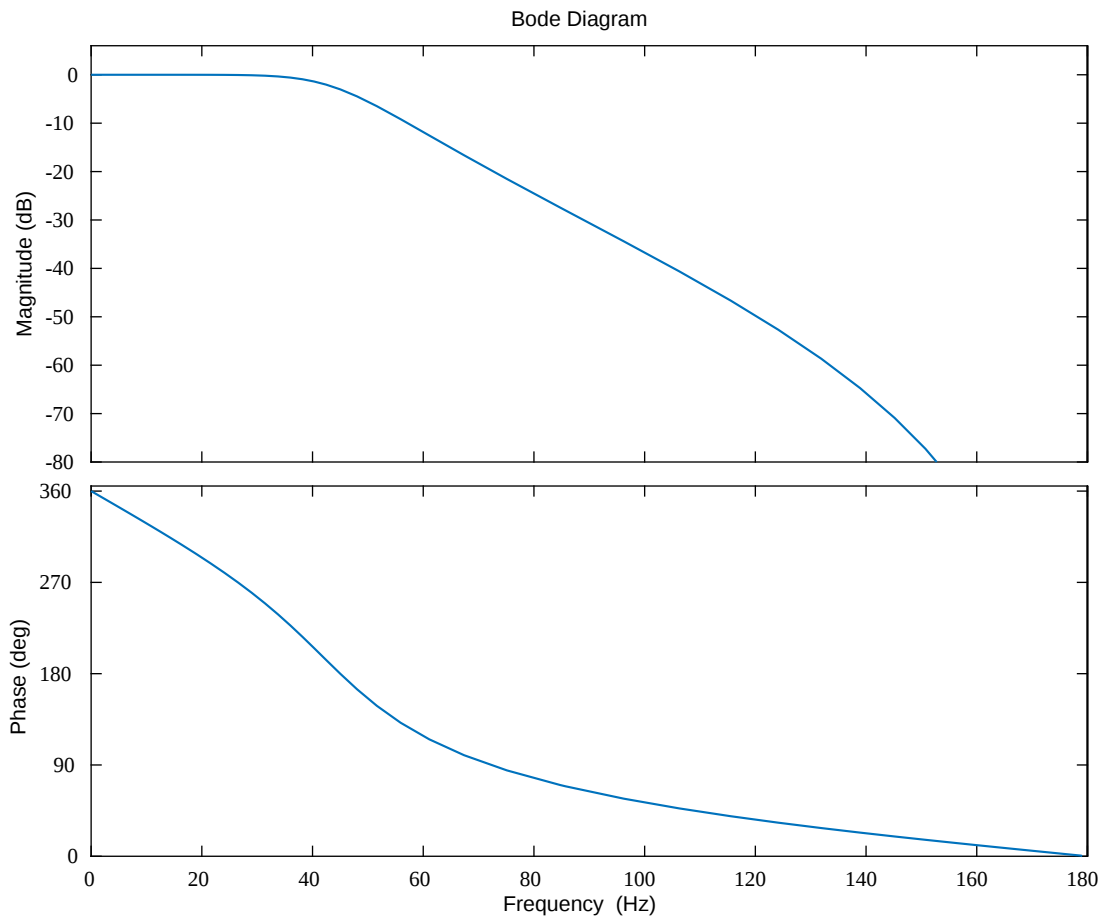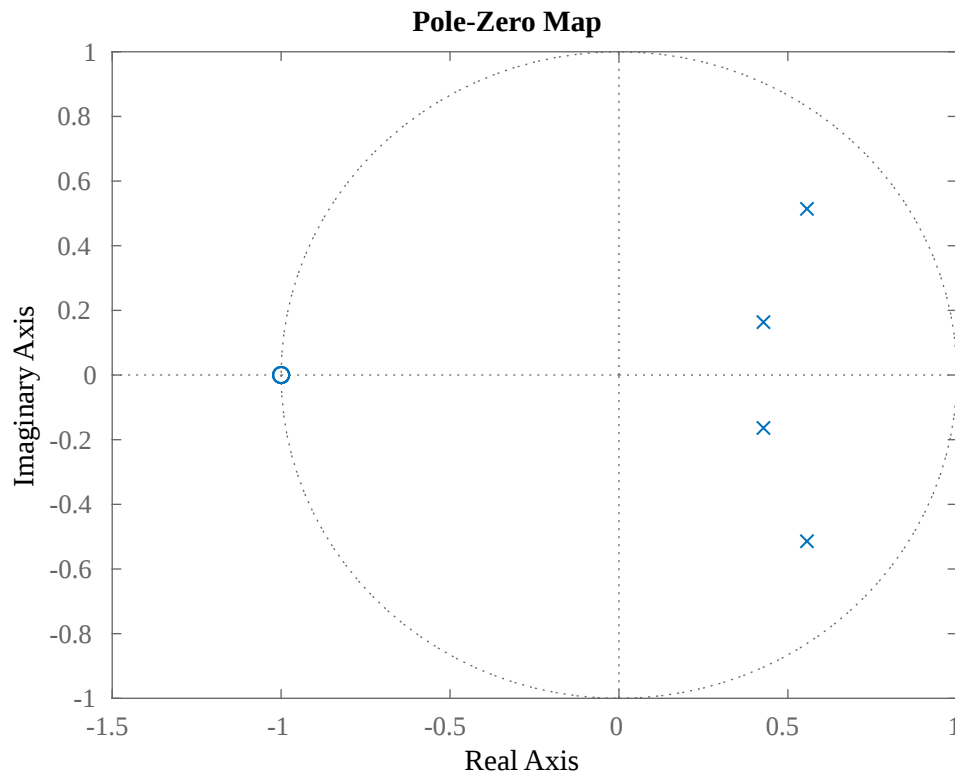
You can see that the corner frequency lies around $45\,\mathrm{Hz}$. We can check this mathematically:

$$A(\omega_{c,d}) = -3.01\mathrm{dB}$$

MATLAB & GNU Octave

If you have to design many different filters, you don't want to calculate them all by hand. Luckily, MATLAB and GNU Octave come with a command to calculate the coefficients of Butterworth filters.

```
1  f_s = 360;      % Sample frequency in Hz
2  f_c = 45;       % Cut-off frequency in Hz
3  order = 4;      % Order of the butterworth filter
4
5  omega_c = 2 * pi * f_c;        % Cut-off angular frequency
6  omega_c_d = omega_c / f_s;     % Normalized cut-off frequency (digital)
7
8  [b, a] = butter(order, omega_c_d / pi);    % Design the Butterworth filter
9  disp('a = '); disp(a);                     % Print the coefficients
10 disp('b = '); disp(b);
11 H = tf(b, a, 1 / f_s);                     % Create a transfer function
12 bode(H);                                   % Show the Bode plot
```

Note that MATLAB expects the normalized frequency as a number from 0 to 1, so we have to divide by π before passing it to the `butter` function.
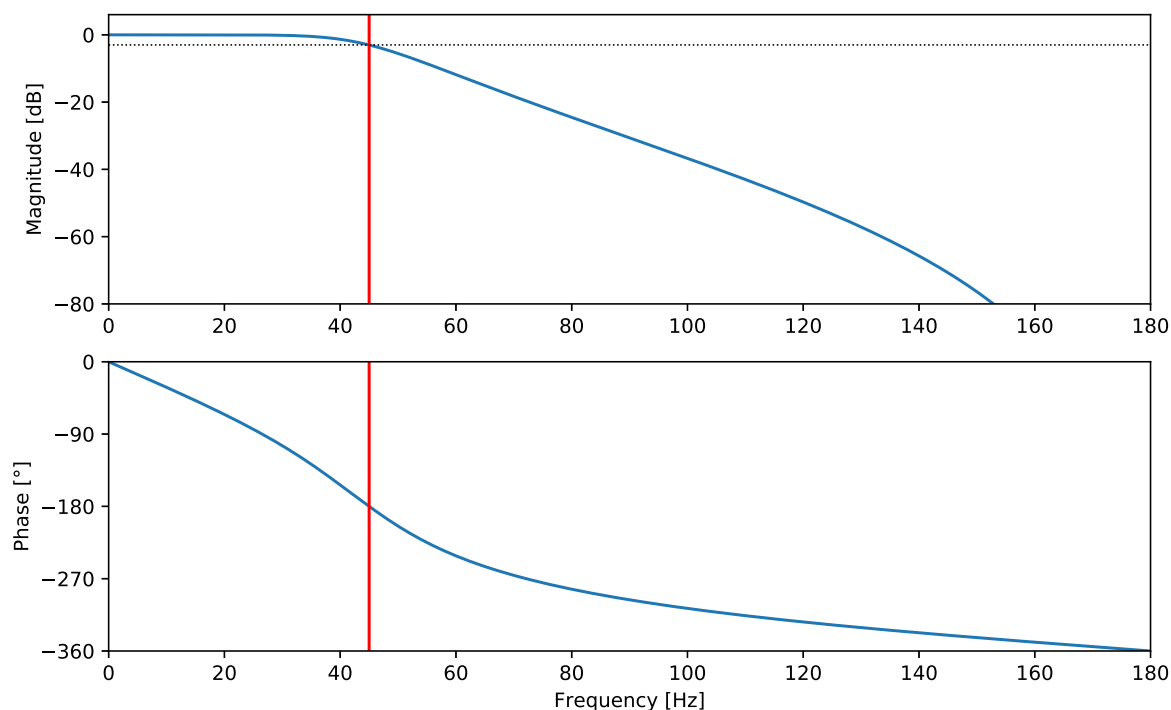
Python

A similar function is available in the SciPy signal package: `butter`.

```
1   from scipy.signal import butter, freqz, freqs
2   import matplotlib.pyplot as plt
3   from math import pi
4   import numpy as np
5
6   f_s = 360    # Sample frequency in Hz
7   f_c = 45     # Cut-off frequency in Hz
8   order = 4    # Order of the butterworth filter
9
10  omega_c = 2 * pi * f_c      # Cut-off angular frequency
11  omega_c_d = omega_c / f_s   # Normalized cut-off frequency (digital)
12
13  # Design the digital Butterworth filter
14  b, a = butter(order, omega_c_d / pi)
15  print('Coefficients')
16  print("b =", b)                        # Print the coefficients
17  print("a =", a)
18
19  w, H = freqz(b, a, 4096)               # Calculate the frequency response
20  w *= f_s / (2 * pi)                    # Convert from rad/sample to Hz
21
22  # Plot the amplitude response
23  plt.subplot(2, 1, 1)
24  plt.suptitle('Bode Plot')
25  H_dB = 20 * np.log10(abs(H))           # Convert modulus of H to dB
26  plt.plot(w, H_dB)
27  plt.ylabel('Magnitude [dB]')
28  plt.xlim(0, f_s / 2)
29  plt.ylim(-80, 6)
30  plt.axvline(f_c, color='red')
31  plt.axhline(-3, linewidth=0.8, color='black', linestyle=':')
32
33  # Plot the phase response
34  plt.subplot(2, 1, 2)
35  phi = np.angle(H)                      # Argument of H
36  phi = np.unwrap(phi)                   # Remove discontinuities
37  phi *= 180 / pi                        # and convert to degrees
38  plt.plot(w, phi)
39  plt.xlabel('Frequency [Hz]')
40  plt.ylabel('Phase [°]')
41  plt.xlim(0, f_s / 2)
42  plt.ylim(-360, 0)
43  plt.yticks([-360, -270, -180, -90, 0])
44  plt.axvline(f_c, color='red')
45
46  plt.show()
```


Bode Plot

## Comparison Between the Analog and Digital Filter

We can easily plot the Bode plots of the two filters on top of each other, in order to compare their properties.

```
1   from scipy.signal import butter, freqz, freqs
2   import matplotlib.pyplot as plt
3   from math import pi
4   import numpy as np
5
6   f_s = 360    # Sample frequency in Hz
7   f_c = 45     # Cut-off frequency in Hz
8   order = 4    # Order of the butterworth filter
9
10  omega_c = 2 * pi * f_c      # Cut-off angular frequency
11  omega_c_d = omega_c / f_s   # Normalized cut-off frequency (digital)
12
13  # Design the digital Butterworth filter
14  b_d, a_d = butter(order, omega_c_d / pi)
15  print('Digital Coefficients')
16  print("b =", b_d)                      # Print the coefficients
17  print("a =", a_d)
18
19  w, H_d = freqz(b_d, a_d, 4096)         # Calculate the frequency response
20  w *= f_s / (2 * pi)                    # Convert from rad/sample to Hz
21
22  # Design the analog Butterworth filter
23  b_a, a_a = butter(order, f_c, analog=True)
24  print('Analog Coefficients')
25  print("b =", b_a)                      # Print the coefficients
26  print("a =", a_a)
27
28  w, H_a = freqs(b_a, a_a, w)            # Calculate the frequency response
29
30  # Plot the amplitude response
31  plt.subplot(2, 1, 1)
32  plt.suptitle('Bode Plot')
33  H_d_dB = 20 * np.log10(abs(H_d))       # Convert modulus of H to dB
34  H_a_dB = 20 * np.log10(abs(H_a))
35  plt.plot(w, H_d_dB, color='blue', label='Digital')
36  plt.plot(w, H_a_dB, color='green', label='Analog')
37  plt.legend()
38  plt.ylabel('Magnitude [dB]')
39  plt.xlim(0, f_s / 2)
40  plt.ylim(-80, 6)
41  plt.axvline(f_c, color='red')
42  plt.axhline(-3, linewidth=0.8, color='black', linestyle=':')
43
44  # Plot the phase response
45  plt.subplot(2, 1, 2)
46  phi_d = np.angle(H_d)                  # Argument of H
47  phi_a = np.angle(H_a)
48  phi_d = np.unwrap(phi_d) * 180 / pi    # Remove discontinuities
49  phi_a = np.unwrap(phi_a) * 180 / pi    # and convert to degrees
50  plt.plot(w, phi_d, color='blue')
51  plt.plot(w, phi_a, color='green')
52  plt.xlabel('Frequency [Hz]')
53  plt.ylabel('Phase [°]')
54  plt.xlim(0, f_s / 2)
55  plt.ylim(-360, 0)
56  plt.yticks([-360, -270, -180, -90, 0])
57  plt.axvline(f_c, color='red')
58
59  plt.show()
```

Bode Plot
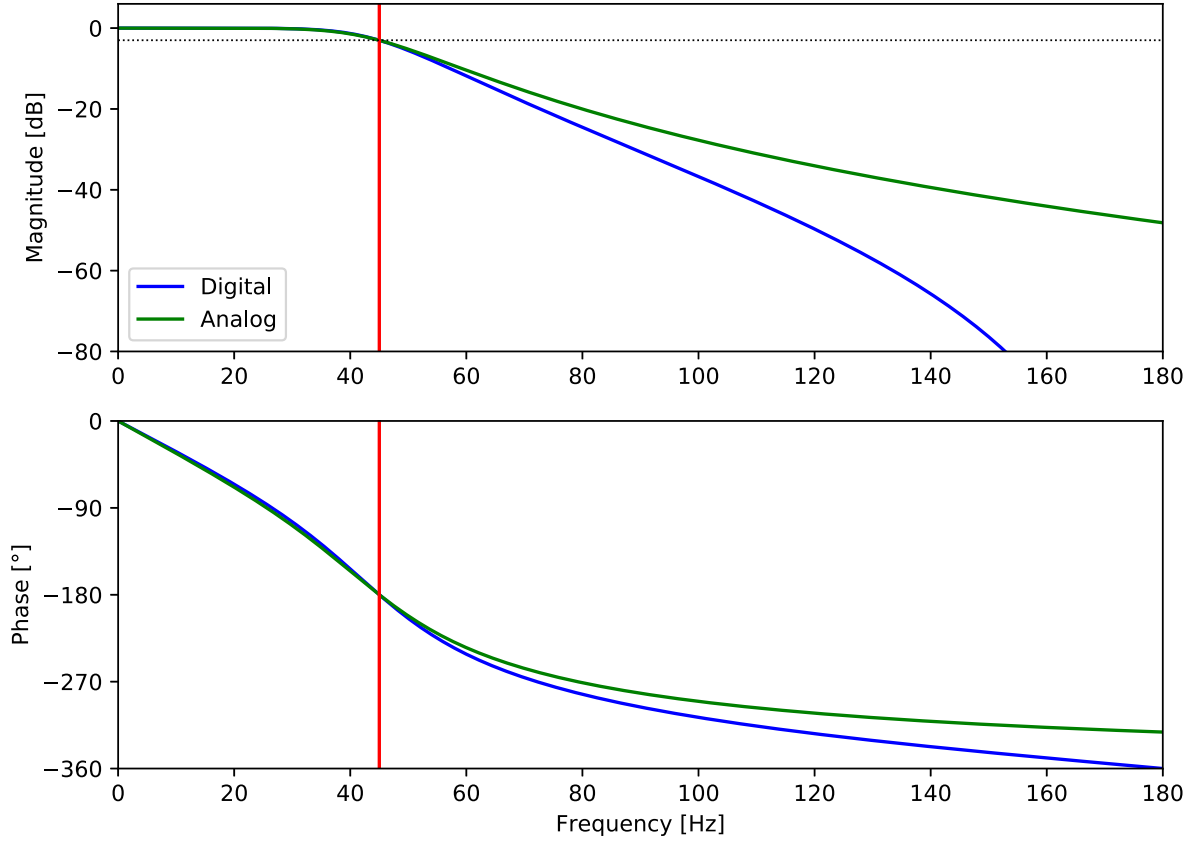
Discretization using Second Order Sections (SOS)

For higher order filters, small quantization errors on the transfer function coefficients can result in large errors on the pole and zero locations.
A solution is to factor the transfer function into second order factors or sections.

Recall from the previous sections, and define α_k, $H_{2,k}$ and H_1 as follows:

$$\gamma \triangleq \cot\left(\pi\frac{f_c}{f_s}\right) \tag{7}$$

$$s' \triangleq \frac{s}{\omega_{c,a}} = \gamma\frac{z-1}{z+1} \tag{8}$$

$$\alpha_k \triangleq 2\cos\left(2\pi\frac{2k+n+1}{4n}\right) \tag{9}$$

$$H_{2,k}\left(s'\right) \triangleq \frac{1}{B_{2,k}\left(s'\right)} \triangleq \frac{1}{s'^2 - \alpha_k s' + 1} \tag{10}$$

$$H_1\left(s'\right) \triangleq \frac{1}{s' + 1} \tag{11}$$

$$H_n\left(s'\right) = \begin{cases} \prod_{k=0}^{\frac{n}{2}-1} H_{2,k}\left(s'\right) & \text{even } n \\ H_1\left(s'\right)\prod_{k=0}^{\frac{n-1}{2}-1} H_{2,k}\left(s'\right) & \text{odd } n \end{cases} \tag{12}$$

Second Order Sections

We'll use the same technique as before to substitute s' into $B_{2,k}\left(s'\right)$ using the pre-warped bilinear transform relation 8 to get the discrete-time Butterworth polynomial $B_{2,k}\left(z\right)$:

$$B_{2,k}\left(s'\right) \triangleq s'^2 - \alpha_k s' + 1$$

$$B_{2,k}\left(z\right) = \gamma^2 \left(\frac{z-1}{z+1}\right)^2 - \alpha_k \gamma \frac{z-1}{z+1} + 1$$

$$= \gamma^2 \frac{(z-1)^2}{(z+1)^2} - \alpha_k \gamma \frac{(z-1)(z+1)}{(z+1)^2} + \frac{(z+1)^2}{(z+1)^2}$$

$$= \frac{\gamma^2(z-1)^2 - \alpha_k \gamma (z-1)(z+1) + (z+1)^2}{(z+1)^2}$$

$$= \frac{\left(\gamma^2 - \alpha_k \gamma + 1\right) z^2 - \left(2\gamma^2 - 2\right) z + \left(\gamma^2 + \alpha_k \gamma + 1\right)}{(z+1)^2}$$

$$H_{2,k}\left(z\right) = \frac{z^2 + 2z + 1}{\left(\gamma^2 - \alpha_k \gamma + 1\right) z^2 - \left(2\gamma^2 - 2\right) z + \left(\gamma^2 + \alpha_k \gamma + 1\right)}$$

$$= \frac{1 + 2z^{-1} + z^{-2}}{\left(\gamma^2 - \alpha_k \gamma + 1\right) - \left(2\gamma^2 - 2\right) z^{-1} + \left(\gamma^2 + \alpha_k \gamma + 1\right) z^{-2}}$$

The coefficients of the k-th factor of the discrete-time transfer function are thus:

$$\begin{aligned}
b_{k,0} &= 1 \\
b_{k,1} &= 2 \\
b_{k,2} &= 1 \\[1em]
a_{k,0} &= \gamma^2 - \alpha_k \gamma + 1 \\
a_{k,1} &= 2\left(1 - \gamma^2\right) \\
a_{k,2} &= \gamma^2 + \alpha_k \gamma + 1
\end{aligned}$$

First Order Section

For odd orders n, we need $\frac{n-1}{2}$ second order sections and a single first order section.

Again, we'll use the the pre-warped bilinear transform relation 8:

$$H_1\left(s'\right) \triangleq \frac{1}{s' + 1}$$

$$= \frac{1}{\gamma \frac{z-1}{z+1} + 1}$$

$$= \frac{z+1}{\gamma(z-1) + (z+1)}$$

$$= \frac{z+1}{(\gamma+1)z + (1-\gamma)}$$

$$= \frac{1 + z^{-1}}{(\gamma+1) + (1-\gamma)z^{-1}}$$

This gives us the coefficients of the first order factor of the discrete-time transfer function:

$$\begin{aligned}
b_0 &= 1 \\
b_1 &= 1 \\[1em]
a_0 &= \gamma + 1 \\
a_1 &= 1 - \gamma
\end{aligned}$$