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 $360\ \mathrm{Hz}$ and a cut-off frequency of $45\ \mathrm{Hz}$.

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:

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

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 $\omega_{c,a}$:

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

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:

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

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

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:

$$\begin{aligned} \gamma \triangleq \frac{2f_s}{\omega_{a,c}} &= \frac{2f_s}{2f_s \tan\left(\frac{\omega_{c,d}}{2}\right)} = \cot\left(\pi\frac{f_c}{f_s}\right) \\ &= \cot\left(\frac{\pi}{8}\right) = 1 + \sqrt{2} \end{aligned} \tag{5}$$

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

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.

$$
\begin{aligned}
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} \\
&\quad + \gamma^3 \frac{(z-1)^3(z+1)}{(z+1)^4}(\alpha+\beta) \\
&\quad + \gamma^2 \frac{(z-1)^2(z+1)^2}{(z+1)^4}(\alpha\beta+2) \\
&\quad + \gamma \frac{(z-1)(z+1)^3}{(z+1)^4}(\alpha+\beta) \\
&\quad + \frac{(z+1)^4}{(z+1)^4} \\
&= \frac{1}{(z+1)^4} \left[\begin{array}{cc} & \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{array} \right] \\
&= \frac{1}{(z+1)^4} \left[\begin{array}{ccccccccccccccc} & \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{array} \right] \\
&= \frac{1}{(z+1)^4} \left[\begin{array}{ccc} & \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{array} \right] \\
H_4(z) &= \frac{1}{B_4(z)} \\
&= \frac{(z+1)^4}{\left[\begin{array}{ccc} & (\gamma^4 + \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) + \gamma(\alpha+\beta) + 1) & z^4 \\ + & (-4\gamma^4 - 2\gamma^3(\alpha+\beta) + 2\gamma(\alpha+\beta) + 4) & z^3 \\ + & (6\gamma^4 - 2\gamma^2(\alpha\beta+2) + 6) & z^2 \\ + & (-4\gamma^4 + 2\gamma^3(\alpha+\beta) - 2\gamma(\alpha+\beta) + 4) & z \\ + & (\gamma^4 - \gamma^3(\alpha+\beta) + \gamma^2(\alpha\beta+2) - \gamma(\alpha+\beta) + 1) & \end{array} \right]} \\
&\triangleq \frac{b_4z^4 + b_3z^3 + b_2z^2 + b_1z + b_0}{a_4z^4 + a_3z^3 + a_2z^2 + a_1z + a_0}
\end{aligned}
$$

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

$$
\begin{aligned}
b_0 &= 1 \\
b_1 &= 4 \\
b_2 &= 6 \\
b_3 &= 4 \\
b_4 &= 1
\end{aligned}
$$

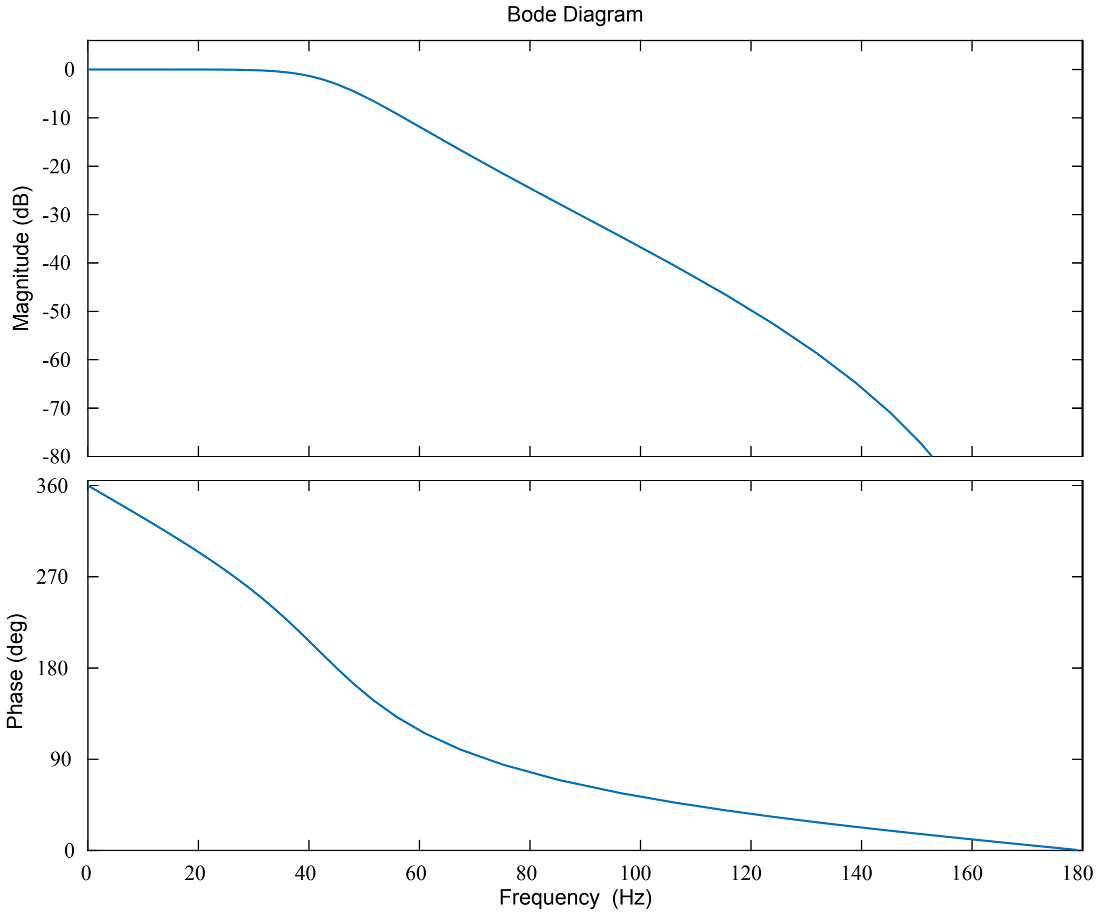
Frequency response & Pole-Zero Map

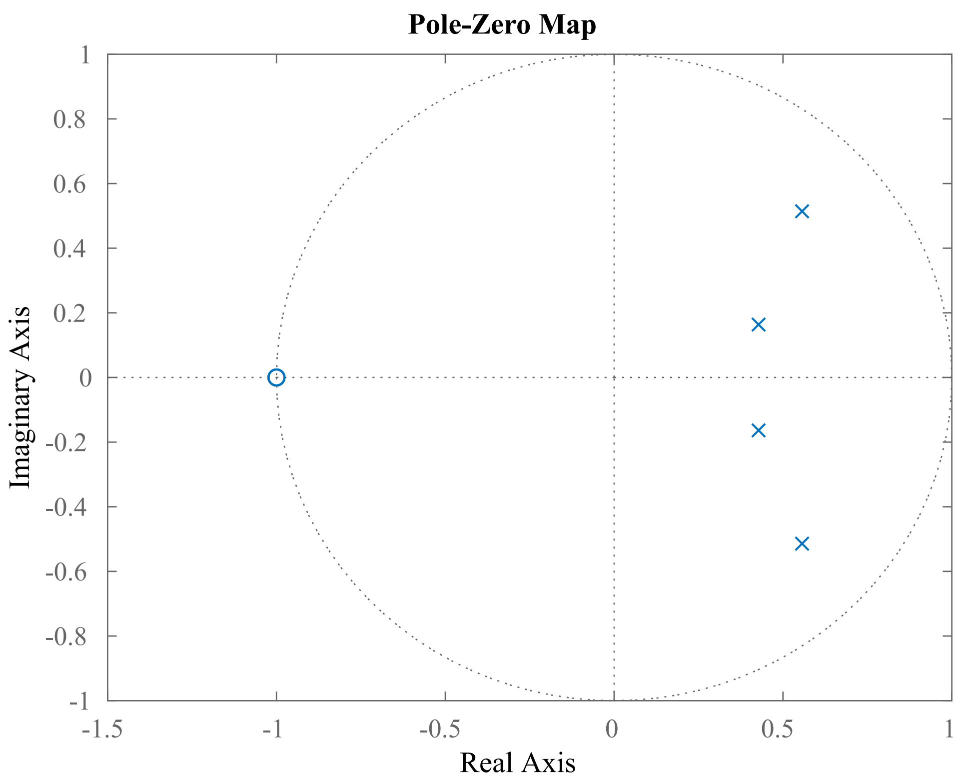
We can check the filter's frequency response to make sure that we didn't make any mistakes. As mentioned in other pages, the frequency response of a digital system can be a obtained by evaluating the transfer function $H(z)$ along the unit circle ($z = e^{j\omega}$). We'll plot the magnitude in decibels.

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

We can also plot the phase angle of the response:

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





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

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.

```
f_s = 360;    % Sample frequency in Hz
f_c = 45;     % Cut-off frequency in Hz
order = 4;    % Order of the butterworth filter

omega_c = 2 * pi * f_c;       % Cut-off angular frequency
omega_c_d = omega_c / f_s;    % Normalized cut-off frequency (digital)

[b, a] = butter(order, omega_c_d / pi);    % Design the Butterworth filter
disp("a = "); disp(a);                     % Print the coefficients
disp("b = "); disp(b);
H = tf(b, a, 1 / f_s);                     % Create a transfer function
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.

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

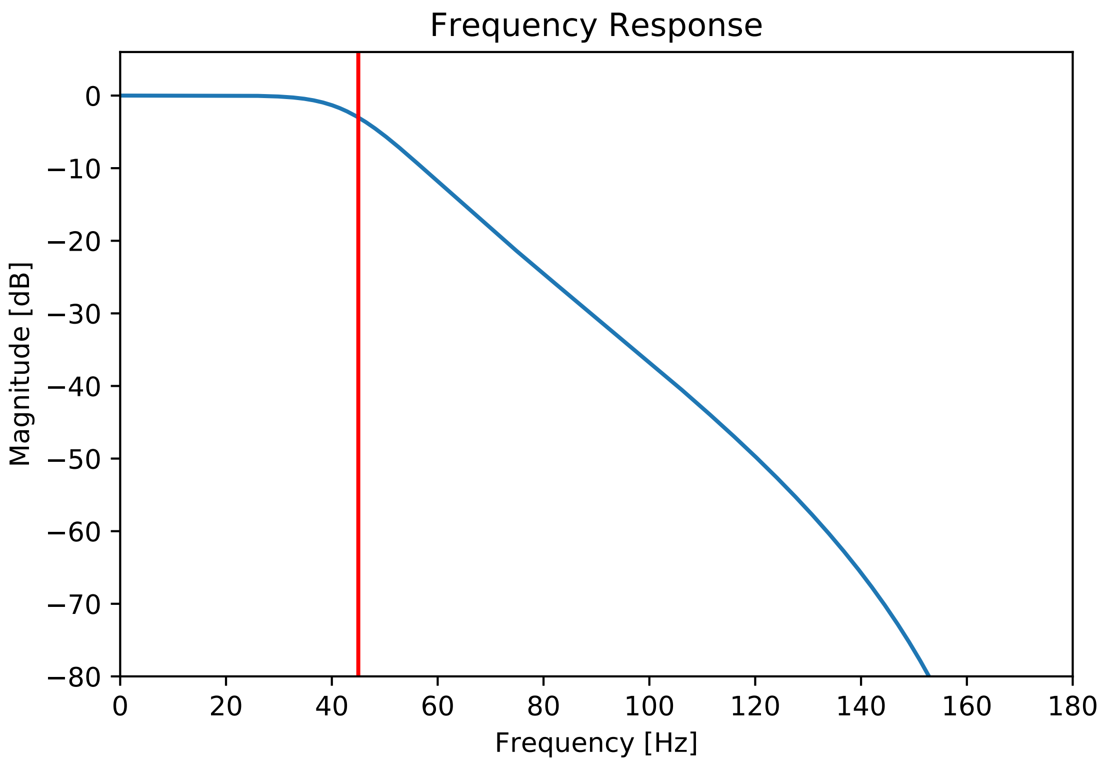
```
from scipy.signal import butter, freqz
import matplotlib.pyplot as plt
from math import pi
import numpy as np

f_s = 360    # Sample frequency in Hz
f_c = 45     # Cut-off frequency in Hz
order = 4    # Order of the butterworth filter

omega_c = 2 * pi * f_c       # Cut-off angular frequency
omega_c_d = omega_c / f_s    # Normalized cut-off frequency (digital)

b, a = butter(order, omega_c_d / pi)    # Design the Butterworth filter
print("a =", a)                         # Print the coefficients
print("b =", b)

w, h = freqz(b, a)                   # Calculate the frequency response
w *= f_s / (2 * pi)                  # Convert from rad/sample to Hz
plt.plot(w, 20 * np.log10(abs(h)))   # Convert to dB
plt.title('Frequency Response')      # Plot the result
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude [dB]')
plt.xlim(0, f_s / 2)
plt.ylim(-80, 6)
plt.axvline(f_c, color='red')
plt.show()
```


Frequency Response
Magnitude [dB]
0
−10
−20
−30
−40
−50
−60
−70
−80
0
20
40
60
80
100
120
140
160
180
Frequency [Hz]