EE 451

IIR Filter Design

One way to design IIR filters is by use of the bilinear transformation. To design a low-pass IIR filter, transform a low-pass continuous-time filter to a discrete-time filter using the transformation

$$s = \frac{1 - z^{-1}}{1 + z^{-1}}$$

To design another type of filter (high-pass, band-pass, or band-stop) another step is needed. There are several ways to do this. One of the simplest is to find different transformations which will map a low-pass continuous-time filter in one of these other types of discrete-time filters. Here are a set of transformations which will do that.

	LP	HP	BP	BS
s	$\frac{1-z^{-1}}{1+z^{-1}}$	$\frac{1+z^{-1}}{1-z^{-1}}$	$\frac{1-2cz^{-1}+z^{-2}}{1-z^{-2}}$	$\frac{1-z^{-2}}{1-2cz^{-1}+z^{-2}}$
z	$\frac{1+s}{1-s}$	$\frac{s+1}{s-1}$	$\frac{-c\pm\sqrt{c^2+s^2-1}}{s-1}$	$\frac{c\pm\sqrt{c^2+\left(\frac{1}{s}\right)^2-1}}{1-\frac{1}{s}}$
Ω	$\tan(\frac{\omega}{2})$	$-\cot(\frac{\omega}{2})$	$\frac{c-\cos(\omega)}{\sin(\omega)}$	$\frac{\sin(\omega)}{c-\cos(\omega)}$
c			$\frac{\sin(\omega_{pl}+\omega_{ph})}{\sin(\omega_{pl})+\sin(\omega_{ph})}$	$\frac{\sin(\omega_{pl}+\omega_{ph})}{\sin(\omega_{pl})+\sin(\omega_{ph})}$
ω_1	0	π	$\cos^{-1}(c)$	0 or π

A filter is usually specified by passband and stopband frequencies ω_{pass} and ω_{stop}, and by passband ripple R_{pass}, and stopband attenuation R_{stop}. Band-pass and band-stop filters have low and high passband and stopband frequencies.

To design a discrete-time IIR filter, do the following:

1. Pre-warp the discrete-time frequencies to continuous-time frequencies using the row labeled Ω:
   (a) If low-pass or high-pass, transform ω_{pass} and ω_{stop} to Ω_{pass} and Ω_{stop}.
   (b) If band-pass or band-stop, find c. Then transform ω_{pl}, ω_{ph}, ω_{sl} and ω_{sh} to Ω_{pl}, Ω_{ph}, Ω_{sl} and Ω_{sh}. Let $\Omega_{pass} = |\Omega_{pl}|$, and let $\Omega_{stop} = \min(|\Omega_{sl}|, |\Omega_{sh}|)$

2. Design a continuous time low-pass filter using Ω_{pass} and Ω_{stop} from step 1. For Butterworth design, find N and Ω_c using the formulas

$$\epsilon = \sqrt{\frac{2\delta_1 - {\delta_1}^2}{(1-\delta_1)^2}}$$

$$\delta = \sqrt{\frac{1-{\delta_2}^2}{\delta_2^2}}$$

$$N = \frac{\log(\epsilon/\delta)}{\log(\Omega_{pass}/\Omega_{stop})}$$

$$\Omega_c = \Omega_{stop}\,(\delta)^{-1/N}$$

   or

$$\Omega_c = \Omega_{pass}\,(\epsilon)^{-1/N}$$

   where $\delta_1 = 1 - 10^{-R_{pass}/20}$, and $\delta_2 = 10^{-R_{stop}/20}$. Since N must be an integer, you must round N up to an integer value.

   This filter will have N poles at

$$s_k = \Omega_c e^{j\left(\frac{\pi}{2}\right)} e^{j\pi\left(\frac{2k+1}{2N}\right)}, \quad k = 0, 1, ..., N-1$$

   and N zeros at infinity.

3. Transform the continuous-time filter to a discrete-time filter using one of the following two methods:

   (a) Replace every s in the continuous-time transfer function as specified with the row labeled s. Do lots of algebra to get it into a usable form.

   (b) Map s-plane poles and zeros to corresponding z-plane poles and zeros using the equations in the row labeled z. (For band-pass and band-stop filters, every s-plane pole maps to two z-plane poles, and every s-pland zero maps to two z-plane zeros.) Don't forget s-plane zeros at infinity. For low-pass filters, an s-plane zero at infinity maps into a z-plane zero at -1. For high-pass filters, an s-plane zero at infinity maps into a z-plane zero at +1. For band-pass filters, an s-plane zero at infinity maps into an z-plane zero at +1 and an z-plane zero at -1. For band-stop filters, and s-plane zero at infinity maps into z-plane zeros at $e^{\pm j\omega_c}$ where $\omega_c = \cos^{-1}(c)$.

   Then find G by

$$G = \left|\frac{\prod_{i=1}^{N}(e^{j\omega_1} - p_i)}{\prod_{i=1}^{N}(e^{j\omega_1} - z_i)}\right|$$

   where ω_1 is the frequency where $|H(\omega_1)| = 1$.

   The discrete-time transfer function is:

$$H(z) = G\frac{\prod_{i=1}^{N}(z - z_i)}{\prod_{i=1}^{N}(z - p_i)}$$

Using MATLAB, steps 1 and 2 can be done with the function `buttord`. The entire design process can be done with the function `butter`.