Lecture 13

IIR filters 2

Introduction

In the previous lecture we learned how to design analog lowpass, highpass, bandpass and bandstop filters based on a Butterworth lowpass prototype. In this lecture we learn how to convert an analog filter with transfer function $H_a(s)$ into an IIR discrete filter with transfer function $H(z)$. Our approach will be to find a transformation $s=g(z)$ that maps the unit circle of the z plane, $z=e^{j\omega}$, onto the imaginary axis of the s plane, $s=j\Omega$ (Fig. 1). Then

$$H(z)=H_a(s)=H_a(g(z)) \tag{1}$$

will generate the desired discrete filter from the analog filter.

Bilinear transformation

We seek a transformation

$$s=g(z) \tag{2}$$

with the property that for any $z=e^{j\omega}$ on the unit circle of the z plane, we will obtain $s=j\Omega$ on the imaginary axis of the s plane. That is

$$j\Omega=g(e^{j\omega}) \tag{3}$$

We want $\omega=0$ to map to $\Omega=0$. Since $z=e^{j0}=1$, this suggests that $g(z)$ should have a zero at $z=1$. It follows that $g(z)$ should have a factor $z-1$. We also want $\omega\rightarrow\pi/2$ to map to $\Omega\rightarrow\infty$ and $\omega\rightarrow-\pi/2$ to map to $\Omega\rightarrow-\infty$. Since $e^{\pm j\pi/2}=-1$, this suggests that $g(z)$ should

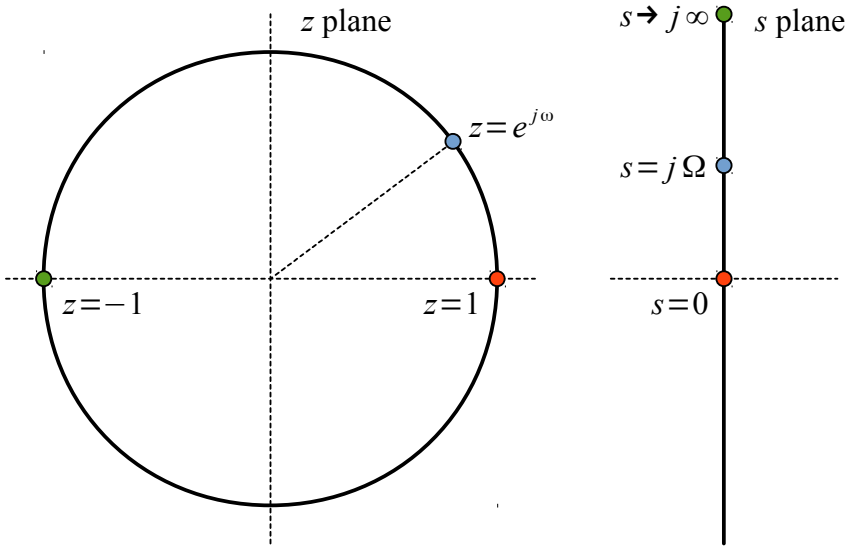

Fig. 1: Mapping between z and s planes.

have a pole at $z=-1$. It follows that $g(z)$ should have a factor $1/(z+1)$. With these two factors we have

$$s=g(z)=\frac{z-1}{z+1} \tag{4}$$

Let's see if this satisfies (3). We find

$$\frac{e^{j\omega}-1}{e^{j\omega}+1}=\frac{e^{j\omega/2}-e^{-j\omega/2}}{e^{j\omega/2}+e^{-j\omega/2}}=j\frac{\sin(\omega/2)}{\cos(\omega/2)}=j\tan(\omega/2)=j\Omega \tag{5}$$

So (3) is satisfied. Moreover, with $\Omega=\tan(\omega/2)$ the interval $-\pi/2\le\omega\le\pi/2$ maps onto $-\infty\le\Omega\le\infty$ with $\omega=0$ mapping to $\Omega=0$. Therefore, all our requirements are satisfied. However, $\tan(\omega/2)$ is dimensionless while Ω needs to have units of s^{-1}. We can rectify this by adding a constant factor with these dimensions. Doing this, and multiplying (4) by z^{-1}/z^{-1} we arrive at

$$s=g(z)=\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \tag{6}$$

Here T is a constant with units of time and the factor of 2 is for convenience in what follows. By expressing $g(z)$ in terms of z^{-1} (delay operator) rather than z (advance operator) we ensure that the resulting $H(z)$ will represent a causal system. The resulting frequency mappings are

$$\begin{aligned}\Omega&=\frac{2}{T}\tan(\omega/2)\\ F&=\frac{1}{\pi T}\tan(\pi f)\end{aligned} \tag{7}$$

The relation between f and F is shown in Fig. 2.

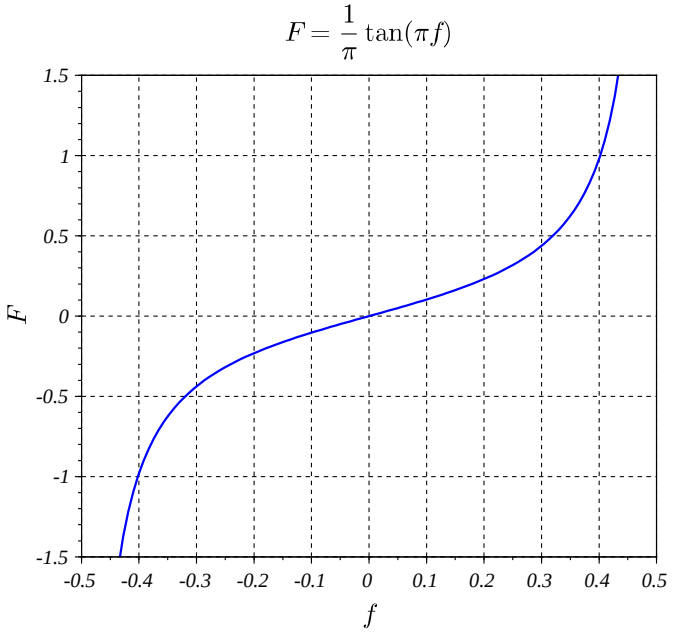

Fig. 2: Analog frequency F vs. discrete frequency f with $T=1\,\mathrm{sec}$.

For small frequencies the approximation $\tan x \approx x$ results in $F \approx f/T$. If we take $T = T_s$ to be the sampling period, this is just the relation $F = f F_s$ between discrete-time and continuous-time frequencies.

Lowpass filter

To design an IIR lowpass filter we first choose a prototype $H_p(s) = 1/B_n(s)$. We then scale the frequency variable to achieve the desired cutoff frequency

$$\tilde{s} = \frac{s}{\Omega_c} \tag{8}$$

What value do we use for Ω_c? Let's leave that question for later. Combining (8) and (6) we get

$$\tilde{s} = \frac{2}{\Omega_c T} \frac{1 - z^{-1}}{1 + z^{-1}} \tag{9}$$

At the cutoff frequency of the prototype filter, $\tilde{s} = j$. Setting this to correspond to the digital cutoff frequency f_c we have

$$j = \frac{2}{\Omega_c T} j \tan(\pi f_c) \tag{10}$$

It follows that

$$\frac{2}{\Omega_c T} = \frac{1}{\tan(\pi f_c)} \tag{11}$$

and the transformation that converts a prototype filter into an IIR discrete lowpass filter is

$$\boxed{H(z) = H_p\left(\frac{1}{\tan(\pi f_c)} \frac{1 - z^{-1}}{1 + z^{-1}}\right)} \tag{12}$$

We didn't have to specify Ω_c directly. Instead the quantity $2/(\Omega_c T)$ is fixed by (11). And example IIR frequency response is shown in Fig. 3.

Example 1: *Design an IIR lowpass filter with cutoff frequency* $f_c = 0.1$ *based on a* 2^{nd} *order Butterworth prototype.*

The prototype analog filter is $H_p(s) = \dfrac{1}{s^2 + 2as + 1}$ with

$$a = -\cos\left(3\frac{\pi}{4}\right) \approx 0.6498$$

Therefore

$$H(z) = \frac{1}{\left(\dfrac{1}{\tan(0.1\pi)} \dfrac{1 - z^{-1}}{1 + z^{-1}}\right)^2 + a \dfrac{1}{\tan(0.1\pi)} \dfrac{1 - z^{-1}}{1 + z^{-1}} + 1}$$

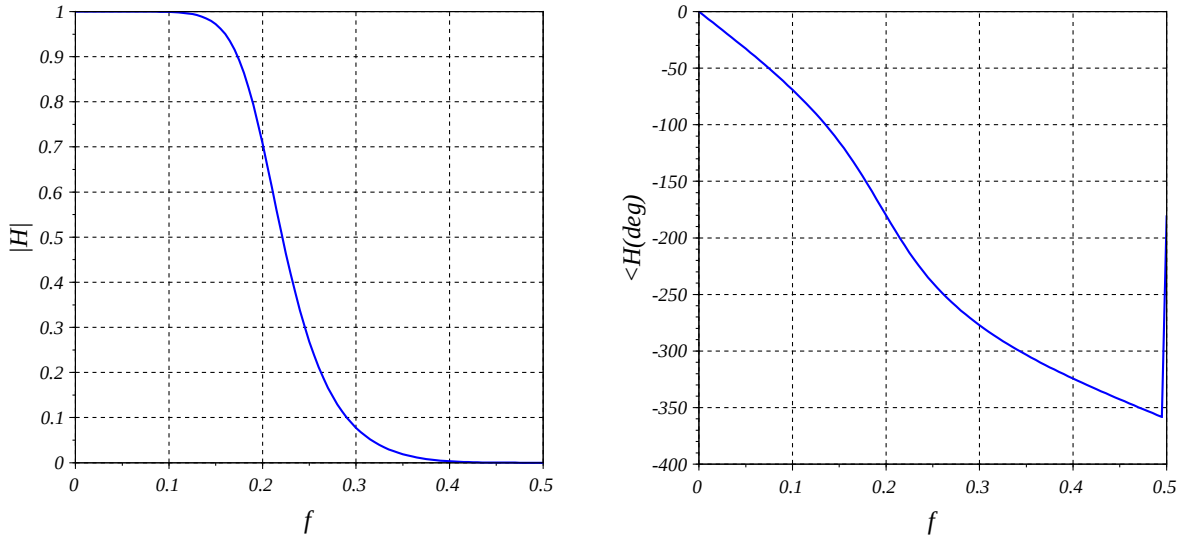

Fig. 3: 4th order IIR lowpass filter with $f_c = 0.2$.

We could do the algebra by hand to put $H(z)$ into standard form, but this can be conveniently done with Scilab. Scilab code to calculate the *a* and *b* coefficients of an IIR lowpass filter is given in the Appendix.

Highpass filter

The highpass transformation

$$\tilde{s} = \frac{\Omega_c}{s} \tag{13}$$

is the inverse of the lowpass transformation (8). This suggests we simply invert (9).

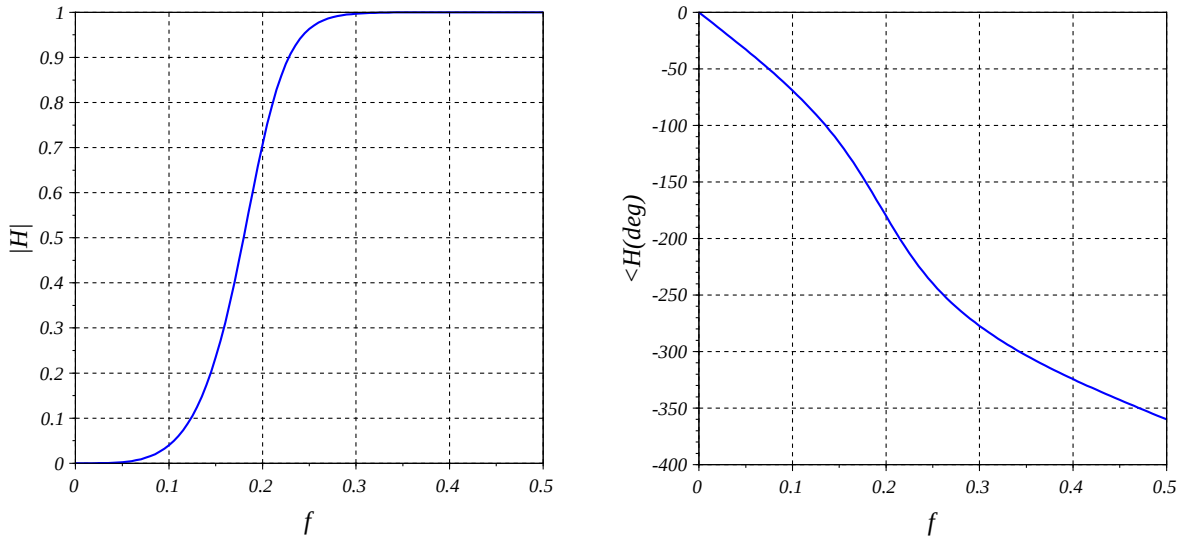

Fig. 4: 4th order IIR highpass filter with $f_c = 0.2$.

The transformation that converts a prototype filter into an IIR discrete highpass filter is then

$$\boxed{H(z)=H_p\left(\tan(\pi f_c)\frac{1+z^{-1}}{1-z^{-1}}\right)} \tag{14}$$

An example IIR highpass response is shown in Fig. 4.

Linkwitz–Riley filter

FIR filters had the property that if $h^{LP}(n)$ and $h^{HP}(n)$ are FIR lowpass and highpass impulse responses, with the same cutoff frequency and number of coefficients, then

$$h^{LP}(n)+h^{HP}(n)=\delta(n) \tag{15}$$

This is the *perfect reconstruction* property. It is very desirable in applications such as crossover filters (Fig. 5) because it ensures that the treble and bass signals coming out of the tweeter and woofer speakers will combine to precisely reproduce the original audio signal (assuming no other EQ effects have been applied).

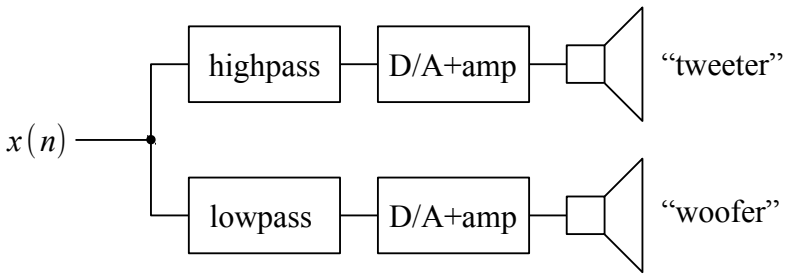

Fig. 5: Crossover filters.

IIR filters *do not* have this property. We can see this by adding the transfer functions of the lowpass and highpass filters shown in Fig. 3 and Fig. 4 and plotting the resulting frequency response. The result is shown in Fig. 6. There is a 3-dB "bump" in the magnitude response at the crossover frequency. This is not acceptable as it would create a "booming" response at this frequency akin to that created by a resonant room structure or other undesirable acoustic effect. In 1976 the Linkwitz-Riley filter was presented [Error: Reference source not found]. This is also known as a *Butterworth-squared filter*. To generate a 4^{th} order Linkwitz-Riley lowpass filter we first design a 2^{nd} order Butterworth lowpass filter with transfer function $H_B^{LP}(z)$. Then the Linkwitz-Riley filter transfer function is simply a cascade of two of the Butterworth filters

$$H_{LR}^{LP}(z)=H_B^{LP}(z)H_B^{LP}(z) \tag{16}$$

Since each Butterworth filters is 2^{nd} order, the Linkwitz-Riley filter is 4^{th} order. The same process is used for the highpass filter

$$H_{LR}^{HP}(z)=H_B^{HP}(z)H_B^{HP}(z) \tag{17}$$

The frequency response of the sum of 4^{th} order Linkwitz-Riley lowpass and highpass filters is shown in Fig. 7. The magnitude response is effectively flat at 0 dB. This is a great improvement over the Butterworth crossover filters. However, it's still not perfect reconstruction as the phase response is not linear.

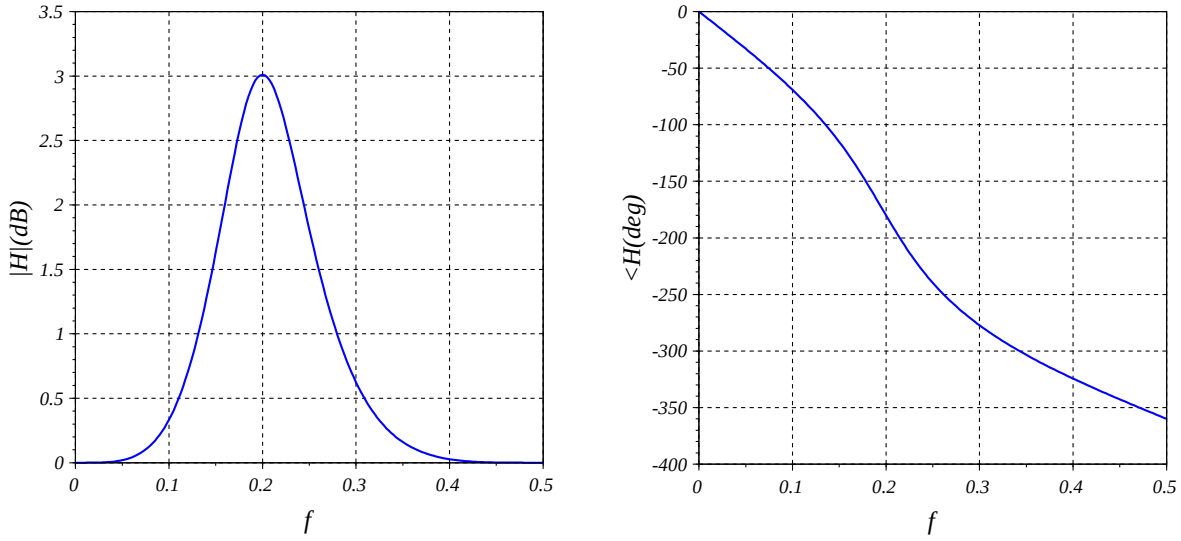

Fig. 6: Frequency response of sum of IIR lowpass (Fig. 3) and highpass (Fig. 4) filter transfer functions (dB scale).

The deviation from perfect reconstruction is best seen by calculating the impulse response of the filter shown in Fig. 7. This is shown in Fig. 8. The response is close to being a delayed impulse, but is not precisely a delta function.

References

1. http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt
2. https://en.wikipedia.org/wiki/Linkwitz%E2%80%93Riley_filter

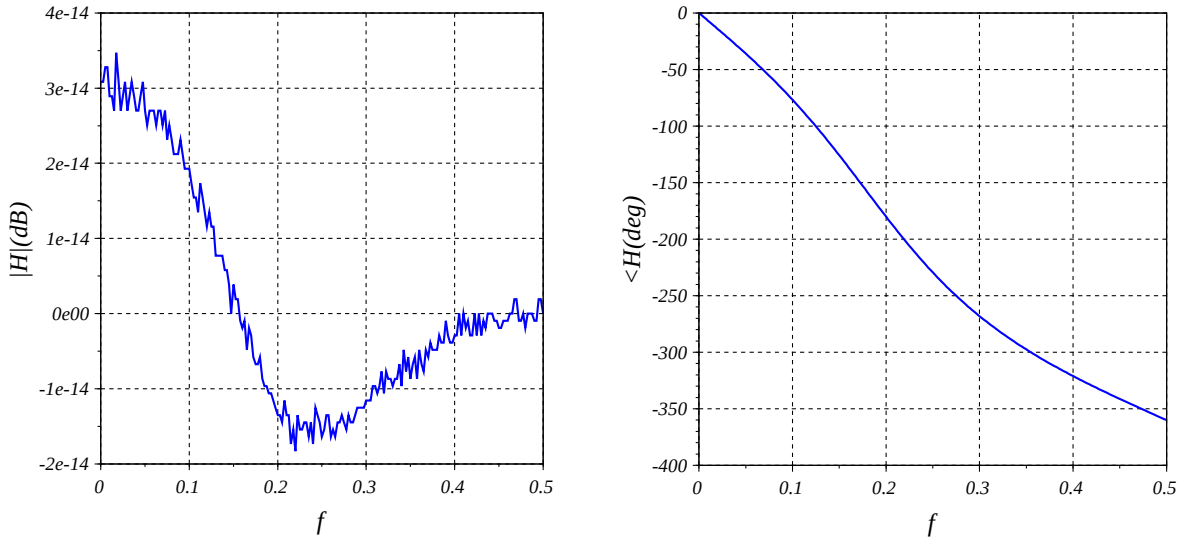

Fig. 7: Frequency response of sum of 4th order Linkwitz-Riley lowpass and highpass transfer functions (dB scale). Note that the total variation of the magnitude response is less than $6 \cdot 10^{-14}$dB *which is at the level of round-off error.*

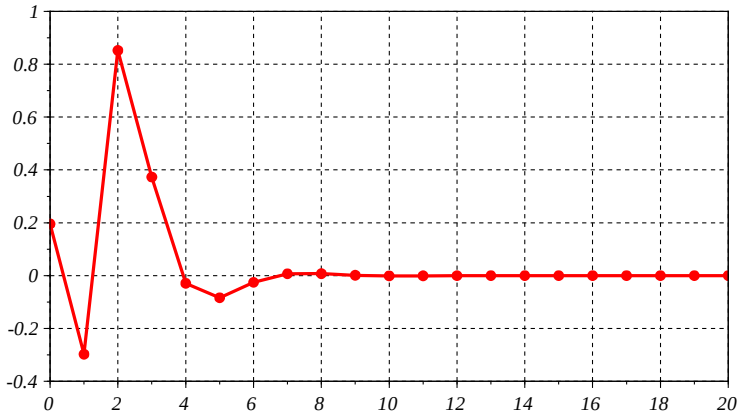

Fig. 8: Impulse response of filter shown in Fig. 7.

Appendix – Scilab code

```
function [b,a] = LPiir(fc,n) //lowpass IIR filter coefficients
  w = poly(0,'w');           //based on n-th order Butterworth
  u = ((1-w)/(1+w))/tan(%pi*fc);
  B = Butterworth(n);
  H = 1/horner(B,u);
  a = coeff(H(3));
  b = coeff(H(2))/a(1);
  a = a/a(1);
endfunction

function [b,a] = HPiir(fc,n) //highpass IIR filter coefficients
  w = poly(0,'w');           //based on n-th order Butterworth
  u = ((1+w)/(1-w))*tan(%pi*fc);
  B = Butterworth(n);
  H = 1/horner(B,u);
  a = coeff(H(3));
  b = coeff(H(2))/a(1);
  a = a/a(1);
endfunction
```