Lecture 12

IIR filters 1

Introduction

The transfer function of an FIR filter has an "all-zeros" form

$$\begin{aligned} H(z) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M} \\ &= b_0 (1 - \zeta_1 z^{-1})(1 - \zeta_2 z^{-1}) \cdots (1 - \zeta_M z^{-1}) \end{aligned} \tag{1}$$

which is a polynomial in z^{-1}. One drawback of FIR filters is that they can require a large number of coefficients to achieve reasonable filter performance. The result is a large computational burden.

The transfer function of an IIR filter has both zeros and poles

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} \tag{2}$$

and is a rational function of z^{-1}. Due to the poles, an IIR filter can often achieve good filter performance with far fewer coefficients (hence with fewer calculations) than an FIR filter requires.

Our approach to IIR filter design will be to first design an analog filter $H_a(s)$ and then apply a continuous-to-discrete frequency transformation to obtain a discrete filter

$$H(z) = H_a(s(z)) \tag{3}$$

Rational transfer functions

We will limit consideration to transfer functions that are rational functions of z^{-1}. Let's see why we don't we consider non-rational transfer functions such as

$$H(z) = e^{-z^{-1}}$$

The transfer function relates the z transform of the input to the z transform of the output as

$$Y(z) = H(z) X(z) \tag{4}$$

If $H(z)$ has the form (2) then

$$\left(1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}\right) Y(z) = \left(b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}\right) X(z) \tag{5}$$

Since z^{-1} is the delay operator, the corresponding time-domain expression is

$$\begin{aligned} y(n) + a_1 y(n-1) + a_2 y(n-2) + \cdots + a_N y(n-N) \\ = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \cdots + b_M x(n-M) \end{aligned} \tag{6}$$

Therefore, any filter that can be realized by a finite difference equation has a rational transfer function. Conversely, if $H(z)$ cannot be expressed as a rational function of z^{-1} then it cannot be realized by a finite difference equation.

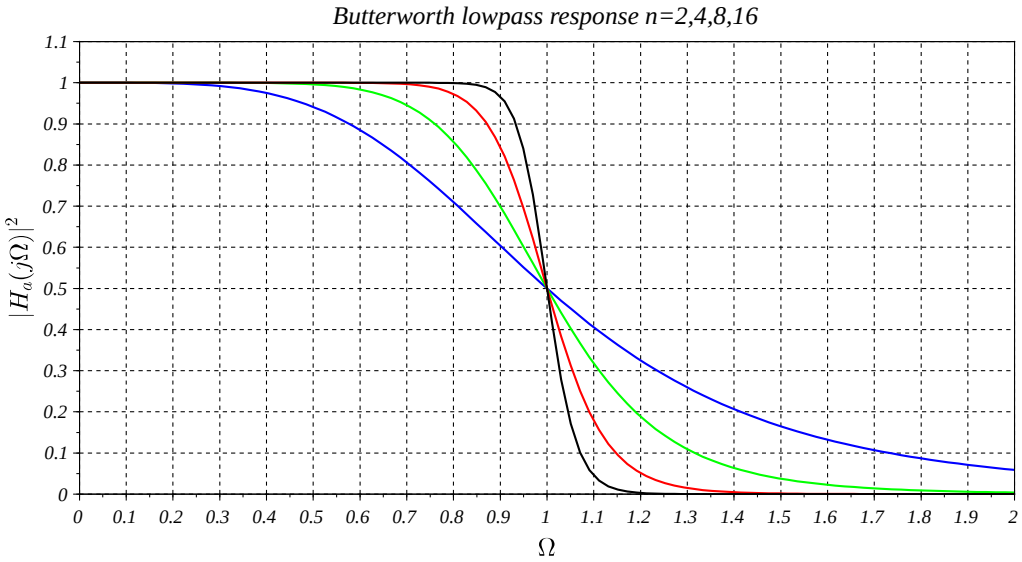

Fig. 1: Butterworth lowpass prototype filter magnitude-squared response for various values of n.

Butterworth lowpass filter prototypes

The transfer function of a *Butterworth lowpass filter prototype* has the form

$$H_a(s) = \frac{1}{B_n(s)} \tag{7}$$

where $B_n(s)$ is the n^{th} order *Butterworth polynomial*. These polynomials have the property that

$$|B_n(j\Omega)|^2 = 1 + \Omega^{2n} \tag{8}$$

Fig. 1 shows the Butterworth prototype response for various values of n. Note that in all cases $|H_a(j)|^2 = 1/2$. We say that $\Omega = 1$ is the *half-power (or –3 dB) frequency*.

We call (7) a prototype because we will use it as a starting point to create lowpass, highpass, bandpass, and bandstop filters with arbitrary cutoff frequencies. We will do this by applying different *filter transformations*.

For larger values of n the transition from passband to stopband is more rapid. If for some $\Omega > 1$ we wish to have

$$\frac{1}{1 + \Omega^{2n}} \le A^2 < 1 \tag{9}$$

then we require

$$n \ge \frac{1}{2\log(\Omega)} \log\left(\frac{1}{A^2} - 1\right) \tag{10}$$

Example 1: What order Butterworth prototype filter is required to achieve a response of $-30\,\text{dB}$ at $\Omega = 1.5$?

The response is $A^2 = 10^{-30/10} = 0.001$, so

$$n \geq \frac{1}{2\log(1.5)}\log(1000-1) = 8.517$$

Rounding up we have $n = 9$.

Therefore, the *filter order n* is determined by how rapid we want the transition from passband to stopband to be.

Butterworth polynomials

The first five Butterworth polynomials are

$$\begin{aligned} B_1(s) &= 1 + s \\ B_2(s) &= 1 + 1.4142136\,s + s^2 \\ B_3(s) &= 1 + 2\,s + 2\,s^2 + s^3 \\ B_4(s) &= 1 + 2.6131259\,s + 3.4142136\,s^2 + 2.6131259\,s^3 + s^4 \\ B_5(s) &= 1 + 3.236068\,s + 5.236068\,s^2 + 5.236068\,s^3 + 3.236068\,s^4 + s^5 \end{aligned} \tag{11}$$

In general, for even values of *n*

$$B_n(s) = \left(s^2 + 2\sin\left(\frac{\pi}{2n}\right)s + 1\right)\left(s^2 + 2\sin\left(\frac{3\pi}{2n}\right)s + 1\right)\cdots\left(s^2 + 2\sin\left(\frac{(n-1)\pi}{2n}\right)s + 1\right) \tag{12}$$

and for odd values of *n*

$$B_n(s) = \left(s^2 + 2\sin\left(\frac{\pi}{2n}\right)s + 1\right)\left(s^2 + 2\sin\left(\frac{3\pi}{2n}\right)s + 1\right)\cdots\left(s^2 + 2\sin\left(\frac{(n-2)\pi}{2n}\right)s + 1\right) \times (s+1) \tag{13}$$

These expressions give the polynomials in terms of quadratic factors (plus a linear factor for odd orders). We can multiply these factors out and put the polynomials in standard form

$$B_n(s) = s^n + c_1 s^{n-1} + c_2 s^{n-2} + \cdots \tag{14}$$

In practice it can be advantageous to implement IIR filters as a cascade of quadratic (and possibly linear) filters. If so, we can keep the polynomials in factored form. Scilab code to calculate Butterworth polynomial is given in the Appendix. Here is an example of its use.

Example 2: Use `Butterworth` *to calculate the 5th order Butterworth polynomial. Then calculate its zeros.*

```
--> p = Butterworth(5)
 p  =
                              2           3           4   5
   1 +3.236068s +5.236068s  +5.236068s  +3.236068s  +s

--> roots(p)
 ans  =

  -0.309017 + 0.9510565i
  -0.309017 - 0.9510565i
  -1.
  -0.809017 + 0.5877853i
```

```
-0.809017 - 0.5877853i
```

Poles

The zeros of the Butterworth polynomial become the poles of the prototype filter (7). The quadratic factors have the form

$$s^2 + 2\sin\phi\, s + 1 = 0 \tag{15}$$

The roots of this are

$$s = -\sin\phi \pm \sqrt{\sin^2\phi - 1} \tag{16}$$

Since

$$\pm\sqrt{\sin^2\phi - 1} = \pm j\sqrt{1 - \sin^2\phi} = \pm j\cos\phi \tag{17}$$

we have

$$s = -\sin\phi \pm j\cos\phi = e^{\pm j(\phi + \pi/2)} \tag{18}$$

For odd-order filters we also have a factor $s + 1$ which has root $s = -1$. We see that the s-plane poles of a Butterworth prototype filter fall on the unit circle $|s| = 1$.

Writing the polynomial in factored form

$$B_n(s) = (s - p_1)(s - p_2)\cdots(s - p_n) \tag{19}$$

we can represent the frequency response magnitude as

$$|H_p(j\Omega)| = \frac{1}{|j\Omega - p_1|\,|j\Omega - p_2|\cdots|j\Omega - p_n|} \tag{20}$$

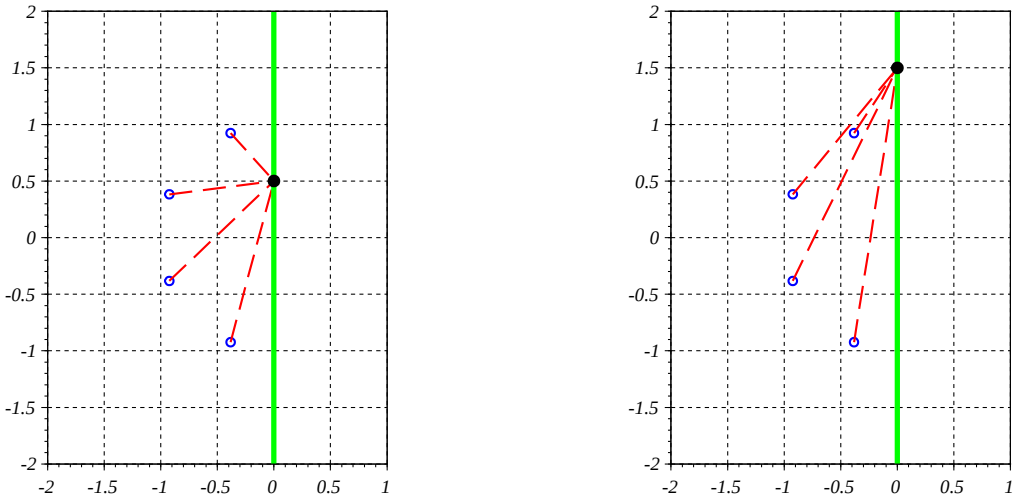

Fig. 2: The magnitude of the filter response is the inverse of the product of the distances from the poles to the frequency axis. (Left) as we move through the passband ($|\Omega| < 1$) some distances increase and some decrease. The product of distances more-or-less remains constant. (Right) in the stopband $|\Omega > 1|$ all distances increase with increasing $|\Omega|$ and the response decreases monotonically.

The factor $|j\Omega - p_k|$ is the distance in the *s* plane from the pole p_k to the point $j\Omega$. Therefore the filter response magnitude is the inverse of the product of the distances from the poles to $j\Omega$ (Fig. 2).

Other filter types

Although we only consider Butterworth filters in this course, these are not the only filter type or even necessarily the best filter type. Any real filter will only approximate an ideal filter response. This approximation involves trade-offs between different desirable characteristics, such as steepness of passband to stopband transition, flatness of passband response, and so on. Different filter types represent different trade-offs. Reference [1] describes some of the other classic filter types, such as Chebyshev, Cauer (elliptic), and Bessel. Butterworth filters are a good "all-round" choice, but certain applications may call for another type of filter.

Filter transformations

Given a lowpass filter prototype $H_p(s)$, we seek an analog filter $H_a(s)$ having a desired frequency response (lowpass, highpass, bandpass, bandstop) by applying a transformation to the *s* variable

$$\widetilde{s} = g_s(s) \tag{21}$$

where g_s is some function. The result is

$$H_a(s) = H_p(\widetilde{s}) = H_p(g_s(s)) \tag{22}$$

On the frequency axes $s = j\Omega$, $\widetilde{s} = j\widetilde{\Omega}$, so the frequency relation is

$$\widetilde{\Omega} = g_\Omega(\Omega) \tag{23}$$

where g_Ω is a function derived from g_s. The resulting frequency response is

$$H_a(j\Omega) = H_p(j\widetilde{\Omega}) = H_p(j\, g_\Omega(\Omega)) \tag{24}$$

For each type of filter we need to find the a corresponding function g_s that causes (24) to have the desired form.

Lowpass with arbitrary cutoff frequency

Since the prototype filter is lowpass, the simplest case is a lowpass filter with arbitrary cutoff frequency. The prototype filter has cutoff frequency $\widetilde{\Omega} = 1$. The scaling transformation

$$\widetilde{s} = \frac{s}{\Omega_c} \tag{25}$$

results in

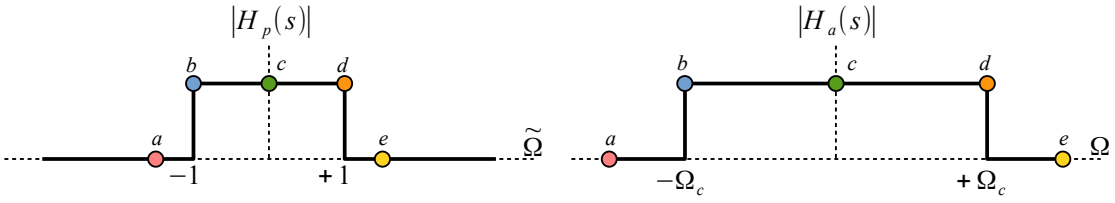

Fig. 3: $H_a(s) = H_p(s/\Omega_c)$ results in a lowpass filter with cutoff frequency Ω_c.

$$\widetilde{\Omega} = \frac{\Omega}{\Omega_c} \tag{26}$$

and $\Omega = \Omega_c$ corresponds to $\widetilde{\Omega} = 1$. Therefore the response of the filter

$$H_a(s) = H_p(s/\Omega_c) \tag{27}$$

at $\Omega = \Omega_c$ equals the prototype response at $\widetilde{\Omega} = 1$ as illustrated in Fig. 3. This allows us to design a lowpass filter with arbitrary cutoff frequency. Other types of filters involve more subtle transformations.

Example 3: *Design an analog lowpass filter with cutoff frequency 20 kHz based on a 3rd order Butterworth prototype.*

The prototype filter (in factored form) is

$$H_p(s) = \frac{1}{(s^2 + s + 1)(s + 1)}$$

$H_a(s) = H_p(s/\Omega_c)$ so

$$H_a(s) = \frac{1}{\left(\frac{1}{\Omega_c^2}s^2 + \frac{1}{\Omega_c}s + 1\right)\left(\frac{1}{\Omega_c}s + 1\right)} = \frac{\Omega_c^3}{\left(s^2 + \Omega_c s + \Omega_c^2\right)(s + \Omega_c)}$$

with $\Omega_c = 2\pi(20{,}000) = 40{,}000\pi$ rad/s .

Lowpass to highpass transformation

The prototype lowpass filter passes low frequencies and blocks high frequencies. A highpass filter blocks low frequencies and passes high frequencies. Therefore we want the response of a highpass filter at high frequencies to correspond to the prototype response at low frequencies and conversely. This suggests the inverse transformation

$$\widetilde{s} = \frac{\Omega_c}{s} \tag{28}$$

which results in the transfer function

$$H_a(s) = H_p(\widetilde{s}) = H_p\left(\Omega_c/s\right) \tag{29}$$

On the frequency axis $j\widetilde{\Omega} = \Omega_c/(j\Omega) = -j\Omega_c/\Omega$, so the frequency relation is

$$\widetilde{\Omega} = -\frac{\Omega_c}{\Omega} \tag{30}$$

The filter response at $\Omega = \pm\Omega_c$ is the prototype response at $\widetilde{\Omega} = \mp 1$, so we have the correct cutoff frequency. Now consider the limiting cases

$$\begin{aligned} &\Omega \to -\infty \quad && \widetilde{\Omega} \to 0^+ \\ &\Omega \to 0^- \quad && \widetilde{\Omega} \to \infty \\ &\Omega \to 0^+ \quad && \widetilde{\Omega} \to -\infty \\ &\Omega \to \infty \quad && \widetilde{\Omega} \to 0^- \end{aligned} \tag{31}$$

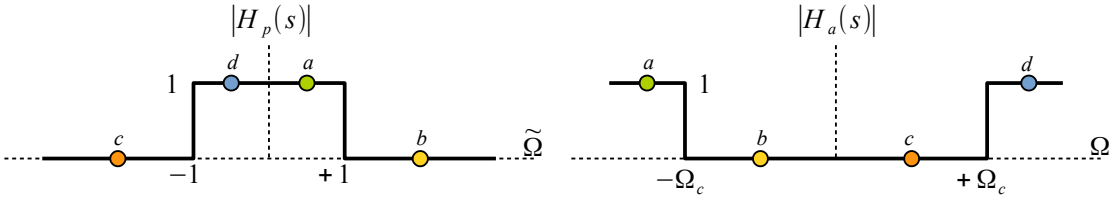

Fig. 4: $H_a(s)=H_p(\Omega_c/s)$ results in a highpass filter with cutoff frequency Ω_c.

These points are diagrammed in Fig. 4. Consider the 2nd order prototype lowpass filter

$$H_p(\widetilde{s})=\frac{1}{\widetilde{s}^2+2\sin\phi\,\widetilde{s}+1} \tag{32}$$

The transformation

$$\widetilde{s}=\frac{1}{s} \tag{33}$$

results in

$$H_a(s)=\frac{1}{\frac{1}{s^2}+2\sin\phi\frac{1}{s}+1}=\frac{s^2}{1+2\sin\phi\,s+s^2} \tag{34}$$

We have the same poles as the lowpass filter, but we now have a double zero at $s=0$.

Lowpass to bandpass transformation

Let's look for a transformation that combines the lowpass and highpass transformations. A candidate is

$$\widetilde{s}=\frac{s}{a}+\frac{b}{s}=\frac{s^2+ab}{as} \tag{35}$$

where *a* and *b* are constants to be determined. Let's rewrite this as

$$\widetilde{s}=\frac{s^2+\Omega_l\Omega_h}{(\Omega_h-\Omega_l)s} \tag{36}$$

where $\Omega_l\Omega_h=ab$ and $\Omega_h-\Omega_l=a$. Substituting $\widetilde{s}=j\widetilde{\Omega}$ and $s=j\Omega$ gives

$$j\widetilde{\Omega}=\frac{-\Omega^2+\Omega_l\Omega_h}{j(\Omega_h-\Omega_l)\Omega} \tag{37}$$

Dividing both sides by *j* we obtain the frequency mapping

$$\widetilde{\Omega}=\frac{\Omega^2-\Omega_l\Omega_h}{(\Omega_h-\Omega_l)\Omega} \tag{38}$$

We have the limiting cases

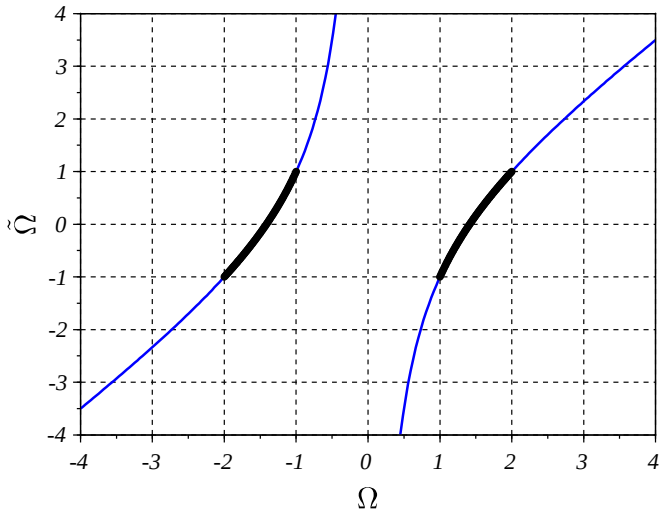

Fig. 5:Lowpass to bandpass transformation with $\Omega_l=1, \Omega_h=2$ *. Thick lines correspond to pass band.*

$$\begin{aligned} \Omega &\to -\infty & \tilde{\Omega} &\to -\infty \\ \Omega &\to 0^- & \tilde{\Omega} &\to \infty \\ \Omega &\to 0^+ & \tilde{\Omega} &\to -\infty \\ \Omega &\to \infty & \tilde{\Omega} &\to \infty \end{aligned} \tag{39}$$

We see that $-\infty<\Omega<0^-$ maps to $-\infty<\tilde{\Omega}<\infty$ and $0^+<\Omega<\infty$ also maps to $-\infty<\tilde{\Omega}<\infty$. Moreover

$$\begin{aligned} \Omega &= -\Omega_h & \tilde{\Omega} &= -1 \\ \Omega &= -\Omega_l & \tilde{\Omega} &= 1 \\ \Omega &= -\Omega_l & \tilde{\Omega} &= -1 \\ \Omega &= \Omega_h & \tilde{\Omega} &= 1 \end{aligned} \tag{40}$$

This is illustrated for $\Omega_l=1, \Omega_h=2$ in Fig. 5. Therefore, the passband of the lowpass prototype filter, $-1\le\tilde{\Omega}\le 1$ gets mapped onto both $-\Omega_h\le\Omega\le-\Omega_l$ and $\Omega_l\le\Omega\le\Omega_h$, the negative and positive-frequency parts of the passband of the bandpass filter.

Lowpass to bandstop transformation

The highpass filter is in a sense the "inverse" of the lowpass filter. Likewise, a bandstop filter is in a sense the "inverse" of a bandpass filter. "Inverting" (36) we have

$$\tilde{s}=\frac{(\Omega_h-\Omega_l)s}{s^2+\Omega_l\Omega_h} \tag{41}$$

The corresponding frequency mapping is

$$\tilde{\Omega}=\frac{(\Omega_h-\Omega_l)\Omega}{\Omega_l\Omega_h-\Omega^2} \tag{42}$$

It is left as an exercise to show that (42) maps the passband of the lowpass prototype filter, $-1 \le \widetilde{\Omega} \le 1$, onto the desired passband of the bandstop filter.

Exercise 1: Show that (42) maps the passband of the lowpass prototype filter, $-1 \le \widetilde{\Omega} \le 1$, onto the desired passband of the bandstop filter.

References

1. https://en.wikipedia.org/wiki/Network_synthesis_filters

Scilab: Butterworth polynomial of order n

```
function B = Butterworth(n) //Butterworth polynomial of order n
  s = poly(0,'s');
  kMax = floor(n/2);
  B = 1;
  for k=1:kMax
    a = sin((2*k-1)*%pi/(2*n));
    B = B*(s^2+2*a*s+1);
  end
  if (modulo(n,2)) //odd order case
    B = B*(s+1);
  end
endfunction
```

Scilab: Butterworth polynomial of order n in factored form

```
function Bq = ButterworthQuads(n)
  s = poly(0,'s');
  kMax = floor(n/2);
  for k=1:kMax
    a = sin((2*k-1)*%pi/(2*n));
    Bq(k) = s^2+2*a*s+1;
  end
  if (modulo(n,2)) //odd order case
    Bq(k+1) = s+1;
  end
endfunction
```