Lab.6+7. Tutorial on IIR Filters

Draft Handouts

Definition

As discussed earlier, FIR filters are relatively easy to design. The textbox below summarizes the characteristics of FIR filters as well as the most popular design techniques.

CHARACTERISTICS OF FIR FILTERS

- Impulse Response has a Finite Duration (N Cycles)
- Linear Phase, Constant Group Delay (N Must be Odd)
- No Analog Equivalent
- Unconditionally Stable
- Can be Adaptive
- Computational Advantages when Decimating Output
- Easy to Understand and Design
  - Windowed-Sinc Method
  - Fourier Series Expansion with Windowing
  - Frequency Sampling Using Inverse FFT - Arbitrary Frequency Response
  - Parks-McClellan Program with Remez Exchange Algorithm

One of the drawbacks of FIR filters is that they require a large filter order to meet some design specifications. By using feedback, it is possible to meet a set of design specifications with a far smaller filter order than a comparable FIR filter. This is the idea behind IIR filter design. IIR filters get their name because their impulse response extends for an infinite period of time. This is because they are recursive, i.e., they utilize feedback. Although they can be implemented with fewer computations than FIR filters, IIR filters do not match the performance achievable with FIR filters, and do not have linear phase. Textbox below summarizes the IIR filter characteristics.

INFINITE IMPULSE RESPONSE (IIR) FILTERS

- Uses Feedback (Recursion)
- Impulse Response has an Infinite Duration
- Potentially Unstable
- Non-Linear Phase
- More Efficient than FIR Filters
- No Computational Advantage when Decimating Output
- Usually Designed to Duplicate Analog Filter Response
- Usually Implemented as Cascaded Second-Order Sections (Biquads)
Design of digital IIR filters is heavily dependent on that of their analog counterparts because there are plenty of resources, works and straightforward design methods concerning analog feedback filter design while there are hardly any for digital IIR filters. As a result, usually, when a digital IIR filter is going to be implemented, an analog filter (e.g. Chebyshev, Butterworth, Elliptic, etc) is first designed and then is converted to a digital filter by applying discretization techniques such as: 1) Bilinear transform or 2) Impulse invariance.

**Transfer function derivation**

Digital filters are often described and implemented in terms of the difference equation that defines how the output signal is related to the input signal:

\[
y[n] = \frac{1}{a_0} \left( b_0 x[n] + b_1 x[n-1] + \cdots + b_P x[n-P] - a_1 y[n-1] - a_2 y[n-2] - \cdots - a_Q y[n-Q] \right)
\]

where:

- \( P \) is the feedforward filter order
- \( b_i \) are the feedforward filter coefficients
- \( Q \) is the feedback filter order
- \( a_i \) are the feedback filter coefficients
- \( x[n] \) is the input signal
- \( y[n] \) is the output signal.

A more condensed form of the difference equation is:

\[
y[n] = \frac{1}{a_0} \left( \sum_{i=0}^{P} b_i x[n-i] - \sum_{j=1}^{Q} a_j y[n-j] \right)
\]

which, when rearranged, becomes:

\[
\sum_{j=0}^{Q} a_j y[n-j] = \sum_{i=0}^{P} b_i x[n-i]
\]

To find the transfer function of the filter, we first take the Z-transform of each side of the above equation, where we use the time-shift property to obtain:
We define the transfer function to be:

\[
H(z) = \frac{Y(z)}{X(z)} - \frac{\sum_{i=0}^{P} b_i z^{-i}}{\sum_{j=0}^{Q} a_j z^{-j}}
\]

Considering that in most IIR filter designs coefficient \( a_0 = 1 \), the IIR filter transfer function takes the more traditional form:

\[
H(z) = \frac{\sum_{i=0}^{P} b_i z^{-i}}{1 + \sum_{j=1}^{Q} a_j z^{-j}}
\]

note that the positive sign indicates negative feedback

**Description of block diagram**

A typical block diagram of an IIR filter looks like the following. The \( z^{-1} \) block is a unit delay. The coefficients and number of feedback/feedforward paths are implementation-dependent.

Fig. Simple IIR filter block diagram (Direct Form I)
Stability

The transfer function allows us to judge whether or not a system is stable. To be specific, the stability criteria requires that the ROC of the system includes the unit circle. For example, for a causal system, all poles of the transfer function have to have an absolute value smaller than one. In other words, all poles must be located within a unit circle in the \( \mathbb{z} \)-plane.

The poles are defined as the values of \( z \) which make the denominator of \( H(z) \) equal to 0:

\[
0 = \sum_{j=0}^{Q} a_j z^{-j}
\]

Clearly, if \( a_j \neq 0 \) then the poles are not located at the origin of the \( z \)-plane. This is in contrast to the FIR filter where all poles are located at the origin, and is therefore always stable.

**Note:** IIR filters are sometimes preferred over FIR filters because an IIR filter can achieve a much sharper transition region roll-off than FIR filter of the same order.

Example

Let the transfer function of a filter \( H \) be

\[
H(z) = \frac{B(z)}{A(z)} = \frac{1}{1 - az^{-1}}
\]

with ROC \( a < |z| \) and \( 0 < a < 1 \)

which has a pole at \( a \), is stable and causal. The time-domain impulse response is

\[
h(n) = a^n u(n)
\]

which is non-zero for \( n \geq 0 \).

**IIR Common Analog Filters**

A popular method for IIR filter design is to first design the analog-equivalent filter and then mathematically transform the transfer function \( H(s) \) into the \( z \)-domain, \( H(z) \). Multiple pole designs are implemented using cascaded biquad sections. The most popular analog filters are the Butterworth, Chebyshev, Elliptical, and Bessel. There are many CAD programs available to generate the Laplace transform, \( H(s) \), for these filters.

- Butterworth (also called maximally flat) has no ripple in the passband or stopband and has monotonic response in both regions. Butterworth filters are maximally-flat IIR
filters. For this reason, the only design parameters are the cutoff frequency and the filter order.

**Example:** Design a 7th order Butterworth filter with a 3 dB point at 0.3\(\pi\).

\[
\begin{align*}
Hf &= \text{fdesign.lowpass}('N,F3db',7,0.3); \\
Hb &= \text{design}(Hf,'butter');
\end{align*}
\]

- Chebyshev Type 1 filter has a faster rolloff than the Butterworth (for the same number of poles) and has ripple in the passband. Chebyshev type I filters can attain a smaller transition width than a Butterworth filter of the same order by allowing for ripples in the passband of the filter. The stopband is, as with Butterworth filters, maximally flat. For a given filter order, the trade-off is thus between passband ripple and transition width.

**Example:** Compare the 7th-order Butterworth filter from previous examples with a 7th-order Chebyshev type I filter with 1 dB of peak-to-peak passband ripple.
- The Chebyshev Type 2 filter is rarely used, but has ripple in the stopband rather than the passband.

**Example** Design a 6th order filter with a 3-dB point of $0.45\pi$. The filter must have an attenuation of at least 80 dB at frequencies above $0.75\pi$ and the passband ripple must not exceed 0.8 dB.

```matlab
Hf1 = fdesign.lowpass('N,F3db',6,0.45);
Hf2 = fdesign.lowpass('N,F3db,Ap',6,0.45,0.8);
Hf3 = fdesign.lowpass('N,F3db,Ast',6,0.45,80);
Hb = design(Hf1,'butter');
Hc1 = design(Hf2,'cheby1');
Hc2 = design(Hf3,'cheby2');
```
The three designs are shown in Figure. Only the Chebyshev type II filter reaches the required attenuation of 80 dB by 0.75π. Also, Chebyshev type II, the latter’s group-delay\textsuperscript{1} is smaller than the former’s for most of the passband. Even though all three designs in the previous example are of 6th order, the Chebyshev type II implementation actually requires more multipliers.

**Methods for IIR design**

**A. Impulse Invariance Transformation Method**

Traditionally, IIR filter design is based on the concept of transforming a continuous-time, or analog, design into the discrete-time domain. In impulse invariance method, the impulse response of the digital filter is the samples of the impulse response of the continuous-time filter:

$$h[n]=T \cdot h[nT]$$

where $T$ represents the sampling interval.

This method is based on the concept of mapping each $s$-plane pole of the continuous time filter to a corresponding $z$-plane pole using the substitution $(1-e^{πkTz^{-1}})$ for $(s + p_k)$ in $H(s)$. This can be achieved by several different means. Partial fraction expansion of $H(s)$ and substitution of $(1-e^{πkTz^{-1}})$ for $(s + p_k)$ is a direct method to do.

**B. Bilinear Transform Method of Digital Filter Implementation**

The bilinear transform method of converting an analog filter design to discrete time is relatively straightforward, often involving less algebraic manipulation than the impulse invariant method. It is achieved by making the substitution:

\textsuperscript{1} Group delay Characterizes the phase distortion (waveform distortion) introduced by the filter.
\[ s = \frac{2(z - 1)}{T(z + 1)} \]

In \( H(s) \), where \( T \) is the sampling period of the digital filter; that is,

\[ H(z) = H(s) \bigg|_{s=\frac{2(z-1)}{T(z+1)}} \]

The concept behind the bilinear transform is that of compressing the frequency response of an analog filters design such that its response over the entire range of frequencies from zero to infinity is mapped into the frequency range zero to half the sampling frequency of the digital filter. This may be represented by:

\[ f_D = \frac{\arctan(\pi f_A T_s)}{\pi T_s} \quad \text{or} \quad \omega_D = \frac{2}{T_s} \arctan\left(\frac{\omega_A T_s}{2}\right) \]

\[ f_A = \frac{\tan(\pi f_D T_s)}{\pi T_s} \quad \text{or} \quad \omega_A = \frac{2}{T_s} \tan\left(\frac{\omega_D T_s}{2}\right) \]

As a result of the frequency warping inherent in the bilinear transform, the cutoff frequency of the discrete-time filter obtained is not equal to the cutoff frequency of the analog filter. A technique called pre-warping the prototype analog design (used by default in the MATLAB filter design and analysis tool fdatool) can be used in such a way that the bilinear transform maps an analog frequency \( \omega_A = \omega_c \), in the range 0 to \( \omega_s/2 \), to exactly the same digital frequency the same digital frequency \( \omega_D = \omega_c \). This technique is based on the selection of \( T \) according to \( T = 2 \tan(\pi \omega_c/\omega_D) / \omega_c \).