The Z-Transform

Mathematically, the z-transform is a mapping between complex sequences and analytical functions on the complex plane. Given a discrete-time signal $x[n]$, the z-transform of $x[n]$ is formally defined as the complex function of a complex variable $z \in \mathbb{C}$

$$X(z) = \mathscr{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \tag{6.1}$$

Contrary to the Fourier transform (as well as to other well-known transforms such as the Laplace transform or the wavelet transform), the z-transform is not an analysis tool *per se*, in that it does not offer a new physical insight on the nature of signals and systems. The z-transform, however, derives its status as a fundamental tool in digital signal processing from two key features:

- Its mathematical formalism, which allows us to easily solve constant-coefficient difference equations as algebraic equations (and this was precisely the context in which the z-transform was originally invented).
- Its close association to the DTFT, which provides us with easy stability criteria for the design and the use of digital filters. (It is evident that the z-transform computed on the unit circle, i.e. for $z = e^{j\omega}$, is nothing but the DTFT of the sequence).

Probably the best approach to the z-transform is to consider it as a clever mathematical *transformation* which facilitates the manipulation of complex sequences; for discrete-time filters, the z-transform bridges the algorithmic side (i.e. the CCDE) to the analytical side (i.e. the spectral properties) in an extremely elegant, convenient and ultimately beautiful way.

6.1 Filter Analysis

To see the usefulness of the z-transform in the context of the analysis and the design of realizable filters, it is sufficient to consider the following two formal properties of the z-transform operator:

- **Linearity:** given two sequences $x[n]$ and $y[n]$ and their respective z-transforms $X(z)$ and $Y(z)$, we have

$$\mathscr{Z}\{\alpha x[n]+\beta y[n]\}=\alpha X(z)+\beta Y(z)$$

- **Time-shift:** given a sequence $x[n]$ and its z-transform $X(z)$, we have

$$\mathscr{Z}\{x[n-N]\}=z^{-N}X(z)$$

In the above, we have conveniently ignored all convergence issues for the z-transform; these will be addressed shortly but, for the time being, let us just make use of the formalism as it stands.

6.1.1 Solving CCDEs

Consider the generic filter CCDE (Constant-Coefficient Difference Equation) in (5.46):

$$y[n]=\sum_{k=0}^{M-1} b_k x[n-k]-\sum_{k=1}^{N-1} a_k y[n-k]$$

If we apply the z-transform operator to both sides and exploit the linearity and time-shifting properties, we have

$$Y(z)=\sum_{k=0}^{M-1} b_k z^{-k} X(z)-\sum_{k=1}^{N-1} a_k z^{-k} Y(z) \tag{6.2}$$

$$=\frac{\sum_{k=0}^{M-1} b_k z^{-k}}{1+\sum_{k=1}^{N-1} a_k z^{-k}} X(z) \tag{6.3}$$

$$=H(z)X(z) \tag{6.4}$$

$H(z)$ is called the *transfer function* of the LTI filter described by the CCDE. The following properties hold:

- The transfer function of a realizable filter is a *rational transfer function* (i.e. a ratio of finite-degree polynomials in z).
- The transfer function evaluated on the unit circle is the frequency response of the filter. In other words, the z-transform gives us the possibility of obtaining the frequency response of a filter *directly from the underlying CCDE*; in a way, we will no longer need to occupy ourselves with the actual impulse response.
- The transfer function is the z-transform of the filter's impulse response (which follows immediately from the fact that the impulse response is the filter's output when the input is $x[n] = \delta[n]$ and that $\mathscr{Z}\{\delta[n]\} = 1$).
- The result in (6.4) can be extended to general sequences to yield a z-transform version of the convolution theorem. In particular, given the square-summable sequences $x[n]$ and $h[n]$ and their convolution $y[n] = x[n] * h[n]$, we can state that

$$\mathscr{Z}\{y[n]\} = Y(z) = X(z)H(z) \tag{6.5}$$

  which can easily be verified using an approach similar to the one used in Section 5.4.2.

6.1.2 Causality

As we saw in Section 5.7.1, a CCDE can be rearranged to express either a causal or a noncausal realization of a filter. This ambiguity is reflected in the z-transform and can be made explicit by the following example. Consider the sequences

$$x_1[n] = u[n] \tag{6.6}$$
$$x_2[n] = \delta[n] - u[-n] \tag{6.7}$$

where $u[n]$ is the unit step. For the first sequence we have

$$X_1(x) = \sum_{n=0}^{\infty} z^{-n} = \frac{1}{1-z^{-1}} \tag{6.8}$$

(again, let us neglect convergence issues for the moment). For the second sequence we have

$$X_2(x) = -\sum_{n=1}^{\infty} z^{n} = 1 - \frac{1}{1-z} = \frac{1}{1-z^{-1}} \tag{6.9}$$

so that, at least formally, $X_1(z) = X_2(z)$. In other words, the z-transform is not an invertible operator or, more precisely, it is invertible up to a causality specification. If we look more in detail, the sum in (6.8) converges only for $|z| > 1$ while the sum in (6.9) converges only for $|z| < 1$. This is actually a general fact: the values for which a z-transform exists define the causality or anticausality of the underlying sequence.

6.1.3 Region of Convergence

We are now ready to address the convergence issues that we have put aside so far. For any given sequence $x[n]$, the set of points on the complex plane for which $\sum x[n]z^{-n}$ exists and is finite, is called the *region of convergence* (ROC) for the z-transform. In order to study the properties of this region, it is useful to split the sum in (6.1) as

$$X(z) = \sum_{n=-N}^{-1} x[n]z^{-n} + \sum_{n=0}^{M} x[n]z^{-n} \tag{6.10}$$

$$= \sum_{n=1}^{N} x[n]z^{n} + \sum_{n=0}^{M} \frac{x[n]}{z^n} \tag{6.11}$$

$$= X_a(z) + X_c(z) \tag{6.12}$$

where $N, M \geq 0$ and where both N and M can be infinity. Now, for $X(z_0)$ to exist and be finite, both power series $X_a(z)$ and $X_c(z)$ must converge in z_0; since they are power series, when they do converge, they converge *absolutely*. As a consequence, for all practical purposes, we define the ROC in terms of absolute convergence:[1]

$$z \in \mathrm{ROC}\{X(z)\} \iff \sum_{n=-\infty}^{\infty} \left| x[n]z^{-n} \right| < \infty \tag{6.13}$$

Then the following properties are easily derived:

- *The ROC has circular symmetry.* Indeed, the sum in (6.13) depends only on the magnitude of z; in other words, if $z_0 \in \mathrm{ROC}$, then the set of complex points $\{z \mid |z| = |z_0|\}$ is also in the ROC, and such a set defines a circle.

- *The ROC for a finite-support sequence is the entire complex plane (with the possible exception of zero and infinity).* For a finite-support se-

[1] This definition excludes the points on the boundary of the ROC from the ROC itself, but this has no consequence on the results we will derive and use in what follows.

quence, both N and M in (6.10) are finite. The z-transform is therefore a simple polynomial which exists and is finite for all values of z (except for $z=0$ if $N>0$ and/or $z=\infty$ if $M>0$).

- *The ROC for a causal sequence is a circularly symmetric region in the complex plane extending to infinity* (Fig. 6.1a). For a causal sequence, $M=\infty$ while N is finite (and equal to zero for a strictly causal sequence). In this case, $X_a(z)$ is a finite-degree polynomial and poses no convergence issues (i.e. ROC$\{X_a(z)\}=\mathbb{C}\setminus\{\infty\}$). As for $X_c(z)$, assume $X_c(z_0)$ exists and is finite and take any z_1 so that $|z_1|>|z_0|$; we have that for all n:

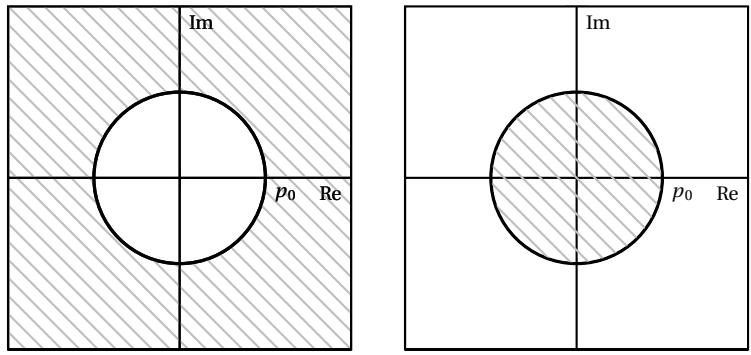
$$\left|\frac{x[n]}{z_1^n}\right| \leq \left|\frac{x[n]}{z_0^n}\right|$$

so that $X_c(z)$ is absolutely convergent in z_1 as well.

- *The ROC for an anticausal sequence is a disk in the complex plane, centered in the origin* (Fig. 6.1b). For an anticausal sequence, $N=\infty$ while M is finite so that $X_c(z)$ poses no convergence issues (i.e. ROC$\{X_c(z)\}=\mathbb{C}\setminus\{0\}$). As for $X_a(z)$, assume $X_a(z_0)$ exists and is finite and take any z_1 so that $|z_1|<|z_0|$; we have that for all n:

$$\left|x[n]\,z_1^n\right| \leq \left|x[n]z_0^n\right|$$

so that $X_a(z)$ is absolutely convergent in z_1 as well.



(a) (b)

Figure 6.1 ROC shapes (hatched area): (a) causal sequence; (b) anticausal sequence.

6.1.4 ROC and System Stability

The z-transform provides us with a quick and easy way to test the stability of a linear system. Recall from Section 5.2.2 that a necessary and sufficient condition for an LTI system to be BIBO stable is the absolute summability of its impulse response. This is equivalent to saying that a system is BIBO stable if and only if the z-transform of its impulse response is absolutely convergent in $|z| = 1$. In other words, *a system is BIBO stable if the ROC of its transfer function includes the unit circle.*

6.1.5 ROC of Rational Transfer Functions and Filter Stability

For rational transfer functions, the analysis of the ROC is quite simple; indeed, the only "trouble spots" for convergence are the values for which the denominator of (6.3) is zero. These values are called the *poles* of the transfer functions and clearly they must lie outside of the ROC. As a consequence, we have an extremely quick and practical rule to determine the stability of a realizable filter.

Consider a causal filter:

- Find the roots of the transfer function's denominator (considered as a polynomial in z). These are the system's poles. Call p_0 the pole with the largest magnitude.
- The ROC has circular symmetry, it extends outwards to infinity and it cannot include any pole; therefore the ROC will simply be the area on the complex plane outside of a circle of radius $|p_0|$.
- For the ROC to include the unit circle we must have $|p_0| < 1$. Therefore, in order to have stability, *all the poles must be inside the unit circle.*

For an anticausal system the procedure is symmetrical; once the smallest-magnitude pole is known, the ROC will be a disk of radius $|p_0|$ and therefore in order to have stability, all the poles will have to be outside of the unit circle.

6.2 The Pole-Zero Plot

The rational transfer function derived in (6.3) can be written out explicitly in terms of the CCDEs coefficients, as follows:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_{M-1} z^{-(M-1)}}{1 + a_1 z^{-1} + \cdots + a_{N-1} z^{-(N-1)}} \tag{6.14}$$

The transfer function is the ratio of two polynomials in z^{-1} where the degree of the numerator polynomial is $M-1$ and that of the denominator polynomial is $N-1$. As a consequence, the transfer function can be rewritten in factored form as

$$H(z) = b_0 \frac{\prod_{n=1}^{M-1}(1 - z_n z^{-1})}{\prod_{n=1}^{N-1}(1 - p_n z^{-1})} \tag{6.15}$$

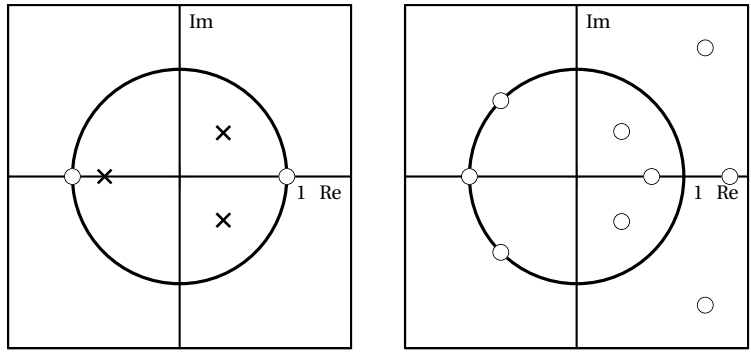
where the z_n are the $M-1$ complex roots of the numerator polynomial and are called the *zeros* of the system; the p_n are the $N-1$ complex roots of the denominator polynomial and, as we have seen, they are called the poles of the system. Both poles and zeros can have arbitrary multiplicity. Clearly, if $z_i = p_k$ for some i and k (i.e. if a pole and a zero coincide) the corresponding first-order factors cancel each other out and the degrees of numerator and denominator are both decreased by one. In general, it is assumed that such factors have already been removed and that the numerator and denominator polynomials of a given rational transfer function are coprime.

The poles and the zeros of a filter are usually represented graphically on the complex plane as crosses and dots, respectively. This allows for a quick visual assessment of stability which, for a causal system, consists of checking whether all the crosses are inside the unit circle (or, for anticausal systems, outside).

6.2.1 Pole-Zero Patterns

The pole-zero plot can exhibit distinctive patterns according to the properties of the filter.

Real-Valued Filters. If the filter coefficients are real-valued (and this is the only case that we consider in this text book) both the numerator and denominator polynomials are going to have real-valued coefficients. We can now recall a fundamental result from complex algebra: the roots of a polynomial with real-valued coefficients are either real or they occur in complex-conjugate pairs. So, if z_0 is a complex zero of the system, z_0^* is a zero as well. Similarly, if p_0 is a complex pole, so is p_0^*. The pole-zero plot will therefore shows a symmetry around the real axis (Fig. 6.2a).



(a) (b)

Figure 6.2 Examples of pole-zero patterns: (a) real-valued IIR filter (note the symmetry around the x-axis); (b) linear phase FIR (each zero appears with its reciprocal).

Linear-Phase FIR Filters. First of all, note that the pole-zero plot for an FIR filter is actually just a zero plot, since FIR's have no poles.[(2)] A particularly important case is that of linear phase FIR filters; as we will see in detail in Section 7.2.2, linear phase imposes some symmetry constraints on the CCDE coefficients (which, of course, coincide with the filter taps). These constraints have a remarkable consequence: if z_0 is a (complex) zero of the system, $1/z_0$ is a zero as well. Since we consider real-valued FIR filters exclusively, the presence of a complex zero in z_0 implies the existence of three other zeros, namely in $1/z_0$, z_0^* and $1/z_0^*$ (Fig. 6.2b). See also the discussion in Section 7.2.2

6.2.2 Pole-Zero Cancellation

We have seen in Section 5.2.1 that the effect of a cascade of two or more filters is that of a single filter whose impulse response is the convolution of all of the filters' impulse responses. By the convolution theorem, this means that the overall transfer function of a cascade of K filters $\mathscr{H}_i$, $i = 1,\ldots,K$ is simply the product of the single transfer functions $H_i(z)$:

$$H(z) = \prod_{i=1}^{K} H_i(z)$$

If all filters are realizable, we can consider the factored form of each $H_i(z)$ as in (6.15). In the product of transfer functions, it may happen that some of

[(2)] Technically, since we use the notation z^{-1} to express a delay, causal FIR filters have a multiple pole in the origin ($z = 0$). This is of no consequence for stability, however, so we will not consider it further.

the poles of a given $H_i(z)$ coincide with the zeros of another transfer function, which leads to a pole-zero cancellation in the overall transfer function. This is a method that can be used (at least theoretically) to stabilize an otherwise unstable filter. If one of the poles of the system (assuming causality) lies outside of the unit circle, this pole can be compensated by cascading an appropriate first- or second-order FIR section to the original filter. In practical realizations, care must be taken to make sure that the cancellation is not jeopardized by numerical precision problems.

6.2.3 Sketching the Transfer Function from the Pole-Zero Plot

The pole-zero plot represents a convenient starting point in order to estimate the shape of the magnitude for a filter's transfer function. The basic idea is to consider the absolute value of $H(z)$, which is a three-dimensional plot ($\left|H(z)\right|$ being a real function of a complex variable). To see what happens to $\left|H(z)\right|$ it is useful to imagine a "rubber sheet" laid over the complex plane; then,

- every zero corresponds to a point where the rubber sheet is "glued" to the plane,
- every pole corresponds to a "pole" which is "pushing" the rubber sheet up (to infinity),

so that the shape of $\left|H(z)\right|$ is that of a very lopsided "circus tent". The magnitude of the transfer function is just the height of this circus tent measured around the unit circle.

In practice, to sketch a transfer function (in magnitude) given the pole-zero plot, we proceed as follows. Let us start by considering the upper half of the unit circle, which maps to the $[0,\pi]$ interval for the ω axis in the DTFT plot; for real-valued filters, the magnitude response is an even function and, therefore, the $[-\pi,0]$ interval need not be considered explicitly. Then:

1. Check for zeros on the unit circle; these correspond to points on the frequency axis in which the magnitude response is exactly zero.
2. Draw a line from the origin of the complex plane to each pole and each zero. The point of intersection of each line with the unit circle gives the location of a local extremum for the magnitude response.
3. The effect of each pole and each zero is made stronger by their proximity to the unit circle.

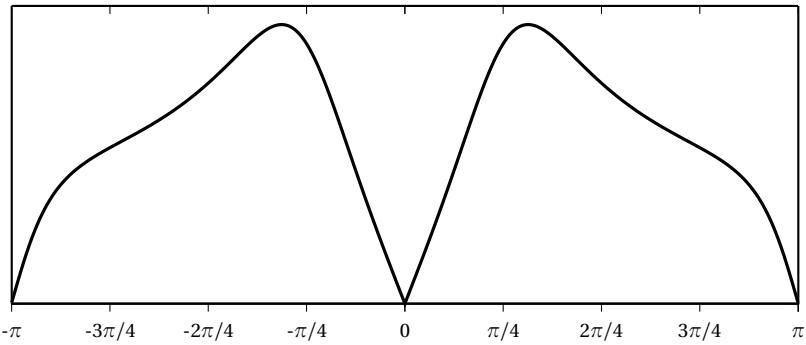


Figure 6.3 Sketch of the magnitude response for the pole-zero plot of Figure 6.2(a).

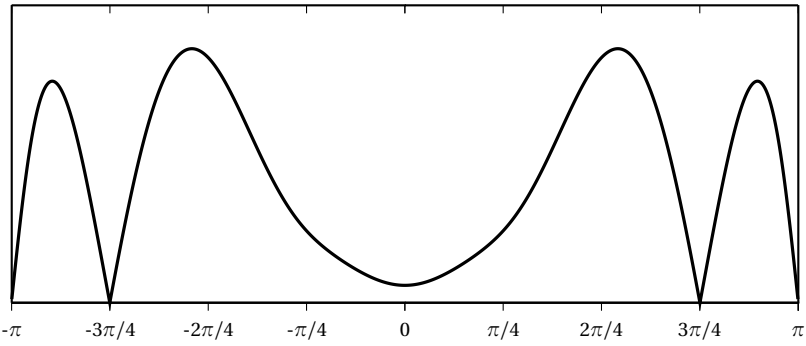


Figure 6.4 Sketch of the magnitude response for the pole-zero plot of Figure 6.2(b).

As an example, the magnitude responses of the pole-zero plots in Figure 6.2 are displayed in Figures 6.3 and 6.4.

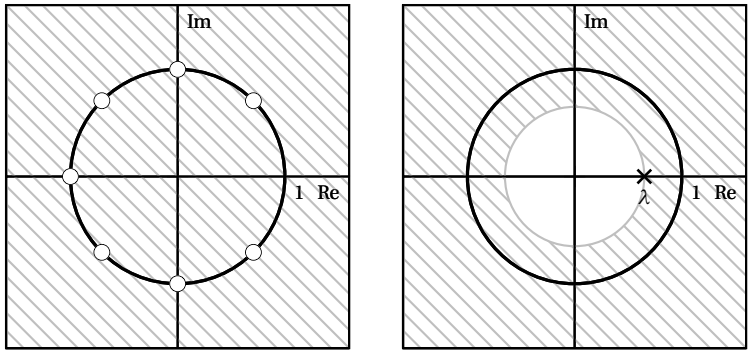
6.3 Filtering by Example – Z-Transform

We will quickly revisit the examples of the previous chapter to show the versatility of the z-transform.

Moving Average. From the impulse response in (5.14), the transfer function of the moving average filter is

$$H(z) = \frac{1}{N}\sum_{k=0}^{N-1} z^{-k} = \frac{1}{N}\,\frac{1-z^{-N}}{1-z^{-1}} \tag{6.16}$$

from which the frequency response (5.26) is easily derived by setting $z = e^{j\omega}$. It is easy to see that the zeros of the filter are on all the roots of unity



(a) (b)

Figure 6.5 Pole-zero plots and ROC: (a) moving average filter with $N = 8$; (b) leaky integrator with $\lambda = 0.65$.

except for $z = 1$, where the numerator and denominator in (6.17) cancel each other out. A factored representation of the transfer function for the moving average is therefore

$$H(z) = \frac{1}{N} \prod_{k=1}^{N-1} (1 - W_N^k z^{-k}) \tag{6.17}$$

and the pole-zero plot (for $N = 8$) is shown in Figure 6.5(a). There being no poles, the filter is unconditionally stable.

Leaky Integrator. From the CCDE for the leaky integrator (5.16) we immediately have

$$Y(z) = \lambda z^{-1} Y(z) + (1 - \lambda) X(z) \tag{6.18}$$

from which

$$H(z) = \frac{1 - \lambda}{1 - \lambda z^{-1}} \tag{6.19}$$

The transfer function has therefore a single real pole in $z = \lambda$; for a causal realization, this implies that the ROC is the region of the complex plane extending outward from the circle of radius λ. The causal filter is stable if λ lies inside the unit circle, i.e. if $\lambda < 1$. An example of pole-zero plot together with the associated ROC is shown in Figure 6.5(b) for the (stable) case of $\lambda = 0.65$.

Examples

Example 6.1: Transform of periodic functions

The z-transform converges without fuss for infinite-energy sequences which the Fourier transform has some difficulties dealing with. For instance, the

z-transform manages to "bring down" the unit step because of the vanishing power of z^{-n} for $|z|>1$ and n large and this is the case for all one-sided sequences which grow no more than exponentially. However, if $|z^{-n}|\to 0$ for $n\to\infty$ then necessarily $|z^{-n}|\to\infty$ for $n\to-\infty$ and this may pose a problem for the convergence of the z-transform in the case of two-sided sequences. In particular, the z-transform does not converge in the case of periodic signals since only one side of the repeating pattern is "brought down" while the other is amplified limitlessly. We can circumvent this impasse by "killing" half of the periodic signal with a unit step. Take for instance the one-sided cosine:

$$x[n]=\cos(\omega_0 n)\,u[n]$$

its z-transform can be derived as

$$\begin{aligned}
X(z) &= \sum_{n=-\infty}^{\infty} z^{-n}\cos(\omega_0 n)\,u[n] \\
&= \sum_{n=0}^{\infty} z^{-n}\cos(\omega_0 n) \\
&= \frac{1}{2}\sum_{n=0}^{\infty} e^{j\omega_0 n}z^{-n} + \frac{1}{2}\sum_{n=0}^{\infty} e^{-j\omega_0 n}z^{-n} \\
&= \frac{1}{2}\left(\frac{1}{1-e^{j\omega_0}z^{-1}} + \frac{1}{1-e^{-j\omega_0}z^{-1}}\right) \\
&= \frac{1-\cos(\omega_0)z^{-1}}{1-2\cos(\omega_0)z^{-1}+z^{-2}}
\end{aligned}$$

Similar results can be obtained for signals such as $x[n]=\sin(\omega_0 n)\,u[n]$ or $x[n]=\alpha^n\cos(\omega_0 n)\,u[n]$.

Example 6.2: The impossibility of ideal filters

The z-transform of an FIR impulse response can be expressed as a simple polynomial $P(z)$ of degree $L-1$ where L is the number of nonzero taps of the filter (we can neglect leading factors of the form z^{-N}). The fundamental theorem of algebra states that such a polynomial has at most $L-1$ roots; as a consequence, the frequency response of an FIR filter can never be identically zero over a frequency interval since, if it were, its z-transform would have an infinite number of roots. Similarly, by considering the polynomial $P(z)-C$, we can prove that the frequency response can never be constant C over an interval which proves the impossibility of achieving ideal (i.e. "brick-wall") responses with an FIR filter. The argument can be easily extended to rational transfer functions, confirming the impossibility of a realizable filter whose characteristic is piecewise perfectly flat.

Filter Design

In discrete-time signal processing, filter design is the art of turning a set of requirements into a well-defined numerical algorithm. The requirements, or *specifications*, are usually formulated in terms of the filter's frequency response; the design problem is solved by finding the appropriate coefficients for a suitable difference equation which implements the filter and by specifying the filter's architecture. Since realizable filters are described by rational transfer functions, filter design can usually be cast in terms of a polynomial optimization procedure for a given error measure. Additional design choices include the *computational cost* of the designed filters, i.e. the number of mathematical operations and storage necessary to compute each output sample. Finally, the structure of the difference equation defines an explicit operational procedure for computing the filter's output values; by arranging the terms of the equation in different ways, we can arrive at different algorithmic structures for the implementation of digital filters.

7.1 Design Fundamentals

As we have seen, a realizable filter is described by a rational transfer function; designing a filter corresponds to determining the coefficients of the polynomials in transfer function with respect to the desired filter characteristics. For an FIR filter of length M, there are M coefficients that need to be determined, and they correspond directly to the filter's impulse response. Similarly, for an IIR filter with a numerator of degree $M-1$ and a denominator of degree $N-1$, there are $M+N-1$ coefficients to determine (since we always assume $a_0 = 1$). The main questions are the following:

- *How do we specify the characteristics of the desired filter?* This question effectively selects the domain in which we will measure the difference (i.e. the *error*) between the desired filter and the achieved implementation. This can be the time domain (where we would be comparing impulse responses) or the frequency domain (where we would be comparing frequency responses). Usually the domain of choice is the frequency domain.

- *What are the criteria to measure the quality of the obtained filter?* This question defines the way in which the above-mentionned error is measured; again, different criteria are possible (such as minimum square error or minimax) and they do depend on the intended application.

- *How do we choose the filter's coefficients in order to obtain the desired filtering characteristics?* This question defines an optimization problem in a parameter space of dimension $M+N-1$ with the optimality criterion chosen above; it is usually answered by the existence of a numerical recipe which performs the task.

- *What is the best algorithmic structure (software or hardware) to implement a given digital filter?* This last question concerns the algorithmic design of the filter itself; the design is subject to various application-dependent constraints which include computational speed, storage requirement and arithmetic precision. Some of these design choices will be addressed at the end of the Chapter.

As is apparent, real-world filters are designed with a variety of practical requirements in mind, most of which are conflicting. One such requirement, for instance, is to obtain a low "computational price" for the filtering operation; this cost is obviously proportional to the number of coefficients in the filter, but it also depends heavily on the underlying hardware architecture. The tradeoffs between disparate requirements such as cost, precision or numerical stability are very subtle and not altogether obvious; the art of the digital filter designer, although probably less dazzling than the art of the *analog* filter designer, is to determine the best design strategy for a given practical problem.

7.1.1 FIR versus IIR

Filter design has a long and noble history in the analog domain: a linear electronic network can be described in terms of a differential equation linking, for instance, the voltage as a function of time at the input of the network to the voltage at the output. The arrangement of the capacitors,

inductances and resistors in the network determine the form of the differential equation, while their values determine its coefficients. A fundamental difference between an analog filter and a digital filter is that the transformation from input to output is almost always considered *instantaneous* (i.e. the propagation effects along the circuit are neglected). In digital filters, on the other hand, the delay is always explicit and is actually the fundamental building block in a processing system. Because of the physical properties of capacitors, which are ubiquitous in analog filters, the transfer function of an analog filter (expressed in terms of its Laplace transform) is "similar" to the transfer function of an IIR filter, in the sense that it contains both poles and zeros. In a sense, IIR filters can be considered as the discrete-time counterpart of classic analog filters. FIR filters, on the other hand, are the flagship of digital signal processing; while one could conceive of an analog equivalent to an FIR, its realization would require the use of analog delay lines, which are costly and impractical components to manufacture. In a digital signal processing scenario, on the other hand, the designer can freely choose between two lines of attack with respect to a filtering problem, IIR or FIR, and therefore it is important to highlight advantages and disadvantages of each.

FIR Filters. The main *advantages* of FIR filters can be summarized as follows:

- unconditional stability;
- precise control of the phase response and, in particular, exact linear phase;
- optimal algorithmic design procedures;
- robustness with respect to finite numerical precision hardware.

While their *disadvantages* are mainly:

- longer input-output delay;
- higher computational cost with respect to IIR solutions.

IIR Filters. IIR filters are often an afterthought in the context of digital signal processing in the sense that they are designed by mimicking established design procedures in the analog domain; their appeal lies mostly in their compact formulation: for a given computational cost, i.e for a given number of operations per input sample, they can offer a much better magnitude

response than an equivalent FIR filter. Furthermore, there are a few fundamental processing tasks (such as DC removal, as we will see later) which are the natural domain of IIR filters. The drawbacks of IIR filters, however, mirror in the negative the advantages of FIR's. The main advantages of IIR filters can be summarized as follows:

- lower computational cost with respect to an FIR with similar behavior;
- shorter input-output delay;
- compact representation.

While their disadvantages are mainly:

- stability is not guaranteed;
- phase response is difficult to control;
- design is complex in the general case;
- sensitive to numerical precision.

For these reasons, in this book, we will concentrate mostly on the FIR design problem and we will consider of IIR filters only in conjunction with some specific processing tasks which are often encountered in practice.

7.1.2 Filter Specifications and Tradeoffs

A set of filter specifications represents a set of guidelines for the design of a realizable filter. Generally, the specifications are formulated in the frequency domain and are cast in the form of boundaries for the magnitude of the frequency response; less frequently, the specifications will take the phase response into account as well.

A set of filter specifications is best illustrated by example: suppose our goal is to design a half-band lowpass filter, i.e. a lowpass filter with cutoff frequency $\pi/2$. The filter will possess a *passband*, i.e. a frequency range over which the input signal is unaffected, and a *stopband*, i.e. a frequency range where the input signal is annihilated. In order to turn these requirements into specifications the following practical issues must be taken into account:

- **Transition band.** The range of frequencies between passband and stopband is called the *transition band*. We should know by now (and we shall see again shortly) that we cannot obtain an instantaneous

transition in a realizable filter[1]. Therefore, we must be willing to allow for a gap between passband and stopband where we renounce control over the frequency response; suppose we estimate that we can tolerate a transition band width up to 20% of the total bandwidth: since the cutoff is supposed to be at 0.5π, the transition band will thus extend from 0.4π to 0.6π.

- **Tolerances.** Similarly, we cannot impose a strict value of 1 for the passband and a value of 0 for the stopband (again, this has to do with the fact that the rational transfer function, being analytical, cannot be a constant over an interval). So we must allow for *tolerances*, i.e. minimum and maximum values for the frequency response over passband and stopband (while, in the transition band, we don't care). In our example, suppose that after examining the filter usage scenario we decide we can afford a 10% error in the passband and a 30% error in the stopband. (Note that these are *huge* tolerances, but they make the plots easier to read).

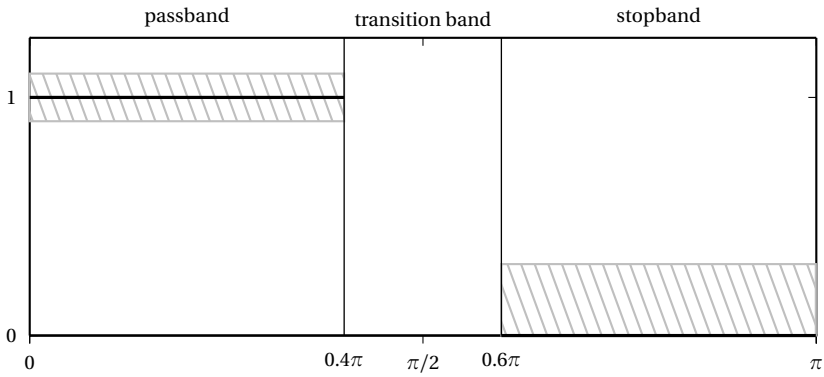


Figure 7.1 Typical lowpass filter specifications.

These specifications can be represented graphically as in Figure 7.1; note that, since we are dealing with real-valued filter coefficients, it is sufficient to specify the desired frequency response only over the $[0,\pi]$ interval, the magnitude response being symmetric. The filter design problem consists now in finding the minimum size FIR or IIR filter which fulfills the required

[1] To get an initial intuition as to why this is, consider that an instantaneous (vertical) transition constitutes a jump discontinuity in the frequency response. But the frequency response is just the transfer function computed on the unit circle and, for a realizable filter, the transfer function is a rational function which must be continuous.

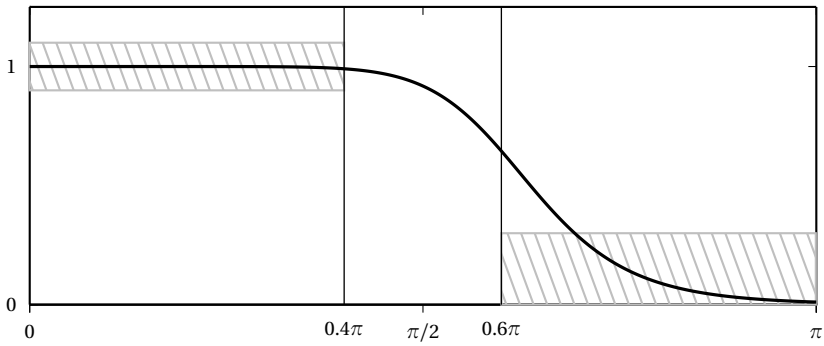


Figure 7.2 Example of monotonic filter which does not satisfies the specifications.

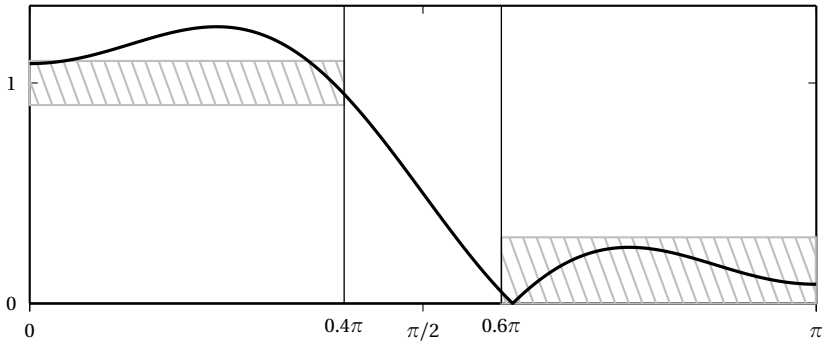


Figure 7.3 Example of FIR filter which does not satisfies the specifications.

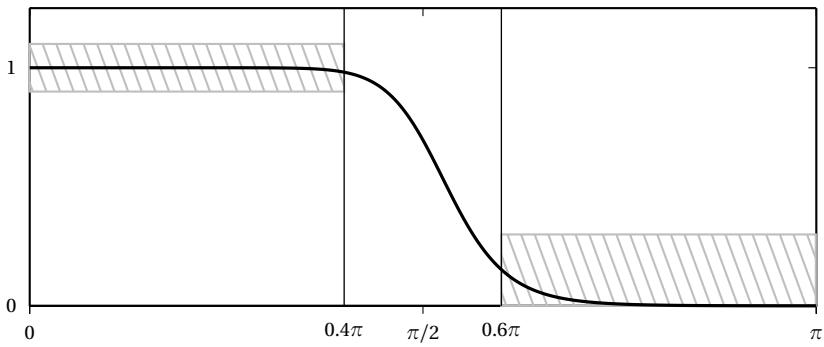


Figure 7.4 Example of monotonic filter which satisfies (and exceeds) the specifications.

specifications. As an example, Figure 7.2 shows an IIR filter which does not fulfill the specifications since the stopband error is above the maximum error at the beginning of the stopband. Similarly, Figure 7.3 shows an FIR filter which breaks the specifications in the passband. Finally, Figure 7.4 shows a monotonic IIR filter which matches and exceeds the specifications (i.e. the error is always smaller than the maximum error).

7.2 FIR Filter Design

In this section we will explore two fundamental strategies for FIR filter design, the window method and the minimax (or Parks-McClellan) method. Both methods seek to minimize the error between a desired (and often ideal) filter transfer function and the transfer function of the designed filter; they differ in the error measure which is used in the minimization. The window method is completely straightforward and it is often used for quick designs. The minimax method, on the other hand, is the procedure of choice for accurate, optimal filters. Both methods will be illustrated with respect to the design of a lowpass filter.

7.2.1 FIR Filter Design by Windowing

We have already seen in Section 5.6 that if there are no constraints (not even realizability) the best lowpass filter with cutoff frequency ω_c is the ideal lowpass. The impulse response is therefore the inverse Fourier transform of the desired transfer function:

$$\begin{aligned} h[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega})e^{j\omega n}\,d\omega \\ &= \frac{1}{2\pi}\int_{-w_c}^{w_c} e^{j\omega n}\,d\omega \\ &= \frac{1}{2\pi j n}\left[e^{j\omega_c n} - e^{-j\omega_c n}\right] \\ &= \frac{\sin(\omega_c n)}{\pi n} \\ &= \frac{\omega_c}{\pi}\,\mathrm{sinc}\left(\frac{\omega_c}{\pi}\,n\right) \end{aligned}$$

The resulting filter, as we saw, is an ideal filter and it cannot be represented by a rational transfer function with a finite number of coefficients.

Impulse Response Truncation. A simple idea to obtain a realizable filter is to take a finite number of samples from the ideal impulse response and use them as coefficients of a (possibly rather long) FIR filter:[2]

$$\hat{h}[n] = \begin{cases} h[n] & -N \le n \le N \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

This is a $(2N+1)$-tap FIR obtained by truncating an ideal impulse response (Figs 5.10 and 5.11). Note that the filter is noncausal, but that it can be made causal by using an N-tap delay; it is usually easier to design FIR's by considering a noncausal version first, especially if the resulting impulse response is symmetric (or antisymmetric) around $n = 0$. Although this approximation was derived in a sort of "intuitive" way, it actually satisfies a very precise approximation criterion, namely the minimization of the mean square error (MSE) between the original and approximated filters. Denote by E_2 this error, that is

$$E_2 = \int_{-\pi}^{\pi} \left| H(e^{j\omega}) - \hat{H}(e^{j\omega}) \right|^2 d\omega$$

We can apply Parseval's theorem (see (4.59)) to obtain the equivalent expression in the discrete-time domain:

$$E_2 = 2\pi \sum_{n \in \mathbb{Z}} \left| h[n] - \hat{h}[n] \right|^2$$

If we now recall that $\hat{h}[n] = 0$ for $|n| > N$, we have

$$E_2 = 2\pi \left[\sum_{n=-N}^{N} \left| h[n] - \hat{h}[n] \right|^2 + \sum_{n=N+1}^{\infty} \left| h[n] \right|^2 + \sum_{n=-\infty}^{-N-1} \left| h[n] \right|^2 \right]$$

Obviously the last two terms are nonnegative and independent of $\hat{h}[n]$. Therefore, the minimization of E_2 with respect to $\hat{h}[n]$ is equivalent to the minimization of the first term only, and this is easily obtained by letting

$$\hat{h}[n] = h[n], \qquad \text{for } n = -N, \ldots, N$$

In spite of the attractiveness of such a simple and intuitive solution, there is a major drawback. If we consider the frequency response of the approximated filter, we have

$$\hat{H}(e^{j\omega}) = \sum_{n=-N}^{N} h[n]\, e^{-j\omega}$$

[2] Here and in the following the "hat" notation will denote an approximated or otherwise *derived* filter.

which means that $\hat{H}(e^{j\omega})$ is an approximation of $H(e^{j\omega})$ obtained by using only $2N+1$ Fourier coefficients. Since $H(e^{j\omega})$ has a jump discontinuity in ω_c, $\hat{H}(e^{j\omega})$ incurs the well-known Gibbs phenomenon around ω_c. The Gibbs phenomenon states that, when approximating a discontinuous function with a finite number of Fourier coefficients, the maximum error in an interval around the jump discontinuity is actually independent of the number of terms in the approximation and is always equal to roughly 9% of the jump. In other words, we have no control over the maximum error in the magnitude response. This is apparent in Figure 7.5 where $\left|\hat{H}(e^{j\omega})\right|$ is plotted for increasing values of N; the maximum error does not decrease with increasing N and, therefore, there are no means to meet a set of specifications which require less than 9% error in either stopband or passband.

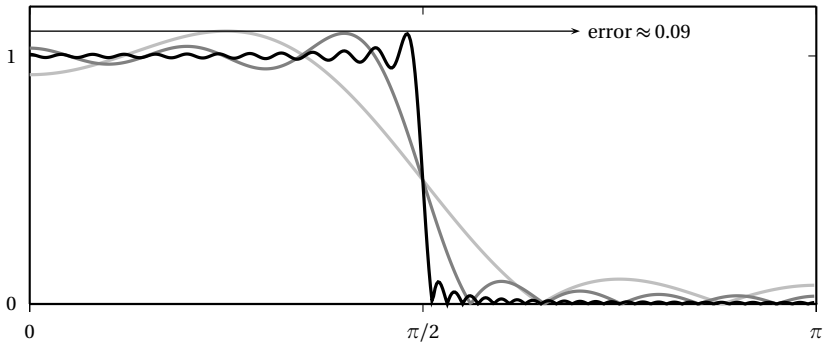


Figure 7.5 Gibbs phenomenon in lowpass approximation; magnitude of the approximated lowpass filter for $N = 4$ (light gray), $N = 10$ (dark gray) and $N = 50$ (black).

The Rectangular Window. Another way to look at the resulting approximation is to express $\hat{h}[n]$ as

$$\hat{h}[n] = h[n]w[n] \tag{7.2}$$

with

$$w[n] = \text{rect}\left(\frac{n}{2N}\right) = \begin{cases} 1 & -N \le n \le N \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

$w[n]$ is called a rectangular *window* of length $(2N+1)$ taps, which in this case is centered at $n = 0$.

We know from the modulation theorem in (5.22) that the Fourier transform of (7.2) is the convolution (in the space of 2π-periodic functions) of the Fourier transforms of $h[n]$ and $w[n]$:

$$\hat{H}(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\theta})W(e^{j(\omega-\theta)})\,d\theta$$

It is easy to compute $W(e^{j\omega})$ as

$$W(e^{j\omega}) = \sum_{n=-N}^{N} e^{-j\omega n} = \frac{\sin\left(\omega\left(N+\frac{1}{2}\right)\right)}{\sin\left(\frac{\omega}{2}\right)} \tag{7.4}$$

An example of $W(e^{j\omega})$ for $N = 6$ is shown in Figure 7.6. By analyzing the form of $W(e^{j\omega})$ for arbitrary N, we can determine that:

- the first zero crossing of $W(e^{j\omega})$ occurs at $\omega = 2\pi/(2N+1)$;
- the width of the main lobe of the magnitude response is $\Delta = 4\pi/(2N+1)$;
- there are multiple *sidelobes*, an oscillatory effect around the main lobe and there are up to $2N$ sidelobes for a $2N+1$-tap window.

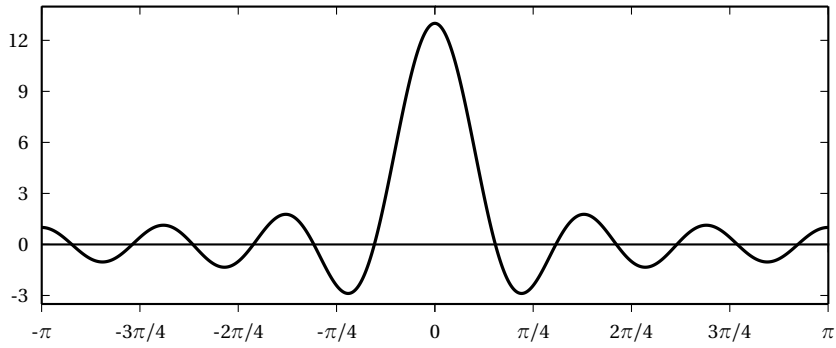


Figure 7.6 Fourier transform of the rectangular window for $N = 6$.

In order to understand the shape of the approximated filter, let us go back to the lowpass filter example and try to visualize the effect of the convolution in the Fourier transform domain. First of all, since all functions are 2π-periodic, everything happens circularly, i.e. what "goes out" on the right of the $[-\pi,\pi]$ interval "pops" immediately up on the left. The value at ω_0 of $\hat{H}(e^{j\omega})$ is the integral of the product between $H(e^{j\omega})$ and a version of

$W(e^{j\omega})$ circularly shifted by ω_0. Since $H(e^{j\omega})$ is zero except over $[-\omega_c, \omega_c]$, where it is one, this value is actually:

$$\hat{H}(e^{j\omega_0}) = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} W(e^{j(\omega_0-\theta)})d\theta$$

When ω_0 is such that the first right sidelobe of $W(e^{j\omega})$ is *outside* of the $[-\omega_c, \omega_c]$ interval, then the integral reaches its maximum value, since the sidelobe is negative and it's the largest. The maximum value is dependent on the shape of the window (a rectangle in this case) but not on its length. Hence the Gibbs phenomenon.

To recap, the windowing operation on the ideal impulse response, i.e. the circular convolution of the ideal frequency response with $W(e^{j\omega})$, produces two main effects:

- The sharp transition from passband to stopband is smoothed by the convolution with the main lobe of width Δ.
- Ripples appear both in the stopband and the passband due to the convolution with the sidelobes (the largest ripple being the Gibbs phenomenon).

The sharpness of the transition band and the size of the ripples are dependent on the shape of the window's Fourier transform; indeed, by carefully designing the shape of the windowing sequence we can trade mainlobe width for sidelobe amplitude and obtain a more controlled behavior in the frequency response of the approximation filter (although the maximum error can *never* be arbitrarily reduced).

Other Windows. In general, the recipe for filter design by windowing involves two steps: the analytical derivation of an ideal impulse response followed by a suitable windowing to obtain an FIR filter. The ideal impulse response $h[n]$ is obtained from the desired frequency response $H(e^{j\omega})$ by the usual DTFT inversion formula

$$h[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega})e^{j\omega n}\,d\omega$$

While the analytical evaluation of the above integral may be difficult or impossible in the general case, for frequency responses $H(e^{j\omega})$ which are *piecewise linear*, the computation of $h[n]$ can be carried out in an exact (if nontrivial) way; the result will be a linear combination of modulated sinc and sinc-squared sequences.[3] The FIR approximation is then obtained by applying a finite-length window $w[n]$ to the ideal impulse response as in (7.2).

[3] For more details one can look at the Matlab `fir1` function.

The shape of the window can of course be more sophisticated than the simple rectangular window we have just encountered and, in fact, a hefty body of literature is devoted to the design of the "best" possible window. In general, a window should be designed with the following goals in mind:

- the window should be short, as to minimize the length of the FIR and therefore its computational cost;
- the spectrum of the window should be concentrated in frequency around zero as to minimize the "smearing" of the original frequency response; in other words, the window's main lobe should be as narrow as possible (it is clear that for $W(e^{j\omega}) = \delta(\omega)$ the resulting frequency response is identical to the original);
- the unavoidable sidelobes of the window's spectrum should be small, so as to minimize the rippling effect in the resulting frequency response (Gibbs phenomenon).

It is clear that the first two requirements are openly in conflict; indeed, the width of the main lobe Δ is inversely proportional to the length of the window (we have seen, for instance, that for the rectangular window $\Delta = 4\pi/M$, with M, the length of the filter). The second and third requirements are also in conflict, although the relationship between mainlobe width and sidelobe amplitude is not straightforward and can be considered a design parameter. In the frequency response, reduction of the sidelobe amplitude implies that the Gibbs phenomenon is decreased, but at the "price" of an enlargement of the filter's transition band. While a rigorous proof of this fact is beyond the scope of this book, consider the simple example of a triangular window (with N odd):

$$w_t[n] = w[n] = \begin{cases} \dfrac{N-n}{N} & |n| < N \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

It is easy to verify that $w_t[n] = w[n] * w[n]$, with $w[n] = \text{rect}\big(2n/(N-1)\big)$ (i.e. the triangle can be obtained as the convolution of a half-support rectangle with itself) so that, as a consequence of the convolution theorem, we have

$$W_t(e^{j\omega}) = W(e^{j\omega})W(e^{j\omega}) = \left[\frac{\sin(\omega N/2)}{\sin(\omega/2)}\right]^2 \tag{7.6}$$

The net result is that the amplitude of the sidelobes is quadratically reduced but the amplitude of the mainlobe Δ is roughly doubled with respect to an

equivalent-length rectangular window; this is displayed in Figure 7.7 for a 17-point window (values are normalized so that both frequency responses are equal in $\omega = 0$). Filters designed with a triangular window therefore exhibit a much wider transition band.

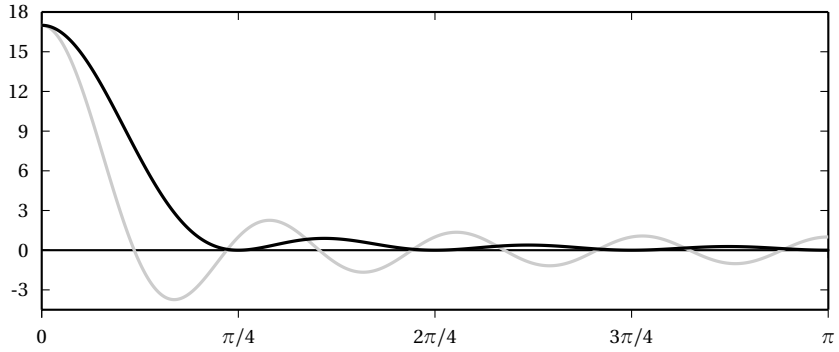


Figure 7.7 Fourier transform of the 17-point rectangular window (gray) vs. an equal-length triangular window (black).

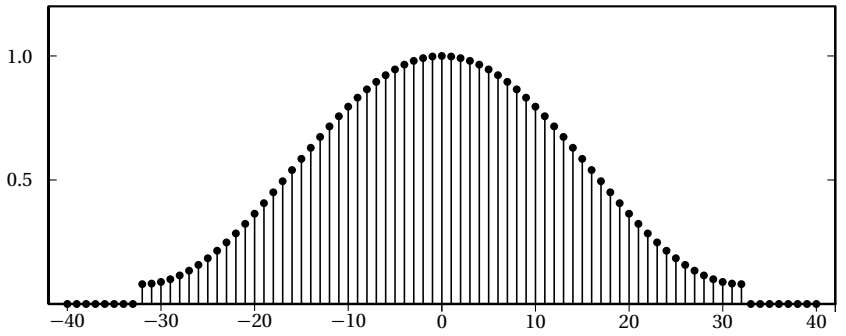


Figure 7.8 Hamming window ($N = 32$).

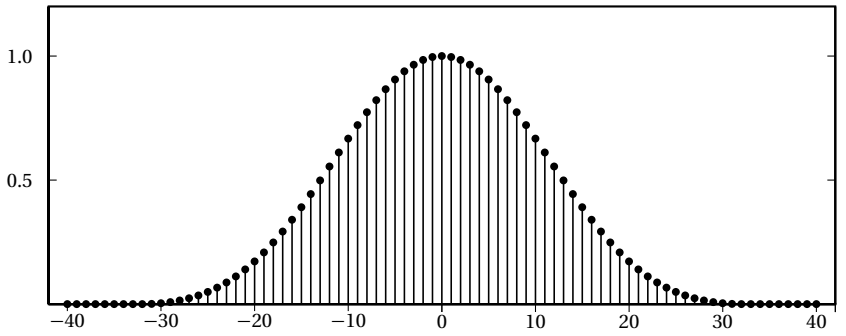


Figure 7.9 Blackman window ($N = 32$).

Other commonly used windows admit a simple parametric closed form representation; the most important are the Hamming window (Fig. 7.8):

$$w(n) = 0.54 - 0.46\cos\left(2\pi\frac{n+N}{2N}\right), \qquad |n| \leq N-1$$

and the Blackman window (Fig. 7.9):

$$w(n) = 0.42 - 0.5\cos\left(2\pi\frac{n+N}{2N}\right) + 0.08\cos\left(4\pi\frac{n+N}{2N}\right), \quad |n| \leq N-1$$

The magnitude response of both windows is plotted in Figure 7.10 (on a log scale so as to enhance the difference in sidelobe amplitude); again, we can remark the tradeoff between mainlobe width (translating to a wider transition band in the designed filter) and sidelobe amplitude (influencing the maximum error in passband and stopband).

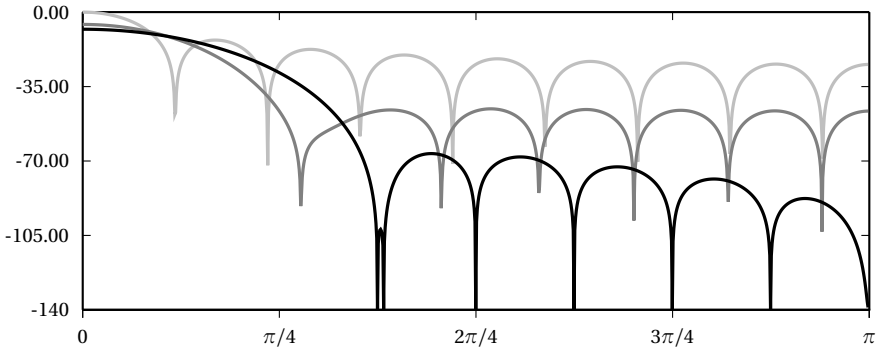


Figure 7.10 Magnitude response (dB scale) of the 17-point rectangular (light gray), Hamming (dark gray) and Blackman (black) windows.

Limitations of the Window Method. Lack of total control on passband and stopband error is the main limitation inherent to the window method; this said, the method remains a fundamental staple of practical signal processing as it yields perfectly usable filters via a quick, flexible and simple procedure. The error characteristic of a window-designed filter can be particularly aggravating in sensitive applications such as audio processing, where the peak in the stopband error can introduce unacceptable artifacts. In order to improve on the filter performance, we need to completely revise our design approach. A more suitable optimization criterion may, for instance, be the *minimax* criterion, where we aim to explicitly minimize the *maximum* approximation error over the entire frequency support; this is

thoroughly analyzed in the next section. We can already say, however, that while the minimum square error is an integral criterion, the minimax is a pointwise criterion; or, mathematically, that the MSE and the minimax are respectively $L_2([-\pi,\pi])$- and $L_\infty([-\pi,\pi])$-norm minimizations for the error function $E(\omega)=\hat{H}(e^{j\omega})-H(e^{j\omega})$. Figure 7.11 illustrates the typical result of applying both criteria to the ideal lowpass problem. As can be seen, the minimum square and minimax solutions are very different.

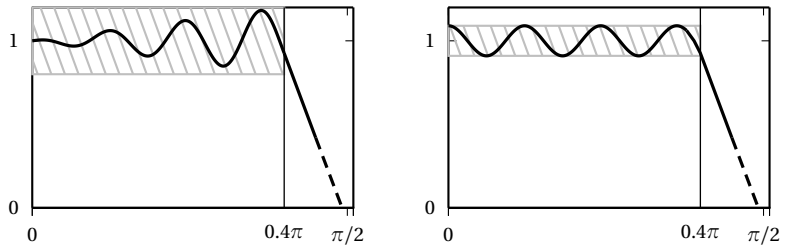


Figure 7.11 Error shapes in passband for MSE and minimax optimization methods.

7.2.2 Minimax FIR Filter Design

As we saw in the opening example, FIR filter design by windowing minimizes the overall mean square error between the desired frequency response and the actual response of the filter. Since this might lead to a very large error at frequencies near the transition band, we now consider a different approach, namely the design by minimax optimization. This technique minimizes the maximum allowable error in the filter's magnitude response, both in the passband and in the stopband. Optimality in the minimax sense requires therefore the explicit stating of a set of *tolerances* in the prototypical frequency response, in the form of design specifications as seen in Section 7.1.2. Before tackling the design procedure itself, we will need a series of intermediate results.

Generalized Linear Phase. In Section 5.4.3, we introduced the concept of linear phase; a filter with linear phase response is particularly desirable since the phase response translates to just a time delay (possibly fractional) and we can concentrate on the magnitude response only. We also introduced the notion of group delay and showed that linear phase corresponds to constant group delay. Clearly, the converse is not true: a frequency response of the type

$$H(e^{j\omega})=\left|H(e^{j\omega})\right|e^{-j\omega d+j\alpha}$$

has constant group delay but differs from a linear phase system by a constant phase factor $e^{j\alpha}$. We will call this type of phase response *generalized linear phase*. Important cases are those for which $\alpha = 0$ (strictly linear phase) and $\alpha = \pi/2$ (generalized linear phase used in differentiators).

FIR Filter Types. Consider a causal, M-tap FIR filter with impulse response $h[n]$, $n = 0, 1, \ldots, M-1$; in the following, we are interested in filters whose impulse response is *symmetric or antisymmetric around the "midpoint"*. If the number of taps is odd, the midpoint of the impulse response coincides with the center tap $h[(M-1)/2]$; if the number of taps is even, on the other hand, the midpoint is still at $(M-1)/2$ but this value does not coincide with a tap since it is located "right in between" taps $h[M/2-1]$ and $h[M/2]$. Symmetric and antisymmetric FIR filters are important since their frequency response has generalized linear phase. The delay introduced by these filters is equal to $(M-1)/2$ samples; clearly, this is an integer delay if M is odd, and it is fractional (half a sample more) if M is even. There are four different possibilities for linear phase FIR impulse responses, which are listed here with their corresponding generalized linear phase parameters :

Type	Nb. of Taps	Symmetry	Delay	Phase
Type I	odd	symmetric	integer	$\alpha = 0$
Type II	even	symmetric	fractional	$\alpha = 0$
Type III	odd	antisymmetric	integer	$\alpha = \pi/2$
Type IV	even	antisymmetric	fractional	$\alpha = \pi/2$

The generalized linear phase of (anti)symmetric FIRs is easily shown. Consider for instance a Type I filter, and define $C = (M-1)/2$, the location of the center tap; we can compute the transfer function of the shifted impulse response $h_d[n] = h[n+C]$, which is now symmetric around zero. i.e. $h_d[-n] = h_d[n]$:

$$\begin{aligned} H_d(z) &= \sum_{n=-C}^{C} h_d[n] z^{-n} \\ &= h_d[0] + \sum_{n=-C}^{-1} h_d[n] z^{-n} + \sum_{n=1}^{C} h_d[n] z^{-n} \\ &= h_d[0] + \sum_{n=1}^{C} h_d[n] (z^n + z^{-n}) \end{aligned} \tag{7.7}$$

By undoing the time shift we obtain the original Type I transfer function:

$$H(z) = z^{-\frac{M-1}{2}} H_d(z) \tag{7.8}$$

On the unit circle (7.7) becomes

$$H_d(e^{j\omega}) = h_d[0] + \sum_{n=1}^{C} h_d[n](e^{j\omega n} + e^{-j\omega n})$$
$$= h_d[0] + 2\sum_{n=1}^{C} h_d[n]\cos n\omega \tag{7.9}$$

which is a *real* function; the original Type I frequency response is obtained from (7.8):

$$H(e^{j\omega}) = \left[h\left[\frac{M-1}{2}\right] + 2\sum_{n=(M+1)/2}^{M-1} h[n]\cos n\omega \right] e^{-j\omega\frac{M-1}{2}}$$

which is clearly linear phase with delay $d = (M-1)/2$ and $\alpha = 0$. The generalized linear phase of the other three FIR types can be shown in exactly the same way.

Zero Locations. The symmetric structures of the four types of FIR filters impose some constraints on the locations of the zeros of the transfer function. Consider again a Type I filter; from (7.7) it is easy to see that $H_d(z^{-1}) = H_d(z)$; by using (7.8) we therefore have

$$\begin{cases} H(z) = z^{-\frac{M-1}{2}} H_d(z) \\ H(z^{-1}) = z^{\frac{M-1}{2}} H_d(z) \end{cases}$$

which leads to

$$H(z^{-1}) = z^{M-1} H(z) \tag{7.10}$$

It is easy to show that the above relation is also valid for Type II filters, while for Type III and Type IV (antisymmetric filters) we have

$$H(z^{-1}) = -z^{M-1} H(z) \tag{7.11}$$

These relations mean that if z_0 is a zero of a linear phase FIR, then so is z_0^{-1}. This result, coupled with the usual fact that all complex zeros come in conjugate pairs, implies that if z_0 is a zero of $H(z)$, then:

- If $z_0 = \rho \in \mathbb{R}$ then ρ and $1/\rho$ are zeros.
- If $z_0 = \rho e^{j\theta}$ then $\rho e^{j\theta}$, $(1/\rho)e^{j\theta}$, $\rho e^{-j\theta}$ and $(1/\rho)e^{-j\theta}$ are zeros.

Consider now equation (7.10) again; if we set $z = -1$,

$$H(-1) = (-1)^{M-1} H(-1) \tag{7.12}$$

for Type II filters, $M-1$ is an odd number, which leads to the conclusion that $H(-1)=0$; in other words, Type II filters *must* have a zero at $\omega=\pi$. Similar results can be demonstrated for the other filter types, and they are summarized below:

Filter Type	Relation	Constraint on Zeros
Type I	$H(z^{-1})=z^{M-1}H(z)$	No constraints
Type II	$H(z^{-1})=z^{M-1}H(z)$	Zero at $z=-1$ (i.e. $\omega=\pi$)
Type III	$H(z^{-1})=-z^{M-1}H(z)$	Zeros at $z=\pm 1$ (i.e. at $\omega=0$, $\omega=\pi$)
Type IV	$H(z^{-1})=-z^{M-1}H(z)$	Zero at $z=1$ (i.e. $\omega=0$)

These constraints are important in the choice of the filter type for a given set of specifications. Type II and Type III filters are not suitable in the design of highpass filters, for instance; similarly, Type III and Type IV filters are not suitable in in the design of lowpass filters.

Chebyshev Polynomials. Chebyshev polynomials are a family of orthogonal polynomials $\{T_k(x)\}_{k\in\mathbb{N}}$ which have, amongst others, the following interesting property:

$$\cos n\omega = T_n(\cos\omega) \tag{7.13}$$

in other words, the cosine of an integer multiple of an angle ω can be expressed as a polynomial in the variable $\cos\omega$. The first few Chebyshev polynomials are

$$\begin{aligned}
T_0(x) &= 1\\
T_1(x) &= x\\
T_2(x) &= 2x^2-1\\
T_3(x) &= 4x^3-3x\\
T_4(x) &= 8x^4-8x^2+1
\end{aligned}$$

and, in general, they can be derived from the recursion formula:

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \tag{7.14}$$

From the above table it is easy to see that we can write, for instance,

$$\cos(3\omega) = 4\cos^3\omega - 3\cos\omega$$

The interest in Chebyshev polynomials comes from the fact that the zero-centered frequency response of a linear phase FIR can be expressed as a linear combination of cosine functions, as we have seen in detail for Type I

filters in (7.9). By using Chebyshev polynomials we can rewrite such a response as just one big polynomial in the variable $\cos\omega$. Let us consider an explicit example for a length-7 Type I filter with nonzero coefficients $h[n] = [d\ c\ b\ a\ b\ c\ d]$; we can state that

$$H_d(e^{j\omega}) = a + 2b\cos\omega + 2c\cos 2\omega + 2d\cos 3\omega$$

and by using the first four Chebyshev polynomials we can write

$$\begin{aligned} H_d(e^{j\omega}) &= a + 2b\cos\omega + 2c(2\cos^2\omega - 1) + 2d(4\cos^3\omega - 3\cos\omega) \\ &= (a - 2c) + (2b - 6d)\cos\omega + 4c\cos^2\omega + 8d\cos^3\omega \end{aligned} \tag{7.15}$$

In this case, $H_d(e^{j\omega})$ turns out to be a third degree polynomial in the variable $\cos\omega$. This is the case for any Type I filter, for which we can always write

$$H_d(e^{j\omega}) = \sum_{k=0}^{(M-1)/2} c_k \cos^k\omega \tag{7.16}$$

$$= P(x)\Big|_{x=\cos\omega} \tag{7.17}$$

where $P(x)$ is a polynomial of degree $(M-1)/2$ whose coefficients c_k are derived as linear combinations of the original filter coefficients a_k as illustrated in (7.15). For the other types of linear phase FIR, a similar representation can be obtained after a few trigonometric manipulations. The general expression is

$$\begin{aligned} H_d(e^{j\omega}) &= f(\omega)\sum_{k=0}^{L} c_k \cos^k\omega \\ &= f(\omega)P(x)\Big|_{x=\cos\omega} \end{aligned} \tag{7.18}$$

where the c_k are still linear combinations of the original filter coefficients and where $f(\omega)$ is a weighting trigonometric function. Both $f(\omega)$ and the polynomial degree L vary as a function of the filter type.[4] In the following Sections, however, we will concentrate only on the design of Type I filters, so these details will be overlooked; in practice, since the design is always

[4] For the sake of completeness, here is a summary of the details:

Filter Type	L	$f(\omega)$
Type I	$(M-1)/2$	1
Type II	$(M-2)/2$	$\cos(\omega/2)$
Type III	$(M-3)/2$	$\sin(\omega)$
Type IV	$(M-2)/2$	$\sin(\omega/2)$

carried out using numerical packages, the appropriate formulation for the filter expression is taken care of automatically.

Polynomial Optimization. Going back to the filter design problem, we stipulate that the FIR filters are (generalized) linear phase, so we can concentrate on the real frequency response of the zero-centered filter, which is represented by the trigonometric polynomial (7.18). Moreover, since the impulse response is real and symmetric, the aforementioned real frequency response is also symmetric around $\omega = 0$. The filter design procedure can thus be carried out only for values of ω over the interval $[0, \pi]$, with the other half of the spectrum obtained by symmetry. For these values of ω, the variable $x = \cos\omega$ is mapped onto the interval $[1, -1]$ and the mapping is invertible. Therefore, *the filter design problem becomes a problem of polynomial approximation over intervals*.

To illustrate the procedure by example, consider once more the set of filter specifications in Figure 7.1 and suppose we decide to use a Type I filter. Recall that we required the prototype filter to be lowpass, with a transition band from $\omega_p = 0.4\pi$ to $\omega_s = 0.6\pi$; we further stated that the tolerances for the realized filter's magnitude must not exceed 10% in the passband and 1% in the stopband. This implies that the maximum magnitude error between the prototype filter and the FIR filter response $H(e^{j\omega})$ must not exceed $\delta_p = 0.1$ in the interval $[0, \omega_p]$ and must not exceed $\delta_s = 0.01$ in the interval $[\omega_s, \pi]$. We can formulate this as follows: the frequency response of the desired filter is

$$H_D(e^{j\omega}) = \begin{cases} 1 & \omega \in [0, \omega_p] \\ 0 & \omega \in [\omega_s, \pi] \end{cases}$$

(note that $H_D(e^{j\omega})$ is not specified in the transition band). Since the tolerances on passband and stopband are different, they can be expressed in terms of a weighting function $H_W(\omega)$ such that the tolerance on the error is constant over the two bands:

$$H_W(\omega) = \begin{cases} 1 & \omega \in [0, \omega_p] \\ \dfrac{\delta_p}{\delta_s} = 10 & \omega \in [\omega_s, \pi] \end{cases} \tag{7.19}$$

With this notation, the filter specifications amount to the following:

$$\max_{\omega \in [0,\omega_p] \cup [\omega_s,\pi]} \left\{ H_W(\omega) \left| H_d(e^{j\omega}) - H_D(e^{j\omega}) \right| \right\} \leq \delta_p = 0.1 \tag{7.20}$$

and the question now is to find the coefficients for $h[n]$ (their number M and their values) which minimize the above error. Note that we leave the

transition band unconstrained (i.e. it does not affect the minimization of the error).

The next step is to use (7.18) to reformulate the above expression as a polynomial optimization problem. To do so, we replace the frequency response $H_d(e^{j\omega})$ with its polynomial equivalent and set $x = \cos\omega$; the passband interval $[0, \omega_p]$ and the stopband interval $[\omega_s, \pi]$ are mapped into the intervals for x:

$$I_p = [\cos\omega_p, 1]$$
$$I_s = [-1, \cos\omega_s]$$

respectively; similarly, the desired response becomes:

$$D(x) = \begin{cases} 1 & \omega \in I_p \\ 0 & \omega \in I_s \end{cases} \tag{7.21}$$

and the weighting function becomes:

$$W(x) = \begin{cases} 1 & \omega \in I_p \\ \delta_p/\delta_s & \omega \in I_s \end{cases} \tag{7.22}$$

The new set of specifications are shown in Figure 7.12. Within this polynomial formulation, the optimization problem becomes:

$$\max_{x \in I_p \cup I_s} \left\{ W(x) \left| P(x) - D(x) \right| \right\} = \max \left\{ \left| E(x) \right| \right\} \leq \delta_p \tag{7.23}$$

where $P(x)$ is the polynomial representation of the FIR frequency response as in (7.18).

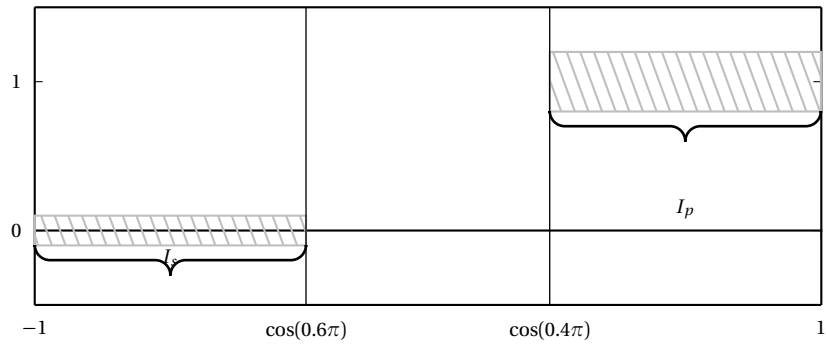


Figure 7.12 Filter specifications as in Figure 7.1 formulated here in terms of polynomial approximation, i.e. for $x = \cos\omega$, $\omega \in [0, \pi]$.

Alternation Theorem. The optimization problem stated by (7.23) can be solved by using the following theorem:

Theorem 7.1 *Consider a set* $\{I_k\}$ *of* closed, *disjoint intervals on the real axis and their union* $I = \bigcup_k I_k$. *Consider further:*

- *a polynomial* $P(x)$ *of degree* L, $P(x) = \sum_{n=0}^{L} a_n x^n$;
- *a desired function* $D(x)$, *continuous over* I;
- *a positive weighting function* $W(x)$.

Consider now the approximation error function

$$E(x) = W(x)\big[D(x) - P(x)\big]$$

and the associated maximum approximation error over the set of closed intervals

$$E_{\max} = \max_{x \in I} \Big\{ \big| E(x) \big| \Big\}$$

Then $P(x)$ *is the* unique *order-*L *polynomial which minimizes* $E_{\max}$ *if and only if there exist* at least $L+2$ *successive values* x_i *in* I *such that* $\big|E(x_i)\big| = E_{\max}$ *and*

$$E(x_i) = -E(x_{i+1})$$

In other words, the error function must have at least $L+2$ *alternations between its maximum and minimum values. Such a function is called* equiripple.

Going back to our lowpass filter example, assume we are trying to design a 9-tap optimal filter. This theorem tells us that if we found a polynomial $P(x)$ of degree 4 such that the error function (7.23) over I_p and I_s as is shown in Figure 7.13 (6 alternations), then the polynomial would be the *optimal*

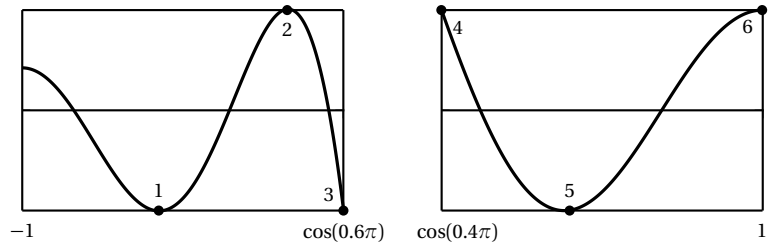


Figure 7.13 Approximation error function $E(x)$ for a 9-tap lowpass prototype; alternations are marked by a dot.

and *unique* solution. Note that the extremal points (i.e. the values of the error function at the edges of the optimization intervals) *do* count in the number of alternations since the intervals I_k are closed.

The above theorem may seem a bit far-fetched since it does not tell us how to find the coefficients but it only gives us a test to verify their optimality. This test, however, is at the core of an *iterative* algorithm which refines the polynomial from an initial guess to the point when the optimality condition is met. Before considering the optimization procedure more in detail, we will state without formal proof, three consequences of the alternation theorem as it applies to the design of Type I lowpass filters:

- The minimum number of alternations for an optimal M-tap lowpass filter is $L+2$, with $L=(M-1)/2$; this is the result of the alternation theorem. The *maximum* number of alternation, however, is $L+3$; filters with $L+3$ alternation are called extraripple filters.
- Alternations always take place at $x=\cos\omega_p$ and $x=\cos\omega_s$ (i.e. at $\omega=\omega_p$ and $\omega=\omega_s$.
- If the error function has a local maximum or minimum, its absolute value at the extremum must be equal to $E_{\max}$ except possibly in $x=-1$ or $x=1$. In other words, all local maxima and minima of the frequency response must be alternating, except in $\omega=0$ or $\omega=\pi$.
- If the filter is extraripple, the extra alternation occurs at either $\omega=0$ or $\omega=\pi$.

Optimization Procedure. Finally, by putting all the elements together, we are ready to state an algorithmic optimization procedure for the design of optimal minimax FIR filters; this procedure is usually called the Parks-McClellan algorithm. Remember, we are trying to determine a polynomial $P(x)$ such that the approximation error in (7.23) is equiripple; for this, we need to determine both the degree of the polynomial and its coefficients. For a given degree L, for which the resulting filter has $2L+1$ taps, the L coefficients are found by an iterative procedure which successively refines an initial guess for the $L+2$ alternation points x_i until the error is equiripple.[5] After the iteration has converged, we need to check that the corresponding

[5] Details about this crucial optimization step can be found in the bibliographic references. While a thorough discussion of the algorithm is beyond the scope of the book, we can mention that at each iteration the new set of candidate extremal points is obtained by exchanging the old set with the ordinates of the current local maxima. This trick is known as the Remez exchange algorithm and that is why, in Matlab, the Parks-McClellan algorithm is named `remez`.

$E_{\max}$ satisfies the upper bound imposed by the specifications; when this is not the case, the degree of the polynomial (and therefore the length of the filter) must be increased and the procedure must be restarted. Once the conditions on the error are satisfied, the filter coefficients can be obtained by inverting the Chebyshev expansion.

As a final note, an initial guess for the number of taps can be obtained using the empirical formula by Kaiser; for an M-tap FIR $h[n]$, $n = 0,\ldots, M-1$:

$$M \simeq \frac{-10\log_{10}(\delta_p\delta_s) - 13}{2.324\Omega} + 1$$

where δ_p is the passband tolerance, δ_s is the stopband tolerance and $\Omega = \omega_s - \omega_p$ is the width of the transition band.

The Final Design. We now summarize the design steps for the specifications in Figure 7.1. We use a Type I FIR. We start by using Kaiser's formula to obtain an estimate of the number of taps: since $\delta_p\delta_s = 10^{-3}$ and $\Omega = 0.2\,\pi$, we obtain $M = 12.6$ which we round up to 13 taps. At this point we can use any numerical package for filter design to run the Parks-McClellan algorithm. In Matlab this would be

```
[h, err] = remez(12, [0 0.4 0.6 1], [1 1 0 0], [1 10]);
```

The resulting frequency response is plotted in Figure 7.14; please note that we are plotting the frequency responses of the zero-centered filter $h_d[n]$, which is a real function of ω. We can verify that the filter has indeed $(M-1)/2 = 6$ alternation by looking at enlarged picture of the passband and the stopband, as in Figure 7.15. The maximum error as returned by Matlab is however 0.102 which is larger than what our specifications called for,

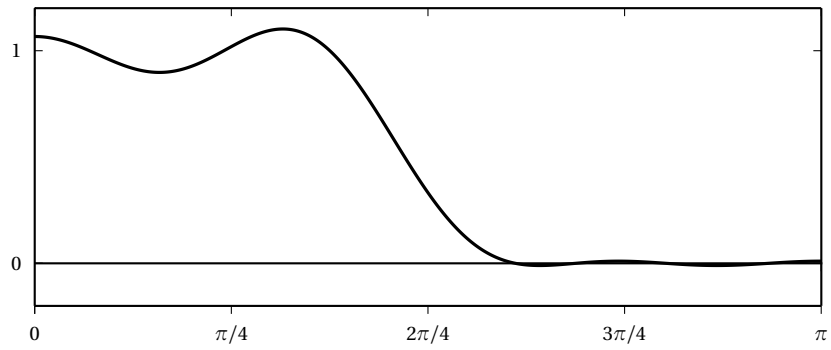


Figure 7.14 An optimal 13-tap Type I filter which does not meet the error specifications.

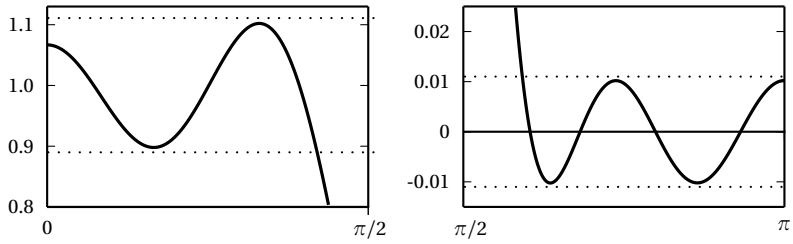


Figure 7.15 Details of passband and stopband of the frequency response in Figure 7.14.

i.e. 0.1. We are thus forced to increase the number of taps; since we are using a Type I filter, the next choice is $M = 15$. Again, the error turns out to be larger than 0.1, since in this case we have $E_{\max} = 0.1006$. The next choice, $M = 17$, finally yields an error $E_{\max} = 0.05$, which exceeds the specifications by a factor of 2. It is the designer's choice to decide whether the computational gains of a shorter filter ($M = 15$) outweigh the small excess error. The impulse response and the frequency response of the 17-tap filter are plotted in Figure 7.16 and Figure 7.17. Figure 7.18 shows the zero locations for the filter; note the typical linear-phase zero pattern as well as the zeros on the unit circle in the stopband.

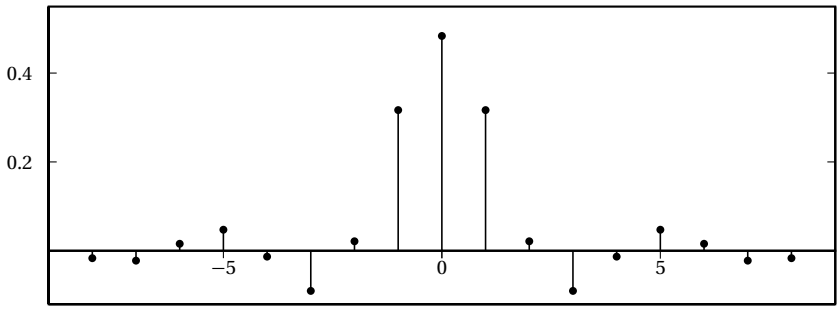


Figure 7.16 Impulse response of the 17-tap filter meeting the specifications.

Other Types of Filters. The Parks-McClellan optimal FIR design procedure can be made to work for arbitrary filter types as well, such as highpass and bandpass, but also for more sophisticated frequency responses. The constraints imposed by the zero locations as we saw on page 181 determine the type of filter to use; once the desired response $H_D(e^{j\omega})$ is expressed as a trigonometric function, the optimization algorithm can take its course. For arbitrary frequency responses, however, the fact that the transition bands are left unconstrained may lead to unacceptable peaks which render the

filter useless. In these cases, visual inspection of the obtained response is mandatory and experimentation with different filter lengths and tolerance may improve the final result.

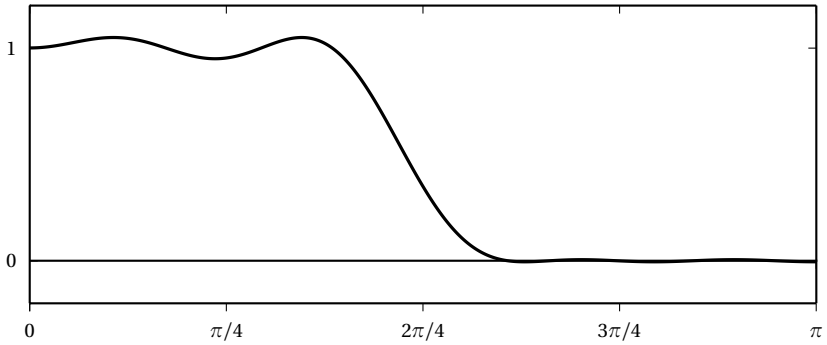


Figure 7.17 Frequency response of the 17-tap filter meeting the specifications.

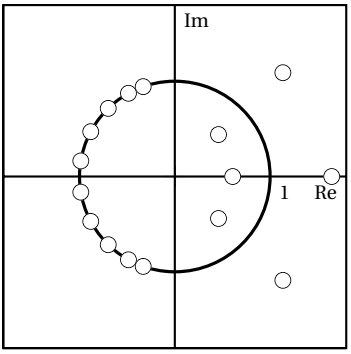


Figure 7.18 Pole-zero plot for the equiripple FIR in Figure 7.17.

7.3 IIR Filter Design

As we mentioned earlier, no optimal procedure exists for the design of IIR filters. The fundamental reason is that the optimization of the coefficients of a rational transfer function is a highly nonlinear problem and no satisfactory algorithm has yet been developed for the task. This, coupled with the impossibility of obtaining an IIR with linear phase response[(6)] makes the design of the IIR filter a much less formalized art. Many IIR designed

[(6)] It can be proved rigorously that an infinite impulse response with linear phase is necessarily not realizable – think of a sinc filter, for instance.

techniques are described in the literature and their origin is usually in tried-and- true analog filter design methods. In the early days of digital signal processing, engineers would own voluminous books with exhaustive lists of capacitance and inductance values to be used for a given set of (analog) filter specifications. The idea behind most digital IIR filter design techniques was to be able to make use of that body of knowledge and to devise formulas which would translate the analog design into a digital one. The most common such method is known as *bilinear transformation*. Today, the formal step through an analog prototype has become unnecessary and numerical tools such as Matlab can provide a variety of routines to design an IIR.

Here we concentrate only on some basic IIR filters which are very simple and which are commonly used in practice.

7.3.1 All-Time Classics

There are a few applications in which simple IIR structures are the design of choice. These filters are so simple and so well behaved that they are a fundamental tool in the arsenal of any signal processing engineer.

DC Removal and Mean Estimation. The DC component of a signal is its mean value; a signal with zero mean is also called an AC signal. This nomenclature comes from electrical circuit parlance: DC is shorthand for *direct current*, while AC stands for *alternating current*;[7] you might be familiar with these terms in relation to the current provided by a battery (constant and hence DC) and the current available from a mains socket (alternating at 50 or 60 Hz and therefore AC).

For a given sequence $x[n]$, one can always write

$$x[n] = x_{\mathrm{AC}}[n] + x_{\mathrm{DC}}$$

where x_{DC} is the mean of the sequence values. Please note the followings:

- The DC value of a finite-support signal is the value of its Fourier transform at $\omega = 0$ divided by the length of the signal's support.
- The DC value of an infinite-support signal must be zero for the signal to be absolutely summable or square summable.

In most signal processing applications, where the input signal comes from an acquisition device (such as a sampler, a soundcard and so on), it is important to remove the DC component; this is because the DC offset is often

[7] And AC/DC for Heavy Metal...

a random offset caused by ground mismatches between the acquisition device and the associated hardware. In order to eliminate the DC component we need to first estimate it, i.e. we need to estimate the mean of the signal.

For finite-length signals, computation of the mean is straightforward since it involves a finite number of operations. In most cases, however, we do not want to wait until the end of the signal before we try to remove its mean; what we need is a way to perform DC removal *on line*. The approach is therefore to obtain, at each instant, an *estimate* of the DC component from the past signal values, with the assumption that the estimate converges to the real mean of the signal. In order to obtain such an estimate, i.e. in order to obtain the average value of the past input samples, both approaches detailed in Section 5.3 are of course valid (i.e. the Moving Average and the Leaky Integrator filters) . We have seen, however, that the leaky integrator provides a superior benefit-cost tradeoff and therefore the output of a leaky integrator with λ very close to one (usually 10^{-3}) is the estimate of choice for the DC component of a signal. The closer λ is to one, the more accurate the estimation; the speed of convergence of the estimate however becomes slower and slower as $\lambda \to 1$. This can easily be seen from the group delay at $\omega = 0$, which is

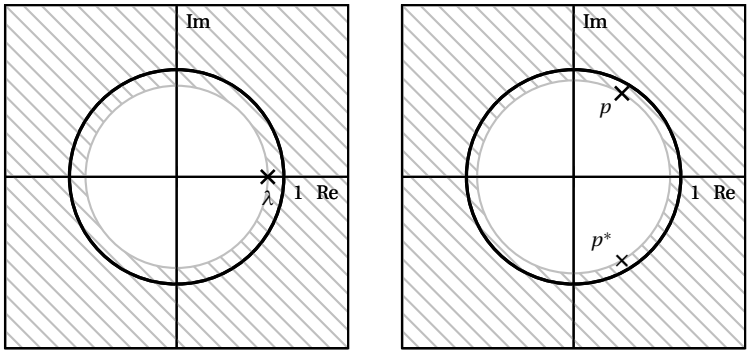
$$\mathrm{grd}\{H(1)\} = \frac{\lambda}{1-\lambda}$$

Resonator Filter. Let us look again at how the leaky integrator works. Consider its z-transform:

$$H(z) = \frac{1-\lambda}{1-\lambda z^{-1}}$$

and notice that what we really want the filter to do is to extract the zero-frequency component (i.e. the frequency component that does not oscillate, that is, the DC component). To do so, we placed a pole near $z = 1$, which of course corresponds to $z = e^{j\omega}$ for $\omega = 0$. Since the magnitude response of the filter exhibits a peak near a pole, and since the peak will be higher, the closer the pole is to the unit circle, we are in fact amplifying the zero-frequency component; this is apparent from the plot of the filter's frequency response in Figure 5.9. The numerator, $1-\lambda$, is chosen such that the magnitude of the filter at $\omega = 0$ is one; the net result is that the zero-frequency component will pass unmodified while all the other frequencies will be attenuated. The value of a filter's magnitude at a given frequency is often called the *gain*.

The very same approach can now be used to extract a signal component at *any* frequency. We will use a pole whose magnitude is still close to one (i.e. a pole near the unit circle) but whose phase is that of the frequency we



(a) (b)

Figure 7.19 Pole-zero plots for the leaky integrator and the simple resonator.

want to extract. We will then choose a numerator so that the magnitude is unity at the frequency of interest. The one extra detail is that, since we want a real-valued filter, we must place a complex conjugate pole as well. The resulting filter is called a resonator and a typical pole-zero plot is shown in Figure 7.19. The z-transform of a resonator at frequency ω_0 is therefore determined by the pole $p = \lambda e^{j\omega_0}$ and by its conjugate:

$$H(z) = \frac{G_0}{(1 - pz^{-1})(1 - p^* z^{-1})} = \frac{G_0}{1 - (2\lambda\cos\omega_0)z^{-1} + \lambda^2 z^{-2}} \tag{7.24}$$

The numerator value G_0 is computed so that the filter's gain at $\pm\omega_0$ is one; since in this case $\left|H(e^{j\omega_0})\right| = \left|H(e^{-j\omega_0})\right|$, we have

$$G_0 = (1-\lambda)\sqrt{1 + \lambda^2 - 2\lambda\cos 2\omega_0}$$

The magnitude and phase of a resonator with $\lambda = 0.9$ and $\omega_0 = \pi/3$ are shown in Figure 7.20.

A simple variant on the basic resonator can be obtained by considering the fact that the resonator is just a bandpass filter with a very narrow passband. As for all bandpass filters, we can therefore place a zero at $z = \pm 1$ and sharpen its midband frequency response. The corresponding z-transform is now

$$H(z) = G_1 \frac{1 - z^{-2}}{1 - (2\lambda\cos\omega_0)z^{-1} + \lambda^2 z^{-2}}$$

with

$$G_1 = \frac{G_0}{\sqrt{2(1 - \cos 2\omega_0)}}$$

The corresponding magnitude response is shown in Figure 7.21.

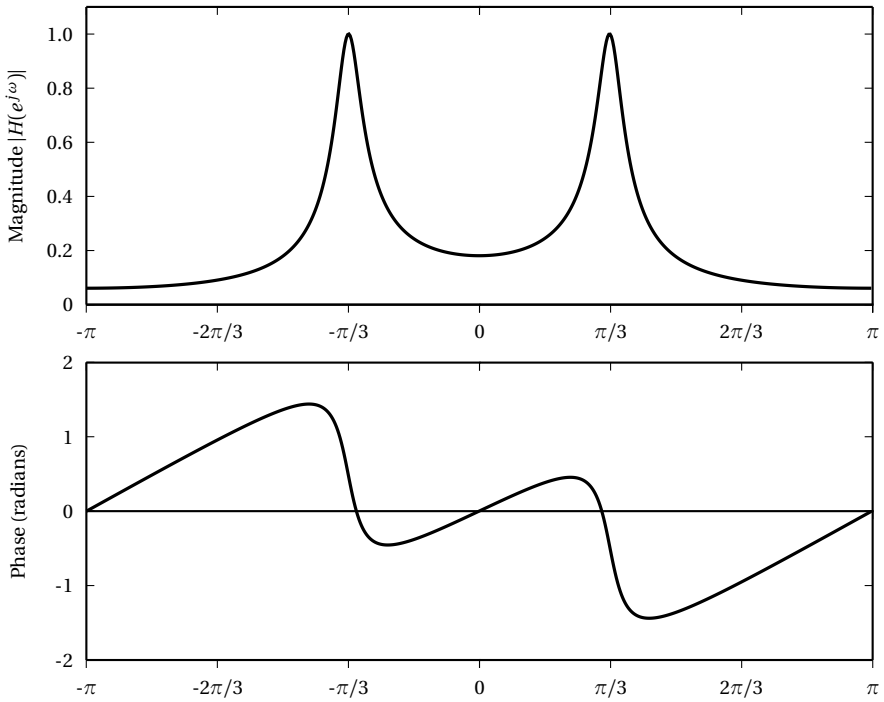


Figure 7.20 Frequency response of the simple resonator.

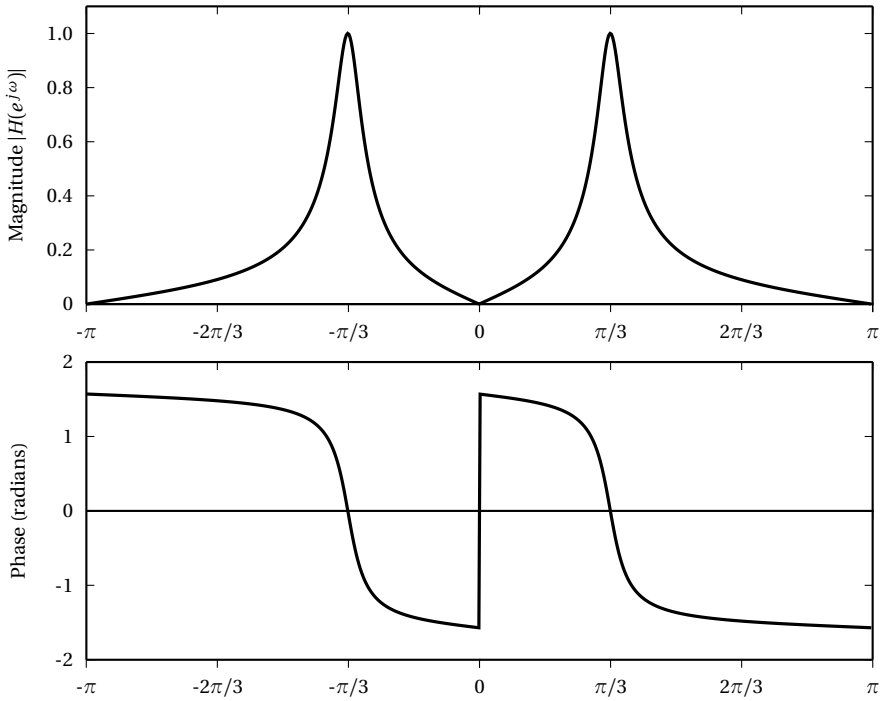


Figure 7.21 Frequency response of the modified resonator.

7.4 Filter Structures

We have seen in Section 5.7.2 a practical implementation of a constant-coefficient difference equation (written in C). That was just one particular way of translating Equation (5.46) into a numerical procedure; in this Section we explore other alternatives for both FIR and IIR and introduce the concept of computational efficiency for filters.

The cost of a numerical filter is dependent on the number of operations per output sample and on the storage (memory) required in the implementation. If we consider a generic CCDE, it is easy to see that the basic building blocks which make up the recipe for a realizable filter are:

- an addition operator for sequence values, implementing $y[n] = x_1[n] + x_2[n]$;
- a scalar multiplication operator, implementing $y[n] = \alpha x[n]$;
- a unit delay operator, implementing $y[n] = x[n-1]$. Note that the unit delay operator is nothing but a memory cell, holding the previous value of a time-varying quantity.

By properly combining these elements and by exploiting the different possible decomposition of a filter's rational transfer function, we can arrive at a variety of different working implementations of a filter. To study the possibilities at hand, instead of relying on a specific programming language, we will use self explanatory block diagrams.

Cascade Forms. Recall that a rational transfer function $H(z)$ can always be written out as follows:

$$H(z) = b_0 \frac{\prod_{n=1}^{M-1}(1 - z_n z^{-1})}{\prod_{n=1}^{N-1}(1 - p_n z^{-1})} \tag{7.25}$$

where the z_n are the $M-1$ (complex) roots of the numerator polynomial and the p_n are the $N-1$ (complex) roots of the denominator polynomial. Since the coefficients of the CCDE are assumed to be real, complex roots for both polynomials always appear in complex-conjugate pairs. A pair of first-order terms with complex-conjugate roots can be combined into a second-order term with real coefficients:

$$(1 - a z^{-1})(1 - a^* z^{-1}) = 1 - 2\,\mathrm{Re}\{a\}\, z^{-1} + |a|^2 z^{-2} \tag{7.26}$$

As a consequence, the transfer function can be factored into the product of first- and second-order terms in which the coefficients are all strictly real; namely:

$$H(z) = b_0 \frac{\prod_{n=1}^{M_r}(1 - z_n z^{-1}) \prod_{n=1}^{M_c}(1 - 2\mathrm{Re}\{z_n\} z^{-1} + |z_n|^2 z^{-2})}{\prod_{n=1}^{N_r}(1 - p_n z^{-1}) \prod_{n=1}^{N_c}(1 - 2\mathrm{Re}\{p_n\} z^{-1} + |p_n|^2 z^{-2})} \tag{7.27}$$

where M_r is the number of real zeros, M_c is the number of complex-conjugate zeros and $M_r + 2M_c = M - 1$ (and, equivalently, for the poles, $N_r + 2N_c = N - 1$). From this representation of the transfer function we can obtain an alternative structure for a filter; recall that if we apply a series of filters in sequence, the overall transfer function is the product of the single transfer functions. Working backwards, we can interpret (7.27) as the cascade of smaller sections. The resulting structure is called a *cascade* and it is particularly important for IIR filters, as we will see later.

Parallel Forms. Another interesting rewrite of the transfer function is based on a partial fraction expansion of the type:

$$\begin{aligned} H(z) = &\sum_n D_n z^{-n} + \sum_n \frac{A_n}{1 - p_n z^{-1}} \\ &+ \sum_n \frac{B_n + C_n z^{-1}}{(1 - p_n z^{-1})(1 - p_n^* z^{-1})} \end{aligned} \tag{7.28}$$

where the multiplicity of the three types of terms as well as the relative coefficients are dependent (in a non-trivial way) on the original filter coefficients. This generates a parallel structure of filters, whose outputs are summed together. The first branch corresponds to the first sum and it is an FIR filter; a further set of branches are associated to each term in the second sum, each one of them a first order IIR; the last set of branches is a collection of second order sections, one for each term of the third sum.

7.4.1 FIR Filter Structures

In an FIR transfer function all the denominator coefficients a_n other than a_0 are zero; we have therefore:

$$H(z) = b_0 + b_1 z^{-1} + \cdots + b_{M-1} z^{-(M-1)}$$

where, of course, the coefficients correspond to the nonzero values of the impulse response $h[n]$, i.e. $b_n = h[n]$. Using the constitutive elements outlined above, we can immediately draw a block diagram of an FIR filter as in

Figure 7.22. In practice, however, additions are distributed as shown in Figure 7.23; this kind of implementation is called a *transversal filter*. Further, ad-hoc optimizations for FIR structures can be obtained in the the case of symmetric and antisymmetric linear phase filters; these are considered in the exercises.

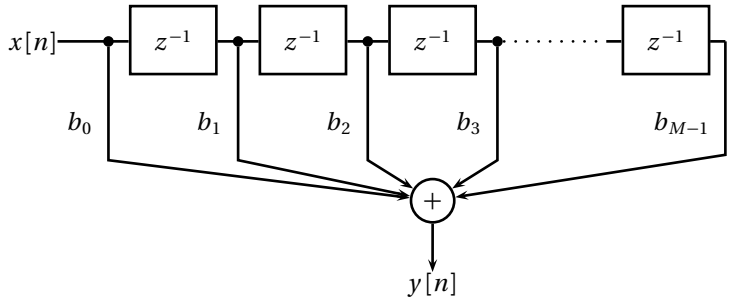


Figure 7.22 Direct FIR implementation.

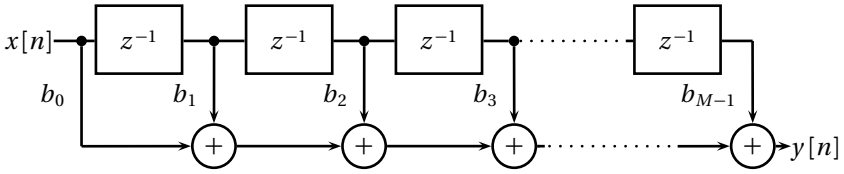


Figure 7.23 Transversal FIR implementation.

7.4.2 IIR Filter Structures

For an IIR filter, all the a_n and b_n in (5.46) are nonzero. One possible implementation based on the direct form of the transfer function is given in Figure 7.24. This implementation is called *Direct Form I* and it can immediately be seen that the C-code implementation in Section 5.7.2 realizes a Direct Form I algorithm. Here, for simplicity, we have assumed $N = M$ but obviously we can set some a_n or b_n to zero if this is not the case.

By the commutative properties of the z-transform, we can invert the order of computation to turn the Direct Form I structure into the structure shown in Figure 7.25 (shown for a second order section); we can then combine the parallel delays together to obtain the structure in Figure 7.26. This implementation is called *Direct Form II*; its obvious advantage is the reduced number of the required delay elements (hence of memory storage).

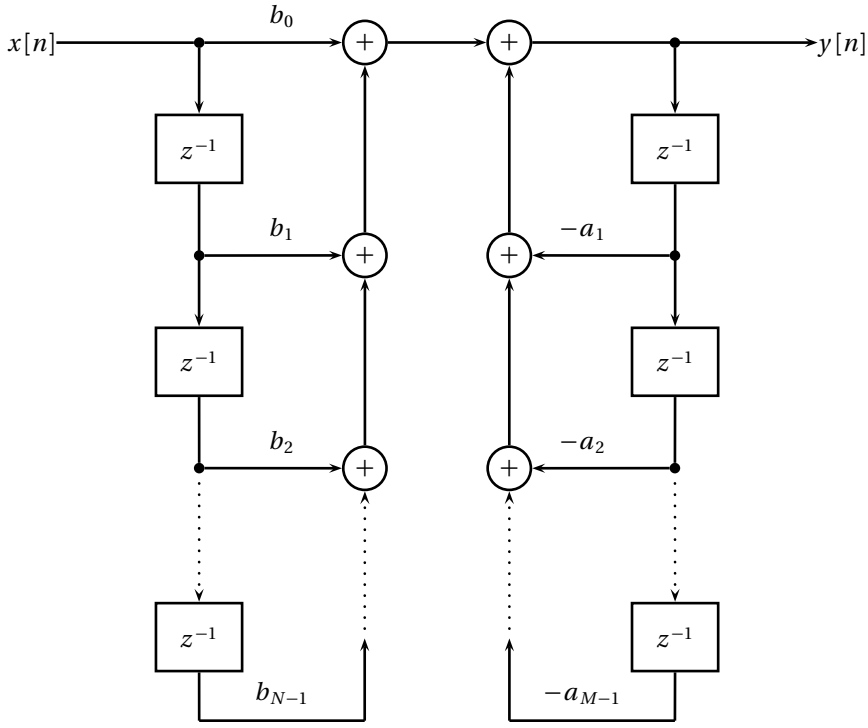


Figure 7.24 Direct Form implementation of an IIR filter.

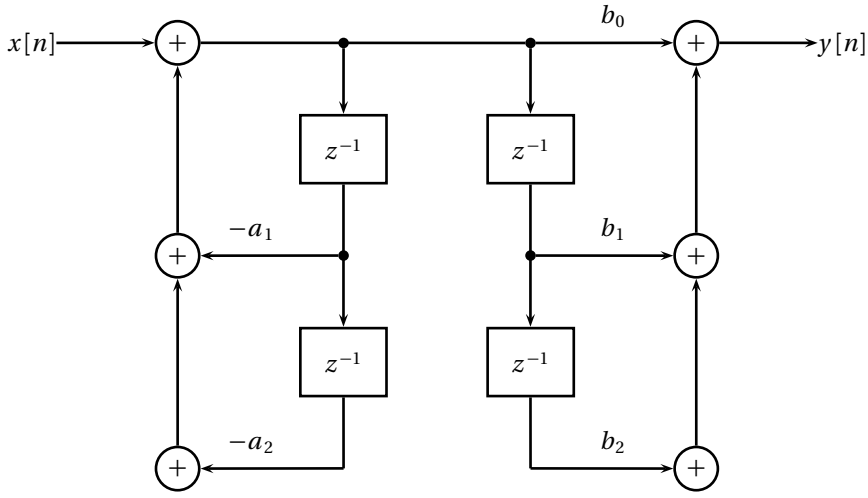


Figure 7.25 Direct form I with inverted order.

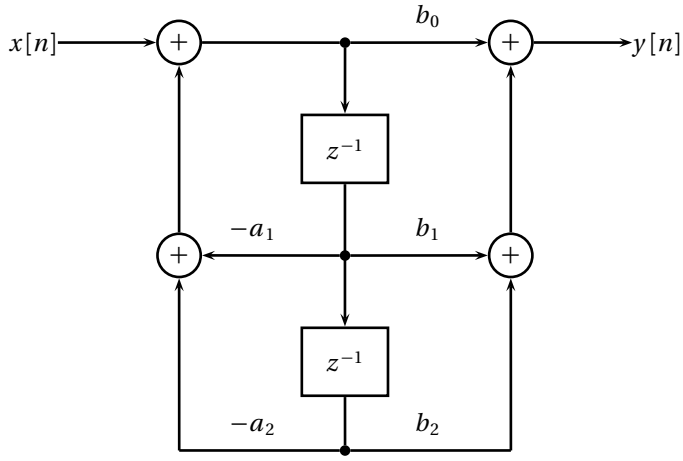


Figure 7.26 Direct Form II implementation of a second-order section.

The second order filter

$$H(z) = \frac{1 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}$$

which gives rise to the second order section displayed in Figure 7.26, is particularly important in the case of cascade realizations. Consider the factored form of $H(z)$ as in (7.27): if we combine the complex conjugate poles and zeros, and group the real poles and zeros in pairs, we can create a modular structure composed of second order sections. For instance, Figure 7.27 represents a 4th order system. Odd order systems can be obtained by setting some of the a_n or b_n to zero.

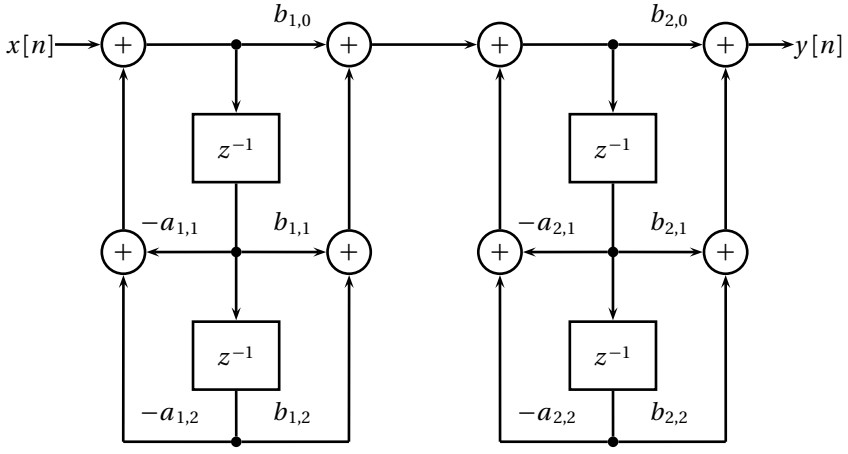


Figure 7.27 4th order IIR: cascade implementation.

7.4.3 Some Remarks on Numerical Stability

A very important issue with digital filters is their numerical behavior for a given implementation. Two key questions are:

- Assume the computations are made with (basically) infinite precision but that the filter coefficients are represented internally with finite precision. How good is the resulting filter? Is it still stable?
- If computations are also made with finite precision arithmetic (which implies rounding and truncation error), what is the resulting numerical behavior of the system?

One important difference is that, in the first case, the system is at least guaranteed to be linear; in the second case, however, we can have non-linear effects such as overflows and limit cycles.

Precision and computational issues are very hard to analyze. Here, we will just note that the direct form implementation is more sensible to precision errors than the cascade form, which is why the cascade form is usually preferred in practice. Moreover, alternative filter structures such as the *lattice* are designed to provide robustness for systems with low numerical precision, albeit at a higher computational cost.

7.5 Filtering and Signal Classes

The filtering structures that we have shown up to now are general-purpose architectures which apply to the most general class of discrete-time signals, (infinite) sequences. We now consider the other two main classes of discrete-time signals, namely finite-length signals and periodic sequences, and show that specialized filtering algorithms can be advantageously put to use.

7.5.1 Filtering of Finite-Length Signals

The convolution sum in (5.3) is defined for infinite sequences. For a finite-length signal of length N we may choose to write simply:

$$y[n] = \mathscr{H}\{x[n]\} = \sum_{k=0}^{N-1} x[k]h[n-k] \tag{7.29}$$

i.e. we let the summation index span only the indices for which the signal is defined. It can immediately be seen, however, that in so doing we are actu-

ally computing $y[n] = \bar{x}[n] * h[n]$, where $\bar{x}[n]$ is the finite support extension of $x[n]$ as in (2.24)); that is, by using (7.29), we are *implicitly* assuming a finite support extension for the input signal.

Even when the input is finite-length, the output of an LTI system is not necessarily a finite-support sequence. When the impulse response is FIR, however, the output has finite support; specifically, if the input sequence has support N and the impulse response has support L, the support of the output is $N+L-1$.

7.5.2 Filtering of Periodic Sequences

For periodic sequences, the convolution sum in (5.3) is well defined so there is no special care to be taken. It is easy to see that, for any LTI system, an N-periodic input produces an N-periodic output. A case of particular interest is the following: consider a length-N signal $x[n]$ and its N-periodic extension $\tilde{x}[n]$. Consider then a filter whose impulse response is FIR with a length-N support; if we call $h[n]$ the length-N signal obtained by considering only the values of the impulse response over its finite support, the impulse response of the filter is $\bar{h}[n]$ (see (2.24)). In this case we can write

$$\tilde{y}[n] = \sum_{k=-\infty}^{\infty} \tilde{x}[k]\bar{h}[n-k] = \sum_{k=0}^{N-1} h[k]\tilde{x}\big[(n-k) \bmod N\big] \tag{7.30}$$

Note that in the last sum, only the first period of $\tilde{x}[n]$ is used; we can therefore define the sum just in terms of the two N-point signals $x[n]$ and $h[n]$:

$$\tilde{y}[n] = \sum_{k=0}^{N-1} h[k]x\big[(n-k) \bmod N\big] \tag{7.31}$$

The above summation is called the *circular convolution* of $x[n]$ and $h[n]$ and is sometimes indicated as

$$\tilde{y}[n] = x[n] \circledast h[n]$$

Note that, for periodic sequences, the convolution as defined in (5.8) and the circular convolution coincide. The circular convolution, just like the standard convolution operator, is associative and commutative:

$$x[n] \circledast h[n] = h[n] \circledast x[n]$$
$$\big(h[n] + f[n]\big) \circledast x[n] = h[n] \circledast x[n] + f[n] \circledast x[n]$$

as is easily proven.

Consider now the output of the filter, expressed using the commutative property of the circular convolution:

$$\tilde{y}[n] = \sum_{k=0}^{N-1} x[k]h\big[(n-k) \bmod N\big]$$

Since the output sequence $\tilde{y}[n]$ is itself N-periodic we can consider the finite-length signal $y[n] = \tilde{y}[n]$, $n = 0, \ldots, N-1$, i.e. the first period of the output sequence. The circular convolution can now be expressed in matrix form as

$$\mathbf{y} = \mathbf{H}\mathbf{x} \tag{7.32}$$

where $\mathbf{y}, \mathbf{x}$ are the usual vector notation for the finite-length signals $y[n], x[n]$ and where

$$\mathbf{H} = \begin{bmatrix} h[0] & h[N-1] & h[N-2] & \ldots & h[2] & h[1] \\ h[1] & h[0] & h[N-1] & \ldots & h[3] & h[2] \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h[N-1] & h[N-2] & h[N-3] & \ldots & h[1] & h[0] \end{bmatrix} \tag{7.33}$$

The above matrix is called a *circulant matrix*, since each row is obtained by a right circular shift of the previous row. A fundamental result, whose proof is left as an exercise, is that the length-N DFT basis vectors $\mathbf{w}^{(k)}$ defined in (4.3) *are left eigenvectors of* $N \times N$ *circulant matrices*:

$$\big(\mathbf{w}^{(k)}\big)^T \mathbf{H} = H[k]\mathbf{w}^{(k)}$$

where $H[k]$ is the k-th DFT coefficient of the length-N signal $h[n]$, $n = 0, \ldots, N-1$. If we now take the DFT of (7.32) then

$$\mathbf{Y} = \mathbf{W}\mathbf{H}\mathbf{x} = \mathbf{\Gamma}\mathbf{W}\mathbf{x} = \mathbf{\Gamma}\mathbf{X}$$

with

$$\mathbf{\Gamma} = \mathrm{diag}\big(H[0], H[1], \ldots, H[N-1]\big)$$

or, in other words

$$Y[k] = H[k]X[k] \tag{7.34}$$

We have just proven a finite-length version of the convolution theorem; to repeat the main points:

- The convolution of an N-periodic sequence with a N-tap FIR impulse response is equal to the periodic convolution of two finite-length sig-

nals of length N, where the first signal is one period of the input and the second signal is the values of the impulse response over the support.

- The periodic convolution can be expressed as a matrix-vector product in which the matrix is circulant.
- The DFT of the circular convolution is simply the product of the DFTs of the two finite-length signals; in particular, (7.34) can be used to easily prove the commutativity and distributivity of the circular convolution.

The importance of this particular case of filtering stems from the following fact: the matrix-vector product in (7.32) requires $O(N^2)$ operations. The same product can however be written as

$$\mathbf{y} = \frac{1}{N}\mathbf{W}^H\mathbf{\Gamma}\mathbf{W}\mathbf{x} = \mathrm{DFT}^{-1}\{\mathbf{\Gamma}\,\mathrm{DFT}\{\mathbf{x}\}\}$$

which, by using the FFT algorithm, requires approximately $N + 2N\log_2 N$ operations and is therefore much more efficient even for moderate values of N. Practical applications of this idea are the *overlap-save* and *overlap-add* filtering methods, for a thorough description of which see [**?**]. The basic idea is that, in order to filter a long input sequence with an N-tap FIR filter, the input is broken into consecutive length-N pieces and each piece, considered as the main period of a periodic sequence, is filtered using the FFT strategy above. The difference between the two methods is in the subtle technicalities which allow the output pieces to bind together in order to give the correct final result.

Finally, we want to show that we could have quickly arrived at the same results just by considering the formal DTFTs of the sequences involved; this is an instance of the power of the DTFT formalism. From (4.43) and (4.44) we obtain:

$$\begin{aligned} Y(e^{j\omega}) &= \bar{H}(e^{j\omega})\tilde{X}(e^{j\omega}) \\ &= \left(\sum_{k=0}^{N-1} H[k]\,\Lambda\left(\omega - \frac{2\pi}{N}k\right)\right)\left(\frac{1}{N}\sum_{k=0}^{N-1} X[k]\,\tilde{\delta}\left(\omega - \frac{2\pi}{N}k\right)\right) \\ &= \frac{1}{N}\sum_{k=0}^{N-1} H[k]X[k]\,\tilde{\delta}\left(\omega - \frac{2\pi}{N}k\right) \end{aligned} \tag{7.35}$$

where the last equality results from the sifting property of the Dirac delta (see (4.31)) and the fact that $\Lambda(0) = 1$. In the last expression, the DTFT of a periodic sequence whose DFS coefficients are given by $H[k]X[k]$, is easily recognezed.

Examples

Example 7.1: The Goertzel filter

Consider the IIR structure shown in Figure 7.28; the filter is called a Goertzel filter, and its single coefficient (which is also the only pole of the system) is the k-th power of the N-th root of unity $W_N = e^{-j2\pi/N}$. Note that, contrary to what we have seen so far, this is a *complex-valued* filter; the analysis of this type of structure however is identical to that of a normal real-valued scheme.

As we said, the only pole of the filter is *on* the unit circle, so the system is not stable. We can nevertheless compute its impulse response, a task which is trivial in the case of a one-pole IIR; we assume zero initial conditions and we use the difference equation directly: by setting $x[n] = \delta[n]$ in

$$y[n] = x[n] + W_N^{-k}\, y[n-1]$$

and by working out the first few iterations, we obtain

$$h[n] = W_N^{-kn}\, u[n]$$

Note that the impulse response is N-periodic (a common trait of sequences whose poles are on the unit circle).

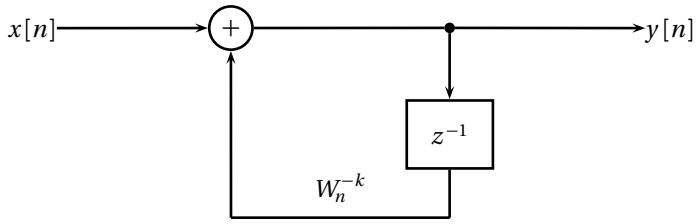


Figure 7.28 The Goertzel filter.

Assume now we have a length-N signal $x[n]$ and we build a finite-support extension $\bar{x}[n]$ so that $\bar{x}[n] = 0$ for $n < 0$, $n \geq N$ and $\bar{x}[n] = x[n]$ otherwise. If we process such a signal with the Goertzel filter we have

$$\begin{aligned}
y[0] &= x[0] \\
y[1] &= x[1] + W_N^{-k}\, x[0] \\
y[2] &= x[2] + W_N^{-k}\, x[1] + W_N^{-2k}\, x[0] \\
y[3] &= x[3] + W_N^{-k}\, x[2] + W_N^{-2k}\, x[1] + W_N^{-3k}\, x[0] \\
&\vdots
\end{aligned}$$

so that finally:

$$y[N] = \sum_{n=0}^{N-1} W_N^{-k(N-n)} x[n] = \sum_{n=0}^{N-1} x[n]\, W_N^{nk} = X[k]$$

that is, the output at time $n = N$ is the k-th DFT coefficient of $x[n]$. The Goertzel filter is therefore a little machine which allows us to obtain one specific Fourier coefficient without needing to compute the whole DFT. As a filter, its usage is nonstandard, since its delay element must be manually reset to zero initial conditions after each group of N iterations. Goertzel algorithm is used in digital detectors of DTMF tones.

Example 7.2: Filtering and numerical precision

Digital filters are implemented on general-purpose microprocessors; the precision of the arithmetics involved in computing the output values depends on the intrinsic word length in the digital architecture, i.e. in the number of bits used to represent both the data and the filter coefficients. To illustrate some of the issues related to numeric precision consider the case of an *allpass* filter. The magnitude response of an allpass filter is *constant* over the entire $[-\pi, \pi]$ interval, hence the name. Such filters are often used in cascade with other filters to gain control on the overall phase response of the system.

Consider the filter described by the following difference equation:

$$y[n] = \alpha y[n-1] - \alpha x[n] + x[n-1]$$

with $0 < \alpha < 1$. The transfer function $H(z)$ is

$$H(z) = \frac{-\alpha + z^{-1}}{1 - \alpha z^{-1}} = -\alpha \frac{1 - (1/\alpha) z^{-1}}{1 - \alpha z^{-1}}$$

and the filter is indeed allpass since:

$$\begin{aligned} \left|H(z)\right|^2 &= H(z) H^*(z) \\ &= \frac{-\alpha + z^{-1}}{1 - \alpha z^{-1}} \cdot \frac{-\alpha + \left(z^{-1}\right)^*}{1 - \alpha \left(z^{-1}\right)^*} \\ &= \frac{\alpha^2 - \alpha \,\mathrm{Re}\{z^{-1}\} + \left|z^{-1}\right|^2}{\alpha^2 \left|z^{-1}\right|^2 - \alpha \,\mathrm{Re}\{z^{-1}\} + 1} \end{aligned}$$

for $z = e^{j\omega}$ (and therefore $|z^{-1}| = 1$):

$$\left|H(e^{j\omega})\right|^2 = \left|H(e^{j\omega})\right| = 1$$

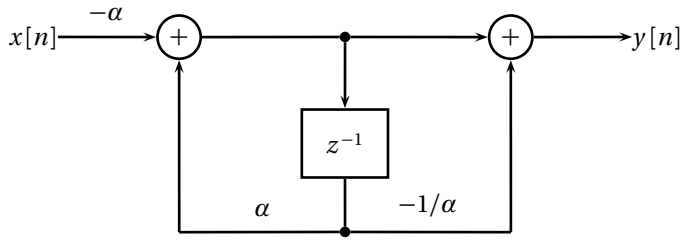


Figure 7.29 Allpass filter Direct Form II implementation.

The filter can be implemented in Direct Form II as in Figure 7.29. Note that the two coefficients of the filter are α and $1/\alpha$ so that, if α is small then $1/\alpha$ will be big, and vice versa. This creates a problem in a digital architecture in which the internal representation has only a small number of bits. Call $\mathcal{Q}\{\cdot\}$ the operator which associates a real number to the closest value in the architecture's internal representation; the process is called *quantization* and we will study it in more detail in Chapter 10. The transfer function with quantized coefficients becomes

$$H_Q(z) = \mathcal{Q}\{-\alpha\}\frac{1-\mathcal{Q}\{1/\alpha\}z^{-1}}{1-\mathcal{Q}\{\alpha\}z^{-1}} = \frac{-\hat{\alpha}+\beta z^{-1}}{1-\hat{\alpha}z^{-1}}$$

where $\beta = \mathcal{Q}\{\alpha\}\mathcal{Q}\{1/\alpha\}$. If the quantization is done with too few bits, $\beta \neq 1$ and the filter characteristic is no longer allpass. Suppose for instance that the filter uses four bits to store its coefficients using an unsigned fixed point 2.2 format; the 16 possible values are listed in Table 7.1.

Table 7.1 Binary-decimal conversion table for fixed-point 2.2 notation.

binary	decimal	binary	decimal
0000	0.00	1000	2.00
0001	0.25	1001	2.25
0010	0.50	1010	2.50
0011	0.75	1011	2.75
0100	1.00	1100	3.00
0101	1.25	1101	3.25
0110	1.50	1110	3.50
0111	1.75	1111	3.75

If $\alpha = 0.4$ we have that $\mathcal{Q}\{0.4\} = 0010 = 0.5$, $\mathcal{Q}\{1/0.4\} = \mathcal{Q}\{2.5\} = 1010 = 2.5$ and therefore $\beta = 0101 = 1.25 \neq 1$.

It is important to point out that the numerical stability of a filter is dependent on the chosen realization. If we rewrite the allpass difference equation as

$$y[n] = \alpha\big(y[n-1] - x[n]\big) + x[n-1]$$

we can a block diagram as in Figure 7.30 which, although a non-canonical form, implements the filter with no quantization issues *independently of* α. Note that the price we pay for robustness is the fact that we have to use two delays instead of one.

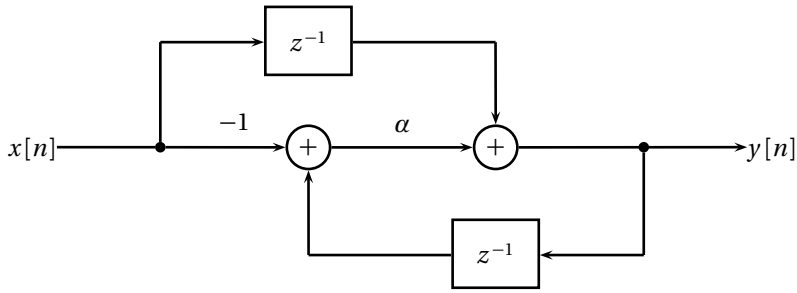


Figure 7.30 Allpass filter Direct Form II implementation.

Example 7.3: A guitar synthesizer

We have encountered the Karplus-Strong algorithm in Example 2.2. A practical implementation of the algorithm is shown in Figure 7.31; it is a quite peculiar filter structure since it has no input! Indeed assume there are N delays in cascade and neglect for a moment the filter $H(z)$; the structure forms a feedback loop in which the N values *contained in the delay units at power-up* are endlessly cycled at the output. By loading the N delay units with all sorts of finite-length sequences we can obtain a variety of different sounds; by changing N we can change the fundamental frequency of the note.

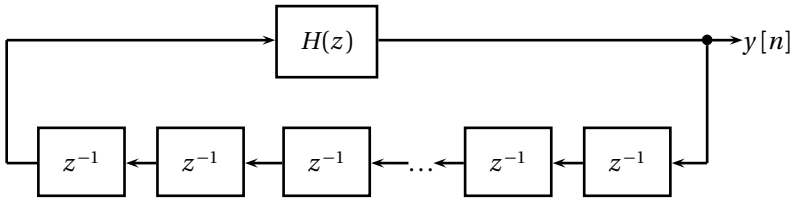


Figure 7.31 Karplus-Strong implementation.

The detailed analysis of a waveform generated by the device in Figure 7.31 is complicated by the fact that the filter does not have zero initial conditions.

Intuitively, however, we can easily appreciate that we can use the filter to simulate a natural decay in the waveform; imagine $H(z) = \alpha$ with $0 < \alpha < 1$: at each passage through the feedback loop the values in the delay line are scaled down exponentially. More complicated filters can be used to simulate different types of acoustic decay as long as $\left|H(e^{j\omega})\right| < 1$ over the entire $[-\pi, \pi]$ interval.

Further Reading

Filter design has been a very popular topic in signal processing, with a large literature, a variety of software designs, and several books devoted to the topic. As examples, we can mention R. Hamming's *Digital Filters* (Dover, 1997), and T. W. Parks and C. S. Burrus, *Digital Filter Design* (Wiley-Interscience, 1987), the latter being specifically oriented towards implementations on a digital signal processor (DSP). All classic signal-processing books cover the topic, for example Oppenheim and Schafer's book *Discrete-Time Signal Processing* (Prentis Hall, 1999) gives both structures and design methods for various digital filters.