RECURSIVE FILTERS

A digital filter is the implementation of an algorithm that computes an output sequence of numbers from an input sequence of numbers. The algorithm may be implemented (or *realized*) using a general-purpose computer running software, a special-purpose computer such as a digital signal processor (DSP) running software, or special purpose hardware such as an application specific integrated circuit (ASIC) or field programmable gate array (FPGA). In this latter case, the digital circuit is designed specifically to accomplish the filtering task. This process is shown in Fig. 1. Here, $x(n)$ is an input sequence of numbers (usually, we read it as the nth sample of the sequence x), $H(z)$ describes the recursive digital filter, and $y(n)$ is the resulting output sequence of numbers. Whether the digital filter is implemented on a general-purpose computer, on a special-purpose computer, or with special-purpose digital circuitry is usually a function of the required processing speeds (sampling rates) or the volume of units needed in production. The digital sequence of numbers $x(n)$ is most typically generated from an analog waveform through the use of an analog-to-digital converter (ADC). The output sequence $y(n)$ is most typically used to generate an analog waveform using a digital-to-analog converter (DAC). For instance, consider the recording and playback system for digital audio commonly called the compact disk. The audio is recorded using microphones, whose analog outputs are digitized using the ADC. These digital samples may be filtered in the processing and mixing of the recording. The resulting digital waveform is stored optically on the compact disk. The compact disk player reads the digital signal, filters these signals digitally, and then produces the analog output for the power amplifier and speakers using a DAC. Some signals are digital by nature and thus do not require the conversions between the analog and digital formats. A classic example of such a signal is the price of a stock or commodity. Consider a digital speech signal that has been converted from an analog signal (a microphone output). The digital sequence representing the sentence "This is sound" is shown graphically in Fig. 2. The amplitudes of the speech vary between 90 and 160, with the average value being approximately 128. This is the result of the speech being quantized to 8 bits, so that the amplitudes are limited to integers between 0 and 255 (which is $2^8 - 1$). The sampling frequency is 8 kHz, so that the samples are spaced 1/8000 s apart. The alternation pattern for various portions of the sequence indicate that the frequency content of the speech varies over the 35,000 samples shown. For instance, the samples in the breaks between "This" and "is" and between "is" and "sound" have flat amplitude values of 128 or 129. These portions of the sequence have a 0 frequency (often called "dc") component only. The words, which have more significantly varying amplitude values, have a much richer frequency content. The highest frequency present in a digital sequence is the *Nyquist frequency*. The Nyquist frequency is

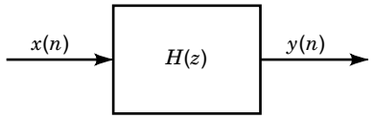


Figure 1. Filter block diagram showing input $x(n)$, output $y(n)$, and transfer function $H(z)$.

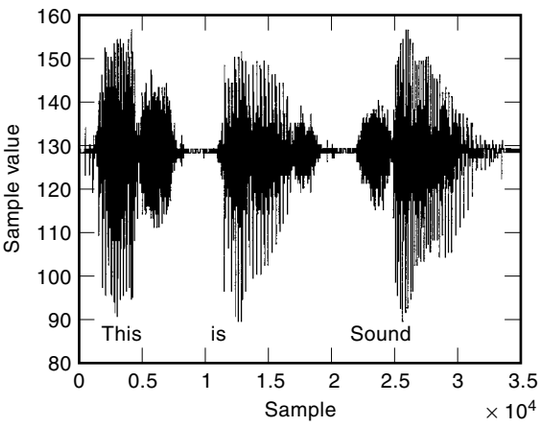


Figure 2. A digital sequence of speech representing the words "This is sound."

the real frequency that is one-half of the sampling frequency. Real frequencies above the Nyquist frequency are folded over into lower frequencies. Once this happens, the signal is said to be *aliased,* and in general the original analog signal cannot be recovered from the digital sequence.

Filtering is required in many products and systems commonly used everyday. Digital filters have been used in telecommunications systems since at least the mid-1970s. As communication systems increasingly become totally digital, this use will only increase. Common communication applications are in echo cancellers, voice codecs (coder/decoder) or vocoders, image and video codecs, and certain switching devices. Medical devices, electronic consumer items, and military devices are major users of digital filters. Common applications are in heartbeat monitoring and radiological imaging, compact disk audio and digital tape, digital cameras and video, and radar and sonar. Digital filters are required to separate an information-bearing signal from extraneous (*noise*) signals, which is the signal-to-noise ratio (SNR) enhancement problem. Digital filters are also used to separate and extract information for use in detection and estimation problems.

Digital filters work on the principle of frequency separation. For a filter to be effective in separating an information-bearing signal from the noise, the frequency content of the information-bearing signal must be sufficiently different from the frequency content of the noise. The *quality* of a filter is the frequency selectivity of the filter—that is, its ability to frequency discriminate. The discrimination is measured using gain (or attenuation). Plots of gain (or attenuation) versus frequency are called the *magnitude response* of the filter. Several classical magnitude responses are the low-pass, high-pass, band-pass, and band-stop filters. Magnitude responses for examples of these filter types are shown in Fig. 3. Content of the sequence that is at frequencies where the gain is 0 are passed without amplitude loss. Other content is attenuated according to the amount of the gain. If the gain is −100, then the amplitude is reduced by 5 orders of magnitude; that is, the amplitude is reduced by a factor of 10^5. The frequencies

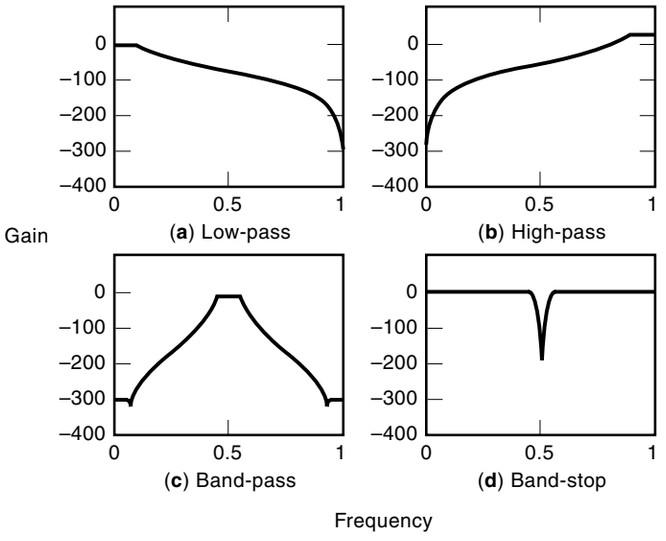


Figure 3. Classical magnitude responses for low-pass, high-pass, bandpass, and bandstop filters.

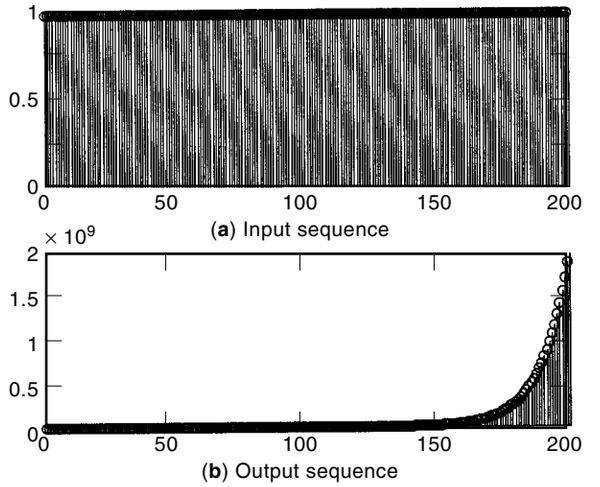


Figure 4. The bounded input and unbounded output of an unstable recursive digital filter.

shown are between 0 and 1, where 0 is dc and 1 is the Nyquist frequency. In the case of the speech sequence shown in Fig. 1, the Nyquist rate is one-half of 8 kHz, or 4 kHz. Each of these filter types is designed to pass unaltered the frequency content of the input signal across a range of frequencies while blocking the frequency content at other frequency ranges. For instance, a low-pass filter is typically used to eliminate broadband noise in communication systems. Such a filter would have unity gain across the frequency range containing the information-bearing signal and zero gain at all other frequencies. Thus, all frequency content of the noise outside the unity gain frequencies will be removed. The magnitude response of the digital filter is controlled by the placement of the poles and zeros of the digital filter. Frequencies near poles are amplified, while frequencies near zeros are attenuated. The number of poles and zeros determines the order of the filter. For instance, a digital filter with five poles and three zeros is a fifth-order filter. The filter stability is a function of the location of the filter poles. For causal filters, all poles must have a magnitude less than one; that is, the complex-valued poles must lie inside the unit circle of the complex plane. Filter stability is usually determined using the bounded-input, bounded-output (BIBO) criterion. BIBO stability means that the output sequence resulting from any input sequence that has all sample values less than some value must also be bounded by some value (not necessarily the same value that bounds all input sequence values). To say this in a less precise but more intuitive manner, a filter is not BIBO stable if for some input sequence whose sample values are all finitely valued, some filter output samples are infinite in magnitude. This case is shown in Fig. 4, where the magnitude of the pole of the filter is 1.1. In this case, the input sequence to the filter has sample values that are all 1. The output sequence values are tending toward infinity as the sample index increases. Clearly then, an unstable filter is undesirable. The order of the filter is also directly related to the number of multiplies and adds required to compute the filter output sequence given an input sequence (i.e., "to run the filter"). This number, of course, directly translates into the amount and capabilities of the digital circuitry required to run the filter.

Prior to the advent of very large scale integration (VLSI) for integrated electronic circuits, the use of digital filters was so restricted that many applications required the use of analog filters. Analog filters require resistors and need either inductors or capacitors (or both). More recently, the use of active components such as operational amplifiers in combination with resistors and capacitors was extremely common. However, with the development of the sigma-delta modulation ($\Sigma\Delta$M) and multirate filtering, the use of all digital systems has increased significantly, to the point that analog filtering may only be used in high-power amplifier systems. Digital filters are superior to analog filters in many respects. They allow high-precision frequency shaping, manufacturing and reuse, system integration, and system design.

While many possible categorization techniques exist, the most common classes of digital filters are the finite impulse response (FIR) and infinite impulse response (IIR) digital filters. FIR filters are also called moving average (MA) or all-zero filters. IIR filters come in two varieties: the autoregressive (AR) or all-pole filters and the autoregressive/moving average (ARMA) filters. The ARMA filters have both poles and zeros. It is the AR and ARMA filters that are recursive, because the filter output depends on both input and past output (the regression). The MA filters are nonrecursive, because the filter output depends only on a finite number of filter input samples. Because MA filters only have poles located at the origin of the complex plane (all poles have magnitude zero, which is less than 1), MA filters are always stable. MA filters have one special property that makes them invaluable in certain high-fidelity systems and products: They can be designed and implemented so that the filter minimally distorts the information-bearing signal. The price of this distortion-free filtering is in the increased requirements of the digital circuitry in terms of both speed and quantity of digital devices. Both speed and quantity translate to the real costs of power and size. Consequently, recursive filters, which do distort the information-bearing signal, are often used. The causes of this distortion are evident from the Payley–Wiener conditions, and we will examine these in a later section. For now, we classify the distortions into two categories: amplitude and time delay. All causal filters produce amplitude distortion, although this distortion can be minimized directly in the design process. The gain cannot be uniform across a range of frequencies; however, the magnitude response is directly specified in the design process and thus can be controlled. All causal IIR filters (unlike causal FIR filters) cause time delay distortion. Because time delay is not part of the basic filter design problem, it cannot be controlled directly in the design process. Time-delay distortion is measured using group delay, and we specifically define this in a later section. However, an intuitive understanding of group delay can be developed. Suppose the input signal is a musical chord consisting of sinusoids of several frequencies with a given phase relationship. The filter processing creates a time delay in the filter output. If the phase relationship present in the input is maintained in the output, we say the filter is without distortion. This dis-

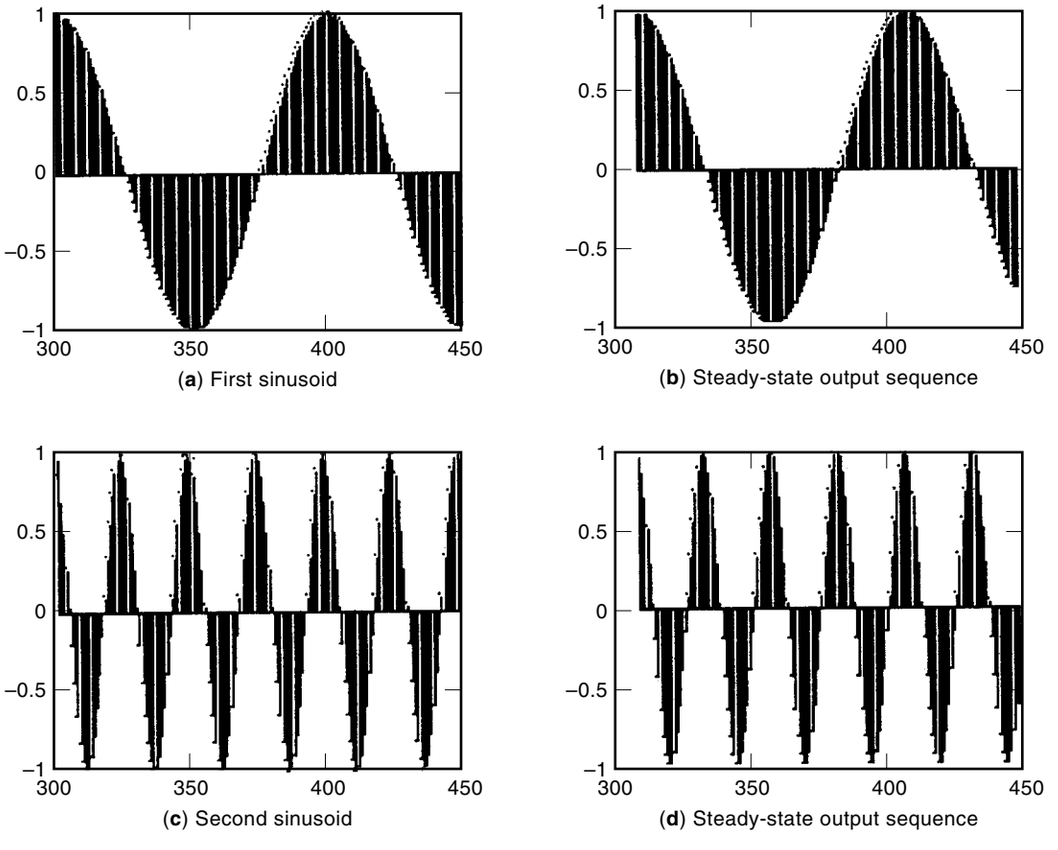


Figure 5. Phase response as time delay for two sinusoids passing through a filter.

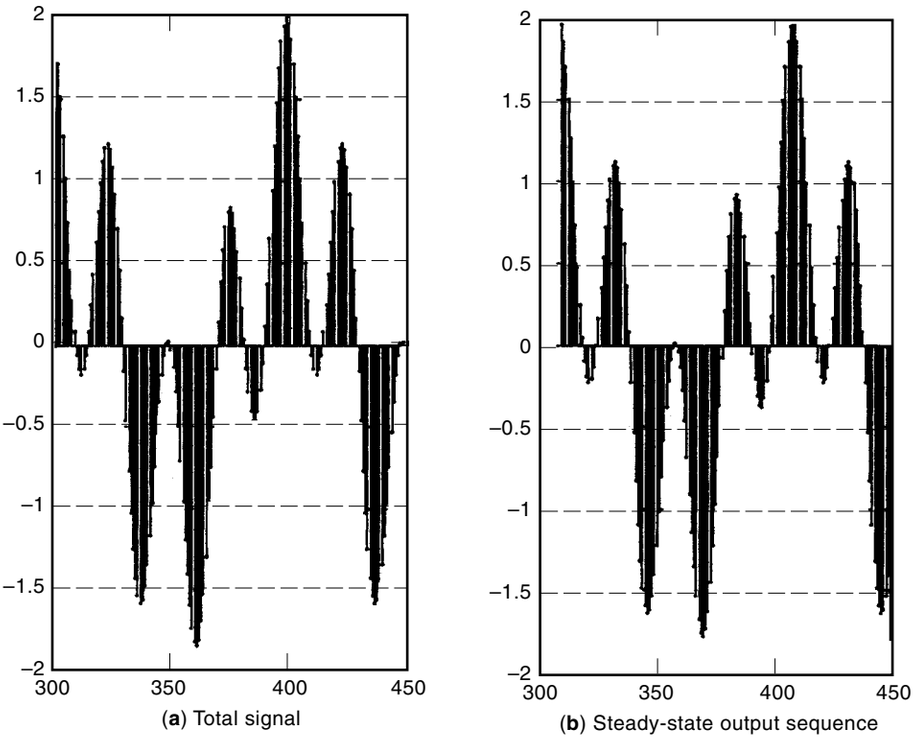


Figure 6. Response showing the phase distortion when the stem of two sinusoids is passed through a filter.

tortion is examined in Figs. 5 and 6. The sequence given in Fig. 6(a) (the total signal) is the sum of the two individual sinusoids given in Figs. 5(a) and 5(c). Both sinusoidal frequencies fall within the region where the filter gain is 0 (they are passed without the amplitudes being altered). Figures 5(b) and 5(d) show the output sequences when only the corresponding individual sinusoids are filtered. Notice that the phase relationship between the two sinusoids is not maintained. The first sinusoid is delayed by nearly 9 samples while the second sinusoid is delayed by nearly 15 samples. To see how this causes phase distortion, examine Fig. 6(b). Because both sinusoid frequencies are within the range of frequencies where the amplitude is not altered, one would expect the output and input sequences to be identical. However, close examination and comparison of the input and output sequences shows that the output sequence is a distorted version of the input sequence. In particular, examine the amplitude of the next to last peak shown in the figure (we have included the dashed lines to assist in the comparison). If the phase relationship is not maintained (as in our example), we say the filter causes distortion. However, the fact that for a given set of filter specifications or requirements a recursive filter will always use less digital circuitry than a nonrecursive filter indicates that recursive filters will always find extensive application.

Filters are commonly characterized in several ways: difference equation, impulse response, signal flow graph, and state space. A difference equation is a mathematical equality that relates past and present inputs to past and present outputs of causal filters. A noncausal filter adds future inputs to the causal equality. The various terms are weighted by the coefficients. The impulse response of a filter is the filter output in response to a very particular input sequence that is zero everywhere except at the zeroth sample. The impulse responses for the four classical filter types whose magnitude responses are shown in Fig. 3 are given in Fig. 7. Notice that the low-pass filter impulse response smoothly decays toward zero, while the high-pass filter impulse response alternates between positive and negative values as it decays toward zero. The duration of the bandpass filter impulse response is long when compared to the other three filter types. Signal flow graphs represent the filter graphically. More importantly, they detail the order of computations required to determine the filter output. State-space descriptions of digital filters are a matrix version of the difference equation. Different characterizations (often called representations) of a digital filter are often needed either to dovetail the required computations to the computational device(s) or to reduce the impact of finite precision arithmetic.

Methods for designing recursive filters can be classified into two main types: direct designs using computer optimization and designs obtained from classical analog filter designs. Many of these techniques rely on direct manipulation of the magnitude response only. Direct designs of both AR and ARMA digital filters are performed using least-squares techniques that we will develop. Such designs use all the techniques of computer optimization, including multivariate and multicriterion methods. Designs obtained by transforming analog filters were the first design methods developed for digital filters because a large body of literature and many design methods had been developed over the many years of practical analog filter use. These methods rely on certain analytical optimization methods of polynomials. These methods also require conversion techniques that transform the designed analog filter to a digital filter suitable for implementation. We will develop the two most commonly used methods of conversion, *impulse-invariance* and *bilinear transformation.* Furthermore, since the analog design methods are based on the

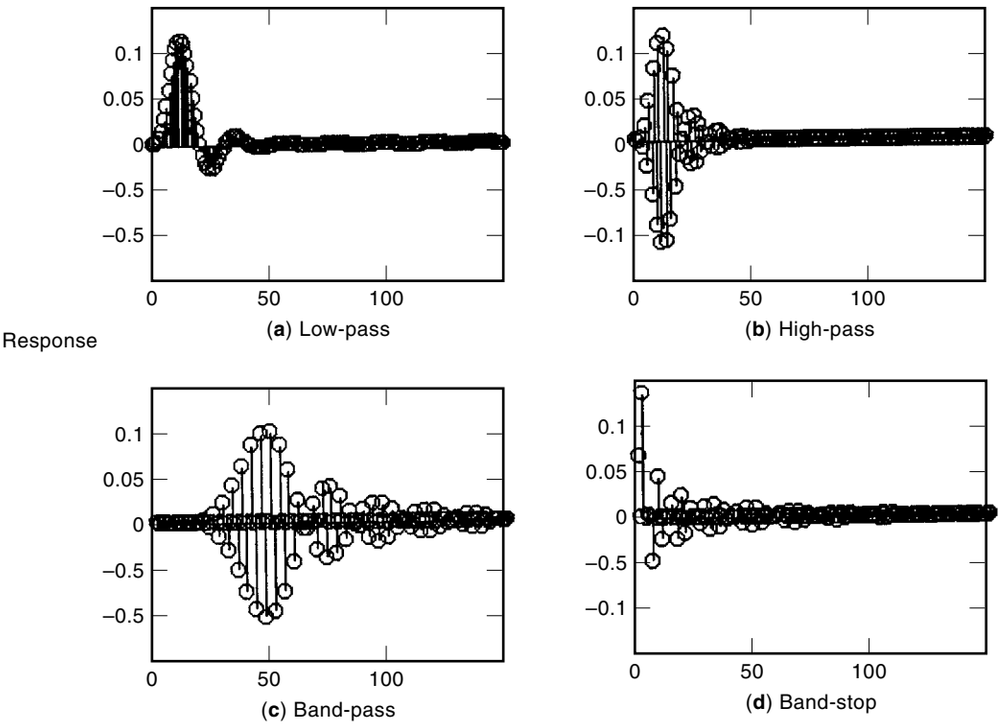


Figure 7. Classical filter impulse responses for low-pass, high-pass, bandpass, and bandstop filters.

design of low-pass filters, frequency transformations are required to change filter magnitude responses between low-pass and the high-pass, bandpass, and bandstop designs that may be desired.

We examine methods and techniques relevant to the implementation of digital filters. The designs of digital filters are most commonly performed on general-purpose computers, most often on machines with at least 64 bits of floating point representations. However, digital filters are most often implemented with finite precision arithmetic hardware that uses much fewer bits (e.g., 16 or 32 bits). By this, we mean that all filter coefficients and input, output, and internal signals are represented using a fixed number of bits. When the filter coefficients are quantized from the designed representation using the higher number of bits to the realized representation using a lower number of bits, the poles and zeros of the realized filter are different from the poles and zeros of the designed filter. One critical issue of utmost practical concern to the filter designer is to determine whether or not this realized filter is still stable. As an example, consider the input and output sequences shown in Fig. 8. The low-pass filter has the magnitude response given in Fig. 8(a). The single-frequency (sinusoid) input is given in Fig. 8(b). The designed filter should have the output sequence given in Fig. 8(c) as the response to the input sequence. However, when the coefficients are quantized to 40 bits (the most available in any current DSP microprocessor), the actual output [shown in Fig. 8(d)] is the output of an unstable system. Furthermore, even if the realized filter is stable, the changes in the poles and zeros might still significantly alter the actual magnitude response to the point where the design specifications are not met. One other consideration for the designer is the quantization of the intermediate quantities (often called the internal signals) computed in the course of computing the overall output samples. The hardware may be either fixed- or floating-point. In implementations using fixed-point hardware, the design must incorporate scaling to minimize the quantization distortion of these intermediate quantities. In floating point hardware, the hardware scales the results automatically. However, in either case, quantization, usually in the form of rounding, is still required to "fit" the resulting values into the available memory locations. This nonlinear distortion causes the problematic limit cycle. For example, consider the high-pass filter whose magnitude response is shown in Fig. 9(a). The input to the filter is zero [see Fig. 9(b)], but the initial value $y(0)$ equals 1. The designed output should approach zero, as shown in Fig. 9(c). If the internal signals are quantized coarsely (to the nearest tenth), then the actual output approaches the limit cycle where the samples alternate between ± 0.5 [shown in Fig. 9(d)]. This feature of recursive digital filters is very undesirable. Increasing the resolution (the number of bits) used in the internal representation and calculations will minimize the effect of the limit cycle. Thus, the choice of scale is crucial to the actual performance of the realized digital filter. The order of computations affects the robustness of the filter implementation to these errors. We develop the theory of an *optimal* form, as well as other forms that are commonly used to minimize the effects of the internal quantization distortion on the digital filter output sequence. The basic approach in these forms is the isolation of filter components, usually through the deconstruction of a high-order filter into many low-order filters whose outputs are then recombined to produce the desired output. The most common of these forms is the *parallel* form, wherein a high-

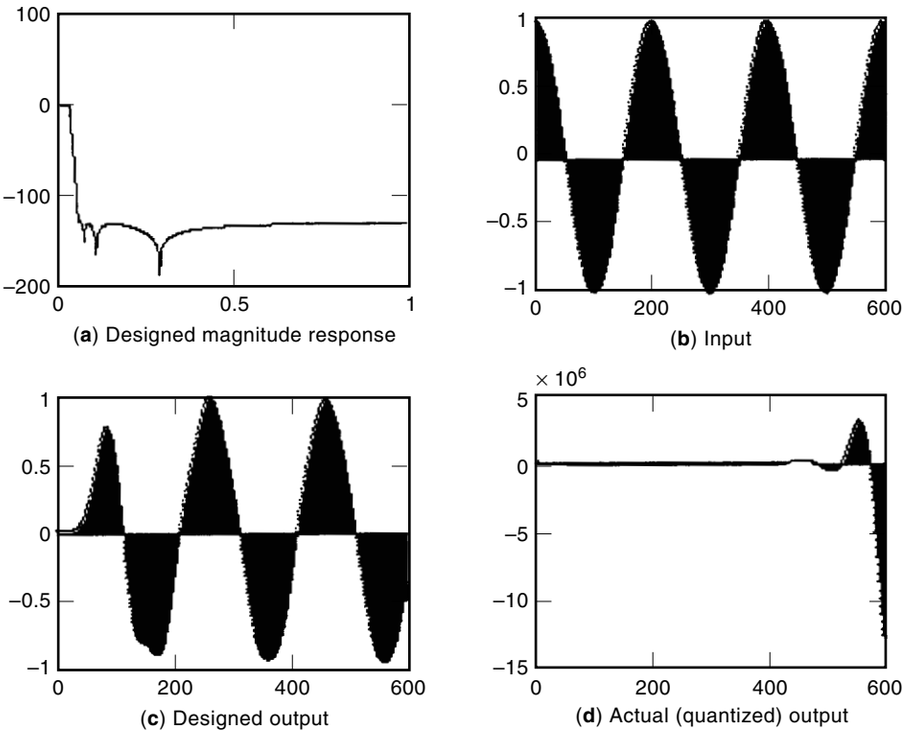


Figure 8. A stable filter becomes unstable when its coefficients are quantized to 40 bits.

order filter is decomposed into many second-order components (called *subfilters*) and possibly one first-order filter if the order of the high-order filter is odd. The outputs of this form are recombined properly by adding the outputs of all of the subfilters. Other forms, such as the cascade and lattice structure, isolate second- and first-order subfilters in a cascade structure. Wave digital filter structures are sometimes used for this purpose.

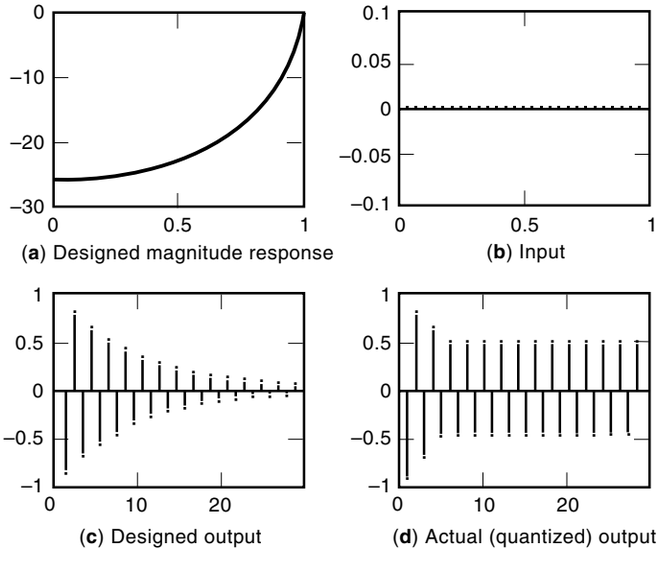


Figure 9. A linear filter becomes nonlinear causing a limit cycle when internal computations are quantized.

FILTERING

Causal, recursive digital filters produce an output sequence using both current and previous inputs and previous outputs only. Referring to Fig. 1, recursive digital filters are most often parameterized in terms of the difference equation

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{1}$$

The filter output $y(n)$ is computed for each sample n. We will see in a later section that these parameters (the a_k, $k = 1, 2, \ldots, N$ and the b_k, $k = 1, 2, \ldots, M$) are not the best for implementation. The filter order is $\max(N, M)$. The unit sample response (or commonly, the impulse response) $h(n)$ is equal to the filter output when the input $x(n) = \delta(n)$, where the unit sample (impulse) sequence is given by

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \tag{2}$$

For a causal filter as described in Eq. (1), we must have $h(n) = 0$ for $n < 0$ and consequently the impulse response sequence $h(n)$ may be directly computed. The impulse responses for the four classical filter magnitude responses are shown in Fig. 7. To ease our notation, we introduce the unit step sequence

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases} \tag{3}$$

Because the filter parameters are constant (i.e., not functions of n) and because the difference equation is linear, we have the convolution result

$$y(n) = x(n) * h(n) \equiv \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \tag{4}$$

The convolution sum given in Eq. (4) is proven as follows. Recognize that the special nature of the unit sample sequence permits any sequence to be rewritten as

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$

Since the filter parameters are constant, the filter response to each individual term in the sum is just $x(k)h(n-k)$; and since the filter is linear, the sum of the responses [as given in the convolution of Eq. (4)] is the digital filter output.

Now, consider what the filter response is to the pure tone (complex exponential) $x(n) = e^{j\omega_0 n}$. From Eq. (4) and using some algebra, we have the filter output

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{\infty} h(k)e^{j\omega_0(n-k)} \\ &= e^{j\omega_0 n} \sum_{k=-\infty}^{\infty} h(k)e^{j\omega_0 k} \equiv e^{j\omega_0 n}H(e^{j\omega_0}) \end{aligned}$$

The complex-valued function

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)e^{-j\omega n} \equiv |H(e^{j\omega})|e^{j\angle H(e^{j\omega})} \tag{5}$$

defines the *discrete-time Fourier transform.* We have written the transform in polar notation, using the magnitude and phase notation, respectively $|\bullet|$ and $\angle\bullet$. Consequently, we see that the output of the filter with impulse response $h(n)$ may be written

$$y(n) = |H(e^{j\omega_0})|e^{j(\omega_0 n + \angle H(e^{j\omega_0}))}$$

The frequency ω_0 of the pure tone is unchanged from the input frequency. However, the magnitude and phase are altered. We define the magnitude and phase responses of the filter at all frequencies as $|H(e^{j\omega})|$ and $\angle H(e^{j\omega})$ respectively. Figure 10 shows the frequency response (magnitude and phase responses), impulse response, and the input and output at two different frequencies for a tenth-order (Butterworth) recursive digital filter. The selectivity of a filter is a function of the magnitude response—that is, which frequencies are "passed through" with unity gain $|H(e^{j\omega})| = 1$ and which frequencies are "rejected" with zero gain $|H(e^{j\omega})| = 0$. The four classical filtering functions have magnitude responses exemplified in Fig. 3. The magnitude response is given in decibel (dB) units, which may be computed from the gain (magnitude) values as

$$\text{gain}_{\text{dB}} = 20\log_{10}(\text{gain}) \tag{6}$$

Thus, a gain of 1 is 0 dB. Alternatively, the magnitude response can be given in terms of attenuation, measured as $-\text{gain}_{\text{dB}}$. Figure 10(c) shows two important characteristics of all filters: (1) The transient or start-up response is caused by the lower limit on the sum in Eq. (5) being 0 instead of $-\infty$, and (2) the phase response of the system is a *time delay* in the output waveform. The *group delay* of a filter is defined using the phase as

$$\tau = -\frac{d\angle H(e^{j\omega})}{d\omega} \tag{7}$$

The group delay is a measure of the amount of phase distortion introduced by the filter. Ideally, the group delay is a constant function of ω, and so, if the input to the filter is a sum of sinusoids of different frequencies, the phase relationship of the sinusoids is maintained at the filter output. In the example shown in Figs. 5 and 6, we see that the filter delays the first sinusoid by 8.6 samples, and the second sinusoid by 14.5 samples. The fractional samples result in the peak sample amplitude being slightly reduced. Also, because the sinusoids have nearly a 6-sample delay difference, the output of the sum is distorted.

The filter transfer function is the Z transform (ZT) (or discrete Laplace transform) of the impulse response and is directly

$$H(z) = \sum_{n=-\infty}^{\infty} h(n)z^{-n} \tag{8}$$

The complex variable z denotes complex frequency; that is, $z = re^{j\theta}$. Comparing Eq. (8) to Eq. (5), we see that the DTFT is the ZT evaluated on the unit circle in the complex plane $z = e^{j\omega}$. Points in the complex plane where $H(z) = 0$ are called zeros of the filter, while points where $H(z) = \infty$ are called poles. For digital filters described by Eq. (1), the transfer function is always a rational polynomial in z of the form

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{9}$$

The coefficients are the same difference equation coefficients in Eq. (1). Because the filter is causal, the sum in Eq. (8) always converges (exists) for points in the complex plane outside the outermost pole. The causal filter is said to be BIBO stable when the outermost pole lies inside the unit circle—that is, when its radius is less than 1. An alternative method for determining the stability of causal digital filters is the absolute summability condition on the impulse response

$$\sum_{n=0}^{\infty} |h(n)| \leq S < \infty \tag{10}$$

The value of S must be finite for the filter to be stable. The impulse response shown in Fig. 4 is that of an unstable digital filter because the sum in Eq. (10) is infinite.

Typically, all recursive filters are designed to match a desired magnitude response. Ideal classical filters fall into four categories: ideal low-pass filters (ILPF), high-pass filters

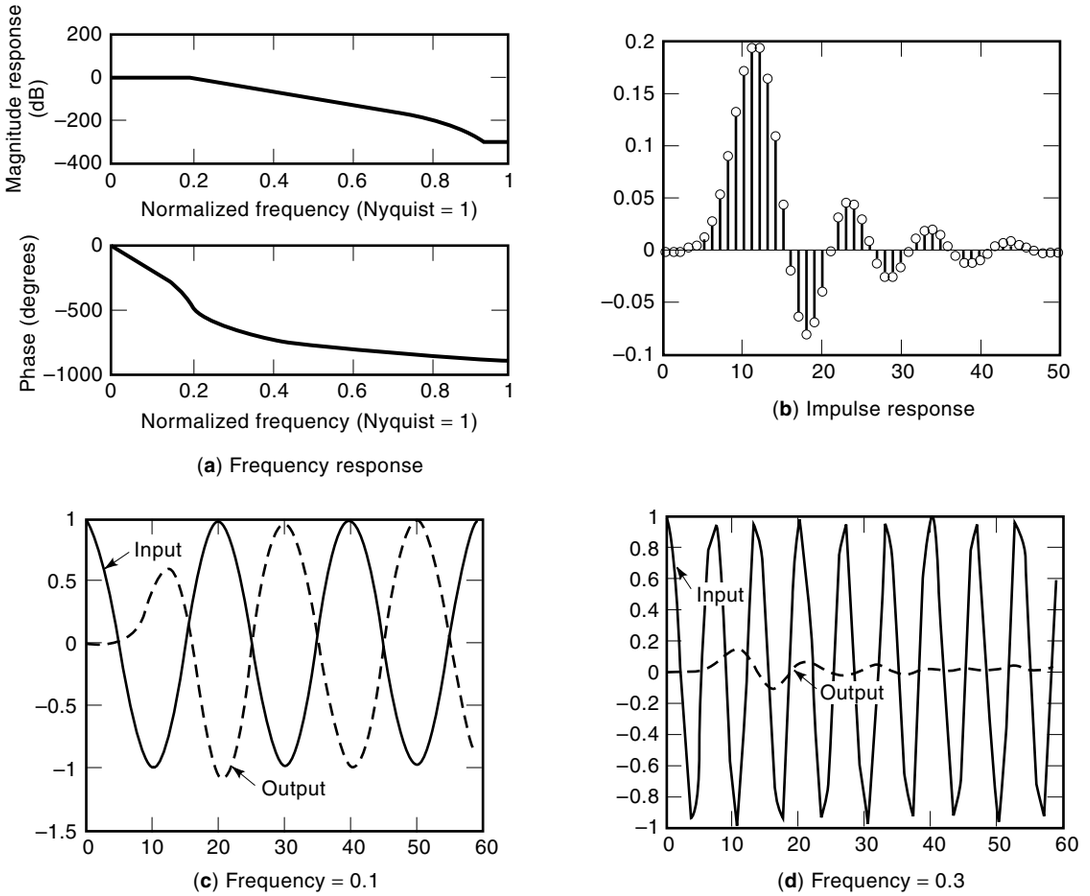


Figure 10. Filtering definitions.

(IHPF), band-pass filters (IBPF), and band-stop filters (IBSF). These four filter types are given by:

1. ILPF: $H_{\text{ILPF}}(e^{j\omega}) = \begin{cases} 1, & \omega < \omega_c \\ 0, & \omega > \omega_c \end{cases}$
2. IHPF: $H_{\text{IHPF}}(e^{j\omega}) = \begin{cases} 0, & \omega < \omega_c \\ 1, & \omega > \omega_c \end{cases}$
3. IBPF: $H_{\text{IBPF}}(e^{j\omega}) = \begin{cases} 1, & \omega_{\text{cl}} < \omega < \omega_{\text{ch}} \\ 0, & \text{otherwise} \end{cases}$
4. IBSF: $H_{\text{IBSF}}(e^{j\omega}) = \begin{cases} 0, & \omega_{\text{cl}} < \omega < \omega_{\text{ch}} \\ 1, & \text{otherwise} \end{cases}$

The *bandwidth* of a filter is specified as the width of the positive frequency range where the gain is greater than some specified value. For example, when using the 3 dB bandwidth measure, the gain threshold is 3 dB below the maximum filter gain. If life were simple, then we would just implement the above filters directly and be done with the problem. However, all causal filters must satisfy the Payley–Wiener (1) conditions:

1. The magnitude response $|H(e^{j\omega})|$ of a filter cannot be zero except at a finite number of frequencies.
2. The magnitude response $|H(e^{j\omega})|$ of a filter cannot be constant over any finite range of frequencies.
3. The magnitude response $|H(e^{j\omega})|$ of a filter cannot be discontinuous at any frequency.
4. The magnitude response $|H(e^{j\omega})|$ and the phase response $\angle H(e^{j\omega})$ of a filter are not independent of each other.

What these conditions mean as far as digital filter designs are concerned is that:

1. Ideal responses cannot be achieved.
2. Optimization techniques must be employed to find the causal filter whose magnitude response is closest to the desired filter response. Because of the general form of a causal recursive digital filter [see Eq. (1)], these optimization techniques are nonlinear, and consequently many different techniques with different convergence and design routes are available for the practicing engineer to use. The engineer must choose the design technique for which available tools are suitable and for which a given set of specifications can be matched.

One other problem in the design of recursive digital filters that must always be considered is the phase response of the designed system. In practice, the phase response of the filter is ignored until the design is complete—that is, until design techniques match only the magnitude response. Consequently, after the magnitude response is matched as closely

as is practical, the phase response must be checked. Sometimes, special remedies like phase compensators are needed. Because the phase responses of Butterworth and Chebyshev II filters are nearly linear over the pass-band of the filter, the group delay is nearly constant. However, the Chebyshev I and elliptic filters are equal-ripple in the pass-band, causing a nonlinear phase response. To reduce the distortion, an all-pass filter can be cascaded with the original recursive filter. The magnitude response of the cascade is unaltered (because the all-pass filter gain is unity), but the phase response is equal to the sum of the phase response of the original and all-pass filters. The phase response (and order) of the all-pass filter can be optimized to minimize the overall phase distortion.

DESIGN METHODS

We discuss two basic approaches to designing recursive digital filters. First, we define certain special filter types that will be useful for developing further ideas or that are useful directly in their own right. These special-type filters are often designed by directly manipulating the pole and zero locations.

Special Types

All-Pass. Consider the first-order system with transfer function

$$H_{\text{ap}}(z) = \frac{z^{-1} - a*}{1 - az^{-1}} = -a^* \frac{z - \frac{1}{a^*}}{z - a} \tag{11}$$

The asterisk (*) indicates conjugation. This system has a pole at $z = a$ and a zero at the conjugate reciprocal location $z = 1/a^*$, both of which lie on the same radius in the complex z plane, mirrored about the unit circle. For instance, if the filter pole is at $a = re^{j\theta}$, then the filter zero is at $1/a^* = r^{-1}e^{j\theta}$. Now, consider that the conjugate of the denominator of the transfer function evaluated along points of the unit circle (i.e., points of the frequency response) is $z(z^{-1} - a^*)$, which is z times the numerator. Consequently, the magnitude response of this filter is 1 at all frequencies. However, the phase response is a nontrivial function of the pole and zero. In this first-order case, we can determine

$$H_{\text{ap}}(e^{j\omega}) = e^{-j(\omega + 2\tan^{-1}[r\sin\theta/(1 - r\cos(\omega - \theta))])}$$

A general nth-order, real all-pass system has the form

$$H_{ap}(z) = \frac{a_n + a_{n-1}z^{-1} + \cdots + a_0 z^{-n}}{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}} \tag{12}$$

Resonators and Oscillators. A *resonator* is a two-pole band-pass filter with a pair of complex-conjugate poles ($z = re^{\pm j\omega_0}$) near the unit circle, which has a transfer function

$$H_R(z) = \frac{b_0}{1 - 2r\cos\omega_0 z^{-1} + r^2 z^{-2}} \tag{13}$$

An oscillator is a resonator whose poles are on the unit circle (i.e., whose poles are $z = e^{\pm j\omega_0}$), with the transfer function

$$H_O(z) = \frac{A\sin\omega_0}{1 - 2\cos\omega_0 z^{-1} + z^{-2}} \tag{14}$$

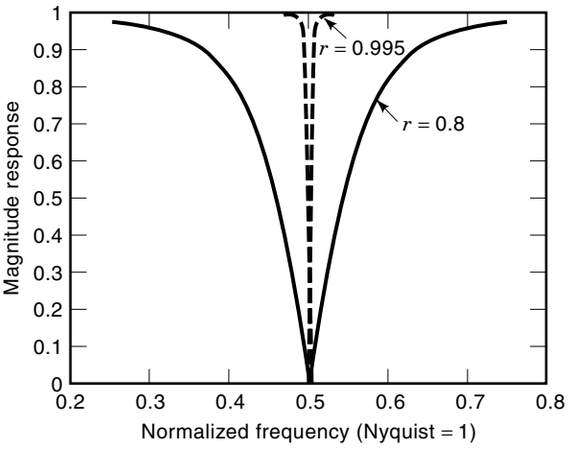


Figure 11. Pole placement effect on the filter quality (frequency selectivity).

Coupled oscillators are used in phase-quadrature modulation communication systems. These modulators generate sine and cosine oscillations simultaneously and are most often implemented using a CORDIC algorithm of Hu (2).

Notch and Comb Filters. A *notch* filter is a band-stop filter with a pair of complex-conjugate zeros ($z = e^{\pm j\omega_0}$) on the unit circle and a pair of resonant poles which has the transfer function

$$H_N(z) = \frac{1 - 2\cos\omega_0 z^{-1} + z^{-2}}{1 - 2r\cos\omega_0 z^{-1} + r^2 z^{-2}} \tag{15}$$

The bandwidth of the notch is determined by the value of r; that is, the closer r is to the unit circle, the narrower the notch, as exemplified in Fig. 11. A *comb* filter is a high order notch filter that has notches uniformly spaced around the unit circle.

Direct Designs

Least-Squares Techniques. Consider the matrix equation

$$Ha = g + e \tag{16}$$

H is an $L \times N$ matrix (i.e., it has L rows and N columns), a is an $N \times 1$ column vector of unknowns (the "solution" if you will), g is an $L \times 1$ column vector of "ideal" values or equalities, and e is an $L \times 1$ column vector of "errors." The matrix equation given has two practical cases:

1. If $L = N$, then the square matrix equation $Ha = g$ has a unique solution, $a = H^{-1}g$ and the error vector is zero (assuming that H^{-1} exists).
2. If $L > N$, then in general no solution yields an error vector equal to zero, and we must find a solution which minimizes the error by some measure.

The most common solution to the second practical case is the method of finding the least-squared error:

$$E \equiv e^t e = (Ha - g)^t (Ha - g) = a^t H^t H a - 2a^t H^t g + g^t g \tag{17}$$

The superscript t denotes transpose. To minimize E, we take its derivative and set it equal to zero; that is,

$$\frac{\partial E}{\partial a} = 0 = 2H^t Ha - 2H^t g \tag{18}$$

Evaluation of Eq. (18) results in the solution

$$a = (H^t H)^{-1} H^t g \equiv H^{\#} g \tag{19}$$

The solution is a minimum because the second derivative is $\partial^2 E/\partial a^2 = H^t H > 0$ (i.e., the second derivative is positive definite). The orthogonality principle of the least-squares problem results from a rewriting of Eq. (18). Consider that $H^t Ha - H^t g = H^t e = 0$. Thus, the error is orthogonal to the columns of H.

The Autocorrelation and Covariance Methods. These methods will determine autoregressive (AR) filters only. Suppose that $H(z) = 1/A(z)$, where $A(z)$ is the denominator polynomial given in Eq. (9). Then we know that $H(z)A(z) = 1$, and taking the inverse ZT we have $h(n)^* a_n = \delta(n)$, which is valid for all $n \geq 0$. Writing the first $L + 1$ equations and putting them in matrix form results in

$$\begin{bmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & h(0) & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ h(N) & n(N-1) & \cdots & h(0) \\ \vdots & \ddots & \ddots & \vdots \\ h(L) & h(L-1) & \cdots & h(L-N) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{20}$$

The LS solution of Eq. (20) is called the *covariance method* of Makhoul (3). Now consider that the covariance method solution can be written as $a = (H^t H)^{-1} h(0) u_1$, where u_1 is the vector on the right-hand side of Eq. (20). Define $\Phi_N = H^t H$ so that $a = h(0)\Phi_N^{-1} u_1$. Now

$$\Phi_N = \begin{bmatrix} \sum_{n=0}^{L} h^2(n) & \sum_{n=0}^{L-1} h(n)h(n+1) & \cdots & \sum_{n=0}^{L-N} h(n)h(n+N) \\ \sum_{n=0}^{L-1} h(n)h(n+1) & \sum_{n=0}^{L-1} h^2(n) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sum_{n=0}^{L-N} h(n)h(n+1) \\ \sum_{n=0}^{L-N} h(n)h(n+N) & \cdots & \sum_{n=0}^{L-N} h(n)h(n+1) & \sum_{n=0}^{L-N} h^2(n) \end{bmatrix} \tag{21}$$

Examining Eq. (21), we can see that on each diagonal the further right and down we go, the fewer terms are present in the sum. More accuracy could be achieved by replacing all diagonal elements to the right and down with the leftmost, top element. Consider

$$R_N \equiv \begin{bmatrix} \sum_{n=0}^{L} h^2(n) & \sum_{n=0}^{L-1} h(n)h(n+1) & \cdots & \sum_{n=0}^{L-N} h(n)h(n+N) \\ \sum_{n=0}^{L-1} h(n)h(n+1) & \sum_{n=0}^{L} h^2(n) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sum_{n=0}^{L-N} h(n)h(n+1) \\ \sum_{n=0}^{L-N} h(n)h(n+N) & \cdots & \sum_{n=0}^{L-1} h(n)h(n+1) & \sum_{n=0}^{L} h^2(n) \end{bmatrix} \tag{22}$$

Then we have the *autocorrelation method* solution (3)

$$a = h(0) R_N^{-1} u_1 \tag{23}$$

To close this subsection, we consider the major performance differences using the desired impulse response $h(n) = 1.1^n u(n)$. Consider first the covariance method design. Application of Eq. (20) with $L = 100$ and $N = 1$ yields $A_{\text{COV}}(z) = 0.9975 - 1.0972z^{-1}$, which has a root (a filter pole) at $z = 1.1$, the exact location needed to match $h(n)$ exactly. However, this is an unstable design. Application of Eq. (23) yields $A_{AC}(z) = 1.000 - 0.9100z^{-1}$, which has a root at $z = 0.91$. While this solution does not precisely match the desired impulse response, it is stable. Note that the pole has been reflected about the unit circle. We state without proof [proof may be found in many places; for example, see Hayes (4)] that the autocorrelation method always yields stable designs, while the covariance method yields the most accurate (with respect to matching the impulse response).

The Prony Method. In the Prony method as developed by Burrus and Parks (5), we allow the filter to have zeros as well as poles. In this situation we rewrite the ARMA transfer function as $H(z)A(z) = B(z)$, which has the inverse transform relation $h(n)^* a_n = b_n$. This entails a rewriting of Eq. (20) to remove the influence of the zeros as

$$\begin{bmatrix} h(M) & h(M-1) & \cdots & h(M-N+1) \\ h(M+1) & h(M) & \cdots & h(M-N+2) \\ \vdots & \ddots & \ddots & \vdots \\ h(M+N) & h(M+N-1) & \cdots & h(M+1) \\ \vdots & \ddots & \ddots & \vdots \\ h(M+L) & h(M+L-1) & \cdots & h(M+L-N+1) \end{bmatrix}$$

$$\begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} = - \begin{bmatrix} h(M+1) \\ h(M+2) \\ \vdots \\ h(M+L+1) \end{bmatrix} \tag{24}$$

To formulate the equations, we have assumed that $a_0 = 1$. The numerator coefficients in this method are determined directly using the relationship

$$b_n = h(n) + \sum_{k=1}^{N} a_k h(n-k), \quad n = 0, 1, \ldots, M \tag{25}$$

The solution of Eq. (25) produces zero error in the first M terms of the impulse response (a Pade approximation). However, this is not the only solution possible, and several variants are possible. The Prony method is not guaranteed to yield stable results.

The Stieglitz–McBride Method. The Stieglitz–McBride method (6,7) minimizes the difference between the desired and designed filter impulse responses in an iterative fashion. This is in contrast to the Prony method solution given above. Define the matrices

$$H = \begin{bmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & h(0) & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ h(N) & h(N-1) & \cdots & h(0) \end{bmatrix}$$

$$\hat{H} = \begin{bmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & h(0) & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ h(M) & h(M-1) & \cdots & h(0) \\ \vdots & \vdots & \ddots & \vdots \\ h(N) & h(N-1) & \cdots & h(N-M) \end{bmatrix}$$

(The first $M+1$ columns of H)

$H([a])$ and $\hat{H}([a])$ are the corresponding matrices using the impulse response of the AR "inverse" filter $1/(a_0 + \sum_{k=1}^{N} a_k z^{-k})$. We have not scaled the zeroth-order coefficient to unity.

The iteration on the AR parameters is

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}_{i+1} = \left(H([a]_i)H\right)^{\#} \hat{H}([a]_i)\left(\hat{H}([a]_i)\right)^{\#} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(N) \end{bmatrix} \tag{26}$$

The numerator coefficients are then determined after convergence of Eq. (26):

$$\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix} = \hat{H}([a])^{\#} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(N) \end{bmatrix} \tag{27}$$

A good initial starting vector in the iteration can be obtained using Eq. (24). Overestimation of the number of poles can result in a failure of the denominator coefficients to converge in the iterations. Otherwise, the Steiglitz–McBride method converges quite rapidly, often after only a few iterations.

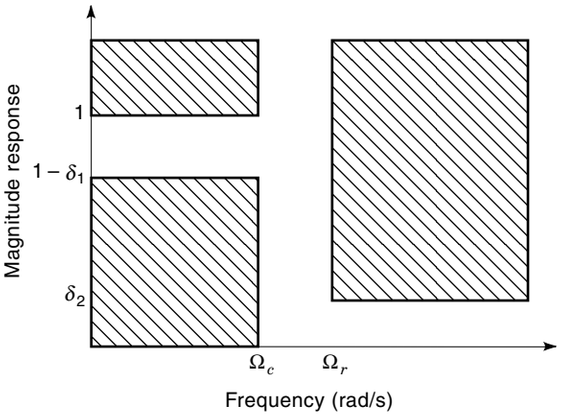


Figure 12. Low-pass filter specifications showing passband ripple and corner frequency and stop-band attenuation and corner frequency.

Designs from Analog Prototypes

Designing Analog Filters. One approach to the design of recursive digital filters is to use the full body of knowledge developed over the many years of analog filter design, which are by nature recursive. The typical approach is to design a low-pass filter according to the magnitude response specifications as indicated in Fig. 12. An analog filter has the transfer function

$$H_a(s) = \frac{\sum_{k=0}^{M} f_k s^k}{1 + \sum_{k=1}^{N} g_k s^k} = \frac{F(s)}{G(s)}$$

The subscript a differentiates a continuous-time function from the discrete-time function. Also,

$$H_a(s) = \int_{-\infty}^{\infty} h_a(t) e^{-st}\, dt$$

is the Laplace transform of the impulse response of the continuous-time filter. Because the filter is analog, $N \geq M$. The filter is stable and causal if the poles of the analog transfer function have negative real parts.

Butterworth. Butterworth filters have a maximally flat magnitude response, and they consequently possess a phase response that is close to linear. The main drawback is the large transition region for any given order. The magnitude-squared response is

$$|H_a(j\Omega)| = \frac{1}{1 + \left(\dfrac{j\Omega}{j\Omega_c}\right)^{2N}} \tag{28}$$

We have used Ω to denote real (analog) frequency. In the s domain, we have

$$H_a(s)H_a(-s) = \frac{1}{1 + \left(\dfrac{s}{j\Omega_c}\right)^{2N}} \tag{29}$$

Ω_c is the 3 dB frequency. To determine the poles, we set the denominator of Eq. (29) equal to zero and solve, yielding

$$s_k = \Omega_c e^{j(\pi/2+\pi(2k-1)/2)}, \qquad k = 1, 2, \ldots, 2N \tag{30}$$

All of the poles lie on a circle in the s-plane of radius Ω_c centered at the origin. Half of the poles have negative real parts. The other half are mirrored about the $j\Omega$ axis and thus have positive real parts. To obtain a stable and causal filter, $H_a(s)$ contains only the poles with negative real parts, with $H_a(-s)$ containing the remaining half. The filter has no finite zeros. To determine the order required to meet design specifications as shown in Fig. 12, consider that

$$\delta_2 = \frac{1}{1 + \left(\dfrac{j\Omega_r}{j\Omega_c}\right)^{2N}}$$

Solving for N, we can see that we must require the integer order N to be such that

$$N \geq \frac{\log_{10}\left(\dfrac{1}{\delta_2^2} - 1\right)}{2\log_{10}\left(\dfrac{\Omega_r}{\Omega_c}\right)} \tag{31}$$

Chebyshev. A more rapid roll-off of the magnitude response at the cut-off frequency can be achieved with the magnitude-squared response of the type I Chebyshev filter:

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \epsilon^2 T_N^2\left(\dfrac{\Omega}{\Omega_c}\right)} \tag{32}$$

This filter has an equal-ripple response in the pass-band and has a flat response in the transition and stop bands. The Chebyshev polynomial $T_N(x) = \cosh(N\cosh^{-1}x)$ can be generated recursively using the relation (and initial polynomials $T_0(x) = 1$ and $T_1(x) = x$)

$$T_{N+1}(x) = 2xT_N(x) - T_{N-1}(x), \qquad N = 2, 3, \ldots \tag{33}$$

Since $T_N(1) = 1$ for all N, we can solve Eq. (32) for ϵ, yielding

$$\epsilon^2 = \frac{1}{(1-\delta_1)^2} - 1 \tag{34}$$

Similarly to the Butterworth derivation, we can also determine that the integer order must satisfy

$$N \geq \frac{\cosh^{-1}\left(\dfrac{1}{\delta_2\epsilon}\right)}{\cosh^{-1}\left(\dfrac{\Omega_r}{\Omega_c}\right)} \tag{35}$$

The poles lie equally spaced on an ellipse centered about the origin of the s plane. The filter has no finite zeros. The pole locations are determined using the variables

$$\alpha = \frac{1}{N}\sinh^{-1}\left(\frac{1}{\epsilon}\right) \tag{36}$$

$$\beta_m = \begin{cases} \pi\dfrac{2m+1}{2N}, & m = 0, 1, \ldots, \dfrac{N}{2} - 1;\ N \text{ even} \\ \pi\dfrac{2m+1}{2N}, & m = 0, 1, \ldots, \dfrac{N-1}{2} - 1;\ N \text{ odd} \end{cases} \tag{37}$$

The poles of the analog filter are then located at the N locations in the s-plane:

$$s_m = -\sinh(\alpha)\sin(\beta_m) \pm j\cosh(\alpha)\cos(\beta_m) \tag{38}$$

If the filter order is odd, then one of the poles is real and located at the s-plane location $s = -\sinh(\alpha)$. A type II Chebyshev filter has magnitude-squared response

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \epsilon^2\left[\dfrac{T_N^2\left(\dfrac{\Omega_r}{\Omega_c}\right)}{T_N^2\left(\dfrac{\Omega_r}{\Omega}\right)}\right]} \tag{39}$$

This filter has an equal-ripple response in the stop-band, with a flat response in the pass-band. The relationships given in Eqs. (34) and (35) remain valid. The poles do not lie on any simple geometric shape as before, and this filter has finite zeros that lie on the imaginary axis of the s-plane. Using the definitions of Eqs. (36) and (37), we have the pole locations

$$\begin{aligned} s_m = &\frac{-\sinh(\alpha)\sin(\beta_m)}{[\sinh(\alpha)\sin(\beta_m)]^2 + [\cosh(\alpha)\cos(\beta_m)]^2} \\ &\pm j\frac{\cosh(\alpha)\cos(\beta_m)}{[\sinh(\alpha)\sin(\beta_m)]^2 + [\cosh(\alpha)\cos(\beta_m)]^2} \end{aligned} \tag{40}$$

If the filter order is odd, then one of the poles is the single real pole at $s = -1/\sinh(\alpha)$. As for the finite zeros, they occur at the locations

$$s = \pm j\sec(\beta_m) \tag{41}$$

If the filter order is odd, then the filter will have one infinite zero in addition to the $N - 1$ listed in Eq. (41).

Elliptic. Elliptic (or Cauer) filters display equal-ripple responses in both the pass and stop bands. For a given filter order, allowable pass-band ripple, and stop-band rejection, no other filter type can provide a narrower transition region. The magnitude-squared response is

$$|H_a(j\Omega)|^2 = \frac{1}{1 + \epsilon^2 U_N^2\left(\dfrac{\Omega}{\Omega_c}\right)} \tag{42}$$

The Jacobian elliptic function $U_N(x)$ must be computed numerically via computer (or looked-up in a table; e.g., see Refs. 8–10). Software programs are now almost exclusively relied

upon to perform these filter designs. To locate the poles and zeros of the filter, we need to first define a few terms:

- *Incomplete elliptic integral of the first kind:* $IEI(\phi, k) = \int_0^{\phi} (1 - k^2 \sin^2(x))^{-1/2}\, dx$
- *Elliptic sine:* $\mathrm{sn}(u, k) = \sin(\phi)$
- *Elliptic cosine:* $\mathrm{cn}(u, k) = \cos(\phi)$
- *Elliptic tangent:* $\mathrm{sc}(u, k) = \tan(\phi)$

The *complete elliptic integral of the first kind* is defined in terms of the *IEI*:

$$CEI(k) = IEI\left(\frac{\pi}{2}, k\right) \tag{43}$$

The order of the elliptic filter required to meet the specifications is

$$N \geq \frac{CEI\left(\frac{\Omega_c}{\Omega_r}\right) CEI\left(\sqrt{1 - \frac{\epsilon^2}{\frac{1}{\delta_2} - 1}}\right)}{CEI\left(\frac{\epsilon}{\sqrt{\frac{1}{\delta_2} - 1}}\right) CEI\left(\sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right)} \tag{44}$$

The pole and zero locations are then determined:

$$\begin{aligned} &\mathrm{cn}\left(\beta_m, \frac{\Omega_c}{\Omega_r}\right) \mathrm{dn}\left(\beta_m, \frac{\Omega_c}{\Omega_r}\right) \\ &s_m = -\frac{\mathrm{sn}\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right) \mathrm{cn}\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right)}{1 - \mathrm{dn}^2\left(\beta_m, \frac{\Omega_c}{\Omega_r}\right) \mathrm{sn}^2\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right)} \\ &\pm j \frac{\mathrm{dn}\left(\beta_m, \frac{\Omega_c}{\Omega_r}\right) \mathrm{dn}\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right)}{1 - \mathrm{dn}^2\left(\beta_m, \frac{\Omega_c}{\Omega_r}\right) \mathrm{sn}^2\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right)} \end{aligned} \tag{45}$$

We have used

$$\alpha = \frac{CEI\left(\frac{\Omega_c}{\Omega_r}\right) \mathrm{sc}^{-1}\left(\frac{1}{\epsilon}, \frac{\epsilon}{\sqrt{\frac{1}{\delta_2} - 1}}\right)}{N \cdot CEI\left(\epsilon\sqrt{\frac{1}{\delta_2} - 1}\right)} \tag{46}$$

$$\beta_m = \begin{cases} \dfrac{(2m+1)CEI\left(\frac{\Omega_c}{\Omega_r}\right)}{N}, & m = 0, 1, \ldots, \dfrac{N-1}{2} - 1, N \text{ odd} \\ \dfrac{(2m+2)CEI\left(\frac{\Omega_c}{\Omega_r}\right)}{N}, & m = 0, 1, \ldots, \dfrac{N}{2} - 1, N \text{ even} \end{cases} \tag{47}$$

In the case where the filter order N is odd, the real pole is located at

$$s_m = -\frac{\mathrm{sn}\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right) \mathrm{cn}\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right)}{1 - \mathrm{sn}^2\left(\alpha, \sqrt{1 - \left(\frac{\Omega_c}{\Omega_r}\right)^2}\right)} \tag{48}$$

The zeros are purely imaginary:

$$s_m = \pm j \frac{1}{\left(\frac{\Omega_c}{\Omega_r}\right) \mathrm{sn}\left(\beta_m, \left(\frac{\Omega_c}{\Omega_r}\right)\right)} \tag{49}$$

Example: Suppose we wish to design low-pass filters with the following specifications:

- Passband ripple of 0.25 dB
- Stopband rejection of 50 dB
- Passband cutoff at 1000 Hz
- Stopband starts at 1500 Hz
- Sampling rate is 10 kHz

The required order for each filter type described above is computed: Butterworth 16, Chebyshev I and II 8, and Elliptic 5. We have shown the resulting magnitude responses in Fig. 13. The equal-ripple natures of the Chebyshev and elliptic filters can be seen in both the pass band and stop band zooms. The elliptic filter is the most efficient design, but the alternations in the magnitude response across the pass-band create significant distortion in the phase response, and thus create significantly different group delays.

Analog-Domain to Digital-Domain Conversions. Once we have an analog prototype, we must somehow convert $H_a(s)$ to the desired digital filter $H(z)$. We desire a mapping of the s-plane into the z-plane that preserves both of the following:

1. *Stability.* The half-plane of s with negative real parts must map to the interior of the unit circle of the z plane.
2. *Frequency response.* The imaginary axis in the s plane must map to the unit circle of the z plane.

The two most commonly used methods [and apparently the most effective according to Jackson (11) and Rabiner and Gold (12)] for accomplishing these tasks are impulse invariance and the bilinear transformation. We consider them in order.

Impulse Invariance. The idea in impulse invariance is relatively straightforward: Let $h(n) = h_a(nT)$; that is, set the impulse response of the discrete-time filter equal to a sampled version of the analog filter impulse response. Because this is sampling, the frequency response of the digital filter is the aliased spectrum

$$H(e^{j\omega}) = \frac{1}{T} \sum_{k=0}^{\infty} H_a\left(j\frac{\Omega}{T} + j\frac{2\pi}{T}k\right) \tag{50}$$

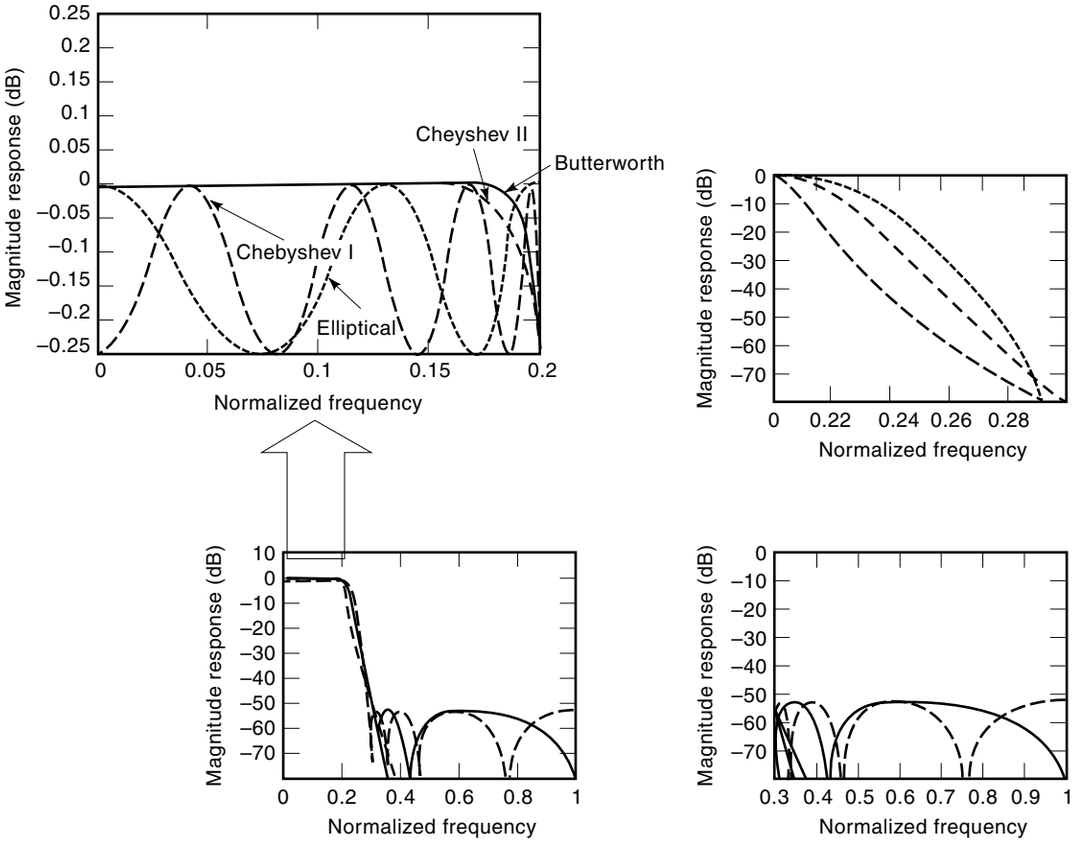


Figure 13. Classical design magnitude responses showing flat and equiripple responses.

The mapping is many-to-one from the s plane to the z plane. Consequently, no inverse mapping exists. With regard to stability, consider that Eq. (50) implies that $z = e^{sT}$. So, assuming that $s = \sigma + j\Omega$, we have $z = e^{\sigma T}e^{j\Omega T} \equiv re^{j\phi}$. Now we can see that for $\sigma < 0$ we have $r < 1$, so the stability is preserved (condition 1 above). One can easily verify that the imaginary axis of the s plane maps to the unit circle of the z plane (condition 2 above). A word of caution: The imaginary axis of the s plane wraps around and around the unit circle of the z plane (the aliasing) without end. To view impulse invariance in another way, consider the partial fraction expansion of the continuous-time transfer function with simple poles (multiple poles can be dealt with; see the discussion on the parallel form in a later section):

$$H_a(s) = \sum_{k=1}^{N} \frac{A_k}{s - s_k}$$

Taking the inverse Laplace transform yields

$$h_a(t) = \sum_{k=1}^{N} A_k e^{s_k t} u(t)$$

so that

$$h(n) = \sum_{k=1}^{N} A_k e^{s_k nT} u(nT)$$

Thus

$$H(z) = \sum_{k=1}^{N} \frac{A_k}{1 - e^{s_k T} z^{-1}}$$

While the poles are mapped from the s plane to the z plane via the transformation $z = e^{sT}$, the zeros are not. They are functions of the A_k and s_k for each k. To preserve scale, we often multiply through by T to eliminate the division in Eq. (50), resulting in

$$H(z) = \sum_{k=1}^{N} \frac{TA_k}{1 - e^{s_k T} z^{-1}} \tag{51}$$

Because of the aliasing inherent to the method, only band-limited filter designs are suitable for impulse invariance designs. Furthermore, pass-band ripple cannot be preserved, so that elliptic and type II Chebyshev filter designs cannot be performed (in general).

Bilinear Transformation. In contrast to the impulse invariant technique, the bilinear transformation is a one-to-one mapping of the s plane onto the z plane. The problem in the impulse invariance method is the aliasing of the spectrum. To eliminate the aliasing potential, we "compress" the s plane into a strip around the real axis of the s plane as shown

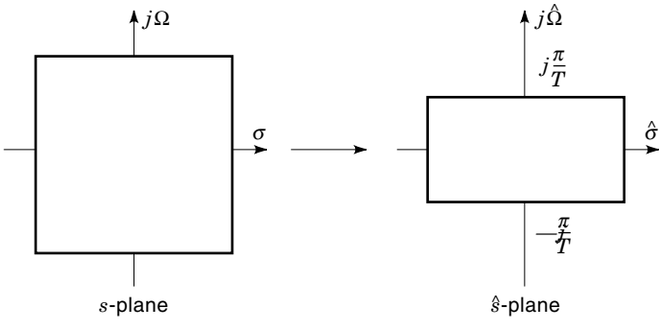


Figure 14. Bilinear transformation mapping of the entire s-plane into the compressed s-plane to prevent aliasing.

in Fig. 14. The compression is a result of the mapping $\hat{s} = (2/T)\tanh^{-1}(sT/2)$. Then $s = \pm\infty \rightarrow \hat{s} = \pm j(\pi/T)$ and $s = 0 \rightarrow \hat{s} = 0$. Now, if we take the mapping $z = e^{\hat{s}T}$, we have $\hat{s} = (1/T)\ln z$. Using the fact that $s = (2/T)\tanh(\hat{s}T/2)$, we see

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{52}$$

Thus, different from the impulse invariance technique, the bilinear transformation is an algebraic one-to-one transformation that is invertible; that is,

$$z = \frac{1 + \dfrac{T}{2}s}{1 - \dfrac{T}{2}s} \tag{53}$$

Let $z = re^{j\phi}$. Then

$$s = \frac{2}{T}\frac{r^2 - 1}{1 + r^2 + 2r\cos\phi} + j\frac{2}{T}\frac{2r\sin\phi}{1 + r^2 + 2r\cos\phi} \equiv \sigma + j\Omega$$

We can now see that for $\sigma < 0$ we have $r < 1$, so the stability is preserved (condition 1 above). Also, the frequency response is preserved because the real part of s vanishes when $r = 1$ (condition 2 above). In this case, we have

$$\Omega = \frac{2}{T}\frac{2\sin\omega}{2(1 + \cos\omega)} = \frac{2}{T}\tan\frac{\omega}{2} \tag{54}$$

Equation (54) is called the *frequency warping* of the bilinear transformation. To account for the warping, the analog frequency design specification must be *prewarped.* One significant problem of the bilinear transformation is that the infinite poles and zeros of analog filters are mapped to $z = -1$ in $H(z)$. Thus, Butterworth designs that have N zeros at infinity will have N zeros at $z = -1$ in $H(z)$.

Frequency Transformations. To design any of the four classical filter shapes, we must determine methods for converting an original low-pass filter to any of the other shapes. Essentially, we look for transformations that accomplish the following:

$$LPF \leftrightarrow \begin{cases} LPF \\ HPF \\ BPF \\ BSF \end{cases}$$

To accomplish these transformations, we need to find a mapping of the z plane such that the designed *LPF* $H(z)$ transforms to the desired frequency response $H_d(z)$. Define the mapping transformation

$$\hat{z} = G(z) \tag{55}$$

This mapping must preserve

1. *Stability:* the interior of the unit circle in z must map to the interior of the unit circle of $\hat{z}$
2. *Causality:* the unit circle in z must map to the unit circle of $\hat{z}$

The second condition implies that $G(z)$ should be an allpass system. This can readily be seen from a close examination of Eq. (55) and taking the absolute value of both sides. Any mapping $G(z)$ that is of the form given in Eq. (12) will accomplish frequency transformation. The most typical transformations are given in Table 1. Derivations of these formulae can be found many places—in particular, Constantinides (13). As an example, consider the *LPF* system with pole $z = c$ and zero $z = -1$:

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

Using the second row of Table 1, we transform the *LPF* to an *HPF* as

$$H_d(z) = \frac{1 - \left(\dfrac{1 - \xi z^{-1}}{z^{-1} - \xi}\right)^{-1}}{1 + c\left(\dfrac{1 - \xi z^{-1}}{z^{-1} - \xi}\right)^{-1}} = \frac{1 + \xi}{1 - c\xi}\frac{1 - z^{-1}}{1 + \dfrac{c - \xi}{1 - c\xi}z^{-1}}$$

The filter $H_d(z)$ has a zero at $z = 1$ and a pole at $z = -(c - \xi)/(1 - c\xi)$. Note that poles and zeros map according to the table transformation. Consequently, the transformations can be accomplished by either (a) transforming the poles and zeros according to the frequency transform and reconstructing the transfer function or (b) direct substitution of the frequency transformation into the original *LPF* transfer function.

FINITE PRECISION EFFECTS AND FILTER STRUCTURES

Because all digital filters are implemented using digital hardware that performs mathematical computations using a fixed number of bits, certain losses due to rounding (quantization) are expected. The ordering of the computations used to compute the output values from the input values and previous

Table 1. Frequency Transformations

$H_d(z)$	$G(z)$	Parameters
Low-pass with cutoff $\hat{\omega}_c$	$\dfrac{1-\alpha z^{-1}}{z^{-1}-\alpha}$	$\alpha = \dfrac{\sin\left(\dfrac{\omega_c - \hat{\omega}_c}{2}\right)}{\sin\left(\dfrac{\omega_c + \hat{\omega}_c}{2}\right)}$
High-pass with cutoff $\hat{\omega}_c$	$-\left(\dfrac{1-\xi z^{-1}}{z^{-1}-\xi}\right)$	$\xi = \dfrac{\cos\left(\dfrac{\omega_c + \hat{\omega}_c}{2}\right)}{\cos\left(\dfrac{\omega_c - \hat{\omega}_c}{2}\right)}$
Band-pass with cutoffs $\hat{\omega}_{c_l}, \hat{\omega}_{c_h}$	$-\left(\dfrac{1+\alpha z^{-1}+\beta z^{-2}}{z^{-2}+\alpha z^{-1}+\beta}\right)$	$K = \cot\left(\dfrac{\hat{\omega}_{c_h} - \hat{\omega}_{c_l}}{2}\right)\tan\left(\dfrac{\hat{\omega}_c}{2}\right)$ $\xi = \dfrac{\cos\left(\dfrac{\omega_c + \hat{\omega}_c}{2}\right)}{\cos\left(\dfrac{\omega_c - \hat{\omega}_c}{2}\right)}$ $\alpha = \dfrac{-2\xi K}{K+1}, \beta = \dfrac{K-1}{K+1}$
Band-stop with cutoffs $\hat{\omega}_{c_l}, \hat{\omega}_{c_h}$	$\dfrac{1+\alpha z^{-1}+\beta z^{-2}}{z^{-2}+\alpha z^{-1}+\beta}$	$K = \cot\left(\dfrac{\hat{\omega}_{c_h} - \hat{\omega}_{c_l}}{2}\right)\tan\left(\dfrac{\hat{\omega}_c}{2}\right)$ $\xi = \dfrac{\cos\left(\dfrac{\omega_c + \hat{\omega}_c}{2}\right)}{\cos\left(\dfrac{\omega_c - \hat{\omega}_c}{2}\right)}$ $\alpha = \dfrac{-2\xi}{K+1}, \beta = \dfrac{1-K}{1+K}$

output values is critical to filter performance. In this section, we first detail mathematical descriptions of filters so that these round-off errors can be analyzed. We then examine the physical natures of the round-off errors and finally close with the filter implementation synthesis methods incorporating the round-off analysis.

Mathematical Descriptions

Round-off errors that occur in the computation of a filter output result from the filter coefficients (the design) and from the internal (partial result) signals that are a natural result of the computational order. These two error types require the filter designer to be able to specify the coefficients, the order of computation, and the internal signals resulting from these computations.

Signal Flow Graphs. A graphical technique for specifying filter implementations is the signal flow graph (SFG). This directed graph has two basic elements: nodes and directed edges called "branches." Nodes are signals, and branches specify computations and timing. For our purposes, only the three groups shown in Fig. 15 are required: sum, amplify, and delay. The SFG describes a method of computing the system output for any input sequence. For example, the second-order filter

$$H(z) = \frac{r}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{56}$$

has an SFG shown in Fig. 16. This SFG is essentially the algorithm for directly computing the difference equation of the digital filter. We do not have to be so restrictive in computing the filter output. To understand how we can be less restrictive, we introduce the concept of state-space digital filters.

State-Space. The state of a system is its memory. Knowing the state of a system means that we know what happened previously, so that future outputs can be computed from the

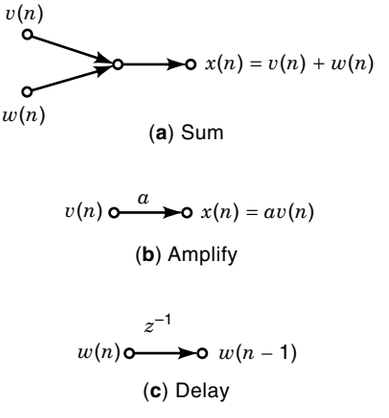


Figure 15. Signal flow graph components.

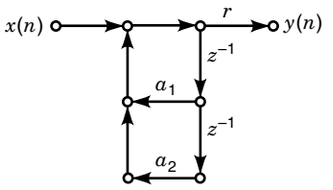


Figure 16. Second-order digital filter SFG.

states and present input without actually having to know what any of the past inputs were. Therefore, since the system in Eq. (56) requires the retention of two past outputs, we need two states to describe the second-order system. The minimum number of states required to describe a filter is equal to the order of the filter, though more states may be used. In any SFG, the natural states are the nodes at the arrowhead ends of the delay branches. In our example, if we use the top delay branch to define state one ($v_1(n)$) and the bottom delay branch to define state two ($v_2(n)$), we can write the two state equations in matrix form:

$$\begin{bmatrix} v_1(n+1) \\ v_2(n+1) \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} + \begin{bmatrix} r \\ 0 \end{bmatrix} x(n) \tag{57}$$

The state equations relate the current state to the "next" state. We also need to define the output in terms of the state and input, which is accomplished via the output equation

$$y(n) = [a_1 \quad a_2] \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} + rx(n) \tag{58}$$

In general, we may write the two equations

$$\begin{aligned} v(n+1) &= Av(n) + bx(n) \\ y(n) &= cv(n) + dx(n) \end{aligned} \tag{59}$$

The state matrix A has N rows and N columns (where N is the number of states), the input vector b has N rows, and the output vector c has N columns. The feed-through d is a scalar. We say the system is $\{A, b, c, d\}$. If we desire the input/output transfer function, we need to take the Z-transform of Eq. (59) and collect like terms to see

$$H(z) = c^t(zI - A)^{-1}b + d \tag{60}$$

I is the $N \times N$ identity matrix (i.e., the $N \times N$ matrix with ones on its diagonal and zeros everywhere else). We state without proof that the poles of the filter are the eigenvalues of A. The zeros of the filter are in general not apparent directly from the state-space equations. Now, the state-space description gives us another implementation of the system because we can compute the output using Eq. (59). An equivalent system to $\{A, b, c, d\}$ with different (new) states $\tilde{v}$ may be obtained using the state transformation matrix $\tilde{v} \equiv T^{-1}v$, where T is any invertible matrix of appropriate dimension. The new system is $\{\tilde{A} = T^{-1}AT, \tilde{b} = T^{-1}b, \tilde{c} = cT, d\}$. The input and output remain unchanged; consequently, the transfer function given in Eq. (60) is valid for both system descriptions.

Quantization Errors

Because digital filters are implemented using either fixed-point or floating-point arithmetic, two basic quantization errors result. Coefficient quantization causes the implemented filter to have poles and zeros slightly different from the designed filter. If the implemented structure is sensitive to the perturbations, the filtering operation performed by the realized filter could be inappropriate, or the filter could become unstable. Also, internal signal quantization can result in limit cycles, which are "stable" oscillations or erroneous outputs. To minimize the distortion caused by internal signal quantization, fixed-point implementations need to be scaled appropriately, while floating-point realizations are auto-scaled.

Coefficient Quantization. Coefficient quantization results in the realized filter having poles and zeros that are different from the (infinite precision) designed filter poles and zeros. For example, consider the second-order all-pole filter

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

Suppose that the filter has a pair of complex-conjugate poles at $z = p, p^*$ such that

$$\begin{aligned} a_1 &= -2\mathrm{Re}\{p\} \\ a_2 &= |p|^2 \end{aligned}$$

Quantization of the parameters of the transfer function results in a quantization of the real part of the poles, as well as the square-root of the pole radii. If we quantize each parameter (coefficient) to b bits, then the realized poles must be located in the z plane at the intersection of b uniformly spaced vertical lines and b nonuniformly spaced circles (due to the squaring). Thus, the density of realizable pole locations is denser near the points $z = \pm j$ than near the points $z = \pm 1$. Oversampling by substantially more than the minimum Nyquist rate increases the coefficient sensitivity because we are "squashing" the low frequencies toward $z = 1$ [see DeBrunner and Beex (14)]. This may seem counterintuitive because as the sampling frequency is increased, we should approach a continuous-time filter. However, even though aliasing is reduced, we now not only require faster hardware, but since the sensitivity is increased we also require more accurate hardware! With higher-order filters, increased sensitivity results. Also, coefficient sensitivity increases as pole magnitudes approach unity—that is, as the poles get close to the unit circle in the z plane. In the example shown in Fig. 8, a tenth-order low-pass elliptic filter with transition region between 0.04 and 0.06, pass-band ripple 0.9 dB, and stop-band attenuation of at least 120 dB is designed. The designed filter has a maximum pole radius of 0.9979. When the coefficients are quantized with as many as 40 bits, the maximum pole radius is 1.0333. With even fewer bits, the maximum pole radius of the implemented filter is even larger.

Internal Signal Quantization. Scaling is required in all fixed-point digital filters. The scaling of all internal signals resulting from a multiplication and add is used to prevent any register or memory location overflow. In the state-space nomenclature, this means scaling to the states. If the internal

Table 2. Limit Cycle Caused by Quantization

Sample	Rounding to Nearest Hundredth	Rounding to Nearest Tenth
0	−0.9	−0.9
1	0.81	0.8
2	−0.72	−0.7
3	0.65	0.6
4	−0.59	−0.5
5	0.53	0.5
6	−0.48	−0.5
7	0.43	0.5

transfer function from the filter input to the ith state is denoted $F_i(z)$ with the corresponding impulse response $f_i(n)$, then a necessary and sufficient condition to ensure that no overflow occurs at the state node is to require for each state that

$$\sum_{n=0}^{\infty} |f_i(n)| \leq 1 \tag{61}$$

This constraint is often overly restrictive, and other scaling bounds provided by Jackson (11) can be used which control the number of instances of overvlow. A practical note when summing more than two numbers in 1's or 2's complement form: If the result is small enough to be represented in the memory, the correct sum will be obtained regardless of the order in which the numbers are added and regardless of whether any intermediate result causes overflow. Consequently, it may be desirable to defeat the normal overflow detection circuitry in the hardware.

Limit-cycle oscillations occur for two reasons: quantization and overflow. Consider the first-order system with difference equation

$$y(n) = x(n) - 0.9y(n-1)$$

This stable system has a pole at $z = -0.9$. Now, suppose the system is run with initial condition $y(0) = 1$ and no input. The output sequence using rounding to the nearest hundredth and rounding to the nearest tenth is given in Table 2. Notice that after the fifth sample, the more coarsely quantized implementation has an output oscillating between -0.5 and $+0.5$. This is the limit cycle. The more finely quantized implementation also has a limit cycle, which occurs when the output reaches ± 0.05. Consequently, we see that the limit cycles caused by quantization can always be reduced in magnitude to acceptable levels by using a sufficient number of bits. This example (with the rounding to the nearest tenth) is shown in Fig. 9. Limit cycles due to overflow (lack of proper scaling) must be stopped in all cases. Again, an example is given to indicate the nonlinear process. Suppose that M is the full-scale value that the memory can hold. Now, consider the total sum $1.1(2M/3) + 0.9(2M/3)$. The overflow result will be $-2M/3$ instead of the correct value $4M/3$ if 2's complement arithmetic is used. Hence, the stable (pole is at radius 0.9847) zero-input AR difference equation $y(n) = 1.1y(n-1) - 0.9y(n-2)$ can support an overflow oscillation of amplitude $2M/3$ and frequency equal to half the sampling frequency. Overflow limit cycles can be eliminated either by proper scaling or by using *saturation arithmetic,* which is achieved by limiting the result of any calculation to be less than M in magnitude.

Structures

The way in which the calculations required to compute the filter output are performed is called the *structure* of the filter. All forms start with the difference equation given in Eq. (1). However, as we have seen, structures based on these direct computations suffer from significant quantization effects. Thus, we introduce several other forms. One definition is required: A *canonical* structure has the minimum number of amplify and delay branches.

Direct Forms. Consider writing the transfer function of Eq. (9) as the cascade of the filter $H_1(z) = \sum_{k=0}^{M} b_k z^{-k}$ followed by the filter $H_2(z) = 1/(1 + \sum_{k=1}^{N} a_k z^{-k})$. The internal signals and the ordering of the computations that result are

$$w(n) = \sum_{k=0}^{M} b_k x(n-k) \tag{62}$$

$$y(n) = w(n) - \sum_{k=1}^{N} a_k y(n-k) \tag{63}$$

The SFG is given in Fig. 17. This direct form I structure requires keeping the most recent M inputs and the most recent N outputs in memory. Also, $N + M + 1$ coefficients are required. Since the filter order is $\max(N, M)$, the direct form I structure is not a canonical structure. However, if we swap the order of the cascaded subfilters so that the AR portion comes first, we can reduce the number of memory locations required to implement the filter to the filter order, resulting in the canonical direct form II structure. The internal signals and the ordering of the computations that result are

$$v(n) = x(n) - \sum_{k=1}^{N} a_k y(n-k) \tag{64}$$

$$y(n) = \sum_{k=0}^{M} b_k v(n-k) \tag{65}$$

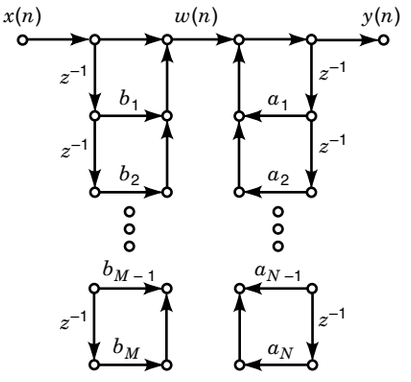


Figure 17. Direct form I structure SFG showing "extra" delay branches.

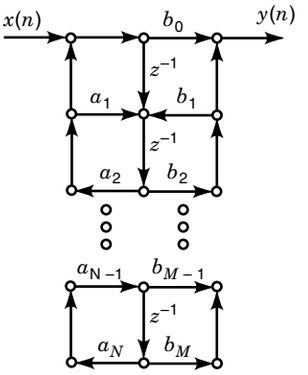


Figure 18. Direct form II canonical structure showing the minimum number of delay branches.

The SFG is shown in Fig. 18. Note that this structure does not maintain delayed versions of either the input or the output signals, but rather maintains delayed versions of the internal signal $v(n)$.

Parallel and Cascade. The *parallel* structure is based on the partial fraction expansion of the transfer function, $H(z) = \sum_{k=1}^{(N+1)/2} H_k(z)$, where

$$H_k(z) = \frac{\gamma_{0k} + \gamma_{1k} z^{-1}}{1 + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}} \tag{66}$$

Assuming that the order of the MA part ($H_1(z)$ from the previous section) is less than the order of the AR part [$H_2(z)$ from the previous section] and that the transfer function $H(z)$ has only single poles, the transfer function may be expanded in the partial-fraction expansion

$$H(z) = \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}} \tag{67}$$

The p_k are the N distinct poles of $H(z)$. The A_k are determined using

$$A_k = [H(z)(1 - p_k z^{-1})]_{z=p_k} \tag{68}$$

Further details on the partial-fraction expansion may be found in Oppenheim et al. (15) or in Oppenheim and Schafer (16). Each second-order section is implemented as a direct form II filter. If the original filter order is odd, then one of the subfilters is a first-order filter. The second-order subfilters are used to allow the realization of complex-conjugate pole pairs with real-valued coefficients. No design method for collecting real poles in the second-order sections exist; however, it is common practice to collect the real poles to provide maximal distance between the poles of each subfilter.

The *cascade* structure uses the factorization of the transfer function, $H(z) = \prod_{k=1}^{(N+1)/2} H_k(z)$, where

$$H_k(z) = \frac{1 + \beta_{1k} z^{-1} + \beta_{2k} z^{-2}}{1 + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}} \tag{69}$$

As in the parallel form, the second-order subfilters are implemented in the direct form II structure. It is common practice to place the subfilters with lowest magnitude later in the cascade.

Lattice. The general ARMA lattice structure SFG is shown in Fig. 19. The parameters are derived using the backward Levinson recursion

$$a_{k-1}(i) = \frac{a_k(i) - a_k(k) a_k^*(k-i)}{1 - |a_k(k)|^2}, \qquad \begin{matrix} i = 1, 2, \ldots, k-1; \\ k = N, N-1, \ldots, 2 \end{matrix} \tag{70}$$

The subscripts denote order; thus the recursion starts with the transfer function denominator coefficients subscripted with N. Then, the reflection coefficients are obtained:

$$\Gamma_i = a_i(i), \qquad i = 1, 2, \ldots, N \tag{71}$$

The zeros are implemented with the tap coefficients [the $c_m(j)$] that are obtained by solving the set of equations

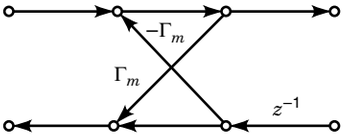
$$b_M(k) = \sum_{j=k}^{M} c_M(j) a_j^*(j-k) \tag{72}$$

This form is often used when a low-sensitivity implementation is desired.

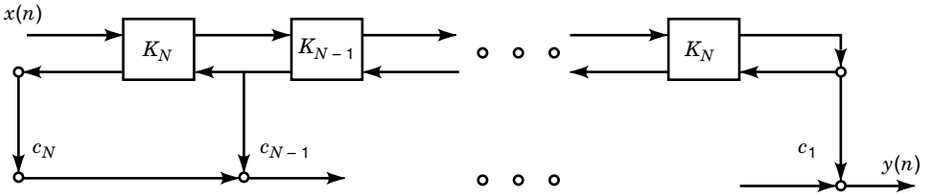
Wave Filters. Wave digital filters (WDF) were introduced by Fettweiss in the early 1970s and described fully in Ref. 17. In their most basic form, the WDF are passive analog ladder and lattice filter designs that have been bilinear-transformed. These forms allow low-delay realizations and filters with coefficients implemented with only a few bits. Thus, this structure has become very important in systems demanding high data rates, such as sonar, video, and microwave frequency communications. The primary benefit of reducing the number of bits in the coefficients is that the multiplication is replaced by a few shifts and one less add. The lattice WDF consists of two parallel signal paths containing cascades of second-order all-pass filters, with a first-order all-pass filter section required if the overall filter order is odd. The two outputs of the last section are added to form the filter output. It is possible to construct a WDF so that two complementary outputs are available from one filter (In fact, the most common construction provides this capability). Thus, for instance, we can construct a WDF that provides one low-pass filtered output and one high-pass filtered output.

Methods to design a WDF are to straightforwardly transform a passive analog filter design with the bilinear transformation. However, Gaszi (18) has provided methods for implementing the Butterworth, Chebyshev, and elliptic filters described in the previous section. WDF design for nontraditional frequency shapes is a current research task.

Optimal Forms. Mullis and Roberts (19,20) first developed the optimal form. It is based on two matrices from system theory: the controllability Grammian $K = AK^tA + bb^t$ and the observability Grammian $W = A^tWA + c^tc$. The names of these two matrices do not have any relative meaning to us. How-



(**a**) m^{th}-order lattice section, K_m



(**b**) ARMA lattice

Figure 19. Lattice structure SFG illustrating its order modularity.

ever, the solutions to these two Nth-order algebraic Lyapunov equations are

$$k_{ij} = \frac{1}{2\pi j}\oint_u \frac{\partial H(z)}{\partial c_i}\frac{\partial H(z^{-1})}{\partial c_j}\frac{dz}{z} \quad \left(\text{Note that } k_{ii} = \sum_{n=0}^{\infty} |f_i(n)|^2\right) \tag{73}$$

$$w_{ij} = \frac{1}{2\pi j}\oint_u \frac{\partial H(z)}{\partial b_i}\frac{\partial H(z^{-1})}{\partial b_j}\frac{dz}{z} \tag{74}$$

One method for constructing the transformation matrix T required to convert the direct form II structure (described in state-space) to the optimal form is given by Hwang (21). In this transformation, the controllability grammian diagonals (the k_{ii}, $i = 1, 2, \ldots, N$) are forced to unity, and so the determination of the optimal form is reduced to the problem of minimizing the trace of the observability grammian tr(W)—that is, minimizing the sum of the diagonal elements of the grammian. While simplifying the problem, the important reason for constraining the controllability grammians is that the resulting filter will be scaled, as can be seen by comparing the defining relation given in Eq. (73) with the scaling condition given in Eq. (61). Consequently, an important property of the optimal forms is that overflow oscillations decay to zero. The construction algorithm is based on the solution of three basic matrix equations (21).

1. Determine the orthogonal matrix R_0 from

$$R_0(\Lambda^*)^{-2}R_0^t = \begin{bmatrix} 1 & \% & \cdots & \% \\ \% & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \% \\ \% & \cdots & \% & 1 \end{bmatrix} \equiv K$$

The % means "don't care." The diagonal matrix

$$\Lambda^* = \begin{bmatrix} \lambda_1^* & 0 & \cdots & 0 \\ 0 & \lambda_2^* & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_N^* \end{bmatrix}$$

is computed from

$$\lambda_i^* = \left[\frac{\sum_{m=i}^{N}\theta_m}{N\theta_i}\right]^{1/2}$$

The θ_i^2 are the eigenvalues of K_0W_0 (the product of the grammians of the state-space structure).

2. Solve for the Cholesky factor T_0 where

$$T_0T_0^t = K_0$$

3. Finally, solve for the orthogonal matrix R_1 where

$$R_1^tT_0^tW_0T_0R_1 = \begin{bmatrix} \theta_1^2 & 0 & \cdots & 0 \\ 0 & \theta_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \theta_N^2 \end{bmatrix}$$

The transformation matrix that converts the present state-space form to the optimal form is then given by

$$T = T_0R_1\Lambda^*R_0^t \tag{75}$$

and the optimal state space structure obtained is $\{T^{-1}AT, T^{-1}b, c^tT\}$. The new grammians are

$$\{T^{-1}K_0T^{-t}, T^tW_0T\} \tag{76}$$

Because the state-space form is full of non-structural parameters, the form is not canonical. Consider that an Nth-order canonical filter requires $2N + 1$ coefficients while the optimal form requires $N^2 + 2N + 1$ coefficients. Recognizing this problem, two attractive alternatives are the block- and section-optimal structures developed by Jackson, et al. (22). These forms are based on the parallel and cascade forms, respectively. Each of the second-order subfilters in the parallel or cascade structures is implemented using a second-order optimal structure. These designs, while suboptimal, often perform very well, with quantization noise power being very close to

optimal. The number of filter parameters has been reduced to $4N$. The parameters of these forms are obtained from the second-order transfer function

$$H(z) = \frac{\gamma_1 z^{-1} + \gamma_2 z^{-2}}{1 + \beta_1 z^{-1} + \beta_2 z^{-2}} + d$$

The second-order optimal state-space form is then constructed using the following:

$$\begin{aligned}
\hat{a}_{11} &= \hat{a}_{22} = -\frac{\beta_1}{2} \\
\hat{a}_{12} &= \frac{1+\gamma_2}{\gamma_1^2}\left[\left(\gamma_2 - \frac{\beta_1\gamma_1}{2}\right) + (\gamma_2^2 - \gamma_1\gamma_2\beta_1 + \gamma_1^2\beta_2)^{1/2}\right] \\
\hat{a}_{21} &= \frac{1}{1+\gamma_2}\left[\left(\gamma_2 - \frac{\beta_1\gamma_1}{2}\right) - (\gamma_2^2 - \gamma_1\gamma_2\beta_1 + \gamma_1^2\beta_2)^{1/2}\right] \\
\hat{b}_1 &= \frac{1+\gamma_2}{2} \\
\hat{b}_2 &= \frac{\gamma_1}{2} \\
\hat{c}_1 &= \frac{\gamma_1}{1+\gamma_2} \\
\hat{c}_2 &= 1
\end{aligned}$$

The optimal form is then the scaled network $\{T^{-1}\hat{A}T,\ T^{-1}\hat{b},\ \hat{c}T,\ d\}$, where

$$T = \begin{bmatrix} \dfrac{1}{2\pi j}\oint_u \dfrac{\partial H(z)}{\partial \gamma_2}\dfrac{\partial H(z^{-1})}{\partial \gamma_2} & 0 \\ 0 & \dfrac{1}{2\pi j}\oint_u \dfrac{\partial H(z)}{\partial \gamma_1}\dfrac{\partial H(z^{-1})}{\partial \gamma_1} \end{bmatrix} \quad (77)$$

Example

We include a design example that shows the effects of finite precision on recursive filter implementation. The example uses the third-order low-pass digital filter with transfer function

$$H(z) = \frac{0.079306721z^{-1} + 0.023016947z^{-2} + 0.0231752363z^{-3}}{1 - 1.974861148z^{-1} + 1.556161235z^{-2} - 0.4537681314z^{-3}}$$

The forms described previously and the corresponding sensitivities as measured using the technique given by DeBrunner and Beex in (14) are as follows:

1. The direct form II is

$$v(n+1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.4537681314 & -1.556161235 & 1.974861148 \end{bmatrix} v(n) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} x(n)$$

$$y(n) = [0.0231752363 \quad 0.023016947 \quad 0.079306721]x(n)$$

The sensitivity is 93.71.

2. The parallel form is

$$v(n+1) = \begin{bmatrix} 0 & 1 & 0 \\ -0.689750194 & 1.316988002 & 0 \\ 0 & 0 & 0.657873146 \end{bmatrix} v(n) + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} x(n)$$

$$y(n) = [0.262118112 \quad -0.20426974 \quad 0.283603691]x(n)$$

The sensitivity is 15.70.

3. The cascade form is

$$v(n+1) = \begin{bmatrix} 0 & 1 & 0 \\ -0.689750194 & 1.316988002 & 0 \\ -0.397527345 & 1.607214942 & 0.657873146 \end{bmatrix} v(n) + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} x(n)$$

$$y(n) = [0 \quad 0 \quad 0.079306721]x(n)$$

The sensitivity is 43.51.

4. The optimal form is

$$v(n+1) = \begin{bmatrix} 0.6672421816 & 0.0588820057 & 0.1297010701 \\ 0.0951152564 & 0.6488117266 & 0.5866572779 \\ 0.089399279 & -0.4660588199 & 0.6588073673 \end{bmatrix} v(n) + \begin{bmatrix} 0.6221731984 \\ -0.1549534962 \\ 0.6111579978 \end{bmatrix} x(n)$$

$$y(n) = [0.2917887397 \quad 0.2806760077 \quad -0.09612048753]x(n)$$

The sensitivity is 8.81.

5. The block-optimal form is

$$v(n+1) = \begin{bmatrix} 0.658494001 & 0.684463705 & 0 \\ -0.3742139062 & 0.658494001 & 0 \\ 0 & 0 & 0.657873146 \end{bmatrix} v(n) + \begin{bmatrix} 0.312887592 \\ -0.652953035 \\ 0.753128756 \end{bmatrix} x(n)$$

$$y(n) = [-0.326470236 \quad 0.156440787 \quad 0.376567338]x(n)$$

The sensitivity is 7.34. This sensitivity is lower than the optimal form sensitivity, probably because of numerical inaccuracies incurred in the computation of the optimal form and because the sensitivity measure provides only a linear upper bound to the output quantization noise power.

6. The section-optimal form is

$$v(n+1) = \begin{bmatrix} 0.658494001 & 0.50628116 & 0 \\ -0.5059162 & 0.658494001 & 0 \\ 1.787303811 & 0.851076483 & 0.657873146 \end{bmatrix} v(n) + \begin{bmatrix} 0.338621722 \\ 0.711122804 \\ 0.753128756 \end{bmatrix} x(n)$$

$$y(n) = [0 \quad 0 \quad 0.105303004]x(n)$$

The sensitivity is 24.79.

We can see from the above that the block-optimal and optimal forms have the lowest coefficient sensitivity by a factor of two better than the next lowest form (the parallel form). Analysis of the output quantization noise power as a function of fixed-point word-length shows that:

1. The section-optimal form possesses the lowest error at 10 bits.
2. The optimal form possesses the lowest error at 8, 16 and 20 bits.
3. The block-optimal form possesses the lowest error at 12, 14, and 18 bits.
4. The parallel form performs well at all word lengths, equaling the performance of the block-optimal form at 14 bits.

From a realization standpoint, it is important for the designer to understand that place-holders in the state-space description are indicated by the 0 and ± 1 coefficients. Consequently, the direct form II, parallel and cascade structures require six multiplications (the minimum number). The block- and section-optimal forms each require 11 multiplications, while the optimal form requires the maximum number of 15 multiplications.

CURRENT RESEARCH AND PRACTICE

Design Methods

General Designs. Three approaches to the general design problem are considered: least-squares optimizations, polynomial methods, and alternative design procedures. We consider these in order.

Least-Squares Methods. In this case, we consider two different approaches: iterative and noniterative. An L_p minimization technique, in particular an L_∞ minimax procedure, is developed by Antoniou (23). This article describes a new class of algorithms developed to improve the performance of two existing optimization algorithms in the filter design case where the frequency response $H(e^{j\omega})$ is matched to a desired frequency response. In particular, the methods use standard optimization techniques from Fletcher (24) and Luenberger (25) with the objective function

$$\min_x \Psi(x) = L_p = \hat{E}(x) \left\{ \sum_{i=1}^{K} \left[\frac{|e_i(x)|}{\hat{E}(x)} \right]^p \right\}^{1/p} \tag{78}$$

The values $e_i(x) = |H(x, e^{j\omega_i})| - |H_d(x, e^{j\omega_i})|$, $i = 1, 2, \ldots, K$ are the difference in the magnitudes of the designed and desired filters at selected frequencies, the x are the recursive filter coefficients, and $\hat{E}(x) = \max_{1 \le i \le K} |e_i(x)|$. The value p is any integer. A variation of the technique uses methods developed by Charalambous (26,27) uses the objective function

$$\Psi(x, \lambda, \xi) = \sum_{i \in I_1} \frac{1}{2} \lambda_i [\phi_i(x, xi)]^2 + \sum_{i \in I_2} \frac{1}{2} [\phi_i(x, \xi)]^2 \tag{79}$$

ξ and the λ_i are constants and $\phi_i(x, \xi) = |e_i(x)| - \xi$. The indices are defined $I_1 = \{i\colon \phi_i(x, \xi) > 0, \lambda_i > 0\}$ and $I_2 = \{i\colon \phi_i(x, \xi) > 0, \lambda_i = 0\}$. The algorithm using the objective function in Eq. (79) is apparently considerably more computationally efficient. Extensive application of the methods indicated that the proposed techniques compare favorably in performance to existing techniques, and that the computational complexity is reduced. Occasional stability problems were noted in the article.

The noniterative approach by Pei and Shyu (28) uses a technique based on the computation of an eigenvalue and its corresponding eigenvector of a real symmetric and positive-definite matrix derived from the appropriate objective function. The particular objective function used in the article is based on a squared impulse response error

$$e = A^t D^t W D A \equiv A^t Q A, Q \equiv D^t W D \tag{80}$$

The recursive filter parameters are incorporated in the vector $A = [a_0, a_1, \ldots, a_N, b_0, b_1, \ldots, b_M]$. Note that in Eq. (1), $a_0 \equiv 1$, a condition that can be enforced by scaling the coefficients appropriately. The matrix D is obtained from the length L desired impulse response in the following manner. Define the following:

1. $\tilde{h}(n) = \begin{cases} \frac{1}{2} h_d(n), & n = 0 \\ h_d(n), & 1 \le n \le L \end{cases}$

2. $D_1 = \begin{bmatrix} \tilde{h}(0) & 0 & 0 & \cdots & 0 \\ \tilde{h}(1) & \tilde{h}(0) & 0 & \cdots & 0 \\ \tilde{h}(2) & \tilde{h}(1) & \tilde{h}(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{h}(N) & \tilde{h}(N-1) & \tilde{h}(N-2) & \cdots & \tilde{h}(0) \end{bmatrix}$

3. $D_2 = \begin{bmatrix} \tilde{h}(N+1) & \tilde{h}(N) & \tilde{h}(N-1) & \cdots & \tilde{h}(1) \\ \tilde{h}(N+2) & \tilde{h}(N+1) & \tilde{h}(N) & \cdots & \tilde{h}(2) \\ \tilde{h}(N+3) & \tilde{h}(N+2) & \tilde{h}(N+1) & \cdots & \tilde{h}(3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{h}(L) & \tilde{h}(L-1) & \tilde{h}(L-2) & \cdots & \tilde{h}(L-N) \end{bmatrix}$

4. $D_3 = -I$

Then,

$$D = \begin{bmatrix} D_1 & D_3 \\ D_2 & 0 \end{bmatrix} \tag{81}$$

The arbitrary diagonal weighting matrix $W \equiv diag(w(0), w(1), \ldots, w(L))$ is provided to allow unequal importance to the impulse response errors. The minimizing solution to Eq. (80) is the eigenvector associated with the smallest eigenvalue of the real, positive definite matrix Q. The method compares favorably to iterative method designs using a desired impulse response.

Polynomial Methods. The polynomial methods use the solution to the Chebyshev approximation problem. In Lee and Chen (29) two algorithms using the generalized Ellacott–Williams algorithm (30) for the design of complex coefficient recursive digital filters. In the algorithms, a linear complex Chebyshev approximation problem is solved at each iteration. These are solved numerically using the procedures developed

by Lim et al. (31). The algorithm may be computed with a lower computational burden than other methods. A minimax filter design method for the direct computation of the lattice form parameters is provided by Lee and Ku (32). This method always provides a stable design, because the lattice reflection coefficients are constrained to fall in the range $(-1, 1)$.

Alternative Design Methods. In an interesting new direction, Hui and Want (33) develop an optimal design technique for recursive digital filters using a highly interconnected Hopfield neural network. These networks are used because they guarantee global solutions for least-square and linear programming problems. The optimization criterion is the impulse response error used in the eigenvector design method discussed previously. The authors conclude that the approach works suitably well, although the benefits and costs are not well-defined at present.

Design Methods for Near Linear Phase. We next consider research in methods to design recursive digital filters that have nearly linear phase responses (or equivalently nearly constant group delay) in the filter pass-band. As we have seen, this feature is important in many high-fidelity filtering applications (recall the example whose results are shown in Figs. 5 and 6). We see several different approaches to the designs. The basic approaches to these designs are:

1. Polynomial methods (using the Chebyshev theory)
2. Methods using direct optimization
3. Methods based on structures using all-pass filters
4. Methods that approximate the good phase response of the analog Butterworth filter using a design technique that is completely in the digital domain

We consider each of these approaches in turn.

Polynomial Methods. Thajchayapong et al. (34) designed a recursive digital filter with prescribed group delay and Chebyshev stop-band using transfer functions with a numerator order higher than the denominator order. The article shows that the transfer function can be determined using the bilinear transformation on analog designs. The method uses analog designs with mirror-image polynomials in the numerator to obtain Chebyshev attenuation (35–38). Direct transformation techniques (36,38) appear to have an advantage over the indirect transformation method described by Thajchayapong and Lomtong (35). However, this article shows that by modifying the first three steps of the indirect method in Ref. 35 and using the bilinear transformation, the results of the direct and indirect methods are identical. The changed steps are as follows:

1. Define the transfer function

$$H(z) = \begin{cases} \dfrac{\prod_{i=1}^{M}(1 - 2\cos\theta_i z^{-1} + z^{-2})}{\sum_{i=0}^{N} a_i z^{-i}}, \\ \qquad M \text{ even and } 2M \geq N \\ \dfrac{(1 + z^{-1})\prod_{i=1}^{M}(1 - 2\cos\theta_i z^{-1} + z^{-2})}{\sum_{i=0}^{N} a_i z^{-i}}, \\ \qquad M \text{ odd and } 2M + 1 \geq N \end{cases} \tag{82}$$

   Multiplying both numerator and denominator in Eq. (82) by z^{2M} and z^{2M+1}, respectively, to eliminate the negative powers of z and using the bilinear transformation of Eq. (53), we obtain the analog transfer function

$$H(s) = \frac{\prod_{i=1}^{M}(s^2 + \omega_i^2)}{(1+s)^i \sum_{i=0}^{N} b_i s^i} \tag{83}$$

2. The coefficients in Eq. (83) can be determined using classical techniques to either produce a maximally flat (39) or an equal-ripple (40) group delay.
3. Prewarp the stop-band frequencies and use the bilinear transformation to get the desired digital filter.

Least-Squares Methods. We present the work from two articles in this section. The first presents a design technique suitable to the design of nearly linear phase filters with adjustable magnitude response at arbitrarily specified frequencies (41). In this article, Trisuwannawat et al. use a power series expansion of the numerator of the frequency response of a general recursive filter to develop a set of linear equations that can be solved by a variety of numerical procedures to yield the optimal filter coefficients. The derivation is straightforward, and the results indicate that the method should be attractive in certain cases.

In the second article, Gu and Shenoi (42) present a sample domain technique that uses an FIR design to yield the desired impulse response. This response is matched optimally using least-squares techniques to the impulse response of a designed recursive digital filter. The authors employ optimal Hankel approximation techniques, although any of the designs by modeling could be used. The method produces causal and stable designs. The design method is intended to reduce the computational complexity of the design, particularly for 2-D linear phase filters for use in image processing.

Design Methods Using All-Pass Filters. The other major techniques incorporate the use of all-pass filters. The classical approach to nearly linear phase designs has been the phase compensation technique previously discussed. However, recent research has centered on the use of parallel combinations of two all-pass filters (43,44) and the recasting of a general classical filtering function design into a special all-pass filter design problem (45). We consider these in turn.

In the first article using the parallel combination that we consider, Jaworski and Saramaki (43) use the structure

$$\begin{aligned} H(z) &= \frac{E(z)}{D(z)} = \frac{1}{2}[A(z) + B(z)] \\ G(z) &= \frac{F(z)}{D(z)} = \frac{1}{2}[A(z) - B(z)] \end{aligned} \tag{84}$$

The transfer functions $A(z)$ and $B(z)$ are stable all-pass filters of orders K and L, respectively. The order of the parallel combination transfer functions is thus $N = K + L$. Causality conditions on the filters require that $H(z)H(z^{-1}) + G(z)G(z^{-1}) = 1$ and that $E(z)$ is a linear phase FIR filter with symmetric impulse response of order N and that $F(z)$ is the same with the exception that its impulse response is antisymmetric. The designs described in this article are limited to low-pass, high-pass pairs, where $E(-1) = 0$ and $F(1) = 0$. Furthermore, $(N - 1)/2$ zeros of $F(z)$ are located on the unit circle of the

complex plane at frequencies ω_k, $0 \le k \le (N-1)/2 - 1$. Selecting the zeros of $E(z)$ appropriately guarantees the second causality condition. The first condition is met by constraining the magnitude response of the low-pass filter appropriately. The approximation is solved using the alternation theory of Chebyshev.

In the second method, Lawson (44) uses the same structure as the method given in (43), but the approach is quite different. In this method, the magnitude and phase responses of the parallel combination are written as functions of the individual all-pass filter phases:

$$\begin{aligned} |H(e^{j\omega})| &= \left|\cos\frac{1}{2}(\phi_1 - \phi_2)\right| \\ \angle H(e^{j\omega}) &= \frac{1}{2}(\phi_1 + \phi_2) \\ \tau(e^{j\omega}) &= \frac{1}{2}(\tau_1 + \tau_2) \end{aligned} \tag{85}$$

The subscripts denote the phase response and group delay of the two all-pass filters. The overall filter is designed by writing the individual phase responses and group delays in terms of the all-pass filter coefficients, where the structure of each all-pass filter is a cascade of the first-order filter with transfer function in Eq. (11). A linear programming algorithm for searching the solution space for the optimal answer is given in the article. The method can be extended to the design of phase compensators using only one all-pass branch.

An interesting approach to the near linear phase IIR filter design problem is given by Song and Gu (45). In this approach, a design method is given to determine the all-pass coefficients of Eq. (12) from a set of phases and frequencies. The procedure relies on the Remez exchange algorithm [for instance, see Braess (46)]. Then, the synthesis of near linear phase IIR filters is recast as an all-pass design method, so that the algorithm developed for all-pass designs is modified slightly to perform the task at hand. This is accomplished by considering designs where

$$H(e^{j\omega}) = \begin{cases} e^{j\omega}, & \text{passband} \\ 0, & \text{stopband} \\ \text{don't care}, & \text{elsewhere} \end{cases} \tag{86}$$

The recasting of the problem is accomplished using the new transfer function

$$G(z) = 2z^D H(z) - 1$$

to design the all-pass filter

$$G(e^{j\omega}) = \begin{cases} 1, & \text{passband} \\ -1, & \text{stopband} \\ \text{don't care} & \text{elsewhere} \end{cases} \tag{87}$$

The filter that we are interested in is then obtained:

$$H(z) = \frac{z^{-D}}{2}\big(1 + G(z)\big) \tag{88}$$

Digital Butterworth–Chebyshev Designs. A design method that approximates an AR filter function directly in the digital domain is given by Stojanovic and Nikolic (47). The resulting designs are called "transitional Butterworth–Chebyshev filters" and share properties with both of the named filters. The motivation of this work is that when analog filters are transformed to digital filters (say, using the bilinear transform) an all-pole analog design becomes a digital function with both poles and zeros. The direct design of digital Butterworth filters [proposed by Rader and Gold (48)] and of Chebyshev filters [proposed by Soltis and Sid-Ahmed (49)] is improved upon in this article. The designs have better phase response than the Chebyshev designs, with better transition band roll-off than the Butterworth designs.

Design Methods Incorporating Both Magnitude and Phase. Two methods using the parallel combination of all-pass filters have been proposed. As such, they are generalizations of the methods described in Refs. 43 and 44. In the first article, Lawson (50) gives methods to design recursive filters with prescribed magnitude and phase characteristics. These methods use the structure from Ref. 44 with slight modifications for the magnitude specification and the arbitrary (not necessarily linear) phase specification. Again, aspects of classical optimization theory (the linear programming from before) are used to determine the optimal filter parameters. Additionally, simulated annealing and genetic algorithms are developed to reduce the computational burden and improve optimization performance. The structure is deemed suitable in many applications because the resulting structure is not too sensitive. This approach is considered further by Lawson and Wicks (51). Further computational complexity reduction is possible through the use of results given by Gregorian and Temes (52), which yield a set of linear equations for each all-pass subfilter.

Conversion and Transient Designs. Erfani, et al. (53) discuss a new technique for converting an analog filter to a digital format. In that article, a generalized bilinear transformation technique is developed that uses the biquadratic approximation

$$\frac{1}{sT} = \frac{1}{3}\left(\frac{z+1}{z-1} + \frac{1}{z-1} + \frac{1}{z+1}\right) \tag{89}$$

The algebraic substitution averages the bilinear transformation, a forward difference, and a corrective term. As in the bilinear substitution, the relationship in Eq. (89) warps the frequencies

$$\Omega = \frac{3}{T}\frac{\sin\omega T}{2 + \cos\omega T} \tag{90}$$

The warping of Eq. (90) is straightforward for low-pass filters, integrators, and differentiators. The mapping may not preserve stability as the bilinear transformation does; consequently, the digital filter should be tested for stability. One other difference when compared to the bilinear transformation is more obvious: The digital filter order is doubled in the application of Eq. (89). However, the multiplication by 4 in the (combined) numerator of Eq. (89) can be accomplished by a two-bit shift to the left.

In filtering cases where the length of the available input is short, the transient response (i.e., the effective duration of the

impulse response) creates a problem. This case often arises in radar applications (54) because extremely narrow-band filters are required. In these cases, we require the filter's steady-state output (recall Fig. 5), but the length of the available input is so short that only the transient response is ever computed. Classically, the approach to solving this problem has been to optimize the initial conditions of the digital filter. In cases where the input duration is not a problem, the initial conditions of the filter are almost always ignored. Sometimes the initial conditions are zeroed, and typically the initial conditions are left to the arbitrary nature of the implemented digital circuitry. Optimizing the initial conditions results in a nonlinear problem. Recently, Musa and Al-Ahmad (55) have developed techniques that optimize the transient response of the filter in combination with a desired frequency response.

Sensitivity and Round-off Noise

Considerable work on sensitivity and designing implementations with low-output quantization round-off noise has taken place over the years. Several areas of active research are methods for analyzing the effects of the nonlinear quantization operations, designs of novel structures, and methods that employ special digital circuitry. We examine each of these areas in turn.

Analysis of Quantization Effects. DeBrunner and Beex (14) provide a thorough review of the background into the analysis of quantization effects. A classic article relating coefficient sensitivity designs and minimal output quantization round-off noise power is given by Tavsonoglu and Thiele (56). Design techniques for producing implementations possessing both properties are given as well. Rao (57) presents further results regarding coefficient sensitivity. All of these articles use L_p norms on some of the internal transfer functions found in Eqs. (73) and (74) and combinations of these equations, although the combinations are slightly different in each. The combination used in Ref. 14 is an exact computation of the upper bound estimate used in Refs. 56 and 57. Some more recent results may be found in Refs. 58–63. In the first of these, Goodall (58) describes methods for designing analog filters (primarily for real-time control applications) that exhibit low sensitivity when bilinear transformed. The main idea is that analog designs should incorporate (through design specifications) the knowledge that the filter implementation is digital. Farison and Kolla (59) extend generalizations to the coefficient sensitivity relations above to time-varying recursive digital filters, such as those used in communication (modulation) systems. Some analysis of existing, thought-to-be low sensitivity filter structures that use error feedback (described below in the section entitled "Error Feedback Systems") is provided by Baez-Lopes et al. (60). Furthermore, Macedo et al. (63) provide methods for examining limit cycles in error feedback systems. Thus, we see the cross-fertilization between these two areas beginning to aid in the development of truly low sensitivity filters. Worthington and Turner (61) provide L_1 bounds (and methods for their computation) on errors resulting from signal quantization at selected frequencies of interest. The output quantization distortion for cascade implementations is provided by Mollova (62).

Structures that Minimize Quantization Effects. Structures that minimize quantization effects are a very active area of current research. Methods of approach include (and we consider them each in turn):

1. Decomposition of high-order filters into subfilter designs (64,65)
2. Decomposing the filter into parallel combinations of all-pass filters (66–68) or, more generally, into orthogonal filters (69)
3. Designs that either minimize coefficient count or maximize the number of coefficients with coefficients that are powers-of-two (70–73)
4. Designs that increase the filter order (75,76)

Incomplete Partial Fraction Expansion. The incomplete partial fraction expansion allows the rational polynomial

$$T(z) = \frac{Q(z)}{F(z)G(z)}$$

to be written as

$$T(z) = \frac{H(z)}{F(z)} + \frac{K(z)}{G(z)} \tag{91}$$

instead of the "complete" PFE given in Eq. (67) by Price and Sandler (64). Two structures based on the partial transfer functions given in Eq. (91) arise because they can be implemented in either cascade or parallel forms (other forms are possible, but are not considered in this article). The methods of the article are not proven, but the performance appears to be satisfactory. The method using the cascade forms, called the parallel interconnection of cascade subfilters (PICS) format, is studied by Sandler (65). The authors consider the cascade structure to be superior because the individual scaling required for each subfilter can be straightforwardly accomplished.

All-Pass Filters. All-pass filters used for low-sensitivity designs were developed at least as early as 1986 by Vaidyanathan et al. (66). Digital all-pass filters can be efficiently realized using lossless structures that possess low-sensitivity, low-output quantization noise power, and are free of overflow limit cycles. Nie et al. (67) show that every rational transfer function, as in Eq. (9), that is stable can be decomposed into a linear combination of stable first-order all-pass filters. Consequently, any stable recursive filter can be implemented as either a parallel or cascade connection of all-pass filters. A technique that allows filters with several pass- and stopband regions (i.e., a restricted class of nonclassical filters) to be implemented with the all-pass structures is developed by Saramaki (68). The developed algorithm uses the Remez algorithm to optimally determine the subfilters.

Nie et al. (69) give an interesting addition to the research of the all-pass filters. Here, the authors show that stable recursive filters can be implemented using an orthogonal expansion of N functions (N being the filter order). The expansion is as follows. Write the filter transfer function as

$$H(z) = \frac{b_N z^N + b_{N-1} z^{N-1} + \cdots + b_1 z + b_0}{(z - p_1)(z - p_2) \cdots (z - p_N)} \tag{92}$$

The p_i, $1 \le i \le N$, are the poles of the filter, and we have assumed that the MA order is $M \le N$. The stable transfer

function $H(z)$ given in Eq. (92) can be written in the expanded form using the orthogonal functions

$$G_k(z) = \frac{\sqrt{1-p_k^2}}{z-p_k}\prod_{i=0}^{k-1} A_i(z), \qquad k = 1, 2, \ldots, N$$
$$G_0(z) = 1 \tag{93}$$

The all-pass transfer functions are defined as follows:

$$A_i(z) = \frac{1-p_i^* z}{z-p_i}$$
$$A_0(z) = 1 \tag{94}$$

Then, we have

$$H(z) = \lambda_0 + \sum_{k=1}^{N} \lambda_k G_k(z) \tag{95}$$

The λ are the projections of $H(z)$ on the orthogonal basis functions

$$\lambda_0 = \lim_{z\to\infty} H(z)$$
$$\lambda_k = \frac{1}{2\pi j}\oint_{|z|=1} G_k(z) H^*(z^{-1})\frac{dz}{z}, \qquad k = 1, 2, \ldots, N \tag{96}$$

Application of the Parseval theorem yields

$$\|H(z)\| = \frac{1}{2\pi j}\oint_{|z|=1} H(z)H^*(z^{-1})\frac{dz}{z} = \sum_{k=0}^{N} |\lambda_k|^2 \tag{97}$$

If $\|H(z)\|_2 \le 1$ (i.e., if the digital filter is properly scaled), then Eq. (97) implies that the filter can be realized using structurally passive component subfilters $G_k(z)$. Thus, the resulting implemented filters can possess very low output quantization noise power and have low sensitivities.

Coefficient Manipulation. The articles by Bomar and Hung (70) and Bomar (71,72) describe several methods for manipulating the filter designs to increase the number of "trivial" coefficients. In these articles, methods are developed that produce near optimal connections of second-order filter structures. The constraints placed on the coefficients attempt to reduce the total number of coefficients in the filter description. These constraints force some of the coefficients to be structural ones and zeros, while others are forced to be exact powers of two, thus making multiplication equivalent to shifting the binary point. Fahmy and Raheem (73) develop a method to design scaled, fixed-point digital filters that are free of limit cycles. The method reduces as many of the filter coefficients to zero as possible using an optimization method based on minimizing the total output quantization noise power.

Filter Order. Of all the techniques described above to reduce digital filter sensitivity, one of the most unusual is based on the use of the filter order. Jackson (74) describes a Chebyshev optimization procedure that designs filters with more zeros than poles. If the MA and AR orders are chosen appropriately, the resulting filter can have low-output quantization noise power. Beex and DeBrunner (75) use direct form II filters as the basis for low-sensitivity designs. In their article, pole/zero cancellation pairs are added to second-order direct form II subfilters to reduce the filter sensitivity and to lower the output quantization noise power. Optimal search algorithms for determining both real and complex-conjugate cancellation pairs are given. Designs are given in the example section that show sensitivities near the optimal form sensitivity with significantly lower coefficient counts. DeBrunner et al. (76) present an optimization method that matches the impulse response and lowers the sensitivity simultaneously in a multicriterion approach. One interesting result from this method is that the designs in Ref 75 are shown to be optimal.

Error Feedback Systems. Error feedback systems give a completely alternative approach to reducing output quantization noise power. Error feedback works by using extra digital circuitry to feed the rounding error back to the next computation. Several possibilities exist. In one recent article, Leung (77) describes a method for designing an accumulator that not only produces low-output quantization noise power, but also increases the density of realizable pole locations near the $z = 1$ location of the complex plane. The method requires only two effective multiplications per output sample for a second-order section. Laakso and Hartimo (78) research the problem of determining the optimal error feedback coefficients. The authors conclude that for high-order filters, the optimization results in solving the classical Wiener–Hopf equations (for instance, see Ref. 4), thus showing that the problem is a special case of Wiener filtering. The article discusses special methods for designing suboptimal error feedback with symmetric or antisymmetric coefficients, as well as methods that incorporate filters with powers-of-two coefficients.

VLSI

Because a separate article will discuss methods of implementing DSP algorithms (including recursive filters) in VLSI, we present only a brief overview of some current research trends of VLSI implementation as they overlap with the design process. For instance, Bowrouzian et al. (79) describe a new digital filter structure based on an equally resistively terminated lossless Jaumann two-port network. This analog filter is transformed into a digital filter using the bilinear transformation with compensation. The derived filter structure is suitable for fast parallel processing because all of the internal states can be computed in two steps. The structure requires the canonical number of multipliers. Alternatively, Kwan (80) describes a structure suitable for implementing a recursive filter on a systolic array. Finally, Dhar (81) describes techniques suitable for realizing very high-speed recursive filters such that the throughput rates that can be achieved are not limited by the atomicity of the device used to implement them. For instance, typically only one delay is tolerated in the feedback path. The article describes a technique that converts the structure into several processing paths that can be implemented by multiprocessing systems using commercial processors. The developed structures consist of modules that are repeated many times over with mostly local interconnections.

BIBLIOGRAPHY

1. N. Wiener and R. E. A. C. Paley, *Fourier Transforms in the Complex Domain,* Providence, RI: American Mathematical Society, 1934.

2. Y. H. Hu, CORDIC-based VLSI architecture for digital signal processing, *IEEE Signal Processing Magazine,* **9** (3): 16–35, 1992.
3. J. Makhoul, Linear prediction: a tutorial review, *Proc. IEEE,* **63**: 561–580, 1975.
4. M. H. Hayes, *Statistical Digital Signal Processing and Modeling,* New York: Wiley, 1996.
5. C. S. Burrus and T. W. Parks, Time domain design of recursive digital filters, *IEEE Trans. Audio Electroacoust.,* **18**: 137–141, 1970.
6. K. Steiglitz, On the simultaneous estimation of poles and zeros in speech analysis. *IEEE Trans. Acoust. Speech Signal Process.,* **25**: 229–234, 1977.
7. K. Steiglitz and L. E. McBride, A technique for the identification of linear systems, *IEEE Trans. Autom. Control,* **10**: 461–464, 1965.
8. A. I. Zverev, *Handbook of Filter Synthesis,* New York: Wiley, 1967.
9. Digital signal processing committee, *Programs for Digital Signal Processing,* New York: IEEE Press, 1979.
10. MATLAB®, *The Mathworks, Inc., User's Manual,* Boston, 1997.
11. L. B. Jackson, *Digital Filters and Signal Processing,* 2nd ed., Boston: Kluwer Academic Publishers, 1989.
12. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1975.
13. A. G. Constantinides, Spectral transformation for digital filters, *Proc. IEE,* **117**: 1585–1590, 1970.
14. V. E. DeBrunner and A. A. (Louis) Beex, Sensitivity analysis of digital filter structures, Invited paper *in SIAM J. Matrix Anal. Appl.,* **9**: 106–125, 1988. This paper originally appeared in B. N. Datta, C. R. Johnson, M. A. Kaashoek, R. Plemmons, and E. Sontag (eds.) *Linear Algebra in Signals, Systems, and Controls,* Philadelphia, PA: SIAM, 1988.
15. A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems,* Englewood Cliffs, NJ: Prentice-Hall, 1983.
16. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1989.
17. A. Fettweiss, Wave digital filters: theory and practice, *Proc. IEEE,* **74**: 270–327, 1986.
18. L. Gazsi, Explicit formulas for lattice wave digital filters, *IEEE Trans. Circuits Syst.,* **32**: 68–88, 1985.
19. C. T. Mullis and R. A. Roberts, Synthesis of minimum roundoff noise fixed point digital filters, *IEEE Trans. Circuits Syst.,* **23**: 551–562, 1976.
20. C. T. Mullis and R. A. Roberts, Roundoff noise in digital filters: Frequency transformations and invariants, *IEEE Trans. Acoust. Speech Signal Process.,* **24**: 538–550, 1976.
21. S. Y. Hwang, Minimum uncorrelated unit noise in state-space digital filtering, *IEEE Trans. Acoust. Speech Signal Process.,* **25**: 273–281, 1977.
22. L. B. Jackson, A. G. Lindgren, and Y. Kim, Optimal synthesis of second-order state-space structures for digital filters, *IEEE Trans. Circuits Syst.,* **26**: 149–153, 1979.
23. A. Antoniou, Improved minimax optimisation algorithms and their application in the design of recursive digital filters. *IEE Proc. G, Circuits, Devices and Syst.,* **138**: 724–730, 1991.
24. R. Fletcher, *Practical Methods of Optimization,* 2nd ed., New York: Wiley, 1987.
25. D. G. Luenberger, *Linear and Nonlinear Programming,* 2nd ed., Reading, MA: Addison-Wesley, 1984.
26. C. Charalambous, Design of 2-dimensional circularly-symmetric digital filters, *IEE Proc. G. Electron. Circuits Syst.,* **129**: 47–54, 1982.
27. C. Charalambous, Acceleration of the least pth algorithm for minimax optimization with engineering applications, *Math. Programming,* **17**: 270–297, 1979.
28. S.-C. Pei and J.-J. Shyu, Design of 1-D and 2-D IIR eigenfilters, *IEEE Trans. Signal Process.,* **42**: 962–966, 1996.
29. J.-H. Lee and C.-K. Chen, Recursive digital filter design in the complex domain using an efficient method, *Proc. IEEE Conf. Custom Integrated Circuits,* **5**: 2429–2432, 1992.
30. S. Ellacott and J. Williams, Rational Chebyshev approximation in the complex plane, *SIAM J. Numerical Anal.,* **13**: 310–323, 1976.
31. Y. C. Lim, J. H. Lee, C. K. Chen, and R. H. Yang, A weighted least squares algorithm for quasi-equiripple FIR and IIR digital filter design, *IEEE Trans. Signal Process.,* **40**: 551–558, 1992.
32. J.-H. Lee and S.-Y. Ku, Minimax design of recursive digital filters with a lattice denominator, *IEE Proc. Vision, Image and Signal Process.,* **143**: 377–382, 1996.
33. Z. Hui and D. Wang, The application of neural nets: The design of recursive digital filters, In *Proceedings of the China 1991 International Conference on Circuits Systems,* 1991, pp. 549–551.
34. P. Thajchayapong, K. Yammun, and A. Khunkitti, Recursive digital filters with predetermined group delay and Chebyshev stopband attenuation, *Electron. Lett.,* **24**: 1547–1549, 1988.
35. P. Thajchayapong and P. Lomtong, A maximally flat group delay recursive digital filter with Chebyshev stopband attenuation, *Proc. IEEE,* **66**: 255–257, 1978.
36. R. Unbehauen, Recursive digital low-pass filters with predetermined phase or group and Chebyshev stopband attenuation, *IEEE Trans. Circuits Syst.,* **28**: 905–912, 1981.
37. S. N. Hazra, Linear phase IIR filter with equiripple stopband, *IEEE Trans. Acoust. Speech Signal Process.,* **31**: 1047–1049, 1983.
38. R. Unbehauen, IIR digital filters with equiripple stopband transmission, *IEEE Trans. Acoust., Speech Signal Process.,* **33**: 744–746, 1985.
39. A. Fettweis, A simple design of maximally flat delay filters, *IEEE Trans. Audio Electroacoust.,* **20**: 112–114, 1972.
40. A. G. Deczky, Recursive digital filters having equiripple group delay, *IEEE Trans. Circuit Theory,* **18**: 664–669, 1971.
41. T. Trisuwannawat, K. Dejhan, and F. Cheevasuvit, A design technique of linear phase recursive digital filter with controllable magnitude at an arbitrary specified frequency, *IEEE Int. Symp. Circuits Syst.,* **5**: 2435–2438, 1991.
42. G. Gu and B. A. Shenoi, A novel approach to the synthesis of recursive digital filters with linear phase, *IEEE Trans. Circuits Syst.,* **38**: 602–612, 1992.
43. B. Jaworski and T. Saramaki, Linear phase IIR filters composed of two parallel allpass sections, *Proc. IEEE Int. Symp. Circuits Syst.,* **2**: 537–540, 1994.
44. S. Lawson, A new direct design technique for ALP recursive digital filters, *Proc. IEEE Int. Symp. Circuits Syst.,* **1**: 499–502, 1993.
45. H. Song and G. Gu, Phase approximation via Remez algorithm. In *Proceedings of the 27th Southeastern Symposium on System Theory,* 1995, pp. 441–444.
46. D. Braess, *Nonlinear Approximation Theory,* Amsterdam: Springer-Verlag, 1986.
47. V. S. Stojanovic and S. V. Nikolic, Direct design of transition Butterworth–Chebyshev recursive digital filters, *Electron. Lett.,* **29**: 286–287, 1993.
48. C. M. Rader and B. Gold, Digital filter design techniques in the frequency domain, *Proc. IEEE,* **55**: 149–171, 1967.
49. J. J. Soltis and M. A. Sid-Ahmed, Direct design of Chebyshev-type recursive digital filters, *Int. J. Electron.,* **70**: 413–419, 1991.

50. S. Lawson, Direct approach to design of PCAS filters with combined gain and phase specification, *IEE Proc. Vis. Image Signal Process.,* **141**: 161–167, 1994.

51. S. Lawson and A. Wicks, Design of efficient digital filters satisfying arbitrary loss and delay specifications, *IEE Proc.-Circuits Dev. and Syst.,* **139**: 611–620, 1992.

52. R. Gregorian and G. C. Temes, Design techniques for digital and analog all-pass circuits, *IEEE Trans. Circuits Syst.,* **25**: 981–988, 1978.

53. S. Erfani, M. Ahmadi, B. Khasnabish, and M. Shridhar, Designing recursive digital filters by inverse Simpson's transform. In *Proceedings of the 35th MWSCAS,* 1992, pp. 942–944.

54. S. Torres, Design and analysis of an adaptive digital notch filter, M.S. Thesis, Norman, OK: The University of Oklahoma, 1997.

55. M. Musa and H. Al-Ahmad, Optimisation of complex recursive digital filters operating in transient mode. In *Proceedings of the IEE 15th SARAGA Colloquium Digital and Analogue Filters and Filtering Systems,* 1995, pp. 16/1–5.

56. V. Tavsonoglu and L. Thiele, Optimal design of state-space digital filters by simultaneous minimization of sensitivity and roundoff noise. *IEEE Trans. Circuits Syst.,* **31**: 884–888, 1984.

57. D. V. B. Rao, Analysis of coefficient quantization errors in state-space digital filters. *IEEE Trans. Acoust. Speech Signal Process.,* **34**: 131–139, 1986.

58. R. M. Godall, A practical method for determining coefficient word length in digital filters, *IEEE Trans. Signal Process.,* **40**: 981–985, 1992.

59. J. B. Farison and S. R. Kolla, Relationship of singular value stability robustness bounds to spectral radius for discrete systems with application to digital filters, *IEE Proc-G,* **138**: pp. 5–8, 1991.

60. D. Baez-Lopez, C. Cabanas-Villa, and M. Hernandez-Apam, Design considerations for very low sensitivity and very low round-off noise recursive digital filters, *Proc. IEEE Pac. Rim Conf. Commun. Comput. Signal Process.,* **2**: 415–418, 1993.

61. S. Worthington and L. E. Turner, A method of evaluating the effects of signal quantization at arbitrary locations in recursive digital filters, *IEEE Int. Symp. Circuits Syst.,* **1**: 615–618, 1993.

62. G. Mollova, Variance of quantization error at the output of recursive digital filter. In *Proceedings of URSI International Symposium on Signal System Electronics,* 1995, pp. 439–442.

63. T. C. Macedo, Jr., T. Laakso, P. S. R. Diniz, and I. Hartimo, Reformulation of Chang's criterion for the absence of limit cycles using bilinear transform, *IEEE Int. Symp. Circuits Syst.,* **1**: 388–391, 1991.

64. M. Price and M. B. Sandler, Performance of novel structures for high order recursive digital filters, *IEE Col. Dig. & Anal. Filters Filtering Syst.,* 1993, 4/1–4.

65. M. B. Sandler, Implementation of high order recursive filters as sub-filters, *IEEE Int. Symp. Circuits Syst.,* **1**: 599–602, 1993.

66. P. P. Vaidyanathan, S. K. Mitra, and Y. Neuvo, A new approach to the realization of low-sensitivity IIR digital filters, *IEEE Trans. Acoust. Speech Signal Process.,* **34**: 350–361, 1986.

67. X. Nie, D. Raghuramireddy, and R. Unbehauen, Allpass expansions of digital transfer functions, *Electron. Lett.,* **27**: 1438–1440, 1991.

68. T. Saramaki, Generalizations of classical recursive digital filters and their design with the aid of a Remez-type algorithm, *IEEE Int. Symp. Circuits Syst.,* **2**: 549–552, 1994.

69. X. Nie, D. Raghuramireddy, and R. Unbehauen, Orthonormal expansion of stable rational transfer functions, *Electron. Lett.,* **27**: 1492–1494, 1991.

70. B. W. Bomar and J. C. Hung, Minimum roundoff noise digital filters with some power-of-two coefficients, *IEEE Trans. Circuits Syst.,* **31**: 833–840, 1984.

71. B. W. Bomar, New second-order state-space structures for realizing low roundoff noise digital filters, *IEEE Trans. Acoust. Speech Signal Process.,* **33**: 106–110, 1985.

72. B. W. Bomar, Computationally efficient low roundoff noise second-order state-space structures, *IEEE Trans. Circuits Syst.,* **33**: 35–41, 1986.

73. M. F. Fahmy and G. A. Raheem, Synthesis of fixed-point low roundoff noise digital filters with no limit cycle, *IEEE Int. Symp. Circuits Syst.,* **2**: 485–488, 1994.

74. L. B. Jackson, An improved Martinet/Parks algorithm for IIR design with unequal numbers of poles and zeros, *IEEE Trans. Signal Process.,* **42**: 1234–1238, 1994.

75. A. A. (Louis) Beex and V. E. DeBrunner, Reducing sensitivities of direct form digital (sub) filter structures by increasing system order, *IEEE Trans. Circuits Syst.,* **36**: 438–442, 1989.

76. V. E. DeBrunner, T. A. Tutunji, and A. R. Corzine, Methods to design low sensitivity canonical digital filters using impulse response data, *Proc. IEEE Int. Conf. Circuits Syst.,* **3**: 445–448, 1996.

77. S.-H. Leung, A realization of narrow-band recursive digital low-pass filter using highly quantized coefficients, *IEEE Trans. Circuits Syst.,* **36**: 618–622, 1989.

78. T. I. Laakso and I. O. Hartimo, Noise reduction in recursive digital filters using high-order error feedback, *IEEE Trans. Signal Process.,* **40**: 1096–1107, 1992.

79. B. Nowrouzian, N. R. Bartley, and L. T. Bruton, Design and DSP-chip implementation of a novel bilinear-LDI digital Jaumann Filter. *IEEE Trans. Circuits Syst.,* **37**: 695–706, 1990.

80. H. K. Kwan, New form of delayed *N*-part recursive digital filters. *Electron. Lett.,* **29**: 736–738, 1993.

81. K. K. Dhar, Very high speed real-time IIR digital filter structures: Suitable for VLSI implementation. *Proc. IEEE Int. Symp. Circuits Syst.,* **1**: 623–626, 1993.

Additional Reading

In addition to the references used and listed above, the author has found the following material to be useful in the design of recursive digital filters. The references are listed in no particular order.

C. B. Rorabaugh, *Digital Filter Designer's Handbook (Featuring C Routines),* New York: McGraw-Hill, 1993.

J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications,* 3rd ed., Upper Saddle River, NJ: Prentice-Hall, 1996.

S. J. Orfanidis, *Introduction to Signal Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1996.

S. K. Mitra and J. F. Kaiser (eds.), *Handbook for Digital Signal Processing,* New York: Wiley, 1993.

VICTOR E. DEBRUNNER
The University of Oklahoma

RECURSIVE FUNCTIONS.

See COMPUTABILITY.