

## The Swiss Army Knife of Digital Networks

This article describes a general discrete-signal network that appears, in various forms, inside many digital signal processing (DSP) applications. So the “DSP Tip” for this column is for every DSP engineer to become acquainted with this network. Figure 1 shows how the network’s structure has the distinct look of a digital filter, a comb filter followed by a second-order recursive network. However, we do not call this unique general network a filter because its capabilities extend far beyond simple filtering. Through a series of examples, we illustrate the fundamental strength of the network: its ability to be reconfigured to perform a surprisingly large number of useful functions based on the values of its seven control parameters.

The general network has a transfer function of

$$H(z) = (1 - c_1 z^{-N}) \times \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1/a_0 - a_1 z^{-1} - a_2 z^{-2}}. \quad (1)$$

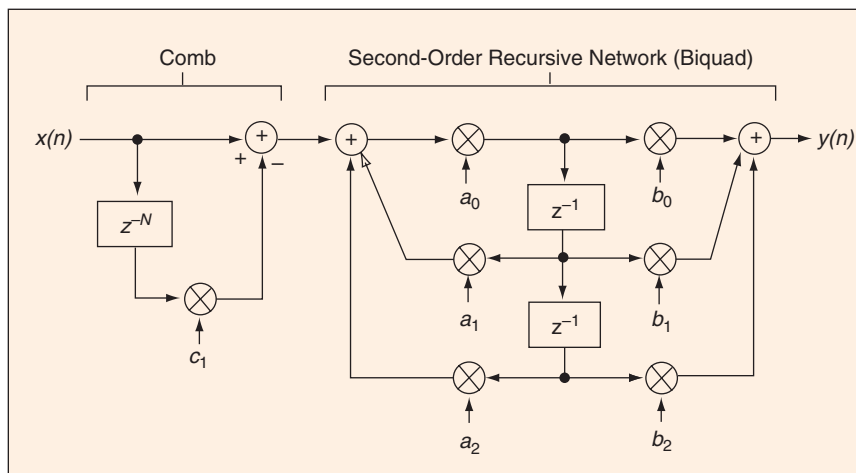
From this point on, we’ll use DSP filter lingo and call the second-order recursive network a “biquad” because its transfer function is the ratio of two quadratic polynomials. The tables in this article list various signal-processing functions performed by the network based on the  $a_n$ ,  $b_n$ , and  $c_1$  coefficients. The variable  $N$  is the order of the comb filter. Included in the tables are depictions of the network’s impulse response,  $z$ -plane pole/zero locations, as well as frequency-domain

magnitude and phase responses. The frequency axis in those tables is normalized such that a value of 0.5 represents a frequency of  $f_s/2$ , where  $f_s$  is the sample rate in hertz.

### Moving Averager

Referring to the first entry in Table 1, this network configuration is a computationally efficient method for computing the  $N$ -point moving average of  $x(n)$ . Also called a *recursive running sum* or *boxcar averager*, this structure is equivalent to an  $N$ -tap direct convolution finite impulse response (FIR) filter with all the coefficients having a value of  $1/N$ . However, this moving averager is efficient because it performs only one add and one subtract per output sample regardless of the value of  $N$  (whereas an  $N$ -tap direct convolution FIR filter must perform  $N-1$  additions per output sample). The moving averager’s transfer function is

$$H_{ma}(z) = \frac{(1/N)(1 - z^{-N})}{(1 - z^{-1})}.$$



▲ 1. General discrete-signal processing network.

“DSP Tips and Tricks” introduces practical tips and tricks of design and implementation of signal processing algorithms so that you may be able to incorporate them into your designs. We welcome readers who enjoy reading this column to submit their contributions. Contact Associate Editors Rick Lyons (r.lyons@ieee.org) or Amy Bell (abell@vt.edu).

$H_{\text{ma}}(z)$ 's numerator results in  $N$  zeros equally spaced around the  $z$ -plane's unit circle located at  $z(k) = e^{j2\pi k/N}$ , where integer  $k$  is  $0 \leq k < N$ .  $H_{\text{ma}}(z)$ 's denominator places a single pole at  $z = 1$  on the unit circle, canceling the zero at that location.

## Differencer

This is a discrete version of a first-order differentiator. An ideal differentiator has a frequency magnitude response that is a linear function of frequency, and this network only approaches that ideal at low frequencies relative to  $f_s$ .

## Integrator

This structure performs the *running summation* of the  $x(n)$  inputs samples, making it the discrete-time equivalent of a continuous-time integrator.

## Leaky Integrator

This network configuration, also called an *exponential averager*, is a venerable structure used in low-pass filter implementations for random noise reduction. It is a first-order infinite impulse response (IIR) filter where, for stable low-pass operation, the constant  $\alpha$  lies in the range  $0 < \alpha < 1$ .

This nonlinear-phase filter has a single pole at  $z = 1 - \alpha$  on the  $z$ -plane and a transfer function of  $H_{\text{li}}(z) = \alpha/[1 - (1 - \alpha)z^{-1}]$ . Small values for  $\alpha$  yield narrow passbands at the expense of increased filter response time. Table 1 shows the filter's behavior for  $\alpha = 0.1$  as solid curves. For comparison, the frequency domain performance for  $\alpha = 0.5$  is indicated by the dashed curves.

## First-Order Delay Network

A subclass of a first-order IIR filter, the coefficients in Table 1 yield an *all-pass* network having a relatively constant group delay at low frequencies. The network's delay is

$D_{\text{total}} = 1 + \Delta_{\text{delay}}$  samples where  $\Delta_{\text{delay}}$ , typically in the range of  $-0.5$  to  $0.5$ , is a fraction of the  $1/f_s$  sample period. For example, when  $\Delta_{\text{delay}}$  is  $0.2$ , the network delay (at low frequencies) is  $1.2$  samples. The real-valued  $R$  coefficient is

$$R = \frac{-\Delta_{\text{delay}}}{\Delta_{\text{delay}} + 2}, \quad (2)$$

producing a  $z$ -plane transfer function of  $H_{1,\text{del}}(z) = (R + z^{-1})/(1 + Rz^{-1})$  with a pole at  $z = -R$  and a zero at  $z = -1/R$ .

Performance for  $\Delta_{\text{delay}} = 0.2$  ( $R = 0.91$ ) is shown in Table 1, where we see the magnitude response being constant. The band, centered at dc, over which the group delay varies no more than  $|\Delta_{\text{delay}}|/10$  from the specified  $D_{\text{total}}$  value, the bar in the group delay plot, ranges roughly from  $0.1f_s$  to  $0.2f_s$  for first-order networks. If your signal is oversampled, making it low in frequency relative to  $f_s$ , this first-order all-pass delay network may be of some use. If you propose its use in a new design, you can impress your colleagues by saying this network is based on the Thiran approximation [1].

## Second-Order Delay Network

A subclass of a second-order IIR filter, the coefficients in Table 1 yield an all-pass network having a relatively constant group at low frequencies (over a wider frequency range, by the way, than the first-order delay network.) This network's delay is  $D_{\text{total}} = 2 + \Delta_{\text{delay}}$  samples, where  $\Delta_{\text{delay}}$  is typically in the range of  $-0.5$  to  $0.5$ . For example, when  $\Delta_{\text{delay}}$  is  $0.3$ , the network delay (at low frequencies) is  $2.3$  samples. The real-valued coefficients are

$$R_1 = \frac{-2\Delta_{\text{delay}}}{\Delta_{\text{delay}} + 3}$$

and

$$R_2 = \frac{(\Delta_{\text{delay}})(\Delta_{\text{delay}} + 1)}{(\Delta_{\text{delay}} + 3)(\Delta_{\text{delay}} + 4)}. \quad (3)$$

The band, centered at dc, over which the group delay varies no more than  $|\Delta_{\text{delay}}|/10$  from the specified  $D_{\text{total}}$  value, the bar in the group delay plot, ranges roughly from  $0.26f_s$  to  $0.38f_s$  for this second-order network. Performance for  $\Delta_{\text{delay}} = 0.3$  ( $R_1 = -0.182$  and  $R_2 = 0.28$ ) is shown in Table 1, where we see the magnitude response being constant.

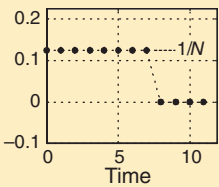
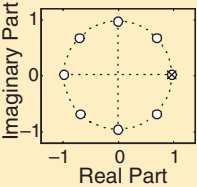
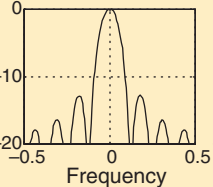
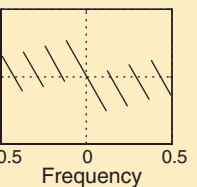
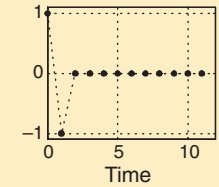
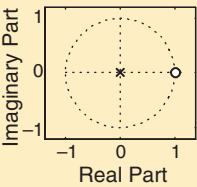
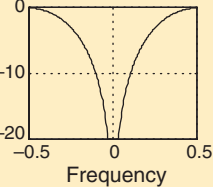
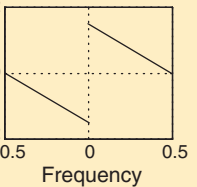
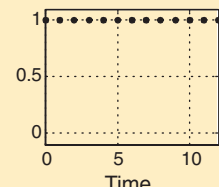
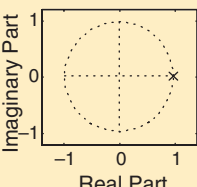
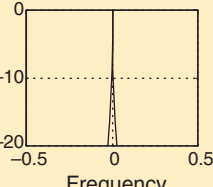
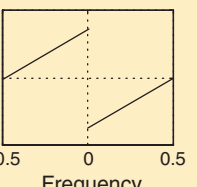
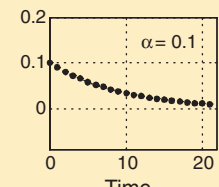
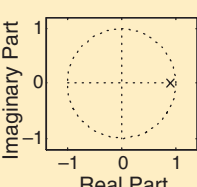
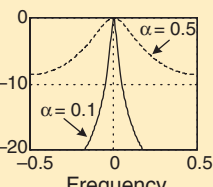
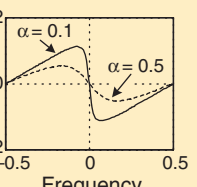
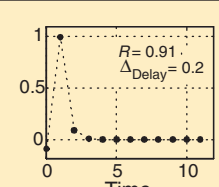
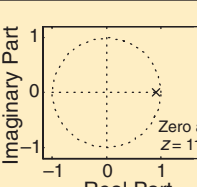
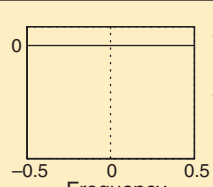
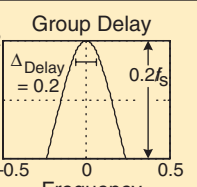
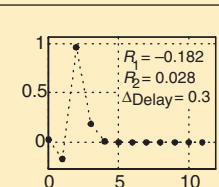
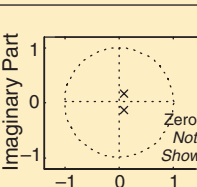
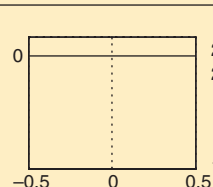
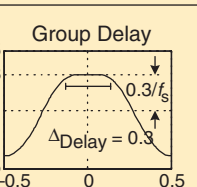
The *flat* group delay band is wider for negative  $\Delta_{\text{delay}}$  than when  $\Delta_{\text{delay}}$  is positive. For example, this means if you desire a group delay of  $D_{\text{total}} = 2.5$  samples, it is better to use an external unit delay and set  $\Delta_{\text{delay}}$  to  $-0.5$  rather than letting  $\Delta_{\text{delay}}$  be  $0.5$ . To ensure stability,  $\Delta_{\text{delay}}$  must be greater than  $-1$ . Reference [1] provides methods for designing higher-order allpass delay networks.

## Goertzel Network

Referring to the first entry in Table 2, this traditional Goertzel network is used for single-tone detection because it computes a single-bin  $N$ -point discrete Fourier transform (DFT) centered at an angle of  $\theta = 2\pi k/N$  rad on the unit circle, corresponding to a cyclic frequency of  $k f_s/N$  Hz. Frequency variable  $k$ , in the range  $0 \leq k < N$ , need not be an integer. The behavior of the network is shown by the solid curves in Table 2. However, the frequency magnitude response of the Goertzel algorithm, for  $N = 8$  and  $k = 1$ , is shown as the dashed curve.

After  $N+1$  input samples are applied,  $y(n)$  is a single-bin DFT result. The DFT computational workload is  $N + 2$  real multiplies and  $2N + 1$  real adds. The network is typically stable because  $N$  is kept fairly low (in the hundreds) in practice before the network is reinitialized [2], [3].

**Table 1. General functions.**

Functions and Coefficients	Impulse Response	$z$ -plane	Magnitude (dB)	Phase (rad.)
<b>Moving Averager</b> $a_0 = 1, a_1 = 1, a_2 = 0,$ $b_0 = 1/N, b_1 = 0, b_2 = 0,$ $c_1 = 1, N = 8$				
<b>Differencer</b> $a_0 = 1, a_1 = 0, a_2 = 0,$ $b_0 = 1, b_1 = -1, b_2 = 0,$ $c_1 = 0$				
<b>Integrator</b> $a_0 = 1, a_1 = 1, a_2 = 0,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_i = 0$				
<b>Leaky Integrator</b> $a_0 = 1, a_1 = 1 - \alpha, a_2 = 0,$ $b_0 = \alpha, b_1 = 0, b_2 = 0,$ $c_1 = 0, \alpha = 0.1$				
<b>First-Order Delay Network</b> $a_0 = 1, a_1 = -R, a_2 = 0,$ $b_0 = R, b_1 = 1, b_2 = 0,$ $c_1 = 0$				
<b>Second-Order Delay Network</b> $a_0 = 1, a_1 = -R_1, a_2 = -R_2,$ $b_0 = R_2, b_1 = R_1, b_2 = 1,$ $c_1 = 0$				

## Sliding DFT Network

This structure computes a single-bin  $N$ -point DFT centered at an angle of  $\theta = 2\pi k/N$  rad on the unit circle, corresponding to a cyclic frequency of  $k f_s/N$  Hz.  $N$  is the DFT

size and integer  $k$  is  $0 \leq k < N$ . The real damping factor  $r$  is kept as close to, but less than, unity as possible to maintain network stability. After  $N$  input samples have been applied, this network will compute

a new follow-on DFT result based on each new  $x(n)$  sample (thus the term *sliding*) at a computational workload of only four real multiplies and four real adds per input sample [2], [3]. Setting coefficient

**Table 2. Analysis and synthesis functions.**

Functions and Coefficients	Network Behavior			
	Impulse Response	$z$ -plane	Magnitude (dB)	Phase (rad.)
<b>Goertzel Network</b> $a_0 = 1, a_1 = 2\cos(\theta),$ $a_2 = -1, b_0 = 1, b_1 = -e^{-j\theta},$ $b_2 = 0, c_1 = 0, \theta = 2\pi k/N$				
<b>Sliding DFT Network</b> $a_0 = re^{j\theta}, a_1 = 1, a_2 = 0,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_1 = r^N, \theta = 2\pi k/N$				
<b>Real Oscillator</b> $a_0 = 1, a_1 = 2\cos(\theta),$ $a_2 = -1, b_0 = 1,$ $b_1 = 0, b_2 = -1$				
<b>Quadrature Oscillator</b> $a_0 = G(n), a_1 = e^{j\theta},$ $a_2 = 0, b_0 = 1,$ $b_1 = 0, b_2 = 0$				
<b>Audio Comb</b> $a_0 = 1, a_1 = 0, a_2 = \alpha,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_1 = 0, \alpha = 0.2$				

$c_1 = -r^N$  allows the analysis band to be centered at an angle of  $\theta = 2\pi (k + 1/2)/N$  rad, corresponding to a cyclic frequency of  $(k + 1/2)f_s/N$  Hz.

## Real Oscillator

There are many possible digital oscillator structures, but this network generates a real-valued sinusoidal  $y(n)$  sequence whose

amplitude is not a function of the output frequency. The argument for coefficient  $a_1$  in Table 2 is  $\theta = 2\pi f_t/f_s$  rad, where  $f_t$  is the oscillator's tuned frequency in hertz. To start the oscillator we set the  $y(n - 1)$  sample driving the  $a_1$  multiplier equal to 1 and compute new output samples as the time index  $n$  advances. For fixed-point implementations, filter coefficients

may need to be scaled so that all intermediate results are in the proper numerical range [4].

## Quadrature Oscillator

Called the *coupled quadrature oscillator*, this structure provides  $y(n) = \cos(n\theta) + j\sin(n\theta)$  outputs for a complex exponential sequence whose tuned frequency is  $f_t$  Hz. The exponent for  $a_1$  in Table 2 is

$\theta = 2\pi f_t/f_s$  rad. To start the oscillator, we set the complex  $y(n-1)$  sample, driving the  $a_1$  multiplier, equal to  $1 + j0$  and begin computing output samples as the time index  $n$  advances. To ensure oscillator output stability in fixed-point arithmetic implementations, instantaneous gain correction  $G(n)$  must be computed for each output sample. The  $G(n)$  sample values will be very close to unity [5], [6].

### Audio Comb

This structure is a second-order (the simplest) version of an IIR comb filter used by audio folks to synthesize the sound of a plucked-string instrument. The input to the filter is random noise samples. The filter has frequency response peaks at dc and  $\pm f_s/2$ , with dips in the response located at  $\pm f_s/4$ . The filter's transfer function is  $H_{ac}(z) = 1/(1 - \alpha z^{-2})$ , resulting in two poles located at  $z = \pm\sqrt{\alpha}$  on the  $z$ -plane. To maintain stability the real-valued  $\alpha$  must be less than unity, and the closer  $\alpha$  is to unity, the more narrow the frequency response peaks.

For a more realistic-sounding synthesis, we can set  $a_1 = \alpha$  and the top delay element of the biquad in Figure 1 may have its delay increased to, say, eight instead of one, yielding more frequency response peaks between 0 and  $f_s/2$  Hz. In this music application, the filter's input is Gaussian white noise samples. Other plucked-string instrument synthesis networks have been used with success [7], [8].

### Comb Filter

Referring to the first entry in Table 3, this standard comb filter is a key component on many filtering applications, as we shall see. Its transfer function,  $H_{comb}(z) = 1 - z^{-N}$ , results in  $N$  zeros equally spaced around the  $z$ -plane's unit circle located at  $z(k) = e^{j2\pi k/N}$ , where integer  $k$  is  $0 \leq k < N$ . Those  $z(k)$

values are the  $N$  roots of unity when we set  $H_{comb}(z)$  equal to zero yielding  $z(k)^N = (e^{j2\pi k/N})^N = 1$ . The  $N$  zeros on the unit circle result in frequency response nulls (infinite attenuation) located at cyclic frequencies of  $m f_s/N$ , where integer  $m$  is  $0 \leq m \leq N/2$ . The peak gain of this linear-phase filter is two.

If we set coefficient  $c_1$  to  $-1$  in the comb filter, making its transfer function  $H_{alt,comb}(z) = 1 + z^{-N}$ , we obtain an alternate linear-phase comb filter having zeros rotated counterclockwise around the unit circle by an angle of  $\pi/N$  rad positioning the zeros at angles of  $2\pi(k + 1/2)/N$  rad on the  $z$ -plane's unit circle. The rotated zeros result in frequency response nulls located at cyclic frequencies of  $(m + 1/2)f_s/N$ . With this filter a frequency magnitude peak is located at 0 Hz (dc).

### Bandpass Filter at $f_s/4$

This network is a bandpass filter centered at  $f_s/4$  having a  $\sin(x)/x$ -like frequency response and linear-phase over the passband. It has poles at  $z = \pm j$ , so for pole/zero cancellation the comb filter's delay ( $N$ ) must be an integer multiple of four. This guaranteed-stable, multiplierless, bandpass filter's transfer function is  $H_{bp}(z) = (1 - z^{-N})/(1 + z^{-2})$ .

### First-Order IIR Filter

This is the direct form II version of a simple first-order IIR filter having a single pole located at a radius of  $R_p$  from the  $z$ -plane's origin at an angle of  $\theta_p$  rad and a single zero at a radius of  $R_z$  at an angle of  $\pi + \theta_z$ . For real-valued coefficients ( $\theta_p = \theta_z = 0$ ), the filter can only exhibit either a low-pass or a high-pass frequency response; no bandpass or bandstop filters are possible. The filter's transfer function is  $H_{1,iir}(z) = (1 + R_p e^{j\theta_p} z^{-1}) / (1 - R_z e^{j\theta_z} z^{-1})$ .

The shape of the filter's frequency magnitude responses are nothing to

write home about; its transition regions are so wide that they don't actually have distinct passbands and stopbands. Of course, to ensure stability,  $R_p$  must be between zero and one to keep the pole inside the  $z$ -plane's unit circle, and the closer  $R_p$  is to unity, the more narrowband is the filter.

### First-Order Equalizer

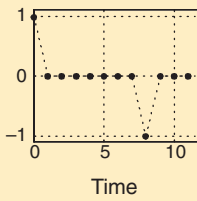
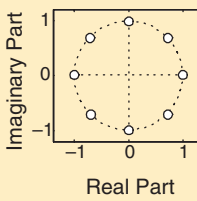
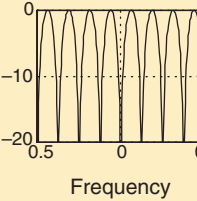
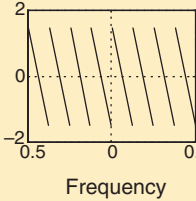
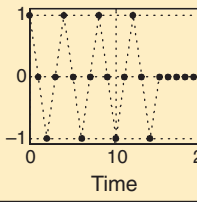
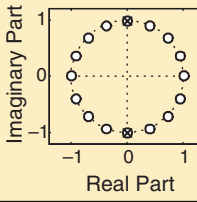
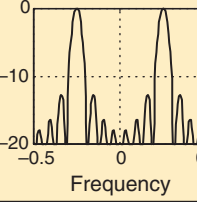
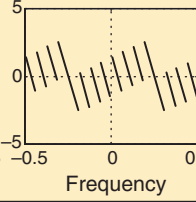
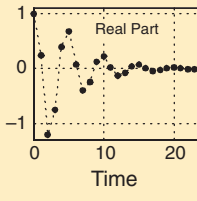
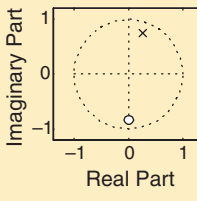
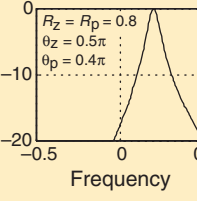
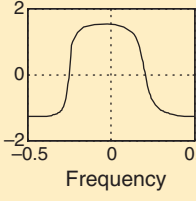
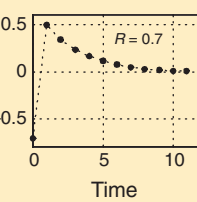
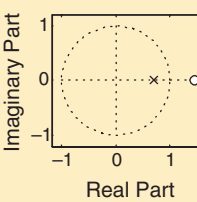
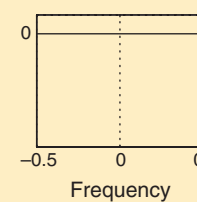
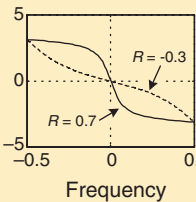
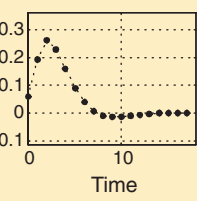
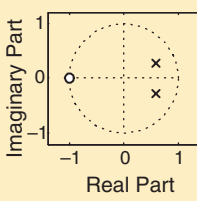
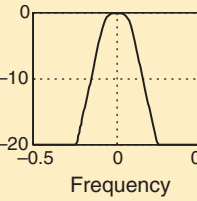
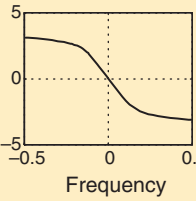
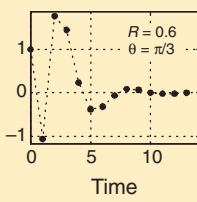
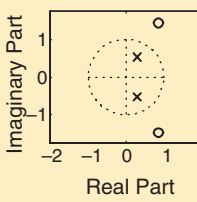
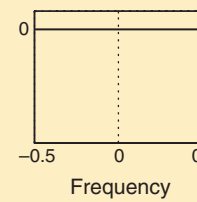
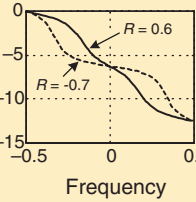
This structure has a frequency magnitude response that is constant across the entire frequency band (an allpass filter). It has a pole at  $z = R$  on the  $z$ -plane and a zero located at  $1/R^*$ , where  $*$  means conjugate. The value of  $R$ , which can be real or complex but whose magnitude must be less than unity to ensure stability, controls the nonlinear-phase response. The equalizer has a transfer function of  $H_{1,eq}(z) = (-R^* + z^{-1}) / (1 - R z^{-1})$ .

These networks can be used as *phase equalizers* by cascading them after a filter or network whose nonlinear phase response requires crude linearization. The goal is to make the cascaded filters' combined phase as linear as possible. Table 3 shows the filter's behavior for  $R = 0.7$  as solid curves. For comparison, the phase response for  $R = -0.3$  is indicated by the dashed curve. These first-order allpass filters can also be used for interpolation and audio reverberation for low-frequency signals.

### Second-Order IIR Filter

This is the direct form II version of a second-order IIR filter, the workhorse of IIR filter implementations. Conjugate pole and zero pairs may be positioned anywhere on the  $z$ -plane to control the filter's frequency response. [There's a terrific piece of MATLAB code (PEZ, created by the talented Dr. Craig Ulmer) that allows us to see the frequency-domain effect of moving multiple poles and zeros, manually

**Table 3. Filter functions.**

Functions and Coefficients	Network Behavior			
	Impulse Response	$z$ -plane	Magnitude (dB)	Phase (rad.)
<b>Comb Filter</b> $a_0 = 1, a_1 = 0, a_2 = 0,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_1 = 1, N = 8$				
<b>Bandpass Filter at <math>f_s/4</math></b> $a_0 = 1, a_1 = 0, a_2 = -1,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_1 = 1, N = 16$				
<b>First-Order IIR Filter</b> $a_0 = 1, a_1 = R_p e^{j\theta_p},$ $a_2 = 0, b_0 = 1,$ $b_1 = R_z e^{j\theta_z}, b_2 = 0,$ $c_1 = 0$				
<b>First-Order Equalizer</b> $a_0 = 1, a_1 = R, a_2 = 0,$ $b_0 = -R^*, b_1 = 1, b_2 = 0,$ $c_1 = 0$				
<b>Second-Order IIR Filter</b> $a_0 = 1, a_1 = 1.194,$ $a_2 = -0.436,$ $b_0 = b_2 = 0.0605,$ $b_1 = 0.121, c_1 = 0$				
<b>Second-Order Equalizer</b> $a_0 = 1, a_1 = 2R\cos(\theta),$ $a_2 = -R^2, b_0 = 1,$ $b_1 = -(2/R)\cos(\theta),$ $b_2 = 1/R^2, c_1 = 0$				

using a mouse, around on the  $z$ -plane. The code is available at <http://www.cspl.umd.edu/spm/tips-n-tricks/>.] Because high-order IIR filters are so susceptible

to coefficient quantization and potential data overflow problems, practitioners typically implement their IIR filters by cascading multiple copies of this second-order

IIR structure to ensure filter stability and avoid limit cycles. The filters have a transfer function of (1) with the  $c_1 = 0$ . Low-pass, high-pass, bandpass, and bandstop



**Table 4. Additional filter functions.**

Functions and Coefficients	Network Behavior			
	Impulse Response	$z$ -plane	Magnitude (dB)	Phase (rad.)
<b>CIC Interpolation Filter</b> $a_0 = 1, a_1 = 1, a_2 = 0,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_1 = 1, N = 8$				
<b>Complex FSF</b> $a_0 = 1, a_1 = e^{j\theta_k}, a_2 = 0,$ $b_0 = (-1)^k, b_1 = 0, b_2 = 0,$ $c_1 = 1, \theta_k = 2\pi k/N$				
<b>Real FSF, Type I</b> $a_0 = 1, a_1 = 2\cos(\theta_k),$ $a_2 = -1, b_0 =  H_k \cos(\phi_k),$ $b_1 = - H_k \cos(\phi_k - \theta_k),$ $b_2 = 0, c_1 = 1, \theta_k = 2\pi k/N$				
<b>Real FSF, Type IV</b> $a_0 = 1, a_1 = 2\cos(\theta_k),$ $a_2 = -1, b_0 = (-1)^k M_k,$ $b_1 = 0, b_2 = (-1)^k (-M_k),$ $c_1 = 1, \theta_k = 2\pi k/N$				
<b>dc Bias Removal</b> $a_0 = 1, a_1 = \alpha, a_2 = 0,$ $b_0 = 1, b_1 = -1, b_2 = 0,$ $c_1 = 0$				

filters are possible. No single example shows all the possibilities of this structure, so Table 3 merely gives a simple low-pass filter example.

If an IIR filter design requires high performance, high Q, it turns out the direct form I version of a second-order IIR filter is less susceptible to coefficient quantization and overflow errors than the direct form II structure given here.

## Second-Order Equalizer

This structure has a frequency magnitude response that's constant across the entire frequency band, making it also an allpass filter. It has two conjugate poles located at a radius of  $R$  from the  $z$ -plane's origin at angles of  $\pm\theta$  rad and two conjugate zeros at a reciprocal radius of  $1/R$  at angles of  $\pm\theta$ . The positioning of the poles and zeros, using real-valued  $R$ , controls the nonlinear-phase response.

Table 3 shows the equalizer's behavior for  $R = 0.6$  and  $\theta = \pi/3$  as solid curves. For comparison, the phase response for  $R = -0.7$  and  $\theta = \pi/3$  is indicated by the dashed curve.

These networks are primarily used for phase equalization by cascading them after a filter or network whose nonlinear phase response requires linearization. It may take multiple cascaded biquad

networks to achieve acceptable equalization, however.

## CIC Interpolation Filter

Referring to the first entry in Table 4, this network is a single-stage cascaded integrator-comb (CIC) interpolation filter used for time-domain interpolation. If a time-domain signal sequence is upsampled by  $N$  (by inserting  $N - 1$  zero-valued samples in between each original sample) and applied to this low-pass filter, the filter's output is an interpolated by  $N$  version of the original signal.  $H_{\text{cic}}(z) = (1 - z^{-N})/(1 - z^{-1})$  is this low-pass filter's transfer function. To improve the attenuation of spectral images, we can cascade  $M$  copies of the comb filter followed by  $M$  cascaded biquad sections. Such cascaded filters will also have narrower passband widths at 0 Hz.

In practice, the upsampling operation (zero stuffing) is performed after the comb filter and before the biquad network. This has the sweet advantage that the comb filter's delay length becomes  $N = 1$ , reducing the necessary comb delay storage requirement to one. CIC filters are typically used as the first stage of multistage low-pass filtering in hardware sample rate increase (interpolation) by  $N$  applications because no multipliers are required [6].

## Complex Frequency Sampling Filter

This structure is a single section of a complex *frequency sampling filter* (FSF) having a  $\sin(x)/x$ -like frequency magnitude response centered at an angle of  $\theta_k = 2\pi k/N$  rad on the unit circle, corresponding to a cyclic frequency of  $k f_s/N$  Hz.  $N$  and  $k$  are integers when  $k$  is  $0 \leq k < N$ . The larger  $N$ , the more narrow the filter's mainlobe width [6].

If multiple biquads are implemented in parallel (all driven by the

single comb filter) with adjacent center frequencies, complex almost-linear phase-bandpass filters can be built. Table 4 shows the behavior of an  $N = 16$ , three-biquad, complex bandpass filter, each centered at  $k = 2, 3$ , and 4, respectively.

## Real Frequency Sampling Filter, Type I

This structure is a single section of a real-coefficient *frequency sampling filter* having a  $\sin(x)/x$ -like frequency magnitude response centered at both  $\pm\theta_k = \pm 2\pi k/N$  rad, where  $N$  is an integer. The larger  $N$  the more narrow than the filter's mainlobe width. Integer  $k$  is  $0 \leq k < N$ .

If multiple biquads are implemented in parallel (all driven by the single comb filter) with adjacent center frequencies, almost-linear phase, low-pass filters can be built. In this case, complex gain factors  $H_k$  are the desired peak frequency response of the  $k$ th biquad. Parameter  $\phi_k$  is the desired relative phase shift, in radians, of  $H_k$ . Table 4 shows the behavior of an  $N = 22$ , three-biquad, low-pass filter each centered at  $k = 0, 1$ , and 2, respectively. In this example,  $|H_0| = 1$ ,  $|H_1| = 2$ , and  $|H_2| = 0.74$ . These bandpass filters can have group delay fluctuations as large as  $2/f_s$  in the passband. This recursive FIR filter is the most common frequency sampling filter discussed in the DSP textbooks [6], [9], [10].

## Real Frequency Sampling Filter, Type IV

This structure is similar in behavior to the type I frequency sampling filter, with important exceptions. First, in multibiquad low-pass filter implementations this filter yields an exactly linear phase response. Also, this filter provides deeper stopband attenuation than the Type I filter.

The real-valued gain factors  $M_k$  are the desired peak frequency mag-

nitude response of the  $k$ th biquad. Table 4 shows the behavior of an  $N = 22$ , three-biquad low-pass filter with the biquads centered at  $k = 0, 1$ , and 2, respectively. In this example,  $M_0 = 1$ ,  $M_1 = 2$ , and  $M_2 = 0.74$ . Here's why you need to know about these filters: with judicious choice of  $M_k$  gain factors, narrowband low-pass linear-phase FIR filters can be built, in some cases, whose computational workload is less than Parks-McClellan-designed FIR filters [6].

## DC Bias Removal

This network, used to remove any dc bias from the  $x(n)$  input, has a transfer function with a pole located at  $z = \alpha$  and a zero at  $z = 1$ . Having a frequency response notch (null) at 0 Hz (dc, hence the name) and the sharpness of the notch is determined by  $\alpha$ , where for stable operation  $\alpha$  lies in the range  $0 < \alpha < 1$ . The closer  $\alpha$  is to unity, the more narrow the notch at dc. This nonlinear-phase filter has a transfer function of  $H_{\text{dc}}(z) = (1 - z^{-1})/(1 - \alpha z^{-1})$ . Table 4 shows the filter's behavior for  $\alpha = 0.8$ .

In those fixed-point implementations where the output  $y(n)$  sequence must be truncated to avoid data overflow [that is,  $y(n)$  must have fewer bits than input  $x(n)$ ], feedback noise shaping can be used to reduce the quantization noise induced by truncation [6], [11].

To avoid overflow, an alternative to truncation is to limit the gain of the filter. For example, we could precede the network with a positive gain element whose gain is less than unity. On the other hand, we could use  $b_0 = G$  and  $b_1 = -G$ , where  $G = (1 + \alpha)/2$ , in our implementation for this purpose, yielding a reduced-gain transfer function of  $H_{\text{alt,dc}}(z) = (G - Gz^{-1})/(1 - \alpha z^{-1})$ .

(continued on page 100)



**Charles:** I'm now working half time (or rather, half effort). I don't know when I will fully retire. My wife has been renovating a property on Cape Cod, which is nearly fin-

ished. I like to joke that it's rising, Phoenixlike, from the ashes of our savings. When I can spend some more time there, I want to try writing about how public policy deci-

sions are made when they are based on technology.

**SPM:** *It has been an honor. Thank you and hope we will do it again.*

## dsp tips &amp; tricks continued from page 97

If the reader has any comments regarding this article, please e-mail one of the authors. Feedback from our readers, either positive or negative, is most welcome.

*Richard Lyons* is a consulting systems engineer and lecturer with Besser Associates in Mt. View, California, and the author of *Understanding Digital Signal Processing*, second edition.

*Amy Bell* is an assistant professor in the department of Electrical and Computer Engineering at Virginia Tech.

## References

- [1] T. Laakso et al., "Splitting the unit delay," *IEEE Signal Processing Mag.*, vol. 13, pp. 30–60, Jan. 1996.
- [2] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Processing Mag.*, vol. 20, pp. 74–80, Mar. 2003.
- [3] E. Jacobsen and R. Lyons, "The sliding DFT, An update," *IEEE Signal Processing Mag.*, vol. 21, pp. 110–111, Jan. 2004.
- [4] D. Grover and J. Deller, *Digital Signal Processing and the Microcontroller*. Upper Saddle River, NJ: Prentice-Hall, 1999.
- [5] C. Turner, "Recursive discrete-time sinusoidal oscillators," *IEEE Signal Processing Mag.*, vol. 20, pp. 103–111, May 2003.
- [6] R. Lyons, *Understanding Digital Signal Processing*, 2nd ed. Upper Saddle River, NJ:

Prentice-Hall, 2004.

- [7] Comb Filters. Available: [http://ccrma-www.stanford.edu/~jos/waveguide/Comb\\_Filters.html](http://ccrma-www.stanford.edu/~jos/waveguide/Comb_Filters.html)
- [8] Texas Instruments, "How can comb filters be used to synthesize musical instruments on a TMS320 DSP?," *TMS320 DSP Designers Notebook*, no. 56, 1995.
- [9] V. Ingle and J. Proakis, *Digital Signal Processing Using MATLAB*. Pacific Grove, CA: Brooks/Cole, 2000, pp. 202–208.
- [10] J. Proakis and D. Manolakis, *Digital Signal Processing-Principles, Algorithms, and Applications*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1996, pp. 630–637.
- [11] C. Dick and F. Harris, "FPGA signal processing using sigma-delta modulation," *IEEE Signal Processing Mag.*, vol. 17, pp. 20–35, Jan. 2000.

## lecture notes continued from page 89

## Concluding Remarks

In view of (2) and (7), the waterbed effect result, also called an uncertainty conservation result in [7], appears to be a fundamental property of both nonparametric and parametric spectral estimation methods. Consequently, an even "more intuitive" or "higher-level" derivation of this property than the one presented herein might exist, but it remains to be discovered.

## Acknowledgments

This work was supported in part by the Swedish Science Council (VR) and the National Science Foundation Grant CCR-0104887.

## References

- [1] M.B. Priestley, *Spectral Analysis and Time Series* (Univariate Series, vol. 1). New York, NY: Academic, 1981.
- [2] P. Stoica and R.L. Moses, *Introduction to Spectral Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1997.
- [3] B. Porat, *Digital Processing of Random Signals—Theory and Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

- [4] P. Stoica and T. Sundin, "On nonparametric spectral estimation," *Circ. Syst. Sign. Process.*, vol. 18, pp. 169–181, 1999.
- [5] K. Berk, "Consistent autoregressive spectral estimates," *Ann. Statist.*, vol. 2, pp. 489–502, 1974.
- [6] L. Ljung, "Asymptotic variance expressions for identified black-box transfer function models," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 9, pp. 834–844, 1985.
- [7] B. Ninness, "The asymptotic CRLB for the spectrum of ARMA processes," *IEEE Trans. Signal Processing*, vol. 51, no. 6, pp. 1520–1531, Nov. 2003.
- [8] P. Whittle, "The analysis of multiple stationary time series," *J. Royal Statist. Soc., ser. b*, vol. 15, pp. 125–139, 1953.