The Swiss Army Knife of Digital Networks

by Richard Lyons and Amy Bell

This article describes a general discrete-signal network that appears, in various forms, inside many 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 2nd-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.

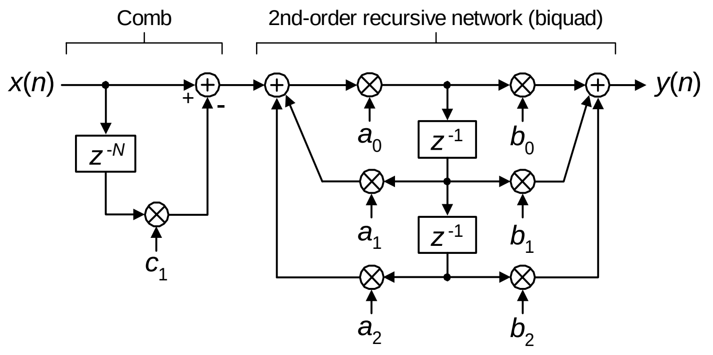


Figure 1. General discrete-signal processing network.

The general network has a transfer function of

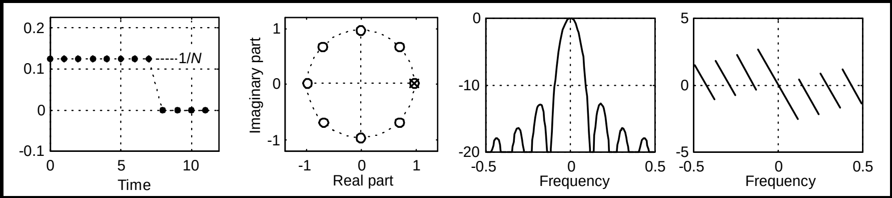
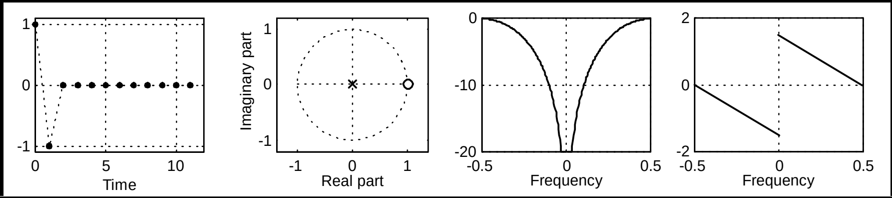
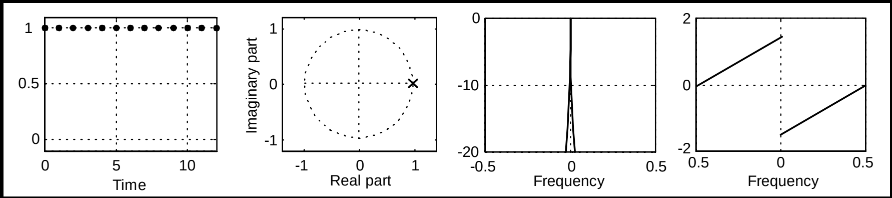
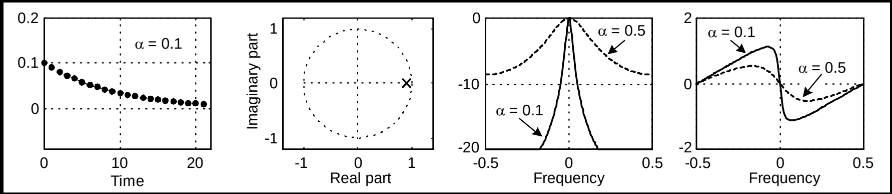
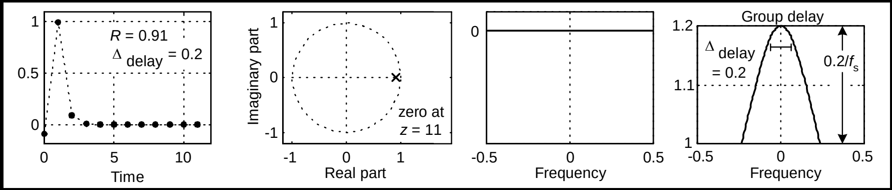
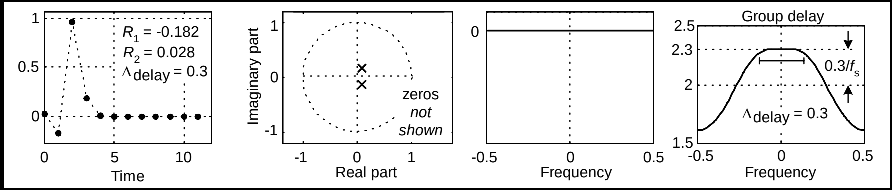
$$H(z) = (1 - c_1 z^{-N}) \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1/a_0 - a_1 z^{-1} - a_2 z^{-2}}. \qquad (1)$$

From here out, we'll use DSP filter lingo and call the 2nd-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. 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 Hz.

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

$H_{ma}(z)$'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 $0 \le k < N$. $H_{ma}(z)$'s denominator places a single pole at $z = 1$ on the unit circle, canceling the zero at that location.

Table 1. General Functions.

Function and coefficients	**Network behavior** impulse response / z-plane / magnitude (dB) / phase (rad.)
Moving Averager $a_0 = 1, a_1 = 1, a_2 = 0,$ $b_0 = 1/N, b_1 = 0, b_2 = 0,$ $c_1 = 1, N = 8$	
Differencer $a_0 = 1, a_1 = 0, a_2 = 0,$ $b_0 = 1, b_1 = -1, b_2 = 0,$ $c_1 = 0$	
Integrator $a_0 = 1, a_1 = 1, a_2 = 0,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_1 = 0$	
Leaky Integrator $a_0 = 1, a_1 = 1\text{-}\alpha, a_2 = 0,$ $b_0 = \alpha, b_1 = 0, b_2 = 0,$ $c_1 = 0, \alpha = 0.1$	
1st-order Delay Network $a_0 = 1, a_1 = -R, a_2 = 0,$ $b_0 = R, b_1 = 1, b_2 = 0,$ $c_1 = 0$	
2nd-order Delay Network $a_0 = 1, a_1 = -R_1, a_2 = -R_2,$ $b_0 = R_2, b_1 = R_1, b_2 = 1,$ $c_1 = 0$	

Differencer: This is a discrete version of a 1st-order differentiator. An ideal differentiator has a frequency magnitude response that's 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 lowpass filter implementations for random noise reduction. It is a 1st-order IIR filter where, for stable lowpass operation, the constant α lies in the range $0<\alpha<1$.

This nonlinear-phase filter has a single pole at $z = 1\text{-}\alpha$ on the z-plane, and a transfer function of $H_{li}(z) = \alpha/[1-(1-\alpha)z^{-1}]$. Small values for α 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.

1st-order Delay Network: A subclass of a 1st-order IIR Filter, the coefficients in Table 1 yield an *allpass* network having a relatively constant group delay at low frequencies. The network's delay is $D_{\text{total}} = 1 + \Delta_{\text{delay}}$ samples where Δ_{delay}, typically in the range of -0.5 to 0.5, is a fraction of the $1/f_s$ sample period. For example, when Δ_{delay} is 0.2, the network delay (at low frequencies) is 1.2 samples. The real-valued R coefficient is

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

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_{total} value, the bar in the group delay plot, ranges roughly from $0.1f_s$ to $0.2f_s$ for 1st-order networks. So if your signal is oversampled, making it low in frequency relative to f_s, this 1st-order allpass 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].

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

$$R_1 = \frac{-2\Delta_{\text{delay}}}{\Delta_{\text{delay}} + 3} \text{ and } R_2 = \frac{(\Delta_{\text{delay}})(\Delta_{\text{delay}}+1)}{(\Delta_{\text{delay}} + 3)(\Delta_{\text{delay}} + 4)}. \qquad (3)$$

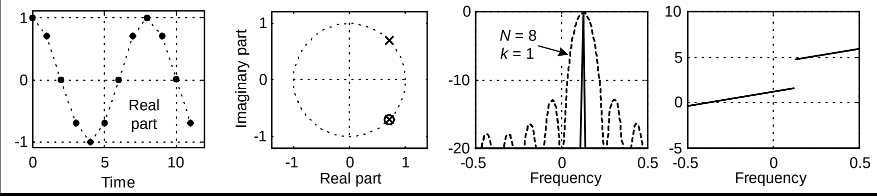
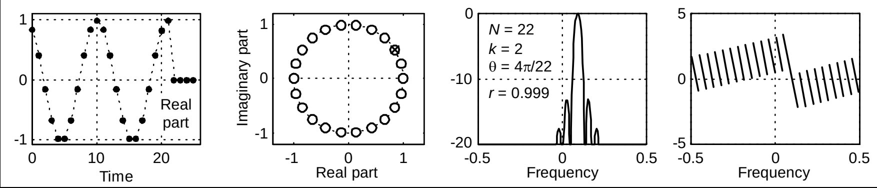
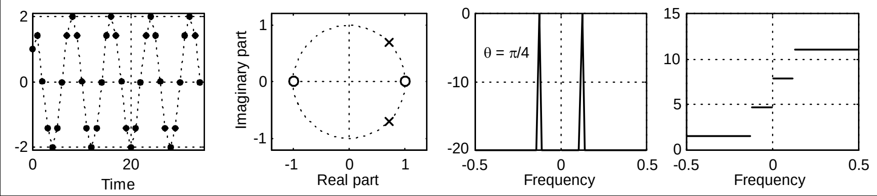
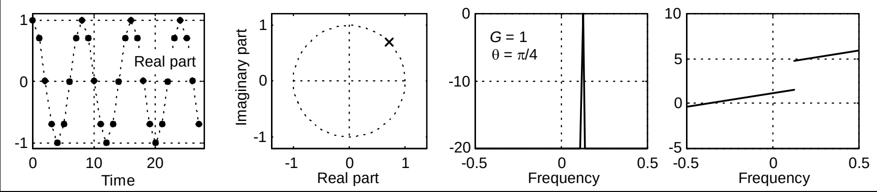
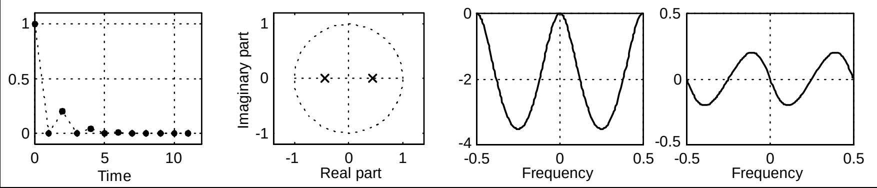
The band, centered at DC, over which the group delay varies no more than $|\Delta_{\text{delay}}|/10$ from the specified D_{total} value, the bar in the group delay plot, ranges roughly from $0.26f_s$ to $0.38f_s$ for this 2nd-order network. Performance for $\Delta_{\text{delay}} = 0.3$ ($R_1 = -0.182$ 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 Δ_{delay} than when Δ_{delay} is positive. This means if you desire, for example, a group delay of $D_{\text{total}} = 2.5$ samples it's better to use an external unit delay and set Δ_{delay} to -0.5 rather than letting Δ_{delay} be 0.5. To ensure stability, Δ_{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 $\theta = 2\pi k/N$ radians on the unit circle, corresponding to a cyclic frequency of kf_s/N Hz. Frequency variable k, in the range $0 \le k < N$, 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 2. Analysis and Synthesis Functions.

Function and coefficients	**Network behavior** impulse response *z*-plane magnitude (dB) phase (rad.)
Goertzel Network $a_0 = 1$, $a_1 = 2\cos(\theta)$, $a_2 = -1$, $b_0 = 1$, $b_1 = -e^{-j\theta}$, $b_2 = 0$, $c_1 = 0$, $\theta = 2\pi k/N$	
Sliding DFT Network $a_0 = re^{j\theta}$, $a_1 = 1$, $a_2 = 0$, $b_0 = 1$, $b_1 = 0$, $b_2 = 0$, $c_1 = r^N$, $\theta = 2\pi k/N$	
Real Oscillator $a_0 = 1$, $a_1 = 2\cos(\theta)$, $a_2 = -1$, $b_0 = 1$, $b_1 = 0$, $b_2 = -1$	
Quadrature Oscillator $a_0 = G(n)$, $a_1 = e^{j\theta}$, $a_2 = 0$, $b_0 = 1$, $b_1 = 0$, $b_2 = 0$	
Audio Comb $a_0 = 1$, $a_1 = 0$, $a_2 = \alpha$, $b_0 = 1$, $b_1 = 0$, $b_2 = 0$, $c_1 = 0$, $\alpha = 0.2$	

Sliding DFT Network: This structure computes a single-bin *N*-point DFT centered at an angle of $\theta = 2\pi k/N$ radians on the unit circle, corresponding to a cyclic frequency of kf_s/N Hz. *N* is the DFT size and integer *k* is $0 \le k < N$. 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 $c_1 = -r^N$ allows the analysis band to be centered at an angle of $\theta = 2\pi(k+1/2)/N$ radians, 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 $\theta = 2\pi f_t/f_s$ radians, where f_t is the oscillator's frequency in Hz. 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 $\theta = 2\pi f_t/f_s$ radians. To start the oscillator, we set the complex $y(n\text{-}1)$ 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 2nd-order (the simplest) version of an infinite impulse response (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 $H_{ac}(z) = 1/(1\text{-}\alpha z^{-2})$ resulting in two poles located at $z = \pm\sqrt{\alpha}$ on the z-plane. To maintain stability the real-valued α must be less than unity, and the closer α 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, $H_{comb}(z) = 1\text{-}z^{-N}$, 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 $0 \le k < N$. Those $z(k)$ values are the N roots of unity when we set (1) 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 mf_s/N where integer m is $0 \le m \le N/2$. The peak gain of this linear-phase filter is 2.

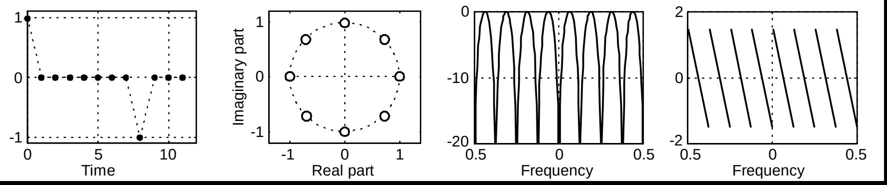
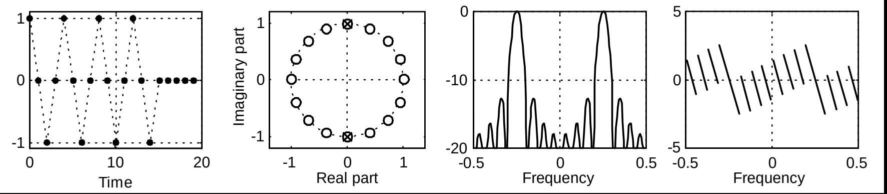
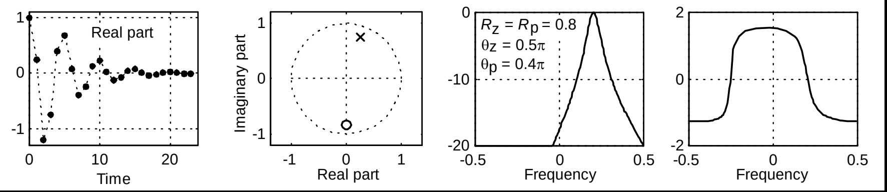
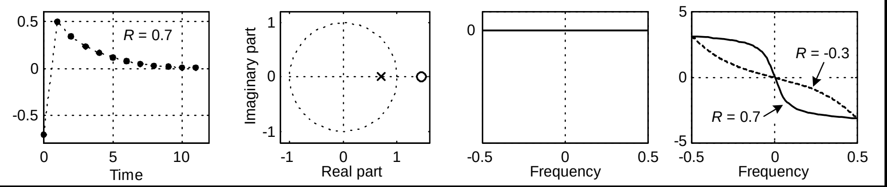
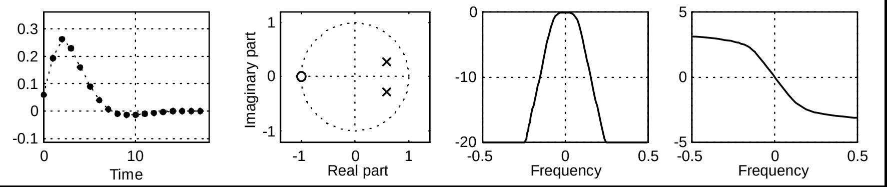
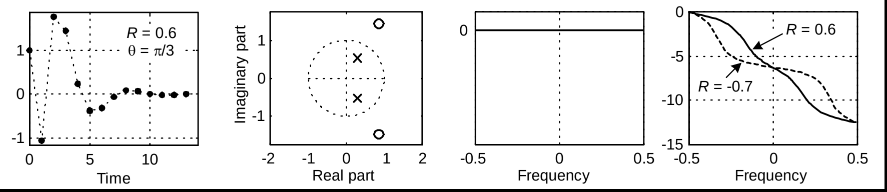
If we set coefficient c_1 to -1 in the comb filter, making its transfer function $H_{alt,comb}(z) = 1+z^{-N}$, we obtain an alternate linear-phase comb filter having zeros rotated counterclockwise around the unit circle by an angle of π/N radians positioning the zeros at angles of $2\pi(k+1/2)/N$ radians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

1st-order IIR Filter: This is the Direct Form II version of a simple 1st-order IIR filter having a single pole located at a radius of R_p from the z-plane's origin at an angle of θ_p radians, 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 lowpass or a highpass frequency response, no bandpass or bandstop filters are possible. The filter's transfer function is $H_{1,iir}(z) = (1+R_z e^{j\theta_z} z^{-1})/(1\text{-}R_p e^{j\theta_p} z^{-1})$.

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 1 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.

Table 3. Filter Functions.

Function and coefficients	**Network behavior** impulse response *z*-plane magnitude (dB) phase (rad.)
Comb Filter $a_0 = 1$, $a_1 = 0$, $a_2 = 0$, $b_0 = 1$, $b_1 = 0$, $b_2 = 0$, $c_1 = 1$, $N = 8$	
Bandpass Filter at $f_s/4$ $a_0 = 1$, $a_1 = 0$, $a_2 = -1$, $b_0 = 1$, $b_1 = 0$, $b_2 = 0$, $c_1 = 1$, $N = 16$	
1st-order IIR Filter $a_0 = 1$, $a_1 = R_p e^{j\theta_p}$, $a_2 = 0$, $b_0 = 1$, $b_1 = R_z e^{j\theta_z}$, $b_2 = 0$, $c_1 = 0$	
1st-order Equalizer $a_0 = 1$, $a_1 = R$, $a_2 = 0$, $b_0 = -R^*$, $b_1 = 1$, $b_2 = 0$, $c_1 = 0$	
2nd-order IIR Filter $a_0 = 1$, $a_1 = 1.194$, $a_2 = -0.436$, $b_0 = b_2 = 0.0605$, $b_1 = 0.121$, $c_1 = 0$	
2nd-order Equalizer $a_0 = 1$, $a_1 = 2R\cos(\theta)$, $a_2 = -R^2$, $b_0 = 1$, $b_1 = -(2/R)\cos(\theta)$, $b_2 = 1/R^2$, $c_1 = 0$	

1st-order Equalizer: This structure has a frequency magnitude response that's 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 $H_{1,\text{eq}}(z) = (-R^*+z^{-1})/(1-Rz^{-1})$.

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 1st-order allpass filters can also be used for interpolation and audio reverberation for low-frequency signals.

2nd-order IIR Filter: This is the Direct Form II version of a 2nd-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.[†] 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 2nd-order IIR structure to ensure filter stability and avoid *limit cycles*. The filters have a transfer function of (1) with the $c_1 = 0$. Lowpass, highpass, bandpass, and bandstop filters are possible. No single example shows all the possibilities of this structure, so Table 3 merely gives a simple lowpass filter example.

If an IIR filter design requires high performance, high Q, it turns out the Direct Form I version of a 2nd-order IIR filter is less susceptible to coefficient quantization and overflow errors than the Direct Form II structure given here.

2nd-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 $\pm\theta$ radians, 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. However, it may take multiple cascaded biquad networks to achieve acceptable equalization.

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 lowpass filter, the filter's output is an interpolated by N version of the original signal. This lowpass filter's transfer function is $H_{\text{cic}}(z) = (1-z^{-K})/(1-z^{-1})$. 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 zero Hz.

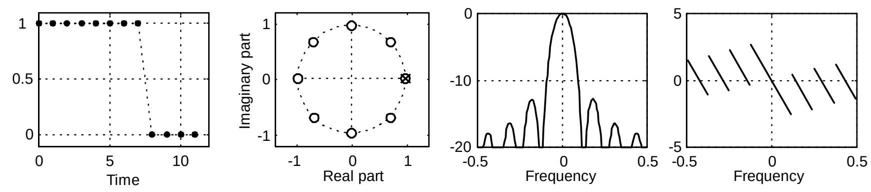
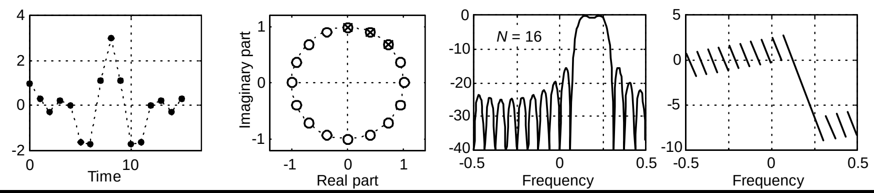
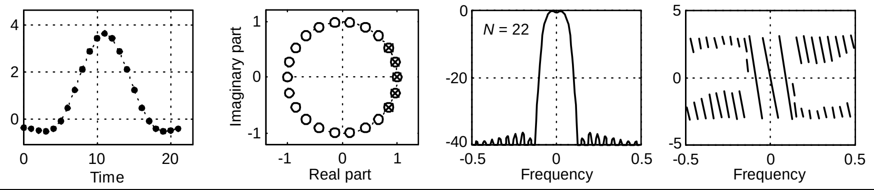
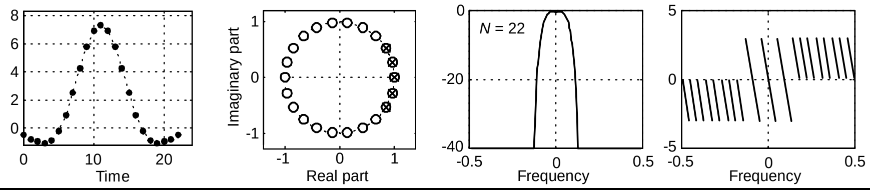
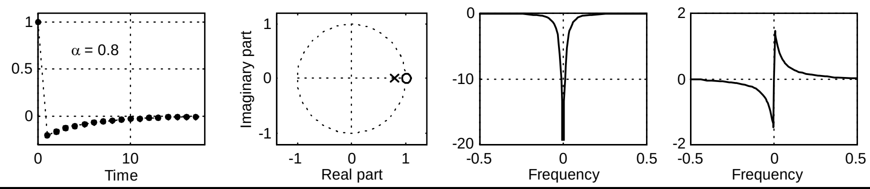
In practice, the upsampling operation (zero stuffing) is performed after the comb filter and before the biquad network. This has the sweet advantage that the comb filter's delay length becomes $N = 1$, reducing the necessary comb delay storage requirement to one. CIC filters are typically used as the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

Complex Frequency Sampling Filter (FSF): This structure is a single section of a complex *frequency sampling filter* having a $\sin(x)/x$-like frequency magnitude response centered at an angle of $\theta_k = 2\pi k/N$ radians on the unit circle, corresponding to a cyclic frequency of kf_s/N Hz. N and k are integers with k is $0 \le k < N$. 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.

[†] There's a terrific piece of MATLAB code (PEZ, created by the talented Dr. Craig Ulmer) allowing us to see the frequency-domain effect of moving multiple poles and zeros, manually using a mouse, around on the z-plane. It's available at http://www.cspl.umd.edu/spm/tips-n-tricks/.

Table 4. Additional Filter Functions.

Function and coefficients	**Network behavior** impulse response *z*-plane magnitude (dB) phase (rad.)
CIC Interpolation Filter $a_0 = 1, a_1 = 1, a_2 = 0,$ $b_0 = 1, b_1 = 0, b_2 = 0,$ $c_1 = 1, N = 8$	
Complex FSF[†] $a_0 = 1, a_1 = e^{j\theta_k}, a_2 = 0,$ $b_0 = (-1)^k, b_1 = 0, b_2 = 0,$ $c_1 = 1, \theta_k = 2\pi k/N$	
Real FSF, Type I $a_0 = 1, a_1 = 2\cos(\theta_k),$ $a_2 = -1, b_0 = \|H_k\|\cos(\phi_k),$ $b_1 = -\|H_k\|\cos(\phi_k - \theta_k),$ $b_2 = 0, c_1 = 1, \theta_k = 2\pi k/N$	
Real FSF, Type IV $a_0 = 1, a_1 = 2\cos(\theta_k),$ $a_2 = -1, b_0 = (-1)^k M_k,$ $b_1 = 0, b_2 = (-1)^k(-M_k),$ $c_1 = 1, \theta_k = 2\pi k/N$	
DC Bias Removal $a_0 = 1, a_1 = \alpha, a_2 = 0,$ $b_0 = 1, b_1 = -1, b_2 = 0,$ $c_1 = 0$	

[†] FSF = Frequency sampling filter

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 responses centered at both $\pm\theta_k = \pm 2\pi k/N$ radians, where *N* is an integer. The larger *N* the more narrow 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 lowpass filters can be built. In this case, complex gain factors H_k are the desired peak frequency response of the *k*th biquad. Parameter ϕ_k is the desired relative phase shift, in radians, of H_k. Table 4 shows the behavior of an $N = 22$, three-biquad, lowpass 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 finite impulse response (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 a multi-biquad lowpass 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 magnitude response of the *k*th biquad. Table 4 shows the behavior of an $N = 22$, three-biquad, lowpass 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 the M_k gain factors, narrowband lowpass 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 having 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 α, where for stable operation α lies in the range $0<\alpha<1$. The closer α is to unity the more narrow the notch at DC. This nonlinear-phase filter has a transfer function of $H_{dc}(z) = (1-z^{-1})/(1-\alpha z^{-1})$. 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].

An alternative to truncation, to avoid overflow, 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 $H_{alt,dc}(z) = (G-Gz^{-1})/(1-\alpha z^{-1})$.

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, CA. He has been the Lead Hardware Engineer for numerous signal processing systems for both the National Security Agency (NSA) and TRW Inc., and has taught at the University of California Santa Cruz Extension. An Associate Editor for the IEEE Signal Processing Magazine, he is also the author of *Understanding Digital Signal Processing 2/E* (Prentice-Hall, 2004). Lyons is a member of the Eta Kappa Nu honor society, and is trying to learn how to strike a cue ball so that it travels where intended on a pool table. He can be contacted at: r.lyons@ieee.org.

Amy Bell is an assistant professor in the department of electrical and computer engineering at Virginia Tech. She received her Ph.D. in electrical engineering from the University of Michigan. Bell conducts research in wavelet image compression, embedded systems, and bioinformatics. She is the recipient of a 1999 NSF CAREER award and a 2002 NSF Information Technology Research award. Bell is an Associate Editor for the IEEE Signal Processing Magazine, and her "best results" to-date include Jacob and Henry: a collaboration with her husband. She can be contacted at: abell@vt.edu.

References

[1] T. Laakso et al., "Splitting the unit delay," *IEEE Signal Proc. Magazine*, pp. 30-60, Jan. 1996.

[2] E. Jacobsen and R. Lyons, "The sliding DFT", *IEEE Signal Proc. Magazine*, *DSP Tips & Tricks Column*, Vol. 20, No. 2, pp. 74-80, Mar. 2003.

[3] E. Jacobsen and R. Lyons, "The sliding DFT, an update", *IEEE Signal Proc. Magazine*, *DSP Tips & Tricks Column*, Vol. 21, No. 1, Jan. 2004.

[4] D. Grover and J. Deller, *Digital Signal Processing and the Microcontroller*, Prentice Hall, Upper Saddle River, New Jersey, 1999.

[5] C. Turner, "Recursive discrete-time sinusoidal oscillators", *IEEE Signal Proc. Magazine*, Vol. 20, No. 3, pp. 103-111, May 2003.

[6] R. Lyons, *Understanding Digital Signal Processing*, 2nd Ed., Prentice Hall, Upper Saddle River, New Jersey, 2004.

[7] 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*, Brookes/Cole Publishing, Pacific Grove, CA, 2000, pp. 202-208.

[10] J. Proakis and D. Manolakis, *Digital Signal Processing-Principles, Algorithms, and Applications*, Third Edition, Prentice Hall, Upper Saddle River, New Jersey, 1996, pp. 630-637.

[11] C. Dick, and F. Harris, "FPGA signal processing using sigma-delta modulation", *IEEE Signal Proc. Magazine*, vol. 17, No. 1, Jan. 2000.