Greg Berchin

Precise Filter Design

You have just been assigned to a new project at work, in which the objective is to replace an existing analog system with a functionally equivalent digital system. Your job is to design a digital filter that matches the magnitude and phase response of the existing system's analog filter over a broad frequency range. You are running out of ideas. The bilinear transform and impulse invariance methods provide poor matches to the analog filter response, particularly at high frequencies. Fast convolution requires more computational resources than you have and creates more input/output latency than you can tolerate. What will you do?

In this article, we describe an obscure but simple and powerful method for designing a digital filter that approximates an arbitrary magnitude and phase response. If applied to the problem above, it can create a filter roughly comparable in computational burden and latency to that created by the bilinear transform method, with fidelity approaching that of fast convolution. In addition, the method described here can also be applied to a wide variety of other system identification tasks.

The filter design method we present is called frequency-domain least-squares (FDLS) [1]–[3]. The FDLS algorithm produces a transfer function that approximates an arbitrary frequency response. The input to the algorithm is a set of magnitude and phase values at a large number (typically thousands) of arbitrary frequencies between 0 Hz and half the sampling rate. The algorithm's output is a set of transfer function coefficients. The FDLS algorithm is quite flexible in that it can create transfer functions containing poles and zeros (infinite response filters), only zeros (finite response filters), or only poles (autoregressive networks). The algorithm uses nothing more esoteric than basic linear algebra. Before we can see how the technique works, we need to review some basic linear algebra and matrix concepts.

BACKGROUND

First let us recall that, in order to uniquely solve a system of equations, we need as many equations as unknowns. For example, the single equation with one unknown $5x = 7$ has the unique solution $x = 7/5$. But the single equation with two unknowns $5x + 2y = 7$ has multiple solutions $x = (7 - 2y)/5$ that depend on the unspecified y-value. If another equation, $-6x + 4y = 9$, is added to the equation above, there are unique solutions for both x and y that can be found algebraically or by matrix inversion (denoted in the following by a "-1" superscript):

$$\begin{bmatrix} 5 & 2 \\ -6 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7 \\ 9 \end{bmatrix}$$

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} 5 & 2 \\ -6 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 7 \\ 9 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{8} & -\frac{1}{16} \\ \frac{3}{16} & \frac{5}{32} \end{bmatrix} \begin{bmatrix} 7 \\ 9 \end{bmatrix} \\ &= \begin{bmatrix} \left(\frac{7}{8} - \frac{9}{16}\right) \\ \left(\frac{21}{16} + \frac{45}{32}\right) \end{bmatrix}. \end{aligned}$$

Let us consider what happens if we add another equation, $x + y = 5$, to the pair that we already have above (we will see later why we might want to do this). There are no values of x and y that satisfy all three equations simultaneously. To address this case, matrix algebra provides the "pseudoinverse," which determines the values of x and y that come, in the least-squares sense, as close as possible to satisfying all three equations. The solution is then given by

$$\begin{bmatrix} 5 & 2 \\ -6 & 4 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7 \\ 9 \\ 5 \end{bmatrix} \quad \text{or}$$

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &\approx \left[\begin{bmatrix} 5 & 2 \\ -6 & 4 \\ 1 & 1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 5 & 2 \\ -6 & 4 \\ 1 & 1 \end{bmatrix} \right]^{-1} \\ &\quad \times \begin{bmatrix} 5 & 2 \\ -6 & 4 \\ 1 & 1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 7 \\ 9 \\ 5 \end{bmatrix} \\ &\approx \begin{bmatrix} 0.3716 \\ 2.8491 \end{bmatrix}, \end{aligned}$$

where T denotes the matrix transpose. Of course, the mathematical derivation of the matrix inverse and pseudoinverse, and the definition of least-squares, can be found in any basic linear algebra text [1]. And while our more mathematically inclined readers will point out that there are better ways than this to compute the pseudoinverse, this method is adequate for our example.

You may also remember that filter specifications are commonly expressed in terms of passband width and flatness, transition band width, and stopband attenuation. There may also be some general specifications about phase response

"DSP Tips and Tricks" introduces practical design and implementation signal processing algorithms that you may wish to incorporate into your designs. We encourage readers to submit their contributions to Associate Editors Rick Lyons (r.lyons@ieee.org) or Britt Rorabaugh (dspboss@aol.com).

1053-5888/07/$25.00©2007IEEE

or time-domain performance, but the exact magnitude and phase responses are usually left to the designer's discretion. However, an important exception occurs when a digital filter is to be used to emulate an analog filter. This is traditionally a very difficult problem, because analog systems are described by Laplace transforms using integration and differentiation, whereas digital systems are described by z-transforms using delay. Since the conversion between them is nonlinear, the response of an analog system can only be approximated by a digital system and vice-versa.

Let us assume that the transfer function of our digital filter (i.e., the mathematical description of the relationship between the filter's input and output) is in a standard textbook form [2] given by

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_D z^{-D}},$$

where $U(z)$ is the z-transform of the input signal, $Y(z)$ is the z-transform of the output signal, and the a and b factors are real-valued coefficients. Furthermore, we assume that the filter is causal, meaning that its response to an input does not begin until after the input is applied. Under these assumptions, the time-domain difference equation that implements our filter is

$$\begin{aligned} y(k) = &- a_1 y(k-1) - \cdots - a_D y(k-D) \\ &+ b_0 u(k) + \cdots + b_N u(k-N), \end{aligned}$$

where the a and b coefficients are exactly the same as in the transfer function above, k is the time index, $u(k)$ and $y(k)$ are the current values of the input and output (respectively), $u(k-N)$ was the input value N samples in the past, and $y(k-D)$ was the output value D samples in the past. We can write the equation above in matrix form as

$$y(k) = [-y(k-1) \ldots - y(k-D) \quad u(k) \ldots u(k-N)] \begin{bmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{bmatrix}.$$

Let us conclude our background section with a comment on what a frequency response value means. In a simple example, if the frequency response of a system at a frequency ω_1 is given in magnitude/phase form as $A_1 \angle \phi_1$, the output amplitude will be A_1 times the input amplitude and the output phase will be shifted an angle ϕ_1 relative to the input phase when a steady-state sine wave of frequency ω_1 is applied to the system. For instance, if the input to the system described above at time k is $u_1(k) = \cos(k\omega_1 t_s)$, where t_s is the sampling period (equal to one over the sampling frequency), then the output will be $y_1(k) = A_1 \cos(k\omega_1 t_s + \phi_1)$. The input and output values at any sample time can be determined in a similar manner. For example, the input sample value N samples in the past was $u_1(k-N) = \cos((k-N)\omega_1 t_s)$ and the output sample value D samples in the past was $y_1(k-D) = A_1 \cos((k-D)\omega_1 t_s + \phi_1)$. For our purposes, since k represents the current sample time, its value can conveniently be set to zero.

FDLS FILTER APPROXIMATION

Based on our review of the pseudoinverse, transfer function, and frequency response, we know that the output is a combination of present and past input and output values, each scaled by a set of b or a coefficients (respectively), the values of which are not yet known. We also know that the relationship between input u and output y at any sample time can be inferred from the frequency response value $A\angle\phi$ at frequency ω. Combining these two ideas, we obtain one equation in $D+N+1$ unknowns

$$y_1(0) = [-y_1(-1) \ldots - y_1(-D) \quad u_1(0) \ldots u_1(-N)] \begin{bmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{bmatrix}.$$

(Note that the current-sample index k has been set to zero.) If we repeat using $A_2\angle\phi_2$ at a different frequency ω_2, we obtain a second equation in $D+N+1$ unknowns as shown in (a) at the bottom of the page. And if we repeat at many more different frequencies M than we have unknowns $D+N+1$, we know from our review of linear algebra that the pseudoinverse will compute values for the set of coefficients $a_1 \ldots a_D$ and $b_0 \ldots b_N$ that come as close as possible to solving all of the equations, which is exactly what we need to design our filter. So now we can write (b), shown at the bottom of the page. We can denote the $y_1(0) \ldots y_M(0)$ column vector above as Y, the matrix as X, and the $a_1 \ldots b_N$ col-

$$\begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} -y_1(-1) & \ldots & -y_1(-D) & u_1(0) & \ldots & u_1(-N) \\ -y_2(-1) & \ldots & -y_2(-D) & u_2(0) & \ldots & u_2(-N) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{bmatrix}. \qquad \text{(a)}$$

$$\begin{bmatrix} y_1(0) \\ y_2(0) \\ \vdots \\ y_M(0) \end{bmatrix} = \begin{bmatrix} -y_1(-1) & \ldots & -y_1(-D) & u_1(0) & \ldots & u_1(-N) \\ -y_2(-1) & \ldots & -y_2(-D) & u_2(0) & \ldots & u_2(-N) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_M(-1) & \ldots & -y_M(-D) & u_M(0) & \ldots & u_M(-N) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{bmatrix}. \qquad \text{(b)}$$

umn vector as Θ. With these notations, $Y = X\Theta$, and the pseudoinverse solves for the vector Θ that contains the desired filter coefficients

$$(X^T X)^{-1} X^T Y \approx \Theta.$$

We can now summarize our filter design trick as follows:

1) Select the numerator order N and the denominator order D, where N and D do not have to be equal and either one (but not both) may be zero. (We have found no "rule of thumb" for defining N and D; they are best determined experimentally.)

2) Define the M separate input u_m cosine sequences, each of length $(N+1)$.

3) Compute the M separate output y_m cosine sequences, each of length D (based on $A_m \angle \phi_m$).

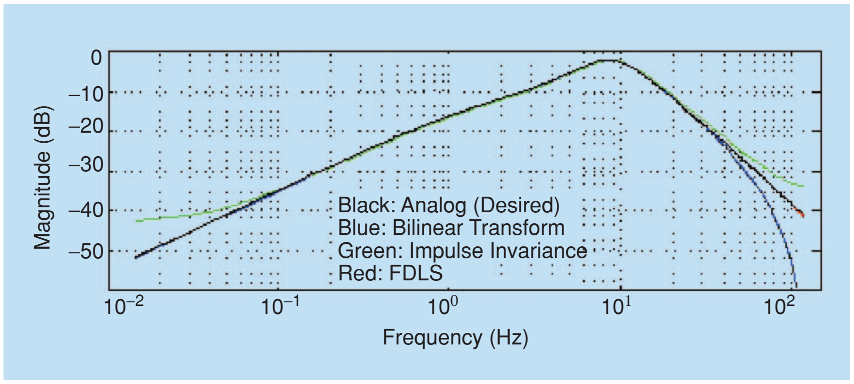
4) Fill the X matrix with the input u_m and output y_m cosine sequences.

5) Fill the Y vector with the M output cosine values, $y_m(0) = A_m \cos(\phi_m)$.

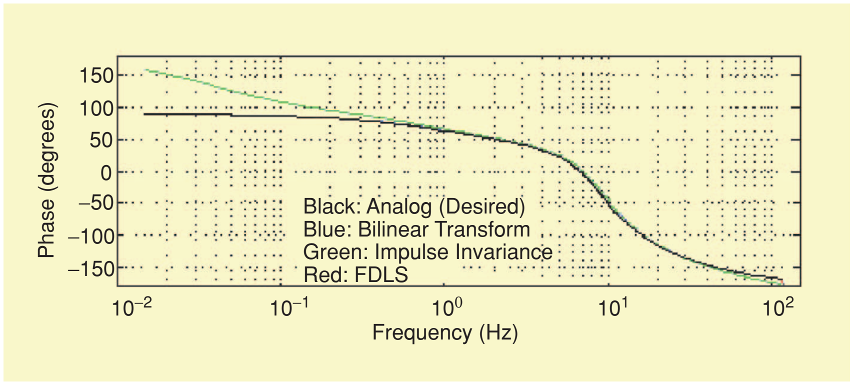
6) Compute the pseudoinverse; the resulting vector Θ contains the filter coefficients.

A numerical example is shown in Figures 1 and 2, which illustrate the magnitude and phase, respectively, of a real-world example analog system (black) and of the associated bilinear transform (blue), impulse invariance (green), and FDLS (red) approximations. The sampling rate is equal to 240 Hz and $D = N = 12$. The red FDLS graphs are almost completely obscured by the black analog system graphs. In this example, the FDLS errors are often three to four orders of magnitude smaller than those of the other methods. (In Figure 2, the bilinear transform curve is obscured by the FDLS and analog curves at low frequencies and by the impulse invariance curve at high frequencies.)

In terms of the computational complexity of an FDIS-designed filter, the number of feedback and feed-forward coefficients is determined by the variables D and N, respectively. As such, an FDIS-designed filter requires $(N+D+1)$ multiplies and $(N+D)$ additions per filter output sample.



[FIG1] **Magnitude responses of the filter designed using the bilinear transform, impulse invariance, and FDLS methods.**



[FIG2] **Phase responses of the filter designed using the bilinear transform, impulse invariance, and FDLS methods.**

CONCLUSION

FDLS is a powerful method for designing digital filters. As is the case with all approximation techniques, there are circumstances in which the FDLS method works well and others in which it does not. The FDLS method does not replace other filter design methods; it provides one more method from which to choose. FDLS is most useful in cases where a specified frequency response must be duplicated to within tight tolerances over a wide frequency range or when the frequency response of an existing system is known but the coefficients of the system's transfer function are unknown. It is up to the designer to determine whether to use it in any given situation. Detailed examples and a MATLAB code implementation of the FDLS algorithm are available at http://apollo.ee.columbia.edu/spm/?i=external/tipsandtricks.

ACKNOWLEDGMENTS

My thanks to Jaime Andrés Aranguren Cardona for providing the example shown in Figures 1 and 2, originally posted on the comp.dsp Newsgroup on 9 January 2005, which ultimately led to this article.

AUTHOR

Greg Berchin (berchin@ieee.org) is a signal processing algorithm engineer who provides contract engineering services from Naperville, Illinois.

REFERENCES

[1] G. Strang, *Linear Algebra and Its Applications*, 2nd ed., Orlando, FL: Academic, pp. 103–152, 1980.

[2] R. Lyons, *Understanding Digital Signal Processing*, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, pp. 232–240, 2004.

[3] G. Berchin, "A new algorithm for system identification from frequency response information," master's thesis, University of California-Davis, 1988 .

[4] G. Berchin and M.A. Soderstrand, "A transform-domain least-squares beamforming technique," in *Proc. IEEE Oceans '90 Conf.*, Arlington, VA, Sept. 1990.