Chapter 7

Precise Filter Design

Greg Berchin
Consultant in Signal Processing

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?

This chapter describes 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 the bilinear transform method, with fidelity approaching that of fast convolution. In addition, the method presented here can be also applied to a wide variety of other system identification tasks.

7.1 PROBLEM BACKGROUND

Filter specifications are commonly expressed in terms of passband width and flatness, transition bandwidth, and stopband attenuation. There may also be some general specifications about phase response or time-domain performance, but the exact magnitude and phase responses are usually left to the designer's discretion.

An important exception occurs, however, 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,

Streamlining Digital Signal Processing: A Tricks of the Trade Guidebook, Edited by Richard G. Lyons

whereas digital systems are described by *Z*-transforms, using delay. Since the conversion between them is nonlinear, the response of an analog system can be only approximated by a digital system, and vice-versa.

7.2 TEXTBOOK APPROXIMATIONS

A common method used to create digital filters from analog prototypes is the *bilinear transform*. This technique can be very effective when specifications are given as passband, transition band, and stopband parameters, as described earlier. And implementation can be very efficient, because the number of coefficients in the digital filter is comparable to the number in its analog prototype. But the bilinear transform suffers from two problems that make a close match to an analog frequency response impossible:

- It squeezes the entire frequency range from zero to infinity in the analog system into the range from DC to half the sampling frequency, inducing *frequency warping*, in the digital system.
- It can match a prototype frequency response at only three frequencies. At all other frequencies the response falls where it may, though its behavior is predictable.

Another design method, called *impulse invariance*, matches the impulse response of the digital filter to the sampled impulse response of the prototype analog filter. Since there is a one-to-one mapping between impulse response and frequency response in the continuous-time case, one might assume that matching the digital impulse response to the analog impulse response will cause the digital frequency response to match the analog frequency response. Unfortunately, aliasing causes large frequency response errors unless the analog filter rolls off steeply at high frequencies.

A popular method for realizing arbitrary-response filters, called *fast convolution*, is implemented in the frequency domain by multiplying the FFT of the signal by samples of the analog filter's frequency response, computing the inverse FFT, and so on. While it can be very effective, it is computationally intensive and suffers from high input-output latency.

7.3 AN APPROXIMATION THAT YOU WON'T FIND IN ANY TEXTBOOK

The filter approximation method we present here is called *frequency-domain least-squares* (FDLS). I developed FDLS while I was a graduate student [1] and described it in some conference papers [2], [3], but the technique was all but forgotten after I left school. 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 zero Hz and half the sampling rate. The algorithm's output is a set of transfer

function coefficients. The technique is quite flexible in that it can create transfer functions containing poles and zeros (IIR), only zeros (FIR), or only poles (autoregressive). Before we can see how the technique works, we need to review some linear algebra and matrix concepts. The FDLS algorithm uses nothing more esoteric than basic linear algebra.

7.4 LEAST SQUARES SOLUTION

Hopefully the reader remembers 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 for x that depend on the unspecified value of y: $x=(7-2y)/5$.

If another equation is added, such as

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

then 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):

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

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

Now let us consider what happens if we add another equation to the pair that we already have (we will see later why we might want to do this), such as

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

There are no values of x and y that satisfy all three equations simultaneously. Well, matrix algebra provides something called the *pseudoinverse* to deal with this situation. It determines the values of x and y that come *as close as possible*, in the least-squares sense, to satisfying all three equations.

Without going into the derivation of the pseudoinverse or the definition of least-squares, let us simply jump to the solution to this new problem:

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

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

(The "T" superscript denotes the matrix transpose.) The pseudoinverse always computes the set of values that comes as close as possible to solving *all* of the equations, when there are more equations than unknowns.

As promised, everything that we have discussed is plain-vanilla linear algebra. The mathematical derivation of the matrix inverse and pseudoinverse, and the definition of least-squares, can be found in any basic linear algebra text. And our more mathematically inclined readers will point out that there are better ways than this to compute the pseudoinverse, but this method is adequate for our example.

Now that we remember how to solve simultaneous equations, we need to figure out how to get the equations in the first place. To do that, we first need to review a little DSP.

We will start with the transfer function, which is a mathematical description of the relationship between a system's input and its output. We will assume that the transfer function of our digital filter is in a standard textbook form:

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

where $U(z)$ is the z-transform of the input signal and $Y(z)$ is the z-transform of the output signal. Furthermore, we assume that the filter is causal, meaning that its response to an input does not begin until after the input is applied. (The filter cannot see into the future.) Under these circumstances the time-domain difference equation that implements our filter is:

$$y(k) = -a_1 y(k-1) - \ldots - a_D y(k-D) + b_0 u(k) + \ldots + b_N u(k-N)$$

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

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

We conclude our review of DSP with a consideration of exactly what a frequency response value means. If, for example, the frequency response of a system

at a frequency ω_1 is given in magnitude/phase form to be $A_1\angle\phi_1$, it means that 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.

Let's look at an example. 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

$$y_1(k-D) = A_1 \cos((k-D)\omega_1 t_s + \phi_1).$$

That's all there is to it. For our purposes, since k represents the current sample time its value can be conveniently set to zero.

That is the end of our review of frequency response, transfer function, and pseudoinverse. Now we will put it all together into a filter design technique. We have just demonstrated 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. We know from our discussion of transfer function 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. Combining the two, our frequency response value $A_1\angle\phi_1$ at ω_1, in the example above, provides us with one equation in $D+N+1$ unknowns:

$$y_1(0) =$$

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

(Note that k, the current-sample index, has been set to zero.) If we do exactly the same thing again, except this time using frequency response $A_2\angle\phi_2$ at a *different* frequency ω_2, we obtain a second equation in $D+N+1$ unknowns:

$$\begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} =$$

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

And if we keep doing this 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:

$$\begin{bmatrix} y_1(0) \\ y_2(0) \\ \vdots \\ y_M(0) \end{bmatrix} =$$

$$\begin{bmatrix} -y_1(-1) & \cdots & -y_1(-D) & u_1(0) & \cdots & u_1(-N) \\ -y_2(-1) & \cdots & -y_2(-D) & u_2(0) & \cdots & u_2(-N) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_M(-1) & \dots & -y_M(-D) & u_M(0) & \dots & u_M(-N) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{bmatrix}.$$

We can employ shortcut matrix notation by defining the following vectors and matrix:

$$Y = \begin{bmatrix} y_1(0) \\ y_2(0) \\ \vdots \\ y_M(0) \end{bmatrix}, \Theta = \begin{bmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{bmatrix}, \quad \text{and}$$

$$X = \begin{bmatrix} -y_1(-1) & \dots & -y_1(-D) & u_1(0) & \dots & u_1(-N) \\ -y_2(-1) & \dots & -y_2(-D) & u_2(0) & \dots & u_2(-N) \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_M(-1) & \dots & -y_M(-D) & u_M(0) & \dots & u_M(-N) \end{bmatrix};$$

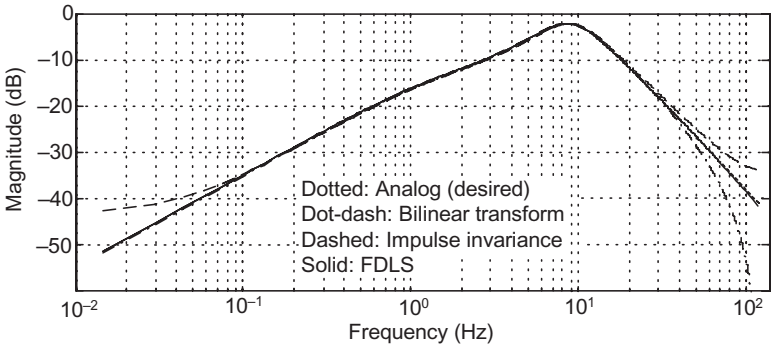

Figure 7-1 Magnitude responses.

then $Y = X\Theta$, and the pseudoinverse is $(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y \approx \Theta$, where vector Θ contains the desired filter coefficients.

That's it. That is the entire algorithm. We can now describe our filter design trick as follows:

1. Choose 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.
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, and the Θ vector contains the filter coefficients.

Figures 7-1 and 7-2 show the magnitude and phase, respectively, of a real-world example analog system (dotted), and of the associated bilinear transform (dot-dash), impulse invariance (dashed), and FDLS (solid) approximations. The sampling rate is 240 Hz, and $D = N = 12$. The dotted analog system curves are almost completely obscured by the solid FDLS curves. In this example, the FDLS errors are often 3 to 4 orders of magnitude smaller than those of the other methods. (In Figure 7-2, the bilinear transform curve is obscured by the FDLS curve at low frequencies and by the impulse invariance curve at high frequencies.)

7.5 IMPLEMENTATION NOTES

- I have found no rule of thumb for defining N and D. They are best determined experimentally.

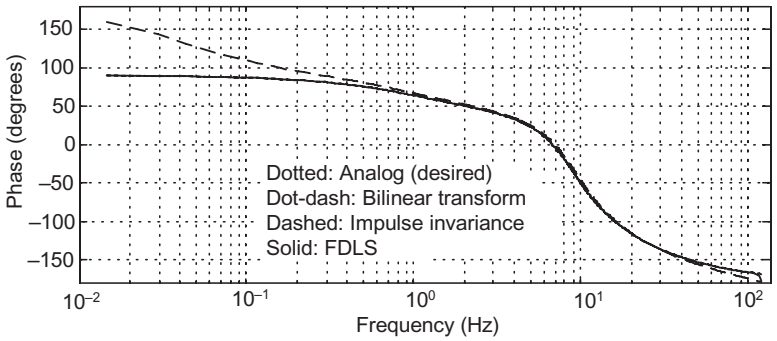

Figure 7-2 Phase responses.

- The selection of a cosine input, rather than sine, was *not* arbitrary. A sine formulation suffers from *zero-crossing* problems at frequencies near half the sampling frequency [1].
- For similar reasons, the algorithm can have some difficulties modeling a system whose phase response approaches odd multiples of 90° near half the sampling frequency. Adding a few samples of artificial delay to the frequency response data, in the form of a linear phase shift, solves this problem. "δ" samples of delay result in a phase shift of "$-\delta\omega_m t_s$," so each equation for $y_m(k)$,

$$y_m(k) = A_m \cos(k\omega_m t_s + \phi_m),$$

  becomes:

$$y_m(k) = A_m \cos(k\omega_m t_s + \phi_m - \delta\omega_m t_s).$$

- The algorithm can be used to design 1-, 2-, or even 3-dimensional beamformers [2]. (*Hint:* Making the delay value p, in equations of the general form $u(k-p) = \cos((k-p)\omega t_s)$, an integer in the range (0, 1, . . . , N) is overly restrictive. Variable p does not have to be an integer and there are some very interesting designs that can be achieved if the integer restriction upon p is removed.)
- A complex-number form of the algorithm exists, in which the inputs and outputs are complex sinusoids $e^{jk\omega t_s}$, the filter coefficients can be complex, and the frequency response can be asymmetrical about 0 Hz (or the beam pattern can be asymmetrical about array-normal).
- A *total-least-squares* formulation exists that concentrates all estimator errors into a single diagonal submatrix for convenient analysis [3].

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

7.6 CONCLUSIONS

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. It does not replace other filter design methods; it provides one more method from which to choose. It is up to the designer to determine whether to use it in any given situation.

7.7 REFERENCES

[1] G. Berchin, "A New Algorithm for System Identification from Frequency Response Information," Master's Thesis, University of California—Davis, 1988.

[2] G. Berchin and M. Soderstrand, "A Transform-Domain Least-Squares Beamforming Technique," *Proceedings of the IEEE Oceans '90 Conference*, Arlington VA, September 1990.

[3] G. Berchin and M. Soderstrand, "A Total Least Squares Approach to Frequency Domain System Identification," *Proceedings of the 32nd Midwest Symposium on Circuits and Systems*, Urbana, IL, August 1989.

EDITOR COMMENTS

Here we present additional examples to illustrate the use of the FDLS algorithm.

Algebraic Example

Recall the FDLS matrix expression

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

which we wrote as $Y = X\Theta$.

Each individual element in the Y column vector is of the form $A_m\cos(\phi_m)$, and each element in the X matrix is of the form $A_1\cos(k\omega_1 t_s + \phi_1)$ or $\cos(k\omega_1 t_s)$. Because all of these elements are of the form of a product (Amplitude)[cos(angle)], each element in Y and X is equal to a constant.

Now if, say, $D = 10$ and $N = 9$, then

$$y_1(0) = A_1\cos(\phi_1)$$

is the first element of the Y column vector and

$$[-y_1(-1) \dots -y_1(-10)\, u_1(0) \dots u_1(-9)]$$

is the top row of the X matrix expression where

$$-y_1(-1) = -A_1\cos[(-1)\omega_1 t_s + \phi_1] = -A_1\cos(-\omega_1 t_s + \phi_1)$$
$$-y_2(-2) = -A_1\cos[(-2)\omega_1 t_s + \phi_1] = -A_1\cos(-2\omega_1 t_s + \phi_1)$$
$$\ldots$$
$$-y_1(-10) = -A_1\cos[(-10)\omega_1 t_s + \phi_1] = -A_1\cos(-10\omega_1 t_s + \phi_1)$$

and

$$u_1(0) = \cos[(0)\omega_1 t_s] = 1$$
$$u_1(-1) = \cos[(-1)\omega_1 t_s] = \cos(-\omega_1 t_s)$$
$$u_1(-2) = \cos[(-2)\omega_1 t_s] = \cos(-2\omega_1 t_s)$$
$$\ldots$$
$$u_1(-9) = \cos[(-9)\omega_1 t_s] = \cos(-9\omega_1 t_s).$$

So the top row of the X matrix looks like:

$$[-A_1\cos(-\omega_1 t_s + \phi_1) \quad -A_1\cos(-2\omega_1 t_s + \phi_1)\ldots -A_1\cos(-10\omega_1 t_s + \phi_1) \quad 1$$
$$\cos(-\omega_1 t_s) \quad \cos(-2\omega_1 t_s)\ldots\cos(-9\omega_1 t_s)].$$

The second row of the X matrix looks like:

$$[-A_2\cos(-\omega_2 t_s + \phi_2) \quad -A_2\cos(-2\omega_2 t_s + \phi_2)\ldots -A_2\cos(-10\omega_2 t_s + \phi_2) \quad 1$$
$$\cos(-\omega_2 t_s) \quad \cos(-2\omega_2 t_s)\ldots\cos(-9\omega_2 t_s)],$$

and so on.

Numerical Example

Here is an example of the above expressions using actual numbers. Suppose we need to approximate the transfer function coefficients for the system whose frequency magnitude and phase response are as shown in Figure 7-3. Assume that our discrete-system sample rate is 1000 Hz, thus $t_s = 10^{-3}$ seconds, and $N = D = 2$. (The $N = D = 2$ values mean that our final filter will be a second-order filter.) Also assume $M = 8$ and we have the eight A_1-to-A_8 magnitude sample values and the eight ϕ_1-to-ϕ_8 phase samples, shown as dots in Figure 7-3, available to us as input values to the FDLS algorithm.

In matrix form, the target analog system parameters are

$$f_m = \begin{bmatrix} 0.0 \\ 19.6850 \\ 35.4331 \\ 51.1811 \\ 59.0551 \\ 66.9291 \\ 106.299 \\ 389.764 \end{bmatrix} \quad \omega_m t_s = \begin{bmatrix} 0.0 \\ 0.1237 \\ 0.2226 \\ 0.3216 \\ 0.3711 \\ 0.4205 \\ 0.6679 \\ 2.449 \end{bmatrix} \quad A_m = \begin{bmatrix} 0.2172 \\ 0.2065 \\ 0.1696 \\ 0.0164 \\ 1.3959 \\ 0.6734 \\ 0.3490 \\ 0.3095 \end{bmatrix} \quad \phi_m = \begin{bmatrix} 0.0 \\ -0.0156 \\ -0.0383 \\ 3.0125 \\ 2.3087 \\ 0.955 \\ 0.0343 \\ 0.0031 \end{bmatrix}$$

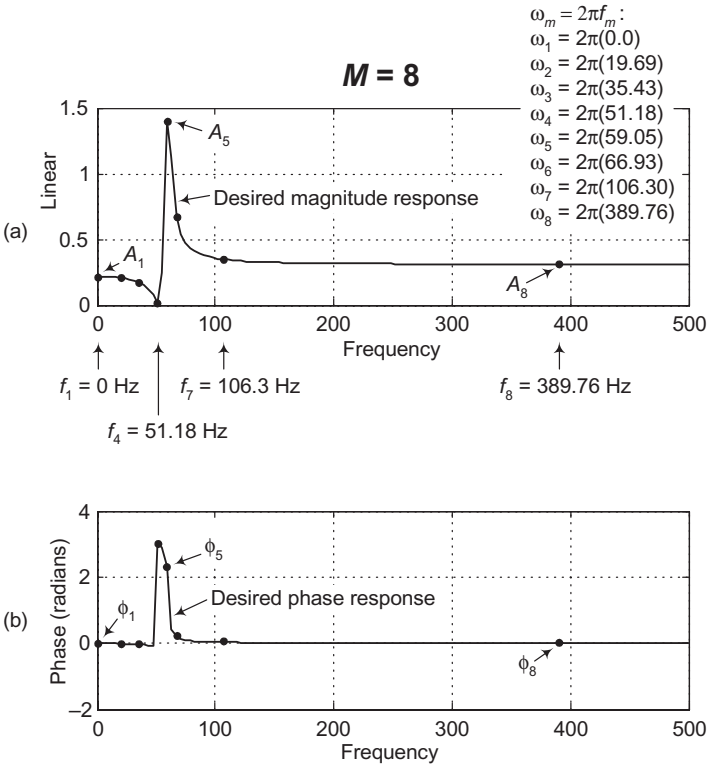

Figure 7-3 Desired magnitude and phase responses.

where the f_m vector is in Hz, the $\omega_m t_s$ vector is in radians, and $1 \le m \le 8$. The first two elements of the Y vector are:

$$y_1(0) = A_1 \cos(\phi_1) = 0.2172 \cos(0) = 0.2172,$$
$$y_2(0) = A_2 \cos(\phi_2) = 0.2065 \cos(-0.0156) = 0.2065.$$

The complete Y vector is:

$$Y = \begin{bmatrix} A_1 \cos(\phi_1) \\ A_2 \cos(\phi_2) \\ A_3 \cos(\phi_3) \\ A_4 \cos(\phi_4) \\ A_5 \cos(\phi_5) \\ A_6 \cos(\phi_6) \\ A_7 \cos(\phi_7) \\ A_8 \cos(\phi_8) \end{bmatrix} = \begin{bmatrix} 0.2172 \\ 0.2065 \\ 0.1695 \\ -0.0162 \\ -0.9390 \\ 0.6605 \\ 0.3488 \\ 0.3095 \end{bmatrix}.$$

The two elements of the "y_1" part of the first row of the X vector are:

$$-y_1(-1) = -A_1\cos(-\omega_1 t_s + \phi_1) = -0.2172\cos(-0+0) = -0.2172,$$
$$-y_1(-2) = -A_1\cos(-2\omega_1 t_s + \phi_1) = -0.2172\cos(-0+0) = -0.2172.$$

The two elements of the "y_8" part of the eighth row of the X vector are:

$$-y_8(-1) = -A_8\cos(-\omega_8 t_s + \phi_8) = -0.3095\cos(-2.449+0.0031) = 0.2376,$$
$$-y_8(-2) = -A_8\cos(-2\omega_8 t_s + \phi_8) = -0.3095\cos(-4.898+0.0031) = -0.562.$$

The three elements of the "u_1" part of the first row of the X matrix are:

$$u_1(0) = \cos(0) = 1$$
$$u_1(-1) = \cos(-\omega_1 t_s) = \cos(-0) = 1$$
$$u_1(-2) = \cos(-2\omega_1 t_s) = \cos(-0) = 1.$$

The three elements of the "u_8" part of the eighth row of the X matrix are:

$$u_8(0) = \cos(0) = 1$$
$$u_8(-1) = \cos(-\omega_8 t_s) = \cos(-2.449) = -0.7696$$
$$u_8(-2) = \cos(-2\omega_8 t_s) = \cos(-4.898) = 0.1845.$$

The complete X matrix is:

$$X = \begin{bmatrix} -A_1\cos(-1\alpha_1+\phi_1) & -A_1\cos(-2\alpha_1+\phi_1) & \cos(0) & \cos(-1\alpha_1) & \cos(-2\alpha_1) \\ -A_2\cos(-1\alpha_2+\phi_2) & -A_2\cos(-2\alpha_2+\phi_2) & \cos(0) & \cos(-1\alpha_2) & \cos(-2\alpha_2) \\ -A_3\cos(-1\alpha_3+\phi_3) & -A_3\cos(-2\alpha_3+\phi_3) & \cos(0) & \cos(-1\alpha_3) & \cos(-2\alpha_3) \\ -A_4\cos(-1\alpha_4+\phi_4) & -A_4\cos(-2\alpha_4+\phi_4) & \cos(0) & \cos(-1\alpha_4) & \cos(-2\alpha_4) \\ -A_5\cos(-1\alpha_5+\phi_5) & -A_5\cos(-2\alpha_5+\phi_5) & \cos(0) & \cos(-1\alpha_5) & \cos(-2\alpha_5) \\ -A_6\cos(-1\alpha_6+\phi_6) & -A_6\cos(-2\alpha_6+\phi_6) & \cos(0) & \cos(-1\alpha_6) & \cos(-3\alpha_6) \\ -A_7\cos(-1\alpha_7+\phi_7) & -A_7\cos(-2\alpha_7+\phi_7) & \cos(0) & \cos(-1\alpha_7) & \cos(-3\alpha_7) \\ -A_8\cos(-1\alpha_8+\phi_8) & -A_8\cos(-2\alpha_8+\phi_8) & \cos(0) & \cos(-1\alpha_8) & \cos(-3\alpha_8) \end{bmatrix}$$

$$= \begin{bmatrix} -0.2172 & -0.2172 & 1.0 & 1.0 & 1.0 \\ -0.2045 & -0.994 & 1.0 & 0.9924 & 0.9696 \\ -0.1639 & -0.1502 & 1.0 & 0.9753 & 0.9025 \\ 0.0147 & 0.0117 & 1.0 & 0.9487 & 0.8002 \\ 0.5007 & -0.0059 & 1.0 & 0.939 & 0.7370 \\ -0.6564 & -0.5378 & 1.0 & 0.9129 & 0.6667 \\ -0.2812 & -0.0928 & 1.0 & 0.7851 & 0.2328 \\ 0.2376 & -0.0562 & 1.0 & -0.7696 & 0.1845 \end{bmatrix}$$

where $\alpha_1 = \omega_1 t_s$, $\alpha_2 = \omega_2 t_s$, . . . $\alpha_8 = \omega_8 t_s$. Given the above Y vector and the X matrix, the FDLS algorithm computes the second-order ($N=D=2$) transfer function coefficients vector $\theta_{M=8}$ as

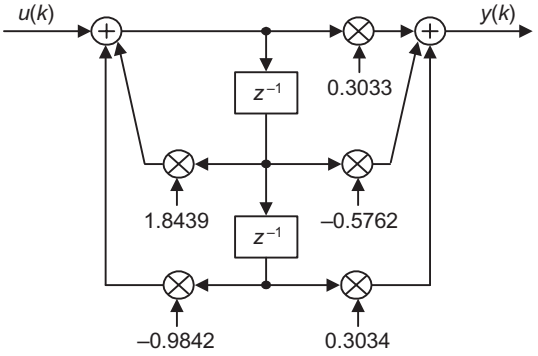

Figure 7-4 Second-order filter.

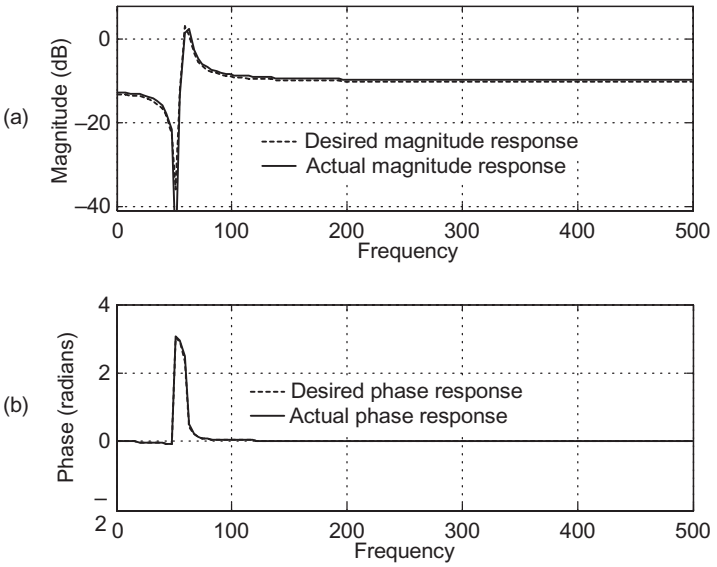

Figure 7-5 Actual versus desired filter magnitude and phase responses.

$$\theta_{M=8} = \begin{bmatrix} -1.8439 \\ 0.9842 \\ 0.3033 \\ -0.5762 \\ 0.3034 \end{bmatrix}.$$

Treated as filter coefficients, we can write vector $\theta_{M=8}$ as

$$a_0 = 1$$
$$a_1 = -1.8439$$

$$a_2 = 0.9842$$
$$b_0 = 0.3033$$
$$b_1 = -0.5762$$
$$b_2 = 0.3034$$

implemented as the recursive filter network shown in Figure 7-4.

The frequency-domain performance of the filter is in the solid curves shown in Figure 7-5. There we see that the $\theta_{M=8}$ coefficients provide an accurate approximation to the desired frequency response in Figure 7-3.