Chapter 10

FIR Filter Fundamentals

Digital filters are usually classified by the duration of their impulse response, which can be either finite or infinite. The methods for designing and implementing these two filter classes differ considerably. *Finite impulse response* (FIR) filters are digital filters whose response to a unit impulse (unit sample function) is finite in duration. This is in contrast to *infinite impulse response* (IIR) filters whose response to a unit impulse (unit sample function) is infinite in duration. FIR and IIR filters each have advantages and disadvantages, and neither is best in all situations. FIR filters can be implemented using either recursive or nonrecursive techniques, but usually nonrecursive techniques are used.

10.1 Introduction to FIR Filters

The general form for a linear time-invariant FIR system's output $y[k]$ at time k is given by

$$y[k] = \sum_{n=0}^{N-1} h[n]\, x[k-n] \tag{10.1}$$

where $h[n]$ is the system's inpulse response. As Eq. (10.1) indicates, the output is a linear combination of the present input and the N previous inputs. The remainder of this chapter is devoted to basic properties and realization issues for FIR filters. Specific design approaches for selecting the coefficients b_n are covered in Chaps. 11, 12, and 13.

FIR advantages

FIR filters have the following advantages:

- FIR filters can easily be designed to have constant phase delay and/or constant group delay.

- FIR filters implemented with nonrecursive techniques will always be stable and free from the limit-cycle oscillations that can plague IIR designs.
- Round-off noise (which is due to finite precision arithmetic performed in the digital processor) can be made relatively small for nonrecursive implementations.
- FIR filters can also be implemented using recursive techniques if this is desired.

FIR disadvantages

Despite their advantages, FIR filters still exhibit some significant disadvantages:

- An FIR filter's impulse response duration, although finite, may have to be very long to obtain sharp cutoff characteristics.
- The design of FIR filters to meet specific performance objectives is generally more difficult than the design of IIR filters for similar applications.

10.2 Evaluating the Frequency Response of FIR Filters

A digital filter's impulse response $h[n]$ is related to the frequency response $H(e^{j\lambda})$ via the DTFT:

$$H(e^{j\lambda}) = \sum_{n=-\infty}^{\infty} h[n]\, e^{-jn\lambda} \tag{10.2}$$

For an FIR filter, $h[n]$ is nonzero only for $0 \le n < N$. Therefore, the limits of the summation can be changed to yield

$$H(e^{j\lambda}) = \sum_{n=0}^{N-1} h[n]\, e^{-jn\lambda} \tag{10.3}$$

Equation (10.3) can be evaluated directly at any desired value of λ.

We now take note of the fact that $\lambda = \omega T$ and that the value of continuous-radian frequency ω_m corresponding to the discrete-frequency index m is given by

$$\omega_m = 2\pi m F \tag{10.4}$$

Substituting $2\pi m F T$ for λ, and $H[m]$ for $H(e^{j\lambda})$ in (10.2) yields the discrete Fourier transform:

$$H[m] = \sum_{n=0}^{N-1} h[n] \exp(-2\pi j n m F T) \tag{10.5}$$

Thus, the DTFT can be evaluated at a set of discrete frequencies $\omega = \omega_m$, $0 \le m < N$, by using the DFT, which in turn may be evaluated in a computationally efficient fashion using one of the various FFT algorithms.

10.3 Linear Phase FIR Filters

As discussed in Sec. 2.8, constant group delay is a desirable property for filters to have since nonconstant group delay will cause envelope distortion in modulated-carrier signals and pulse-shape distortion in base-band digital signals. A filter's frequency response $H(e^{j\omega})$ can be expressed in terms of amplitude response $A(\omega)$ and phase response $\theta(\omega)$ as

$$H(e^{j\omega}) = A(\omega)\, e^{j\theta(\omega)}$$

If a filter has a linear phase response of the form

$$\theta(\omega) = -\alpha\omega \qquad -\pi \le \omega \le \pi \tag{10.6}$$

it will have both constant phase delay τ_p and constant group delay τ_g. In fact, in this case $\tau_p = \tau_g = \alpha$. It can be shown (for example, Rabiner and Gold 1975) that for $\alpha = 0$, the impulse response is an impulse of arbitrary strength:

$$h[n] = \begin{cases} c & n = 0 \\ 0 & n \neq 0 \end{cases}$$

For nonzero α, it can be shown that Eq. (10.6) is satisfied if and only if

$$\alpha = \frac{N-1}{2} \tag{10.7a}$$

$$h[n] = h[N-1-n] \qquad 0 \le n \le N-1 \tag{10.7b}$$

Within the constraints imposed by (10.7), the possible filters are usually separated into two types. Type 1 filters satisfy (10.7) with N odd, and type 2 filters satisfy (10.7) with N even. For type 1 filters, the axis of symmetry for $h[n]$ lies at $n = (N-1)/2$ as shown in Fig. 10.1. For type 2 filters, the axis of symmetry lies midway between $n = N/2$ and $n = (N-2)/2$ as shown in Fig. 10.2.

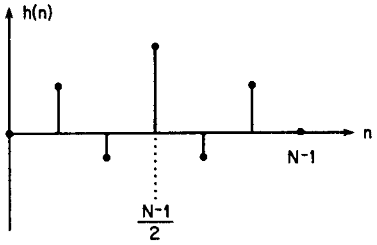

Figure 10.1 Impulse response for a type 1 linear phase FIR filter showing even symmetry about $n = (N-1)/2$.

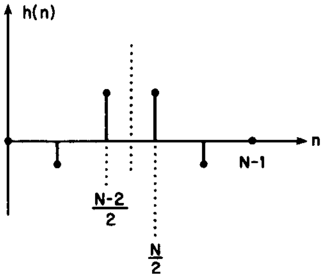

Figure 10.2 Impulse response for a type 2 linear phase FIR filter showing even symmetry about the abscissa midway between $n = (N-2)/2$ and $n = N/2$.

Filters can have constant group delay without having constant phase delay if the phase response is a straight line that does not pass through the origin. Such a phase response is defined as

$$\theta(\omega) = \beta + \alpha\omega \tag{10.8}$$

The phase response of a filter will satisfy (10.8) if

$$\alpha = \frac{N-1}{2} \tag{10.9a}$$

$$\beta = \pm\frac{\pi}{2} \tag{10.9b}$$

$$h[n] = -h[N-1-n] \qquad 0 \le n \le N-1 \tag{10.9c}$$

An impulse response satisfying (10.9*c*) is said to be *odd symmetric*, or *antisymmetric*. Within the constraints imposed by (10.9), the possible filters can be separated into two types that are commonly referred to as type 3 and type 4 *linear phase* filters despite the fact that the phase response is *not truly linear*. [The phase response is a straight line, but it does not pass through the origin, and consequently $\theta(\omega_1 + \omega_2)$ does not equal $\theta(\omega_1) + \theta(\omega_2)$.] Type 3 filters satisfy (10.9) with N odd, and type 4 filters satisfy (10.9) with N even. For type 3 filters, the axis of antisymmetry for $h[n]$ lies at $n = (N-1)/2$ as shown in Fig. 10.3. When $n = (N-1)/2$, with N even, Eq. (10.9*c*) gives

$$h\left[\frac{N-1}{2}\right] = -h\left[\frac{N-1}{2}\right]$$

Therefore, $h[(N-1)/2]$ must always equal zero in type 3 filters. For type 4 filters, the axis of antisymmetry lies midway between $n = N/2$ and $n = (N-2)/2$ as shown in Fig. 10.4.

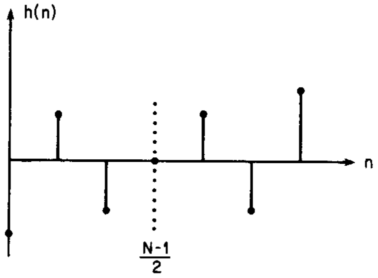

Figure 10.3 Impulse response for a type 3 linear phase FIR filter showing odd symmetry about $n = (N-1)/2$.

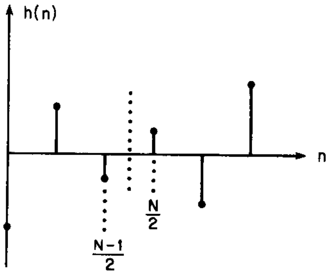

Figure 10.4 Impulse response for a type 4 linear phase FIR filter showing odd symmetry about the abscissa midway between $n = (N-2)/2$ and $n = N/2$.

The discrete-time Fourier transform (DTFT) can be used directly to obtain the frequency response of any FIR filter. However, for the special case of linear phase FIR filters, the symmetry and "realness" properties of the impulse response can be used to modify the general DTFT to obtain dedicated formulas having reduced computational burdens.

The frequency response $H(e^{j\omega T})$ and amplitude response $A(e^{j\omega T})$ are listed in Table 10.1 for the four types of linear phase FIR filters. The properties of these four types are summarized in Table 10.2. A C function, **cgdFirResponse( )**, which implements the equations of Table 10.1 is provided in Listing 10.1. The function **normalizeResponse( )** in Listing 10.2 can be used to normalize the response so that the peak pass-band value is at 0 dB.

At first glance, the fact that $A(\omega)$ is periodic with a period of 4π for type 2 and type 4 filters seems to contradict the fundamental relationship between sampling rate and folding frequency that was established in Chap. 7. The difficulty lies in how we have defined $A(\omega)$. The frequency response $H(\omega)$ is in fact periodic in 2π for all four types as we would expect. Both $\mathrm{Re}[H(\omega)]$ and $\mathrm{Im}[H(\omega)]$ are periodic in 2π, but factors of -1 are allocated between $A(\omega)$ and $\theta(\omega)$ differently over the intervals $(0, 2\pi)$ and $(2\pi, 4\pi)$ so that $\theta(\omega)$ can be made linear [and $A(\omega)$ can be made analytic].

TABLE 10.1 Frequency Response Formulas for Linear Phase FIR Filters

$h(nT)$ symmetry	N	$H(e^{j\omega T})$	$A(e^{j\omega T})$
Even	Odd	$e^{-j\omega(N-1)T/2} A(e^{j\omega T})$	$\sum_{k=0}^{(N-1)/2} a_k \cos \omega k T$
Even	Even	$e^{-j\omega(N-1)T/2} A(e^{j\omega T})$	$\sum_{k=1}^{N/2} b_k \cos[\omega(k - \frac{1}{2})T]$
Odd	Odd	$e^{-j[\omega(N-1)T/2 - \pi/2]} A(e^{j\omega T})$	$\sum_{k=1}^{(N-1)/2} a_k \sin \omega k T$
Odd	Even	$e^{-j[\omega(N-1)T/2 - \pi/2]} A(e^{j\omega T})$	$\sum_{k=1}^{N/2} b_k \sin[\omega(k - \frac{1}{2})T]$

$$a_0 = h\left[\frac{(N-1)T}{2}\right] \qquad a_k = 2h\left[\left(\frac{N-1}{2} - k\right)T\right] \qquad b_k = 2h\left[\left(\frac{N}{2} - k\right)T\right]$$

TABLE 10.2 Properties of FIR Filters Having Constant Group Delay

	Type			
	1	2	3	4
Length, N	Odd	Even	Odd	Even
Symmetry about $\omega = 0$	Even	Even	Odd	Odd
Symmetry about $\omega = \pi$	Even	Odd	Odd	Even
Periodicity	2π	4π	2π	4π

Some of the properties listed in Table 10.2 have an impact on which types can be used in particular applications. As a consequence of odd symmetry about $\omega = 0$, types 3 and 4 always have $A(0) = 0$ and should therefore not be used for lowpass or bandstop filters. As a consequence of their odd symmetry about $\omega = \pi$, types 2 and 3 always have $A(\pi) = 0$ and should therefore not be used for highpass or bandstop filters. Within the bounds of these restrictions, the choice between an odd-length or even-length filter is often made so that the desired transition frequency falls as close as possible to the midpoint between two sampled frequencies. The phase response of types 3 and 4 includes a constant component of 90° in addition to the linear component. Therefore, these types are suited for use as differentiators and Hilbert transformers (see Rabiner and Gold 1975).

Listing 10.1 cgdFirResponse( )

```
/*********************************/
/*                               */
/*   Listing 10.1                */
/*                               */
/*   cgdFirResponse()            */
/*                               */
/*********************************/

void cgdFirResponse( int firType,
                         int numbTaps,
                         real hh[],
                         logical dbScale,
                         int numberOfPoints,
                         real Hd[])
{
int index, L, n;
real lambda, work;
/*printf("in symFirResponse\n");*/

for( L=0; L<=numberOfPoints-1; L++)
    {
    lambda = L * PI / (real) numberOfPoints;
    switch (firType) {
        case 1:              /* symmetric and odd */
            work = hh[(numbTaps-1)/2];
            for( n=1; n<=((numbTaps-1)/2); n++) {
                index = (numbTaps-1)/2 - n;
                work = work + 2.0 * hh[index] * cos(n*lambda);
                }
            break;
        case 2:              /* symmetric and even */
            work = 0.0;
            for( n=1; n<=(numbTaps/2); n++) {
                index = numbTaps/2-n;
                work = work + 2.0 * hh[index] * cos((n-0.5)*lambda);
                }
            break;
        case 3:              /* antisymmetric and odd */
            work = 0.0;
            for( n=1; n<=((numbTaps-1)/2); n++) {
                index = (numbTaps-1)/2 - n;
                work = work + 2.0 * hh[index] * sin(n*lambda);
                }
            break;
        case 4:              /* symmetric and even */
            work = 0.0;
            for( n=1; n<=(numbTaps/2); n++) {
                index = numbTaps/2-n;
                work = work + 2.0 * hh[index] * sin((n-0.5)*lambda);
                }
            break;
        }

    if(dbScale)
        {Hd[L] = 20.0 * log10(fabs(work));}
    else
        {Hd[L] = fabs(work);}
    if(!(L%10)) printf("%3d\r",numberOfPoints-L);
    }
return;
}
```

Listing 10.2 normalizeResponse( )

```
/*********************************/
/*                               */
/*   Listing 10.2                */
/*                               */
/*   normalizeResponse()         */
/*                               */
/*********************************/

void normalizeResponse(   logical dbScale,
                          int numPts,
                          real H[])
{
int n;
real biggest;

if(dbScale)
    {
    biggest = -100.0;
    for( n=0; n<=numPts-1; n++)
        {if(H[n]>biggest) biggest = H[n];}
    for( n=0; n<=numPts-1; n++)
        {H[n] = H[n]-biggest;}
    }
else
    {
    biggest = 0.0;
    for( n=0; n<=numPts-1; n++)
```

```
          {if(H[n]>biggest) biggest = H[n];}
     for( n=0; n<=numPts-1; n++)
          {H[n] = H[n]/biggest;}
     }
return;
}
```