```
        for (j=2;j<=m;j++) {
            j2=j+j;
            p[j] += (SQR(w1[j2])+SQR(w1[j2-1])
                +SQR(w1[m44-j2])+SQR(w1[m43-j2]));
        }
        den += sumw;
    }
    den *= m4;                              Correct normalization.
    for (j=1;j<=m;j++) p[j] /= den;         Normalize the output.
    free_vector(w2,1,m);
    free_vector(w1,1,m4);
}
```

CITED REFERENCES AND FURTHER READING:

Oppenheim, A.V., and Schafer, R.W. 1989, *Discrete-Time Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall). [1]

Harris, F.J. 1978, *Proceedings of the IEEE*, vol. 66, pp. 51–83. [2]

Childers, D.G. (ed.) 1978, *Modern Spectrum Analysis* (New York: IEEE Press), paper by P.D. Welch. [3]

Champeney, D.C. 1973, *Fourier Transforms and Their Physical Applications* (New York: Academic Press).

Elliott, D.F., and Rao, K.R. 1982, *Fast Transforms: Algorithms, Analyses, Applications* (New York: Academic Press).

Bloomfield, P. 1976, *Fourier Analysis of Time Series – An Introduction* (New York: Wiley).

Rabiner, L.R., and Gold, B. 1975, *Theory and Application of Digital Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall).

13.5 Digital Filtering in the Time Domain

Suppose that you have a signal that you want to filter digitally. For example, perhaps you want to apply *high-pass* or *low-pass* filtering, to eliminate noise at low or high frequencies respectively; or perhaps the interesting part of your signal lies only in a certain frequency band, so that you need a *bandpass* filter. Or, if your measurements are contaminated by 60 Hz power-line interference, you may need a *notch filter* to remove only a narrow band around that frequency. This section speaks particularly about the case in which you have chosen to do such filtering in the time domain.

Before continuing, we hope you will reconsider this choice. Remember how convenient it is to filter in the Fourier domain. You just take your whole data record, FFT it, multiply the FFT output by a filter function $\mathcal{H}(f)$, and then do an inverse FFT to get back a filtered data set in time domain. Here is some additional background on the Fourier technique that you will want to take into account.

- Remember that you must define your filter function $\mathcal{H}(f)$ for both positive and negative frequencies, and that the magnitude of the frequency extremes is always the Nyquist frequency $1/(2\Delta)$, where Δ is the sampling interval. The magnitude of the smallest nonzero frequencies in the FFT is $\pm 1/(N\Delta)$, where N is the number of (complex) points in the FFT. The positive and negative frequencies to which this filter are applied are arranged in wrap-around order.
- If the measured data are real, and you want the filtered output also to be real, then your arbitrary filter function should obey $\mathcal{H}(-f) = \mathcal{H}(f)^*$. You can arrange this most easily by picking an $\mathcal{H}$ that is real and even in f.

- If your chosen $\mathcal{H}(f)$ has sharp vertical edges in it, then the *impulse response* of your filter (the output arising from a short impulse as input) will have damped "ringing" at frequencies corresponding to these edges. There is nothing wrong with this, but if you don't like it, then pick a smoother $\mathcal{H}(f)$. To get a first-hand look at the impulse response of your filter, just take the inverse FFT of your $\mathcal{H}(f)$. If you smooth all edges of the filter function over some number k of points, then the impulse response function of your filter will have a span on the order of a fraction $1/k$ of the whole data record.
- If your data set is too long to FFT all at once, then break it up into segments of any convenient size, as long as they are much longer than the impulse response function of the filter. Use zero-padding, if necessary.
- You should probably remove any trend from the data, by subtracting from it a straight line through the first and last points (i.e., make the first and last points equal to zero). If you are segmenting the data, then you can pick overlapping segments and use only the middle section of each, comfortably distant from edge effects.
- A digital filter is said to be *causal* or *physically realizable* if its output for a particular time-step depends only on inputs at that particular time-step or earlier. It is said to be *acausal* if its output can depend on both earlier and later inputs. Filtering in the Fourier domain is, in general, acausal, since the data are processed "in a batch," without regard to time ordering. Don't let this bother you! Acausal filters can generally give superior performance (e.g., less dispersion of phases, sharper edges, less asymmetric impulse response functions). People use causal filters not because they are better, but because some situations just don't allow access to out-of-time-order data. Time domain filters can, in principle, be either causal or acausal, but they are most often used in applications where physical realizability is a constraint. For this reason we will restrict ourselves to the causal case in what follows.

If you are still favoring time-domain filtering after all we have said, it is probably because you have a real-time application, for which you must process a continuous data stream and wish to output filtered values at the same rate as you receive raw data. Otherwise, it may be that the quantity of data to be processed is so large that you can afford only a very small number of floating operations on each data point and cannot afford even a modest-sized FFT (with a number of floating operations per data point several times the logarithm of the number of points in the data set or segment).

Linear Filters

The most general linear filter takes a sequence x_k of input points and produces a sequence y_n of output points by the formula

$$y_n = \sum_{k=0}^{M} c_k \; x_{n-k} + \sum_{j=1}^{N} d_j \; y_{n-j} \tag{13.5.1}$$

Here the $M + 1$ coefficients c_k and the N coefficients d_j are fixed and define the filter response. The filter (13.5.1) produces each new output value from the current and M previous input values, and from its own N previous output values. If $N = 0$, so that there is no second sum in (13.5.1), then the filter is called *nonrecursive* or *finite impulse response (FIR)*. If $N \neq 0$, then it is called *recursive* or *infinite impulse response (IIR)*. (The term "IIR" connotes only that such filters are *capable* of having infinitely long impulse responses, not that their impulse response is necessarily long in a particular application. Typically the response of an IIR filter will drop off exponentially at late times, rapidly becoming negligible.)

The relation between the c_k's and d_j's and the filter response function $\mathcal{H}(f)$ is

$$\mathcal{H}(f) = \frac{\sum\limits_{k=0}^{M} c_k e^{-2\pi i k (f\Delta)}}{1 - \sum\limits_{j=1}^{N} d_j e^{-2\pi i j (f\Delta)}} \tag{13.5.2}$$

where Δ is, as usual, the sampling interval. The Nyquist interval corresponds to $f\Delta$ between $-1/2$ and $1/2$. For FIR filters the denominator of (13.5.2) is just unity.

Equation (13.5.2) tells how to determine $\mathcal{H}(f)$ from the c's and d's. To design a filter, though, we need a way of doing the inverse, getting a suitable set of c's and d's — as small a set as possible, to minimize the computational burden — from a desired $\mathcal{H}(f)$. Entire books are devoted to this issue. Like many other "inverse problems," it has no all-purpose solution. One clearly has to make compromises, since $\mathcal{H}(f)$ is a full continuous function, while the short list of c's and d's represents only a few adjustable parameters. The subject of digital filter design concerns itself with the various ways of making these compromises. We cannot hope to give any sort of complete treatment of the subject. We can, however, sketch a couple of basic techniques to get you started. For further details, you will have to consult some specialized books (see references).

FIR (Nonrecursive) Filters

When the denominator in (13.5.2) is unity, the right-hand side is just a discrete Fourier transform. The transform is easily invertible, giving the desired small number of c_k coefficients in terms of the same small number of values of $\mathcal{H}(f_i)$ at some discrete frequencies f_i. This fact, however, is not very useful. The reason is that, for values of c_k computed in this way, $\mathcal{H}(f)$ will tend to oscillate wildly in between the discrete frequencies where it is pinned down to specific values.

A better strategy, and one which is the basis of several formal methods in the literature, is this: Start by pretending that you are willing to have a relatively large number of filter coefficients, that is, a relatively large value of M. Then $\mathcal{H}(f)$ can be fixed to desired values on a relatively fine mesh, and the M coefficients c_k, $k = 0, \dots, M-1$ can be found by an FFT. Next, truncate (set to zero) most of the c_k's, leaving nonzero only the first, say, K, $(c_0, c_1, \dots, c_{K-1})$ and last $K-1$, $(c_{M-K+1}, \dots, c_{M-1})$. The last few c_k's are filter coefficients at *negative lag*, because of the wrap-around property of the FFT. But we don't want coefficients at negative lag. Therefore we cyclically shift the array of c_k's, to bring everything to positive lag. (This corresponds to introducing a time-delay into the filter.) Do this by copying the c_k's into a new array of length M in the following order:

$$(c_{M-K+1}, \dots, c_{M-1},\ c_0,\ c_1, \dots, c_{K-1},\ 0,\ 0, \dots, 0) \qquad (13.5.3)$$

To see if your truncation is acceptable, take the FFT of the array (13.5.3), giving an approximation to your original $\mathcal{H}(f)$. You will generally want to compare the *modulus* $|\mathcal{H}(f)|$ to your original function, since the time-delay will have introduced complex phases into the filter response.

If the new filter function is acceptable, then you are done and have a set of $2K-1$ filter coefficients. If it is not acceptable, then you can either (i) increase K and try again, or (ii) do something fancier to improve the acceptability for the same K. An example of something fancier is to modify the magnitudes (but not the phases) of the unacceptable $\mathcal{H}(f)$ to bring it more in line with your ideal, and then to FFT to get new c_k's. Once again set to zero all but the first $2K-1$ values of these (no need to cyclically shift since you have preserved the time-delaying phases), then inverse transform to get a new $\mathcal{H}(f)$, which will often be more acceptable. You can iterate this procedure. Note, however, that the procedure will not converge if your requirements for acceptability are more stringent than your $2K-1$ coefficients can handle.

The key idea, in other words, is to iterate between the space of coefficients and the space of functions $\mathcal{H}(f)$, until a Fourier conjugate pair that satisfies the imposed constraints *in both spaces* is found. A more formal technique for this kind of iteration is the *Remes Exchange Algorithm* which produces the best Chebyshev approximation to a given desired frequency response with a fixed number of filter coefficients (cf. §5.13).

IIR (Recursive) Filters

Recursive filters, whose output at a given time depends both on the current and previous inputs and on previous outputs, can generally have performance that is superior to nonrecursive filters with the same total number of coefficients (or same number of floating operations per input point). The reason is fairly clear by inspection of (13.5.2): A nonrecursive filter has a frequency response that is a polynomial in the variable $1/z$, where

$$z \equiv e^{2\pi i(f\Delta)} \tag{13.5.4}$$

By contrast, a recursive filter's frequency response is a *rational function* in $1/z$. The class of rational functions is especially good at fitting functions with sharp edges or narrow features, and most desired filter functions are in this category.

Nonrecursive filters are always stable. If you turn off the sequence of incoming x_i's, then after no more than M steps the sequence of y_j's produced by (13.5.1) will also turn off. Recursive filters, feeding as they do on their own output, are not necessarily stable. If the coefficients d_j are badly chosen, a recursive filter can have exponentially growing, so-called *homogeneous*, modes, which become huge even after the input sequence has been turned off. This is not good. The problem of designing recursive filters, therefore, is not just an inverse problem; it is an inverse problem with an additional stability constraint.

How do you tell if the filter (13.5.1) is stable for a given set of c_k and d_j coefficients? Stability depends only on the d_j's. The filter is stable if and only if all N complex roots of the *characteristic polynomial* equation

$$z^N - \sum_{j=1}^{N} d_j z^{N-j} = 0 \tag{13.5.5}$$

are inside the unit circle, i.e., satisfy

$$|z| \le 1 \tag{13.5.6}$$

The various methods for constructing stable recursive filters again form a subject area for which you will need more specialized books. One very useful technique, however, is the *bilinear transformation method*. For this topic we define a new variable w that reparametrizes the frequency f,

$$w \equiv \tan[\pi(f\Delta)] = i\left(\frac{1-e^{2\pi i(f\Delta)}}{1+e^{2\pi i(f\Delta)}}\right) = i\left(\frac{1-z}{1+z}\right) \tag{13.5.7}$$

Don't be fooled by the i's in (13.5.7). This equation maps real frequencies f into real values of w. In fact, it maps the Nyquist interval $-\frac{1}{2} < f\Delta < \frac{1}{2}$ onto the real w axis $-\infty < w < +\infty$. The inverse equation to (13.5.7) is

$$z = e^{2\pi i(f\Delta)} = \frac{1+iw}{1-iw} \tag{13.5.8}$$

In reparametrizing f, w also reparametrizes z, of course. Therefore, the condition for stability (13.5.5)–(13.5.6) can be rephrased in terms of w: If the filter response $\mathcal{H}(f)$ is written as a function of w, then the filter is stable if and only if the poles of the filter function (zeros of its denominator) are all in the upper half complex plane,

$$\mathrm{Im}(w) \ge 0 \tag{13.5.9}$$

The idea of the bilinear transformation method is that instead of specifying your desired $\mathcal{H}(f)$, you specify only its desired modulus square, $|\mathcal{H}(f)|^2 = \mathcal{H}(f)\mathcal{H}(f)^* = \mathcal{H}(f)\mathcal{H}(-f)$. Pick this to be approximated by some rational function in w^2. Then find all the poles of this function in the w complex plane. Every pole in the lower half-plane will have a corresponding pole in the upper half-plane, by symmetry. The idea is to form a product only of the factors with good poles, ones in the upper half-plane. This product is your *stably realizable* $\mathcal{H}(f)$. Now substitute equation (13.5.7) to write the function as a rational function in z, and compare with equation (13.5.2) to read off the c's and d's.

The procedure becomes clearer when we go through an example. Suppose we want to design a simple bandpass filter, whose lower cutoff frequency corresponds to a value $w = a$, and whose upper cutoff frequency corresponds to a value $w = b$, with a and b both positive numbers. A simple rational function that accomplishes this is

$$|\mathcal{H}(f)|^2 = \left(\frac{w^2}{w^2 + a^2}\right)\left(\frac{b^2}{w^2 + b^2}\right) \tag{13.5.10}$$

This function does not have a very sharp cutoff, but it is illustrative of the more general case. To obtain sharper edges, one could take the function (13.5.10) to some positive integer power, or, equivalently, run the data sequentially through some number of copies of the filter that we will obtain from (13.5.10).

The poles of (13.5.10) are evidently at $w = \pm ia$ and $w = \pm ib$. Therefore the stably realizable $\mathcal{H}(f)$ is

$$\mathcal{H}(f) = \left(\frac{w}{w - ia}\right)\left(\frac{ib}{w - ib}\right) = \frac{\left(\frac{1-z}{1+z}\right) b}{\left[\left(\frac{1-z}{1+z}\right) - a\right]\left[\left(\frac{1-z}{1+z}\right) - b\right]} \tag{13.5.11}$$

We put the i in the numerator of the second factor in order to end up with real-valued coefficients. If we multiply out all the denominators, (13.5.11) can be rewritten in the form

$$\mathcal{H}(f) = \frac{-\frac{b}{(1+a)(1+b)} + \frac{b}{(1+a)(1+b)} z^{-2}}{1 - \frac{(1+a)(1-b)+(1-a)(1+b)}{(1+a)(1+b)} z^{-1} + \frac{(1-a)(1-b)}{(1+a)(1+b)} z^{-2}} \tag{13.5.12}$$

from which one reads off the filter coefficients for equation (13.5.1),

$$\begin{aligned} c_0 &= -\frac{b}{(1+a)(1+b)} \\ c_1 &= 0 \\ c_2 &= \frac{b}{(1+a)(1+b)} \\ d_1 &= \frac{(1+a)(1-b) + (1-a)(1+b)}{(1+a)(1+b)} \\ d_2 &= -\frac{(1-a)(1-b)}{(1+a)(1+b)} \end{aligned} \tag{13.5.13}$$

This completes the design of the bandpass filter.

Sometimes you can figure out how to construct directly a rational function in w for $\mathcal{H}(f)$, rather than having to start with its modulus square. The function that you construct has to have its poles only in the upper half-plane, for stability. It should also have the property of going into its own complex conjugate if you substitute $-w$ for w, so that the filter coefficients will be real.

For example, here is a function for a notch filter, designed to remove only a narrow frequency band around some fiducial frequency $w = w_0$, where w_0 is a positive number,

$$\begin{aligned} \mathcal{H}(f) &= \left(\frac{w - w_0}{w - w_0 - i\epsilon w_0}\right)\left(\frac{w + w_0}{w + w_0 - i\epsilon w_0}\right) \\ &= \frac{w^2 - w_0^2}{(w - i\epsilon w_0)^2 - w_0^2} \end{aligned} \tag{13.5.14}$$

In (13.5.14) the parameter ϵ is a small positive number that is the desired width of the notch, as a

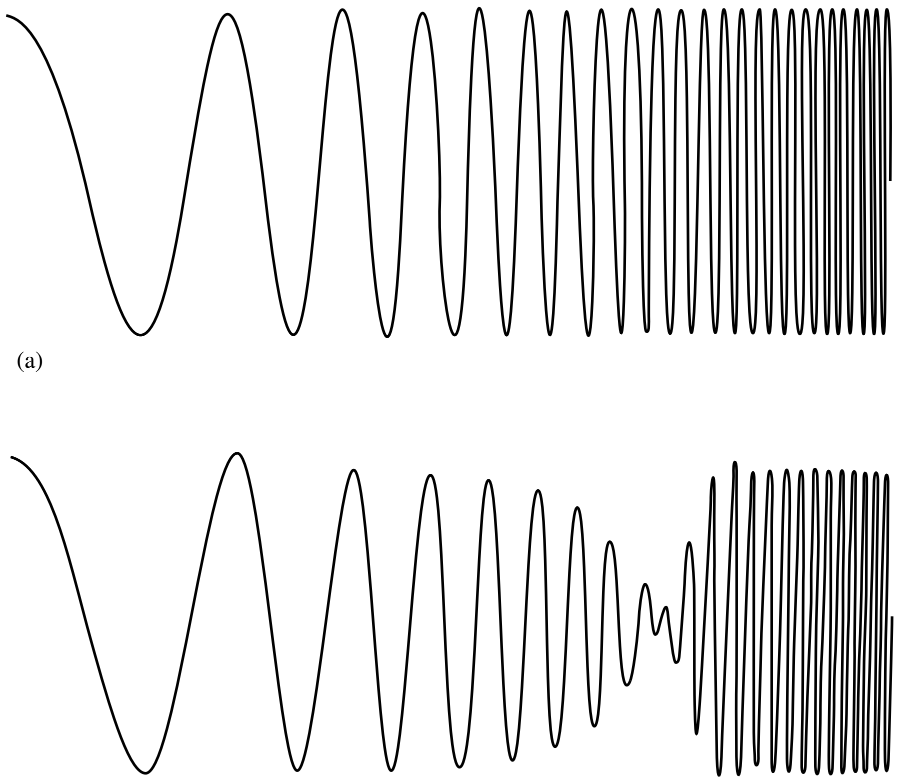

Figure 13.5.1. (a) A "chirp," or signal whose frequency increases continuously with time. (b) Same signal after it has passed through the notch filter (13.5.15). The parameter ϵ is here 0.2.

fraction of w_0. Going through the arithmetic of substituting z for w gives the filter coefficients

$$
\begin{aligned}
c_0 &= \frac{1 + w_0^2}{(1 + \epsilon w_0)^2 + w_0^2} \\
c_1 &= -2\frac{1 - w_0^2}{(1 + \epsilon w_0)^2 + w_0^2} \\
c_2 &= \frac{1 + w_0^2}{(1 + \epsilon w_0)^2 + w_0^2} \\
d_1 &= 2\frac{1 - \epsilon^2 w_0^2 - w_0^2}{(1 + \epsilon w_0)^2 + w_0^2} \\
d_2 &= -\frac{(1 - \epsilon w_0)^2 + w_0^2}{(1 + \epsilon w_0)^2 + w_0^2}
\end{aligned}
\qquad (13.5.15)
$$

Figure 13.5.1 shows the results of using a filter of the form (13.5.15) on a "chirp" input signal, one that glides upwards in frequency, crossing the notch frequency along the way.

While the bilinear transformation may seem very general, its applications are limited by some features of the resulting filters. The method is good at getting the general shape of the desired filter, and good where "flatness" is a desired goal. However, the nonlinear mapping between w and f makes it difficult to design to a desired shape for a cutoff, and may move cutoff frequencies (defined by a certain number of dB) from their desired places. Consequently, practitioners of the art of digital filter design reserve the bilinear transformation for specific situations, and arm themselves with a variety of other tricks. We suggest that you do likewise, as your projects demand.

CITED REFERENCES AND FURTHER READING:

Hamming, R.W. 1983, *Digital Filters*, 2nd ed. (Englewood Cliffs, NJ: Prentice-Hall).

Antoniou, A. 1979, *Digital Filters: Analysis and Design* (New York: McGraw-Hill).

Parks, T.W., and Burrus, C.S. 1987, *Digital Filter Design* (New York: Wiley).

Oppenheim, A.V., and Schafer, R.W. 1989, *Discrete-Time Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall).

Rice, J.R. 1964, *The Approximation of Functions* (Reading, MA: Addison-Wesley); also 1969, *op. cit.*, Vol. 2.

Rabiner, L.R., and Gold, B. 1975, *Theory and Application of Digital Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall).

13.6 Linear Prediction and Linear Predictive Coding

We begin with a very general formulation that will allow us to make connections to various special cases. Let $\{y'_\alpha\}$ be a set of measured values for some underlying set of true values of a quantity y, denoted $\{y_\alpha\}$, related to these true values by the addition of random noise,

$$y'_\alpha = y_\alpha + n_\alpha \tag{13.6.1}$$

(compare equation 13.3.2, with a somewhat different notation). Our use of a Greek subscript to index the members of the set is meant to indicate that the data points are not necessarily equally spaced along a line, or even ordered: they might be "random" points in three-dimensional space, for example. Now, suppose we want to construct the "best" estimate of the true value of some particular point $y_\star$ as a linear combination of the known, noisy, values. Writing

$$y_\star = \sum_\alpha d_{\star\alpha} y'_\alpha + x_\star \tag{13.6.2}$$

we want to find coefficients $d_{\star\alpha}$ that minimize, in some way, the *discrepancy* $x_\star$. The coefficients $d_{\star\alpha}$ have a "star" subscript to indicate that they depend on the choice of point $y_\star$. Later, we might want to let $y_\star$ be one of the existing y_α's. In that case, our problem becomes one of optimal filtering or estimation, closely related to the discussion in §13.3. On the other hand, we might want $y_\star$ to be a completely new point. In that case, our problem will be one of *linear prediction*.

A natural way to minimize the discrepancy $x_\star$ is in the statistical mean square sense. If angle brackets denote statistical averages, then we seek $d_{\star\alpha}$'s that minimize

$$\begin{aligned}\left\langle x_\star^2 \right\rangle &= \left\langle \left[\sum_\alpha d_{\star\alpha}(y_\alpha + n_\alpha) - y_\star \right]^2 \right\rangle \\ &= \sum_{\alpha\beta} (\langle y_\alpha y_\beta \rangle + \langle n_\alpha n_\beta \rangle) d_{\star\alpha} d_{\star\beta} - 2 \sum_\alpha \langle y_\star y_\alpha \rangle \, d_{\star\alpha} + \left\langle y_\star^2 \right\rangle\end{aligned} \tag{13.6.3}$$