spectrum relatively unaffected. We'll see shortly, however, that it's just a toy compared to the really effective filters that are possible. Before we look at more complicated filters, though, we need to look at the limitations imposed by implementing filters on a computer.

3 Digital filters

We're going to concentrate entirely on filters implemented on a computer, which we'll call *digital filters*. The kinds of filters that can be implemented with components like acoustic delay lines, inductors, capacitors, and resistors, behave in similar ways, and are described by similar mathematics. But there are certain characteristics, both advantageous and restrictive, that are specific to digital filters. We'll begin with two crucial peculiarities.

In the previous section we assumed the kind of filter we studied can be implemented with any delay τ whatsoever. But when we use a computer we keep signals in arrays, and we are allowed to delay signals only an integer number of samples, because the time variable corresponds to the array index. On a computer, therefore, the delay τ must be an integer multiple of the sampling period T_s. That's one very important restriction.

There is another important restriction: In the digital world, frequencies above half the sampling frequency, the Nyquist frequency, don't really exist — we think of them as aliased to frequencies below the Nyquist frequency. If a phasor jumps more than π radians between sampling instants, we agree to think of it as jumping less than π radians per sample, thus giving us an unambiguous representation of frequencies less than the Nyquist, at the expense of not being able to represent any above the Nyquist. This means that the frequency plots of filter magnitude responses need not extend beyond the Nyquist frequency. It is usually convenient to normalize the frequency variable in such plots to the sampling rate, making the Nyquist frequency equal to 0.5. I will label the abscissas of such frequency-response plots "frequency, fractions of sampling rate."

I now will adjust notation slightly to simplify many equations in the rest of this book. Examine again the first equation in this chapter, which shows that delaying a phasor one sample, T_s sec, multiplies the phasor by $e^{-j\omega T_s}$. The frequency variable ω here has the units radians per sec. Therefore, ωT_s has the units radians per sample, and is the number of radians that the phasor turns between samples. We can always measure time in the digital domain in terms of the sample number, and we can always convert to actual time by multiplying by T_s sec. Therefore, in the digital world, we might as well think of ω as having the units radians **per sample** to begin with, and not write T_s with it all the time.

So from now on, in the digital domain, we'll measure the frequency ω in radians per sample. (In the continuous domain, we'll continue to use radians per sec.) The digital sampling frequency is then $\omega = 2\pi$ radians per sample (a full cycle between samples), and the Nyquist frequency is $\omega = \pi$ radians per sample (half a cycle between samples). The normalized frequency axis mentioned above, "frequency, fractions of sampling rate,'' can be thought of as measured in the units cycles per sample. To convert from this normalized frequency to actual frequency, multiply by the sampling rate.

A word about phasors. In the continuous world we write a phasor as $x_t = e^{j\omega t}$, where ω has the units radians per sec. In the digital world we'll write it in exactly the same way, remembering that ω is now measured in radians per sample, and interpreting t as the integer sample number. A delay of one sample in the digital domain therefore multiplies a phasor by $e^{-j\omega}$.

For a very simple example of a digital filter, suppose we use a delay of one sampling period in the filter equation Eq. 2.1:

$$y_t = x_t + a_1 x_{t-1} \tag{3.1}$$

where now the signals are indexed by the integer sample number t. When the digital signal x_t is the phasor $e^{j\omega t}$, the output phasor is

$$y_t = e^{j\omega t}\left[1 + a_1 e^{-j\omega}\right] \tag{3.2}$$

and the corresponding magnitude response of this digital filter is, as in Eqs. 2.6 and 2.7,

$$|H(\omega)| = |1 + a_1 e^{-j\omega}| = |1 + a_1^2 + 2a_1 \cos\omega|^{1/2} \tag{3.3}$$

At the Nyquist frequency, ω is π radians per sample, and the cosine in Eq. 3.3 is equal to -1. When $a_1 > 0$ this means there is a dip at that point in the magnitude response. On the other hand, there is a relative peak at zero frequency, so this filter is *lowpass*, meaning it tends to pass low frequencies and reject high frequencies. Fig. 3.1 shows the frequency response of this filter for the value $a_1 = 0.99$. Because this is a digital filter, we need concern ourselves only with the frequencies below the Nyquist.

To summarize notation: In the continuous-time world, we'll use the continuous-time variable t sec and the frequency variable ω radians per sec; in the digital world we'll use the integer time variable t samples and the frequency variable ω radians per sample. The product ωt therefore measures radians per sec or per sample, depending on whether we are in the continuous or discrete domain. We've fussed a fair amount with notation in this section, but it will make life much simpler later on.

4 A big filter

I don't want to leave you with the impression that digital filters usually have only one or two terms. There's no reason we can't implement a filter with hundreds of terms; in fact this is done all the time. How can we possibly know how to select a few hundred coefficients so that the resulting digital filter has some desired, predetermined effect? This question is called the filter design problem. Fortunately, it's almost completely solved for feedforward digital filters. The mathematical problems involved were worked out in the 1960s and 1970s, and design packages are now widely

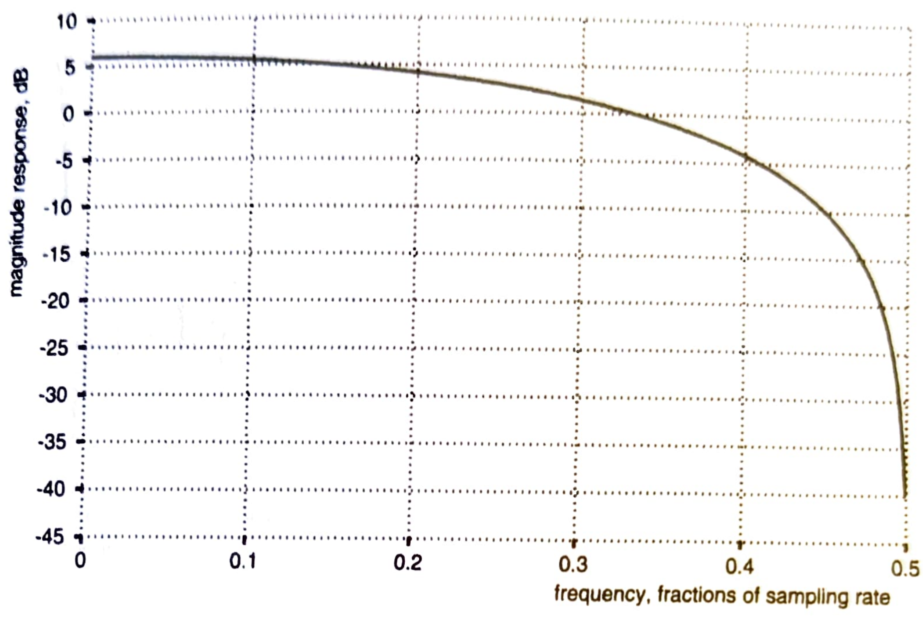

Fig. 3.1 Magnitude response (in dB) of a simple feedforward digital filter. The example shown has the filter parameter $a_1 = 0.99$ and a delay of one sampling period.

available. The particular filter used as an example in this section was designed using METEOR, a program based on linear programming [Steiglitz, et al., 1992].

Let's look at an example. Suppose we want to design a digital *bandstop* filter, which removes a particular range of frequencies, the ***stopband***, but passes all others. The stopband is chosen to be the interval $[0.22, 0.32]$ in normalized frequency (fractions of the sampling rate). We require that the magnitude response be no more than 0.01 in the stopband, and within 0.01 of unity in the passbands. Figures 4.1 and 4.2 show the result of using METEOR for this design problem.

An interesting point comes up when we specify the passbands. Of course we'd like the passbands to extend right up to the very edges of the stopband, so, for example, the filter would reject the frequency 0.35999999 and pass the frequency 0.36. But this is asking too much. It is a great strain on a filter to make such a sharp distinction between frequencies so close together. The filter needs some slack in frequency to get from one value to another, so we need to allow what are called ***transition bands***. The band between the normalized frequency 0.2 and 0.22 in this example is such a band. The narrower the transition bands, and the more exacting the amplitude specifications, the more terms we need in the filter to meet the specifications.

In the next section we'll start to develop a simple system for manipulating digital filters.