Center for Computer Research in Music and Acoustics April 1985

Department of Music

Report No. STAN-M-20

Introduction to Digital Filter Theory

by

Julius Orion Smith III

Tutorial sponsored by

The Fannie and John Hertz Foundation

CCRMA

DEPARTMENT OF MUSIC

Stanford University

Stanford, California 94305

Introduction to Digital Filter Theory

Julius Orion Smith III

Center for Computer Research in Music and Acoustics
Department of Music, Stanford University
Stanford, California 94305

Abstract

This manuscript is an outgrowth of notes for a course in computer music at Stanford University. It provides an elementary introduction to the analysis of digital filters, with emphasis on understanding spectral modifications caused by general filtering algorithms.

Table of Contents

Chapter 1

Introduction to Digital Filter Theory

1.1. Introduction

1.1.1. What Is a Filter?

Musicians have been using filters for thousands of years to shape the sounds of their art in various ways. For example, the evolution of the physical dimensions of the violin constitutes an evolution in filter design. The choice of wood, the shape of the cutouts, the purfling, the geometry of the bridge, and everything that affects resonance all have a bearing on how the violin body filters the signal induced at the bridge by the vibrating strings. Once a sound is airborne there is yet more filtering performed by the listening environment, the pinnae of the ear, and by idiosyncrasies of the hearing process.

Any medium through which the music signal passes, whatever its form, can be regarded as a filter. However, we do not usually think of something as a filter unless it can modify the sound in some way. For example, speaker wire is not considered a filter, but the speaker is (unfortunately). The different vowel sounds in speech are produced primarily by changing the shape of the mouth cavity, which changes the resonances and hence the filtering characteristics of the vocal tract. The tone control circuit in an ordinary car radio is a filter, as are the bass, midrange, and treble boosts in a stereo preamplifier. Graphic equalizers, reverberators, echo devices, phase shifters, and speaker crossover networks are further examples of useful filters in audio. There are also examples of undesirable filtering, such as the uneven reinforcement of certain frequencies in a room with "bad acoustics." A well-known signal processing wizard is said to have remarked, "When you think about it, everything is a filter."

A *digital* filter is just a filter that operates on digital signals, such as sound represented inside a computer. It is a *computation* which takes one sequence of numbers (the input signal) and produces a new sequence of numbers (the filtered output signal). The filters mentioned in the previous paragraph are not digital only because they operate on signals that are not digital. It is important to realize that a digital filter can do anything that a real-world filter can do. That is, all the filters alluded to above can be simulated to an arbitrary degree of precision digitally. Again, a digital filter is only a formula for going from one digital signal to another. It may exist as an equation on paper, as a small loop in a computer subroutine, or as a handful of integrated circuit chips properly interconnected.

1.1.2. Why Should a Musician Learn about Digital Filters?

Serious computer musicians generally use digital filters in every piece of music they create. Without digital reverberation, for example, it is difficult to get rich, full-bodied sound from the computer. However, reverberation is only one example of the capabilities of digital filters. A digital filter can arbitrarily shape the spectrum of a sound. Yet very few musicians are prepared to design the filter they need even when they know exactly what they want in the way of a spectral modification. It is the goal of this paper to assist sound designers by listing the concepts and tools necessary for doing custom filter design.

The reference, *Programs for Digital Signal Processing* [5] contains a collection of Fortran programs which design a wide variety of digital filters. In light of this available code, it is plausible to imagine that only programming skills are required to use digital filters. This perhaps is true for simple applications, but knowledge of how digital filters work will help at every phase of installing and using such software. Also, you must understand a program before you can modify it or extract pieces of it. Even in standard applications, effective use of a filter design program requires an understanding of the design parameters, which in turn requires some understanding of filter theory.

Perhaps most important for composers who design their own sounds, a vast range of imaginative filtering possibilities is available to those who understand how filters affect sounds. In my practical experience, intimate knowledge of filter theory has proved to be a very valuable tool in the design of musical instruments. Typically, a simple yet unusual filter is needed rather than one of the classical designs obtainable by using published software.

1.1.3. Relevant Math

First, it is assumed that the reader has some familiarity with computer music and knows at least conceptually how sound is stored and processed inside the computer. The architectural concepts introduced in [9] still pervade the current practice in computer music synthesis, and there are several introductory articles in the *Computer Music Journal*. An anthology of tutorial articles appears in [17,20].

Next, it is important to know the mathematical representation of a signal (sound). We will need only two types of signal representation: the *time-waveform* and the *spectrum*. These and related concepts are discussed elsewhere in [10], and this paper assumes a working knowledge of Fourier and z transforms. A brief review of some basic quantities such as physical units and various notation is given in appendix A, which should be consulted whenever an unfamiliar (or undefined) notational construct appears in this paper.

If appendix A is $\gamma\rho\epsilon\epsilon\kappa$ to you, then the following references may help to fill in the gaps. A good review of basic mathematics with lots of problems and a fair amount on complex numbers may be found in [18]. Facts about Fourier transforms are derived in [13, 3], and the first three chapters of

[4] give a good development of complex variables. Finally, an introduction to digital systems, on a level comparable to this paper, is given in [19].

The only essential mathematical requirement which is beyond the scope of most highschool curricula is that of complex variables. It is unfortunate that complex numbers are usually taught so late in school, because we need them to represent both time-waveforms and spectra. On the other hand, similarly advanced college math, such as calculus and linear algebra, can be sidestepped entirely in an all-digital treatment of filter theory. As we will see, the central reason for using complex numbers is the ease with which both time-waveforms and spectra may be represented.

1.1.4. The Simplest Lowpass Filter

This section gives a very basic example of the generic problem at hand: understanding the effect of a digital filter on the spectrum of a digital signal. The purpose of this example is to provide motivation for the general theory introduced in chapter 2.

Our example is the simplest possible lowpass filter. A lowpass filter is one which does not affect low frequencies and rejects high frequencies. The gain of the *ideal* lowpass filter is unity (1) for frequencies between 0 Hz and the *cutoff* frequency f_c Hz, and it is 0 for all higher frequencies. This ideal gain versus frequency curve is shown in Figure 1.1. The output spectrum is obtained by multiplying the input spectrum by the function shown.

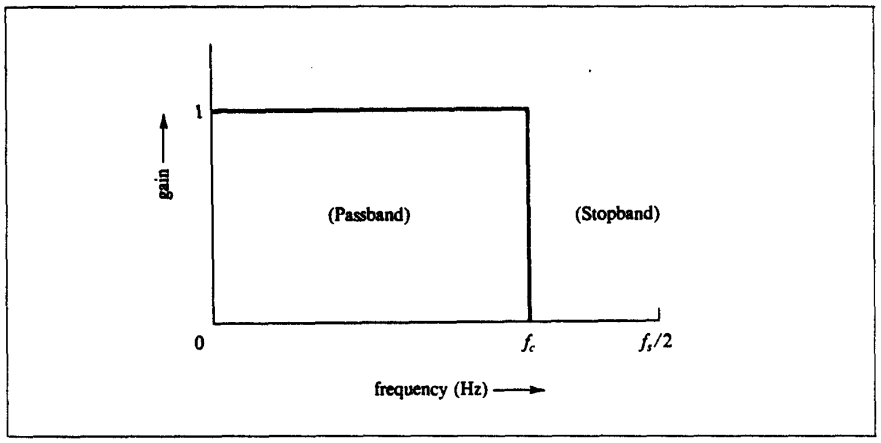

Figure 1.1. Gain vs. frequency specification for the ideal lowpass filter.

1.1.5. Definition of the Filter

The simplest (and by no means ideal) lowpass filter is given by the following *difference equation*:

$$y(n) = x(n) + x(n-1), \qquad n = 0, 1, 2, \ldots \tag{1.1}$$

where $x(n)$ is the filter input amplitude at time (or sample) n, and $y(n)$ is the output amplitude at time n. The *system diagram* for this little filter is given in Figure 1.2. The symbol "z^{-1}" means a delay of one sample; i.e., $z^{-1}x(n) = x(n-1)$.

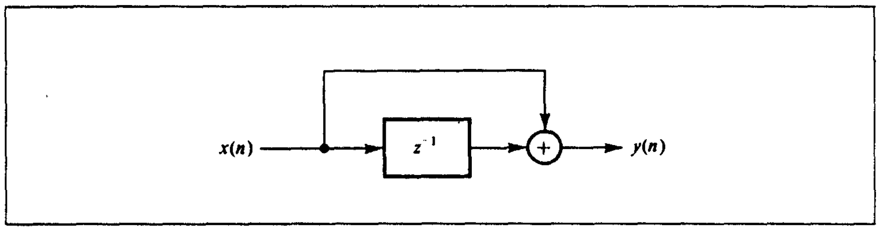

Figure 1.2. System diagram for the filter $y(n) = x(n) + x(n-1)$ (cf. (1).

It is important when working with spectra to be able to convert time from sample numbers, as in (1.1) above, to seconds. Thus, another way of writing the filter equation is

$$y(nT) = x(nT) + x((n-1)T), \qquad n = 0, 1, 2, \ldots \tag{1.1}$$

where T is the sampling interval. It is customary in digital signal processing to leave off T (pretend it equals 1), but anytime you see an n you can translate to *seconds* by thinking nT. Be careful with integer expressions, however, such as $(n-k)$, which would be $(n-k)T$ seconds, not $(nT-k)$.

To further our appreciation of this example, let's write a computer subroutine to implement (1.1). In the computer, $x(n)$ and $y(n)$ are data arrays and n is an array index. Since sound files are usually larger than what the computer can hold in memory all at once, we must process the data in blocks of some reasonable size. Therefore, the complete filtering operation consists of two loops, one within the other. The outer loop fills the input array x and empties the output array y, while the inner loop does the actual filtering on the x array to produce y. Let M denote the block size (i.e., the number of samples to be processed on each iteration of the outer loop). In Pascal, the inner loop of the subroutine might appear as shown in Figure 1.3. The outer loop might read something like "fill $x[1..M]$ from the input file," "call SIMPLP," and "write out $y[1..M]$." However, there is one more important statement whose absence will cause a "glitch" every M samples. Can you guess what it is? After the call to SIMPLP, there must be the line "$x[0] := x[M];$" to make $x[n-1]$ valid in the first iteration of the next call to SIMPLP. (For the very first call to SIMPLP, $x[0] := 0$.)

```
procedure SIMPLP(x,y:array[0..M] of real);
  var n : integer;
  for n := 1 to M do
    y[n] := x[n] + x[n-1];
end.
```

Figure 1.3. Pascal implementation of the simple lowpass filter of (1.1).

You might suspect that since (1.1) is the simplest possible lowpass filter, that it is also somehow the worst possible lowpass filter. How bad is it? In what sense is it bad? How do we even know it is a lowpass at all? Can it blow up?* To answer these questions, we need to find the *frequency response* of this filter.

1.1.6. Finding the Frequency Response

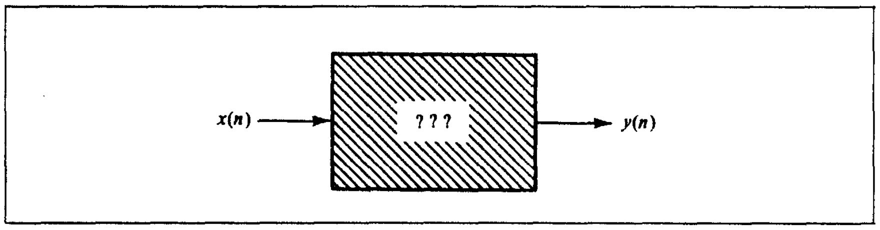

Figure Fig. 1.4. "Black box" representation of an arbitrary filter.

Think of the filter expressed by (1.1) as a "black box" as shown in Fig. Fig. 1.4. We want to know the effect of this black box on the spectrum of $x(\cdot)$, where $x(\cdot)$ represents the entire input signal (see appendix A). Suppose we test its effect at each frequency separately. This is called *sine-wave analysis*. Figure 1.5 shows an example of an input-output pair, for the filter of (1.1), at the frequency $f = f_s/4$ Hz, where f_s is the signal sampling rate. (The continuous-time waveform has been drawn through the samples for clarity.) Figure 1.5a shows the input, and Fig. 1.5b shows the output. The ratio of the peak output amplitude to the peak input amplitude is the filter *gain* at this frequency. From Fig. 1.5, we find that the gain is about 1.414 at the frequency $f_s/4$. The phase of the output minus the phase of the input is called the *phase response* of the filter. Figure 1.5 shows that the filter of (1.1) has a phase response equal to $-2\pi/8$ (minus one-eighth of a cycle) at the frequency $f_s/4$. Continuing in this way, we can input a sinusoid at each frequency (from 0 to $f_s/2$), examine the input and output waveforms as in Fig. 1.5, and record on a graph the peak-amplitude ratio (gain) and phase shift for each frequency. The resultant pair of plots, shown in Figure 1.6, is called the *frequency response*. Note that Fig. Fig. 1.5 specifies the middle point of each graph in Fig. 1.6.

Not every black box *has* a frequency response, however. What good is a pair of graphs such as shown in Fig. 1.6 if for all input sinusoids, the output is 60 Hz hum!? What if the output is not even

* The author can assure you that some digital filters can "blow up." This occurs when the amplitude of a signal internal to the filter *overflows* and causes a prolonged "overflow oscillation". (1.1) cannot fail this badly, but it can cause "clipping" which is momentary overflow.

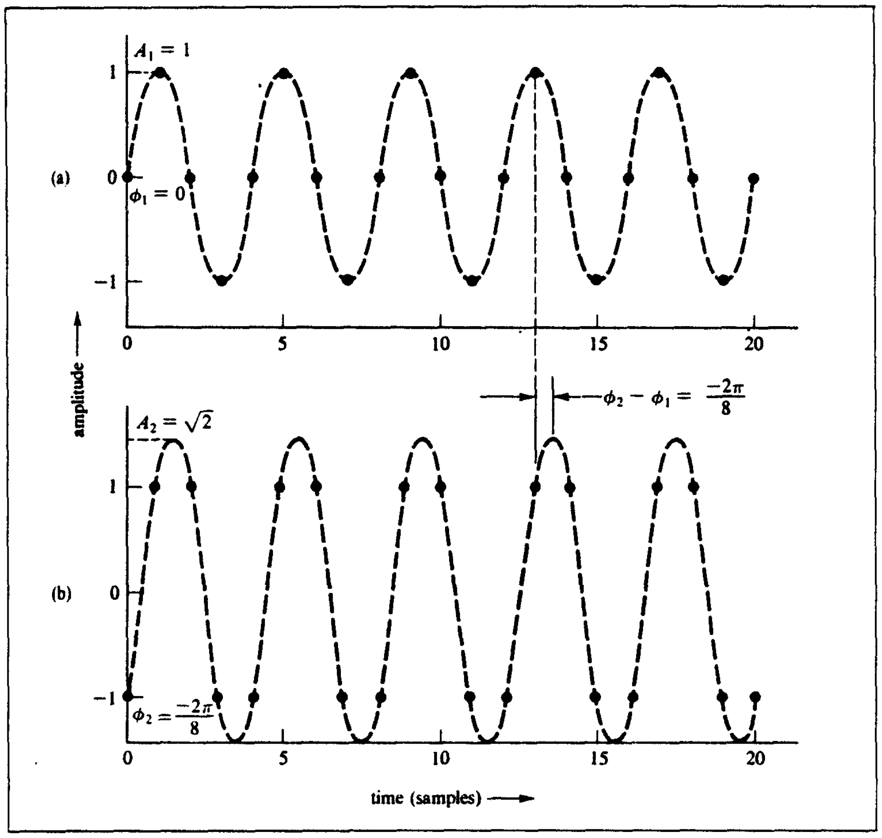

Figure Fig. 1.5. Input and output signals for the filter $y(n) = x(n) + x(n-1)$.
a) Input sinusoid $x(n) = A_1 \sin(\omega nT + \phi_1)$ at amplitude $A_1 = 1$, frequency $\omega = 2\pi f_s/4$, and phase $\phi_1 = 0$.
b) Output sinusoid $y(n) = A_2 \sin(\omega nT + \phi_2)$ at amplitude $A_2 = 1.414$, frequency $\omega = 2\pi f_s/4$, and phase $\phi_2 = -\pi/4$.

a sinusoid? The output $y(n)$ might be a cheeseburger for all we know. We will learn in chapter 2 that the sine-wave analysis procedure for measuring frequency response is guaranteed to work only if the filter is *linear* and *time-invariant*. Linearity means that the output due to a sum of input signals equals the sum of outputs due to each signal alone. Time-invariance means that the filter does not change over time. (Whatever you did to get a cheeseburger out of it yesterday will work today, now that you have witnesses assembled.) We will elaborate on these technical terms and their implications later. For now, just remember that such filters are guaranteed to produce a sinusoid in response to a sinusoid—and at the same frequency.

The above method of finding the frequency response involves physically measuring the amplitude and phase response for input sinusoids of every possible frequency. While this basic idea may be practical for a real black box, it is hardly useful for filter design. Ideally, we wish to arrive at a *formula* for the frequency response of the filter given by (1.1). There are several ways of doing this. The first we consider is exactly analogous to the sine-wave analysis procedure given above.

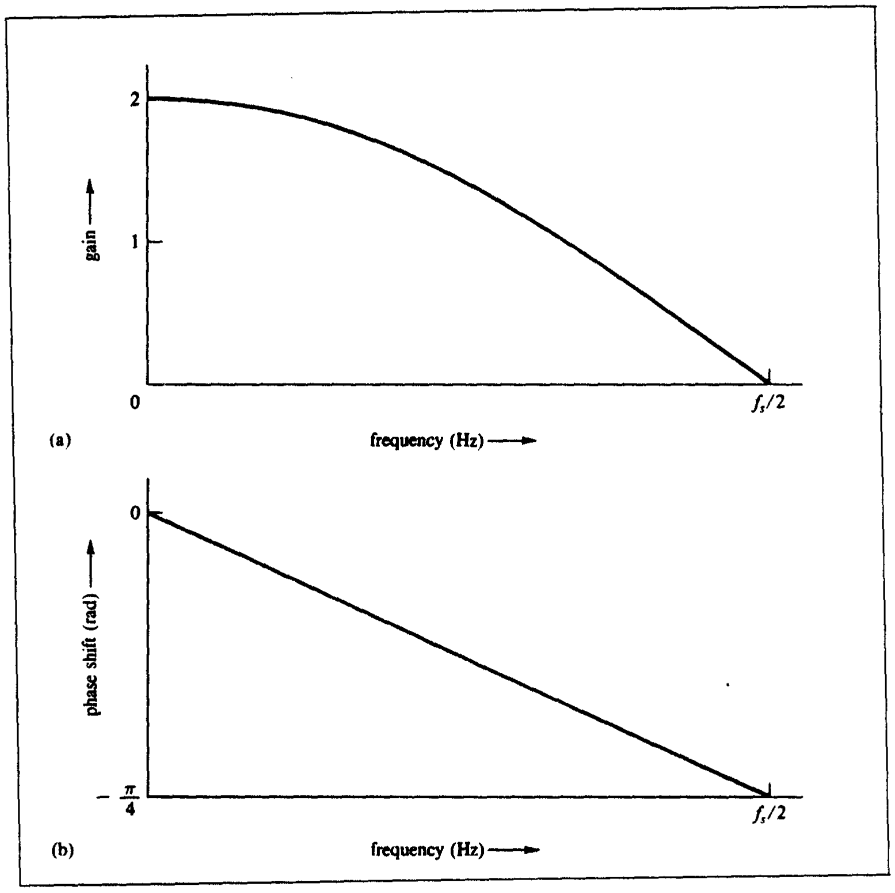

Figure 1.6. Frequency response for the filter $y(n) = x(n) + x(n-1)$.
a) Amplitude response.
b) Phase response.

Assuming (1.1) to be a linear time-invariant filter specification (which it is), let's take a few points in the frequency response by analytically "plugging in" sinusoids at a few different frequencies. Two graphs are required to fully represent the frequency response: gain versus frequency and phase shift versus frequency.

The frequency 0 Hz (often called "DC" for "Direct Current") is always comparatively easy to handle when analyzing a filter. Since plugging in a sinusoid means setting $x(n) = A\cos(2\pi f nT + \phi)$, by setting $f = 0$ we obtain $x(n) = A\cos[2\pi(0)nT + \phi] = A\cos(\phi)$ for all n. The input signal is then the same number $(A\cos\phi)$ over and over again for each sample. It should be clear that the filter output will be $y(n) = x(n) + x(n-1) = A\cos(\phi) + A\cos(\phi) = 2A\cos(\phi)$ for all n. Thus, the gain at frequency $f = 0$ is 2, which we get by dividing $2A$, the output amplitude, by A, the input amplitude. Phase is arbitrary at $f = 0$ Hz, so the phase response can be arbitrarily defined at this frequency. In all cases such as this, where the phase response may be arbitrarily defined, we choose a value which preserves continuity. This means we must analyze at frequencies in a neighborhood of the arbitrary point and take a limit. We will compute the phase response at DC later using different techniques. It is worth noting, however, that at 0 Hz, phase is almost always defined to be zero.

The next easiest frequency to look at is half the sampling rate, $f = f_s/2 = 1/(2T)$. In this case, by using our basic signal processing math and highschool trigonometry*, the *input* x can be simplified as follows:

$$\begin{aligned}
x(n) &= A\cos(2\pi \frac{f_s}{2} nT + \phi), \qquad n = 0,1,2,\ldots \\
&= A\cos(2\pi \frac{1}{2T} nT + \phi) \\
&= A\cos(\pi n + \phi) \\
&= A\cos(\pi n)\cos(\phi) - A\sin(\pi n)\sin(\phi) \\
&= A\cos(\pi n)\cos(\phi) \\
&= A(-1)^n\cos(\phi), \qquad n = 0,1,2,\ldots \\
&= \{A\cos(\phi), -A\cos(\phi), A\cos(\phi), -A\cos(\phi), \ldots\},
\end{aligned}$$

where the beginning of time was arbitrarily set at $n = 0$. Now with this input, the *output* of (1.1) is

$$\begin{aligned}
y(n) &= x(n) + x(n-1) \\
&= (-1)^n A\cos(\phi) + (-1)^{n-1} A\cos(\phi) \\
&= (-1)^n A\cos(\phi) - (-1)^n A\cos(\phi) \\
&= 0.
\end{aligned}$$

The filter of (1.1) thus has a gain of 0 at $f = f_s/2$. Again the phase is not measurable, since the input signal has been snuffed out altogether, and we will have to extrapolate the phase response from surrounding frequencies.

If we back off a bit, the above results for gain response are obvious without any trigonometry. The filter of (1.1) is equivalent (except for a factor of 2) to a simple two-point average, $y(n) = [x(n) + x(n-1)]/2$. Averaging adjacent samples in a signal is intuitively a lowpass filter because at low frequencies the sample amplitudes change slowly so that the average of two neighboring samples is very close to either sample, while at high frequencies the adjacent samples tend to have opposite sign and cancel out when added. The two extremes are frequency 0 Hz, at which the averaging has no effect, and half the sampling rate $f_s/2$, where the samples alternate in sign and exactly add to 0. (This kind of reasoning is good to go through in any analysis in order to eliminate gross mistakes. Sometimes we get two conflicting answers working a problem two different ways, and then we're thankful not to be designing airplanes or bridges.)

We are beginning to see that (1.1) may be a lowpass filter after all, since we found a boost of about 6 dB at the lowest frequency and a null at the highest frequency. (A gain of 2 may be expressed in decibels as $20\log_{10}(2) \approx 6\,\text{dB}$, and a *null* or *notch* is another term for a gain of 0 at a single frequency.) Of course we tried only two out of an infinite number of possible frequencies.

Let's go for broke and plug the general sinusoid into (1.1), confident that our savvy at trigonometry will see us through (after all, this is the simplest filter there is, right?). We let $x(n) = A\cos(2\pi fnT +$

* All the trigonometric identities used in the following derivation as well as in the rest of this paper are given in appendix B.

ϕ), so that the output is given by

$$y(n) = A\cos(2\pi fnT + \phi) + A\cos[2\pi f(n-1)T + \phi].$$

This input can be simplified as follows. Recall from the discussion surrounding Fig. 1.5 that only the peak-amplitude *ratio* and and the phase *difference* between input and output sinusoids are needed to measure the frequency response. The filter phase response does not depend on ϕ above (due to time-invariance), and so we can set ϕ to 0. Also, the filter amplitude response does not depend on A (due to linearity), so we can let $A = 1$. With these simplifications of $x(\cdot)$, the gain and phase response of the filter appear directly as the amplitude and phase of the output $y(\cdot)$. Thus, we input the signal

$$\begin{aligned} x(n) &= \cos(2\pi fnT) \\ &= \cos(\omega nT) \end{aligned}$$

where $\omega \triangleq 2\pi f$, as discussed in appendix A. (The symbol "$\triangleq$" means "is defined as".) With this input, the output of the simple lowpass filter is given by

$$y(n) = \cos(\omega nT) + \cos[\omega(n-1)T].$$

All that remains is to reduce the above expression to a single sinusoid with some frequency-dependent amplitude and phase. (The sum of sinusoids at the same frequency, but different phase and amplitude, can always be expressed as one sinusoid at that frequency with a new phase and amplitude [10].) This will be done first by using standard trigonometric identities [1] in order to avoid introducing complex numbers. Afterward, a much "easier" derivation using complex numbers will be given. We have

$$\begin{aligned} y(n) &= \cos(\omega nT) + \cos[\omega(n-1)T] \\ &= \cos(\omega nT) + \cos(\omega nT)\cos[-\omega T] - \sin(\omega nT)\sin[-\omega T] \\ &= \cos(\omega nT) + \cos(\omega nT)\cos(\omega T) + \sin(\omega nT)\sin(\omega T) \\ &= [1 + \cos(\omega T)]\cos(\omega nT) + \sin(\omega T)\sin(\omega nT) \\ &= a(\omega)\cos(\omega nT) + b(\omega)\sin(\omega nT). \end{aligned}$$

where $a(\omega) \triangleq [1 + \cos(\omega T)]$ and $b(\omega) \triangleq \sin(\omega T)$. We are looking for an answer of the form

$$y(n) = G(\omega)\cos[\omega nT + \Theta(\omega)],$$

where $G(\omega)$ is the filter gain response and $\Theta(\omega)$ is the phase response. This may be expanded as

$$y(n) = G(\omega)\cos[\Theta(\omega)]\cos(\omega nT) - G(\omega)\sin[\Theta(\omega)]\sin(\omega nT).$$

Therefore,

$$\begin{aligned} a(\omega) &= G(\omega)\cos[\Theta(\omega)] \\ b(\omega) &= -G(\omega)\sin[\Theta(\omega)], \end{aligned}$$

We can isolate the filter gain response $G(\omega)$ by squaring and adding the above two equations:

$$\begin{aligned} a^2(\omega) + b^2(\omega) &= G^2(\omega)\cos^2[\Theta(\omega)] \; + \; G^2(\omega)\sin^2[\Theta(\omega)] \\ &= G^2(\omega)\{\cos^2[\Theta(\omega)] \; + \; \sin^2[\Theta(\omega)]\} \\ &= G^2(\omega) \end{aligned}$$

This can then be simplified as follows:

$$\begin{aligned} G^2(\omega) &= a^2(\omega) + b^2(\omega) = [1+\cos(\omega T)]^2 + \sin^2(\omega T) \\ &= 1 + 2\cos(\omega T) + \cos^2(\omega T) + \sin^2(\omega T) \\ &= 2 + 2\cos(\omega T) = 4\cos^2(\omega T/2) \end{aligned}$$

So we have made it to the amplitude response,

$$G(\omega) = 2|\cos(\omega T/2)| = 2\cos(\pi f T), \qquad |f| \le \frac{f_s}{2}.$$

Since $\cos(\pi f T)$ is nonnegative for $-f_s/2 \le f \le f_s/2$ (verify this), it is unnecessary to take the absolute value as long as f is understood to lie in this range.

Now we isolate the filter phase response $\Theta(\omega)$ by taking a ratio of the same two equations used to isolate $G(\omega)$ and noting that $\Theta(\omega)$ appears trapped inside a tangent.

$$\begin{aligned} \tan[\Theta(\omega)] &= -\frac{b(\omega)}{a(\omega)} \\ &= -\frac{\sin(\omega T)}{1+\cos(\omega T)} = -\frac{2\sin(\omega T/2)\cos(\omega T/2)}{1+\cos^2(\omega T/2) - \sin^2(\omega T/2)} \\ &= -\frac{2\sin(\omega T/2)\cos(\omega T/2)}{2\cos^2(\omega T/2)} \\ &= -\frac{\sin(\omega T/2)}{\cos(\omega T/2)} = -\tan[\omega T/2] = \tan[-\omega T/2] \end{aligned}$$

Thus, the filter phase response is

$$\Theta(\omega) = -\omega T/2 = -\pi f T.$$

We have completely solved for the frequency response of the simplest lowpass filter given in (1.1) using only trigonometry. We found that an input sinusoid of the form

$$x(n) = A\cos(2\pi f n T + \phi)$$

produces the output

$$y(n) = 2A\cos(\pi f T)\cos[2\pi f n T + \phi - \pi f T].$$

Thus, the gain versus frequency is $2\cos(\pi f T)$ and the change in phase at each frequency is given by $-\pi f T$ radians. These functions are shown in 1.6.

An Easier Way

Fortunately, there are much more effective (and less arduous) methods for finding the gain and phase-shift as a function of frequency. The first step on the road to Utopia in signal analysis is to understand *Euler's identity*:

$$e^{j\omega n T} = \cos(\omega n T) + j\sin(\omega n T), \quad j = \sqrt{-1} \tag{1.2}$$

I shall refer to $e^{j\omega nT}$ as the *complex sinusoid*. We will see that it is easier to manipulate both *sine* and *cosine* simultaneously in this form than it is to deal with either *sine* or *cosine* separately. One may take the point of view that $e^{j\theta}$ is *simpler* and *more fundamental* than $\sin(\theta)$ or $\cos(\theta)$, as evidenced by the following identities (which follow immediately from (1.2)):

$$\cos\theta = \frac{e^{j\theta}+e^{-j\theta}}{2} \tag{1.3 a}$$

$$\sin\theta = \frac{e^{j\theta}-e^{-j\theta}}{2j}. \tag{1.3 b}$$

Thus, sine and cosine are each regarded as a combination of two complex sinusoids. Another reason for the success of the complex sinusoid is that we will be concerned only with real *linear* operations on signals. This means that j in (1.3) will never be multiplied by j or raised to a power by a linear filter with real coefficients. Therefore, the real and imaginary parts of (1.3) are actually treated independently. Thus, we can feed a complex sinusoid into a filter, and the real part of the output will be the *cosine* response, and the imaginary part of the output will be the *sine* response. Some of you may be thinking questions like "Why e? Where did the imaginary exponent come from? Are imaginary exponents legal?" The interested reader is referred to [4] for the real answers (which can be complex). Here, we will look only at some intuitive connections between complex sinusoids and the more familiar real sinusoids.

Figure 1.7 shows Euler's relation graphically as it applies to sinusoids. A point traveling with uniform velocity around a circle with radius 1 may be represented by $e^{j\omega t} = e^{j2\pi ft}$ where t is time and f is the number of revolutions per second. The projection of this motion onto the horizontal (real) axis is $\cos(\omega t)$ and the projection onto the vertical (imaginary) axis is $\sin(\omega t)$. For discrete-time circular motion, replace t by nT to get $e^{j\omega nT} = e^{j2\pi fnT} = e^{j2\pi(f/f_s)n}$ which may be interpreted as a point which jumps an arc-length $2\pi f/f_s$ along the circle each sampling instant.

Euler's identity says that circular motion is the vector sum of two sinusoidal motions.

For circular motion to ensue, the sinusoidal motions must be at the same frequency, one-quarter cycle out of phase, and along perpendicular lines. (With phase differences other than one-quarter cycle, the motion is generally elliptical.)

The converse of this is also illuminating. Take the usual circular motion $e^{j\omega t}$ which spins counterclockwise, and add to it a similar but clockwise circular motion, $e^{-j\omega t}$. This is shown in Figure 1.8. Next apply Euler's identity to get

$$\begin{aligned} e^{j\omega t}+e^{-j\omega t} &= \cos(\omega t)+j\sin(\omega t)+\cos(-\omega t)+j\sin(-\omega t)\\ &= \cos(\omega t)+j\sin(\omega t)+\cos(\omega t)-j\sin(\omega t)\\ &= 2\cos(\omega t). \end{aligned}$$

Thus,

a cosine is the vector sum of two circular motions with the same angular speed but opposite direction.

This statement is a graphical or geometric interpretation of (1.3a). A similar derivation (subtracting instead of adding) gives the *sine* identity (1.3b).

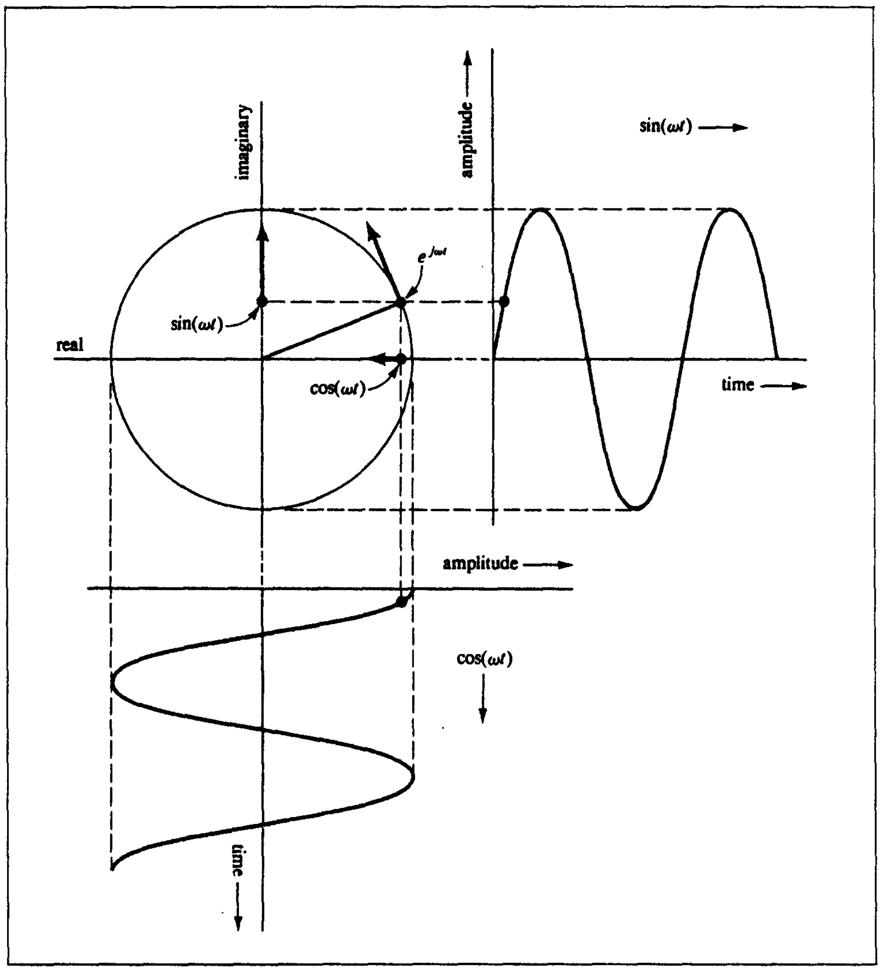

Figure 1.7. Relation of uniform circular motion to sinusoidal motion via Euler's identity, $e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$. The projection of $e^{j\omega t}$ onto the real axis is $\cos(\omega t)$, and the projection onto the imaginary axis is $\sin(\omega t)$.

We call $e^{j\omega nT}$ a *positive-frequency sinusoidal component* when $\omega > 0$, and $e^{-j\omega nT}$ is the corresponding *negative-frequency* component. Note that both sine and cosine have equal-amplitude positive- and negative-frequency components (cf. [10]). This happens to be true of every real signal. To see this, recall that every signal can be represented as a sum of complex sinusoids at various frequencies (the so-called *Fourier expansion*). For the signal to be real, every positive-frequency complex sinusoid must be summed with a negative-frequency sinusoid of equal amplitude. In other words, any counterclockwise circular motion must be matched by an equal and opposite clockwise circular motion in order that the imaginary parts always cancel to yield a real signal (see Fig. 1.8). Thus, a real signal always has a symmetric magnitude spectrum.

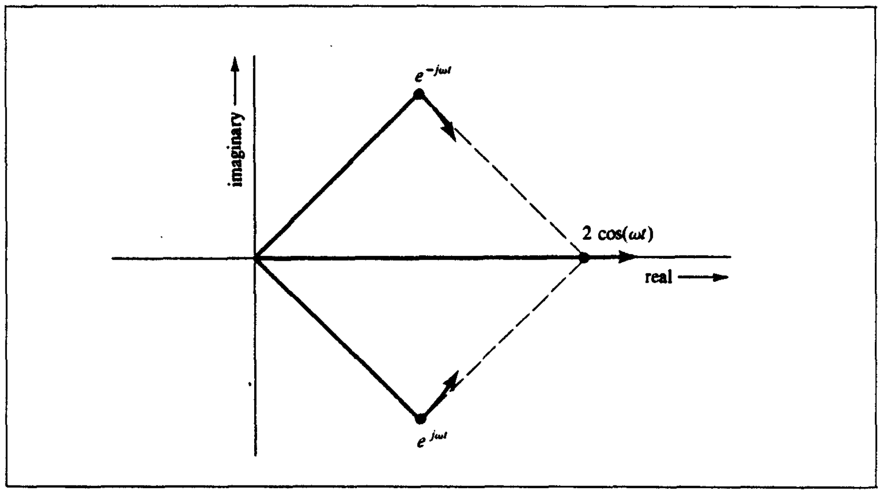

Figure 1.8. Illustration of the way in which opposite circular motions add to give real sinusoidal motion, $e^{j\omega t} + e^{-j\omega t} = 2\cos(\omega t)$.

The Solution

Now that we have a new tool, let's apply it to the simplest lowpass filter given by (1.1). We test the filter response at frequency f by letting $x(n) = Ae^{j(2\pi fnT+\phi)}$. Again, because of time-invariance, the frequency response will not depend on ϕ, so let $\phi = 0$. Similarly, due to linearity, we may normalize A to 1. By virtue of Euler's relation and the linearity of the filter, setting the input to $x(n) = e^{j\omega nT}$ is physically equivalent to putting $\cos(\omega nT)$ into one copy of the filter, and $\sin(\omega nT)$ into a separate copy of the same filter. The signal path where the *cosine* goes in is the *real* part of the signal, and the other signal path is simply *called* the *imaginary* part. Thus, a complex signal in real life is just two ordinary signals side by side, one-quarter cycle out of phase at each frequency.

Using the normal rules for manipulating exponents, we find that the response of the simple lowpass filter to the complex sinusoid at frequency $\omega/2\pi$ Hz is given by

$$\begin{aligned} y(n) &= x(n) + x(n-1) \\ &= e^{j\omega nT} + e^{j\omega(n-1)T} \\ &= e^{j\omega nT} + e^{j\omega nT}e^{-j\omega T} \\ &= \left(1 + e^{-j\omega T}\right)e^{j\omega nT} \\ &= \left(1 + e^{-j\omega T}\right)x(n) \\ &\triangleq H(e^{j\omega T})x(n) \end{aligned} \tag{1.4}$$

This derivation is clearly easier than the trigonometry approach which used tricky identities that few people would think of without staring at the answer for a good while.

What is a little puzzling, however, is that the filter ends up looking like a frequency-dependent complex multiply. What does this mean? Well the theory that we are blindly trusting at this point says it must somehow mean a gain scaling and a phase shift. This is true and easy to see once the complex filter gain is expressed in *polar form*, $H(e^{j\omega T}) = G(\omega)e^{j\Theta(\omega)}$. The gain versus frequency is given by the absolute value of H (the absolute value is just the radius in polar coordinates), and the phase shift in radians versus frequency is given by the complex angle of H (angle in polar coordinates). In other words, we must find

$$G(\omega) \triangleq \left| H(e^{j\omega T}) \right|$$

which is the amplitude response, and

$$\Theta(\omega) \triangleq \angle H(e^{j\omega T})$$

which is the phase response. There is a trick I call "balancing the exponents" which will work nicely for the simple lowpass of (1.1).

$$\begin{aligned} H(e^{j\omega T}) &= \left(1 + e^{-j\omega T}\right) \\ &= \left(e^{j\omega T/2} + e^{-j\omega T/2}\right)e^{-j\omega T/2} \\ &= 2\cos(\omega T/2)e^{-j\omega T/2} \end{aligned}$$

It is now easy to see that

$$\begin{aligned} G(\omega) &= \left| 2\cos(\omega T/2)e^{-j\omega T/2} \right| \\ &= 2|\cos(\omega T/2)| \\ &= 2\cos(\omega T/2), \quad |f| \leq \frac{f_s}{2}, \end{aligned}$$

and

$$\Theta(\omega) = -\omega T/2, \quad |f| \leq \frac{f_s}{2}.$$

We have derived again the graph of 1.6 which shows the complete frequency response of (1.1). The gain of the simplest lowpass filter varies, as cosine varies, from 1 to 0 as the frequency of an input sinusoid goes from 0 to half the sampling rate. In other words, the amplitude response of (1.1) goes sinusoidally from 1 to 0 as ωT goes from 0 to π. It does seem somewhat reasonable to consider it a lowpass, and it *is* a poor one in the sense that it is hard to see which frequency should be called the cutoff frequency. We see that the spectral "rolloff" is very slow, as lowpass filters go, and this is what we pay for the extreme simplicity of (1.1). The phase response $\Theta(\omega) = -\omega T/2$ is linear in frequency, which gives rise to a constant time delay irrespective of the signal frequency (see problem 3).

It deserves to be emphasized that all a linear time-invariant filter can do to a sinusoid is scale its amplitude and change its phase. Since a sinusoid is completely determined by its amplitude A, frequency f, and phase ϕ, the constraint on the filter is that the output must also be a sinusoid, and furthermore it must be *at the same frequency* as the input sinusoid. More explicitly:

If a sinusoid $A_1 \cos(\omega nT + \phi_1)$ is input to a linear time-invariant filter, then the output signal (after start-up transients have died away) will be a sinusoid at the same frequency $A_2 \cos(\omega nT + \phi_2)$. The only possible differences between the input and output are in their relative amplitude and relative phase. Any linear time-invariant filter may thus be completely characterized by its gain A_2/A_1 and phase $\phi_2 - \phi_1$ at each frequency.

Mathematically, a sinusoid has no beginning and no end, so there really are no start-up transients in the theoretical setting. However, in practice, we must approximate eternal sinusoids with finite-time sinusoids whose starting time was so long ago that the filter output is essentially the same as if the input had been applied forever.

Tying it all together, the general output of a linear time-invariant filter with a complex sinusoidal input may be expressed as

$$\begin{aligned} y(n) &= (Complex\,Filter\,Gain) times (Input\,Circular\,Motion\,with\,Radius\,A, Phase\,\phi) \\ &= \left(G(\omega)e^{j\Theta(\omega)}\right)\left(Ae^{j(\omega nT+\phi)}\right) \\ &= (G(\omega)A)e^{j\{\omega nT + [\phi + \Theta(\omega)]\}} \\ &= Circular\,Motion\,with\,Radius\,G(\omega)A,\;Phase\,\phi + \Theta(\omega) \end{aligned}$$

In Summary

We have introduced many of the concepts associated with digital filters such as signal representation, filter representation, and determination of frequency response. We used a simple filter example to motivate the need for more advanced methods to analyze digital filters of arbitrary complexity. We found even in the simple example of (1.1) that complex variables are much more compact and convenient for representing signals and analyzing filters than trigonometric techniques. We employ a complex-valued function of time $Ae^{j(\omega nT+\phi)}$ to represent the amplitude and phase of a complex sinusoid, and we use a complex-valued function of frequency $G(\omega)e^{j\Theta(\omega)}$ to represent the gain and phase response of a linear time-invariant filter,

In chapter 2, we will visit the basic facts necessary to understand the analysis of the general linear time-invariant filter. Time-invariance is not overly restrictive because the static analysis holds very well for filters that change slowly with time. Nor is linearity very restrictive, since, for example, all the filters mentioned in the introduction are linear or nearly so. In other words, the techniques to be discussed apply to the understanding and design of almost all useful audio filters.

1.2. Problems

1. *General Amplitude and Phase in Deriving Frequency Response*

Repeat the derivation of (1.4) without the simplifications $A = 1$, $\phi = 0$. That is, let

$$x(n) = Ae^{j(\omega nT+\phi)}$$

and find $y(n)$ from (1.1). Verify that the frequency response is the same.

2. *A Modification of the Simplest Lowpass Filter*

Consider the filter

$$y(n) = x(n) - x(n-1)$$

which is identical to (1.1) except that adjacent input samples are *subtracted* rather than added. Derive the amplitude response and the phase response. How has the response changed? Would you call this a lowpass filter, a highpass filter, or something else?

3. *Linear Phase means Simple Waveform Delay*

Show that the phase response $\Theta(\omega) = -\omega T/2$ obtained for (1.1) corresponds to a waveform delay of one-half sample ($T/2$ sec) at all frequencies.

4. *Complex Numbers and Trigonometry*

Show how easy it is to derive the identities

$$\begin{aligned}\cos(a+b) &= \cos(a)\cos(b) - \sin(a)\sin(b)\\ \sin(a+b) &= \sin(a)\cos(b) + \cos(a)\sin(b)\end{aligned}$$

by using Euler's identity and the formula

$$e^{j(a+b)} = e^{ja}e^{jb}.$$

How would you derive the $\cos(a+b)$ identity, say, if you did not know the answer and did not use complex numbers?

5. *Filters with Coefficients*

The analysis for (1.1) will work for a larger class of filters. Try to get the frequency response for the filter

$$y(n) = ax(n) - ax(n-2)$$

where a is a constant but arbitrary gain. Plot the amplitude response $G(\omega)$. Would you call this a lowpass filter, a highpass filter, or something else entirely? What is the effect on $G(\omega)$ when a is changed? How does the phase response $\Theta(\omega)$ depend on a?

Chapter 2

Theoretical Foundations of Digital Filters

In this chapter we explore more fully the implications of *linearity* and *time-invariance*. Four basic representations of digital filters are defined—the *difference equation coefficients, impulse response, transfer function,* and *frequency response*. In addition, the concepts of phase delay, group delay, poles and zeros, and filter stability are introduced.

2.1. Linearity and Time-Invariance

Almost all filters of interest are *linear*. Linearity is extremely important for several reasons. From a system theory point of view, there is a lot of useful analysis and general design theory for linear filters that can say nothing about nonlinear filters. From a sound processing point of view, linearity is important because it means that no new spectral components are introduced by the filter (unless the filter is rapidly time-varying, which causes spectral changes not considered here). Nonlinear filters can (and probably will) produce new signal components at all the possible sums and differences of the frequencies present in the input signal. This includes both harmonic and intermodulation distortion. A truly linear filter can have neither type of distortion. *All the examples of filters mentioned in chapter 1 were linear, or approximately linear.* In addition, the z transform and all variations of the Fourier transform are linear.

In everyday terms, the fact that a filter is linear means simply that

L1) *the amplitude of the output is proportional to the amplitude of the input,*

and

L2) *when two signals are added together and fed to the filter, the filter output is the same as if one had put each signal through the filter separately and then added the outputs.*

This second property of linear systems is known as *superposition*. The response of a linear system to a sum of signals is the sum of the responses to each individual input signal. Another view is that the individual signals which have been summed at the input are processed independently inside the filter—i.e., they superimpose and do not interact. (The addition of two signals, sample by sample, is like converting stereo to mono by equally mixing the two channels together.) Another example of a linear signal medium is the Earth's atmosphere. When two sounds are in the air at once, the air pressure fluctuations that convey them simply add (unless they are very loud). Since any finite continuous signal can be represented as a sum (or superposition) of sinusoids, we can predict the filter response to any input signal just by knowing the response for all sinusoids. Without superposition, we have no such general description, and it may be impossible to do any better than to catalog the filter output for each possible input.

While the implications of linearity are far-reaching, the mathematical definition is simple. Let us represent the general linear *time-varying* filter by

$$y(n) = \mathcal{L}_n\{x(\cdot)\} \tag{2.1}$$

where $x(\cdot)$ is the entire input signal, $y(n)$ is the output at time n, and $\mathcal{L}_n$ is the filter expressed as a *real-valued function of a signal* for each n. Think of the subscript n on $\mathcal{L}_n$ as selecting the n^{th} output sample of the filter. In general, *each* output sample can be a function of several or even *all* input samples, and this is why we write $x(\cdot)$ as the filter input. Furthermore, the function $\mathcal{L}_n$ can be different for each n (in the time-varying case only). Now, if for any pair of signals $x_1(\cdot), x_2(\cdot)$ and for all constant gains g, we have

$$\begin{aligned} &L1) \quad \mathcal{L}_n\{gx_1(\cdot)\} = g\mathcal{L}_n\{x_1(\cdot)\} \\ &L2) \quad \mathcal{L}_n\{x_1(\cdot)+x_2(\cdot)\} = \mathcal{L}_n\{x_1(\cdot)\} + \mathcal{L}_n\{x_2(\cdot)\}, \end{aligned} \tag{2.2}$$

for all n, then the filter is said to be *linear*. These two restrictions simply restate, in mathematical notation, the definition previously given in everyday language. Actually, for all practical purposes, property 2) alone is enough to guarantee linearity (see problem 6).

In plain terms, a *time-invariant filter* is one which performs the same operation at all times. It is awkward to express this mathematically by restrictions on (2.1) because of the use of $x(\cdot)$ as the symbol for the filter input. What we want to say is that if the input signal is delayed by, say, N samples, then the output waveform is the same, but also delayed by N samples. Thus, $y(\cdot)$, the output waveform of a time-invariant filter, merely shifts forward or backward in time as the input waveform $x(\cdot)$ is shifted forward or backward in time.

Imagine the formula $\mathcal{L}_n$ as a big brain which reaches out to $x(\cdot)$ with long tentacles, one for each sample. At each time n it utters the output $y(n)$. Then for time-invariant filters the brain is fixed while the tentacles shift. That is for each increase of n by 1, the tentacles shift over one sample, and the brain repeats the same operation on the data it receives from the tentacles.

Another way of expressing time invariance is to say that a filter is time invariant if and only if it commutes with any delay line.

For completeness, the condition on (2.1) which defines time-invariance is given by

$$\mathcal{L}_n\{x(\cdot - N)\} = \mathcal{L}_{n-N}\{x(\cdot)\} = y(n-N), \tag{2.3}$$

where $x(\cdot - N)$ is understood to denote the waveform $x(\cdot)$ shifted right (or delayed) by N samples.

A simple example of a *nonlinear* filtering operation is *compression* in which the average output energy is made nearly constant. Compression works by automatically turning up the volume (increasing gain) when the signal level (average amplitude) is low, and turning it down when the level is high, so as to keep the "loudness" almost constant.

Now, why is compression nonlinear? Multiplying a signal by a constant gain ("volume control") is a linear operation; i.e., for an arbitrary pair of signals x_1, x_2, and for arbitrary constants α, β, it is true that

$$g \cdot [\alpha \cdot x_1(n) + \beta \cdot x_2(n)] = \alpha \cdot [g \cdot x_1(n)] + \beta \cdot [g \cdot x_2(n)].$$

Thus, both L1) and L2) are satisfied. (We have used " $\cdot$ " to explicitly indicate multiplication here, so as to avoid confusion between "A *times* $(B+C)$" and "A *of* $(B+C)$.") However, compression is a nonlinear operation because the gain g, which multiplies the input, depends on the input signal amplitude. This happens because the compressor must estimate the current signal level in order to normalize it. Thus, compression is a filter of the form

$$y(n) = g(x) \cdot x(n),$$

where $g(x)$ denotes a gain that depends on the "current amplitude" of x.* In general,

$$g(x_1+x_2)\cdot[x_1(n)+x_2(n)] \neq g(x_1)\cdot x_1(n)+g(x_2)\cdot x_2(n).$$

That is, the compression of the sum of two signals is not generally the same as the addition of the two signals compressed individually. Therefore, condition L2) of linearity fails. It is also clear that condition L1) fails, since the output level is independent of the input level for an ideal compressor.

Generally, almost any signal operation that includes a multiplication in which both multiplicands depend on the input signal can be shown to be nonlinear. Note, however, that g may vary with time *independently* of x to yield a *linear time-varying filter.* In this case, linearity is demonstrated by writing

$$g(n)\cdot[\alpha\cdot x_1(n)+\beta\cdot x_2(n)] = \alpha\cdot[g(n)\cdot x_1(n)]+\beta\cdot[g(n)\cdot x_2(n)]$$

to show that both L1) and L2) are satisfied. A simple example of a *linear time-varying* filter is a *tremolo* function which can be written as a time-varying gain, $y(n)=g(n)x(n)$. For example, $g(n)=1+\cos(2\pi(4)nT)$ would give a maximally deep tremolo with four swells per second.

From now on, all filters discussed will be linear and time-invariant. For brevity, these will be referred to as *LTI filters.*

2.2. Representations of a Digital Filter

For linear time-invariant (LTI) filters, just as for signals, there are two fundamental domains of representation: the *time domain* and the *frequency domain.* However, there are at least *two* time-domain and *two* frequency-domain representations for a digital filter, and they can all be converted into one another. The two time-domain representations we'll discuss are the *difference equation coefficients* and the *impulse response*, and the two principal frequency-domain forms are the *transfer function* and the *frequency response.*

2.2.1. Difference Equation

The *difference equation* gives a time-domain representation because it tells how to calculate the filter output samples, given the filter input samples. We may write the general case as follows.

$$\begin{aligned} y(n) = a_0x(n)+a_1x(n-1)+\cdots+a_Mx(n-M) \\ -b_1y(n-1)-\cdots-b_Ny(n-N) \end{aligned} \tag{2.4}$$

where x is the input signal, y is the output signal, and the constants $\{a_i,\ i=0,1,2,\ldots,M\}$, $\{b_i,\ i=1,2,\ldots,N\}$ are called *difference equation coefficients* or, more simply, *filter coefficients.* When the a and b coefficients are real numbers, the filter is said to be *real.* All filters discussed in this paper are real.

For example, the difference equation

$$y(n)=0.01x(n)+0.002x(n-1)+0.99y(n-1)$$

* Since many successive samples of x are needed to estimate current amplitude, we cannot correctly write $g[x(n)]$ for the gain function. The most accurate notation is $g_n[x(\cdot)]$, which is too general and cumbersome for present purposes.

specifies a digital filtering operation, and the coefficient sets $\{0.01, 0.002\}$ and $\{0.99\}$ represent the filter. In this example, we have $M = N = 1$.

Equation (2.4) is not entirely general, for it represents only *causal* LTI filters. A filter is said to be *causal* when the output does not depend on any "future" inputs. (In more colorful terms, a filter is causal if it does not "laugh" before it is "tickled.") For example, $y(n) = x(n+1)$ is a noncausal filter because the output anticipates the input one sample before it arrives. Restriction to causal filters is quite natural in real-time situations. Many digital filters, on the other hand, are implemented on a computer where time is artificially represented by an array index. Thus, noncausal filters present no difficulty in an "off-line" situation. It happens that the analysis for noncausal filters is pretty much the same as that for causal filters, so we haven't given up too much with this restriction.

The *maximum time span*, in samples, used in creating each output sample is called the *order* of the filter. In this case the order is the larger of M and N in (2.4). For example, $y(n) = x(n) - x(n-1) - 2y(n-1) + y(n-2)$ specifies a particular *second-order* filter.

If M and N in (2.4) are constrained to be finite (which is, of course, necessary in practice), then (2.4) represents the class of *finite order* causal LTI filters.

Notice that a filter of the form (2.4) can use *"past" output samples* (such as $y(n-1)$) in the calculation of the "present" output $y(n)$. This use of past output samples is called *feedback*. Any filter having one or more feedback paths ($N > 0$) is called *recursive*. (By the way, the minus signs for the feedback in (2.4) are supposed to be mysterious until we get to transfer functions.)

(2.4) may be expressed in different but entirely equivalent notation by

$$y(n) = \sum_{i=0}^{M} a_i x(n-i) - \sum_{j=1}^{N} b_j y(n-j). \tag{2.4}$$

One possible *system diagram* of (2.4) is given in Figure 2.1a for the case $M = 2$, $N = 3$. Hopefully it is easy to see how this diagram represents the difference equation. Soon, we will have sufficient knowledge to see that the filter specified in Fig. 2.1b is exactly equivalent.

2.2.2. Impulse Response and Convolution

In addition to difference equation coefficients, any LTI filter may be represented in the time domain by its response to a specific signal called the *impulse*. The impulse is denoted as $\delta(n)$ and is defined by

$$\delta(n) \triangleq \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}.$$

A plot of $\delta(\cdot)$ is given in Figure 2.2a. In real life, an impulse may be approximated by a swift hammer blow, for example.

We also have a special notation for the impulse *response* of a filter, namely,

$$h(n) \triangleq \mathcal{L}_n\{\delta(\cdot)\},$$

which, in words, is the response of the filter at time n to an impulse occurring at time 0. We assume in this section that the impulse response $h(n)$ approaches zero as n goes to infinity. Such a filter is said to be *stable*.

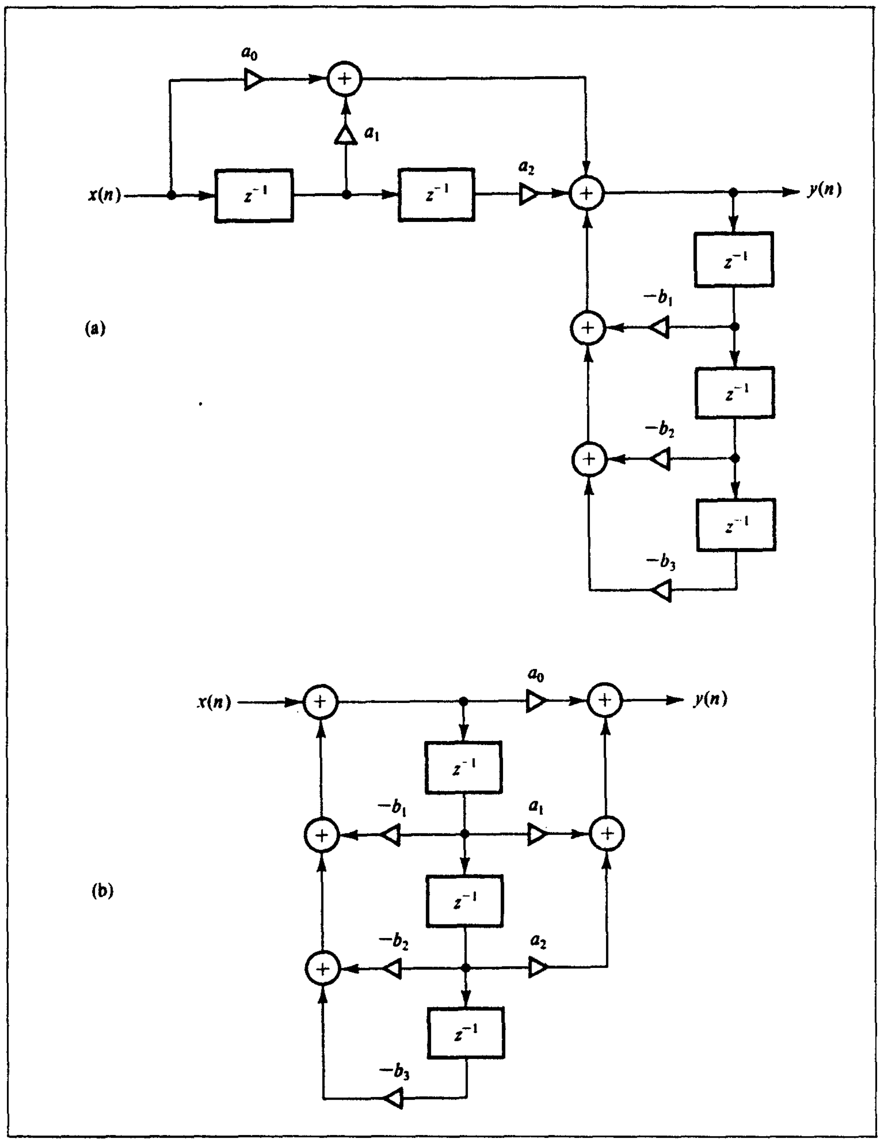

Figure 2.1. System diagram for the filter $y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) - b_1y(n-1) - b_2y(n-2) - b_3y(n-3)$.
a) Direct system diagram.
b) Equivalent system diagram using shared delays.

An example impulse response for the first-order recursive filter

$$y(n) = x(n) + 0.9y(n-1) \tag{2.5}$$

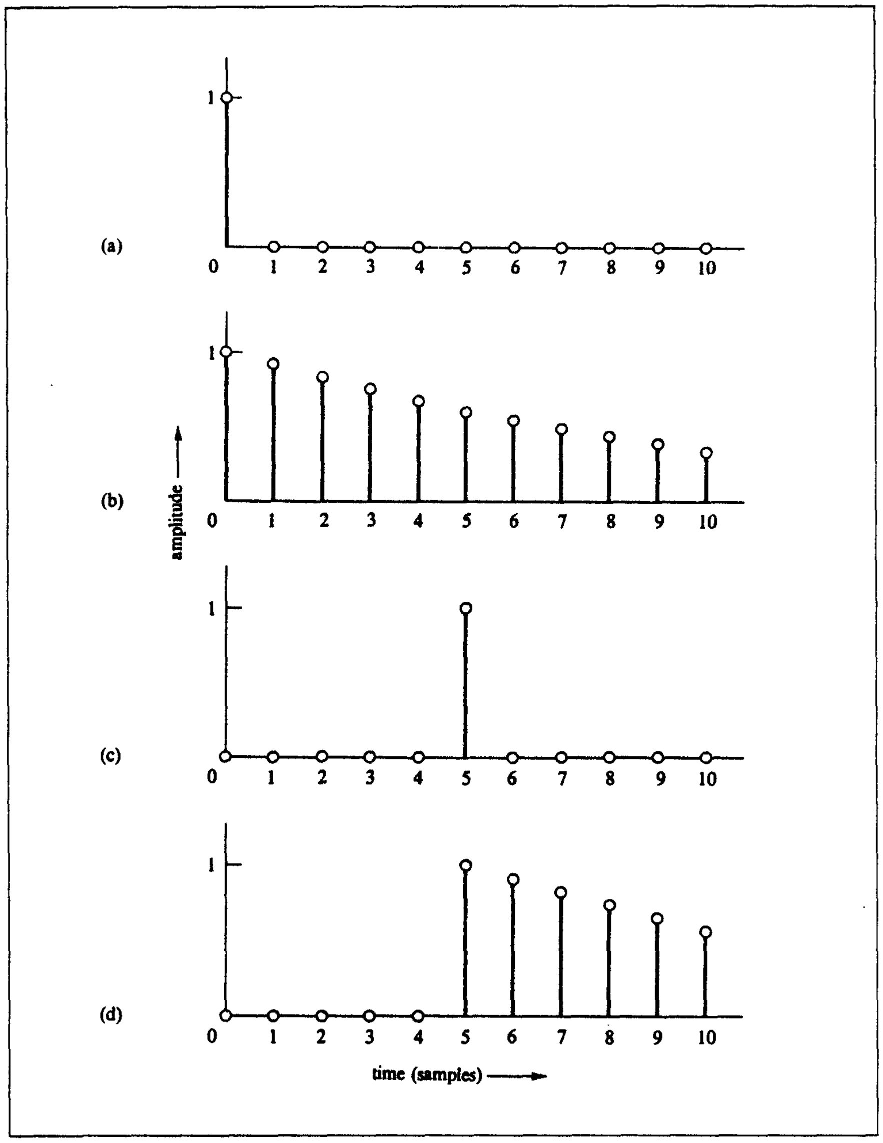

Figure 2.2. Input and output signals for the filter $y(n) = x(n) + 0.9y(n-1)$.
a) Input impulse $\delta(n)$.
b) Output impulse response $h(n)$.
c) Input delayed-impulse $\delta(n-5)$.
d) Output delayed-impulse response $h(n-5)$.

is shown in 2.2b. The impulse response is a simple exponential decay, $\{1, 0.9, 0.81, 0.73, \ldots\}$ or, more formally,

$$h(n) = \begin{cases} (0.9)^n, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

The reason no information is lost in representing an arbitrary LTI filter by its impulse response lies in the fact that the impulse contains energy at all frequencies. To see this we simply observe that the z transform of $\delta(n)$ is 1 (verify). Consequently, the impulse response contains the superposition of the response of the filter to every possible sinusoidal frequency from $-f_s/2$ to $f_s/2$ Hz.

Using the basic properties of linearity and time-invariance, we will derive the *convolution formula* which gives an algorithm for implementing the filter directly in terms of the impulse response. In other words, *convolving the input signal with the filter impulse response gives the filtered output.* The convolution formula plays the role of the difference equation when the impulse response is used in place of the difference equation coefficients as a filter representation.

An outline of the derivation of the convolution formula is as follows: Any signal $x(n)$ may be regarded as a superposition of impulses at various amplitudes and arrival times, i.e., each sample of $x(n)$ is regarded as an impulse with amplitude $x(n)$ and delay n. By the superposition principle for LTI filters, the filter output is simply the superposition of impulse *responses*, each having a scale factor and time-shift given by the amplitude and time-shift of the corresponding input impulse.

Before we proceed to the general case, a simple example will serve to illustrate how these basic mechanisms are expressed mathematically. If an impulse strikes at time $n = 5$ rather than at time $n = 0$, this is represented by writing $\delta(n-5)$. A picture of this delayed impulse is given in 2.2c. When $\delta(n-5)$ is fed to a time-invariant filter, the output will be the impulse response $h(n)$ delayed by 5 samples, or $h(n-5)$. 2.2d shows the response of the example filter of (2.5) to the delayed impulse $\delta(n-5)$.

In the general case, for time-invariant filters we may write

$$\mathcal{L}_n\{\delta(\cdot - N)\} = \mathcal{L}_{n-N}\{\delta(\cdot)\} = h(n-N),$$

where N is the number of samples delay. This equation states that right-shifting the input impulse by N points merely right-shifts the output (impulse response) by N points. Note that this is just a special case of the definition of time-invariance, (2.3).

If *two* impulses arrive at the filter input, the first at time $n = 0$, say, and the second at time $n = 5$, then this input may be expressed as $\delta(n) + \delta(n-5)$. If, in addition, the amplitude of the first impulse is 2, while the second impulse has an amplitude of 1, then the input may be written as $2\delta(n) + \delta(n-5)$. In this case, using *linearity* as well as time-invariance, the response of the general LTI filter to this input may be expressed as

$$\begin{aligned} \mathcal{L}_n\{2\delta(\cdot) + \delta(\cdot - 5)\} &= \mathcal{L}_n\{2\delta(\cdot)\} + \mathcal{L}_n\{\delta(\cdot - 5)\} \\ &= 2\mathcal{L}_n\{\delta(\cdot)\} + \mathcal{L}_{n-5}\{\delta(\cdot)\} \\ &= 2h(n) + h(n-5). \end{aligned}$$

For the example filter of (2.5), given the input $2\delta(n) + \delta(n-5)$ (pictured in Figure 2.3a), the output may be computed by scaling, shifting, and adding together copies of the impulse response $h(n)$. That is, taking the impulse response in 2.2b, multiplying it by 2, and adding it to the delayed impulse response in 2.2d, we obtain the output

$$2h(n) + h(n-5) = \begin{cases} 2(0.9)^n + (0.9)^{n-5}, & n \geq 5 \\ 2(0.9)^n, & 0 \leq n < 5 \\ 0, & n < 0 \end{cases}$$

as shown in Figure 2.3b. Thus, a weighted sum of impulses produces the same weighted sum of impulse responses.

Convolution

We are now ready to derive the convolution formula in its complete generality. The first step is to express an arbitrary signal $x(\cdot)$ as a linear function of shifted impulses, i.e.,

$$x(n) = x * \delta(n) \triangleq \sum_{i=-\infty}^{\infty} x(i)\delta(n-i). \tag{2.6}$$

where "$*$" stands for convolution.

If the above equation is not obvious, here is how it is built up intuitively. Imagine $\delta(\cdot)$ as a 1 in the midst of an infinite string of 0's. Now think of $\delta(\cdot - i)$ as the same pattern shifted over to the right by i samples. Next multiply $\delta(\cdot - i)$ by $x(\cdot)$, which plucks out the sample $x(i)$ and surrounds it on both sides by 0's. An example collection of waveforms $x(i)\delta(\cdot - i)$ for the case $x(i) = i$, $i = -1, 0, 1, 2$, is shown in Figure 2.4a. Now, sum over all i, bringing together all the samples of x one at a time, to obtain $x(\cdot)$. Figure 2.4b shows the result of this addition for the sequences in Fig. 2.4a. Thus, any signal $x(\cdot)$ may be expressed as a weighted sum of shifted impulses.

Equation (2.6) expresses a signal as a linear combination (or weighted sum) of impulses. That is, each sample may be viewed as an impulse at some amplitude and time. As we have already seen, each impulse (sample) arriving at the filter's input will cause the filter to produce an impulse response. If another impulse arrives at the filter's input before the first impulse response has died away, then the impulse response for both impulses will superimpose (add together sample by sample). More generally, since the input is a linear combination of impulses, the output is the *same* linear combination of impulse responses. This is a direct consequence of the superposition principle which holds for any LTI filter.

We repeat this in more precise terms. First linearity is used and then time-invariance is invoked. Using the form of the general linear filter in (2.1), and the definition of linearity, (2.2), we can express the output of any linear (and possibly time-varying) filter by

$$\begin{aligned} y(n) = \mathcal{L}_n\{x(\cdot)\} = \mathcal{L}_n\{x * \delta(\cdot)\} &= \mathcal{L}_n\left\{\sum_{i=-\infty}^{\infty} x(i)\delta(\cdot - i)\right\} \\ = \sum_{i=-\infty}^{\infty} x(i)\mathcal{L}_n\{\delta(\cdot - i)\} &\triangleq \sum_{i=-\infty}^{\infty} x(i)h(n,i) \end{aligned} \tag{2.7}$$

where we have written $h(n,i) \triangleq \mathcal{L}_n\{\delta(\cdot - i)\}$ to denote the filter response at time n to an impulse which occurred at time i. If we are to be completely rigorous mathematically, certain "smoothness" restrictions must be placed on the linear operator $\mathcal{L}_n$ in order that it may be distributed inside

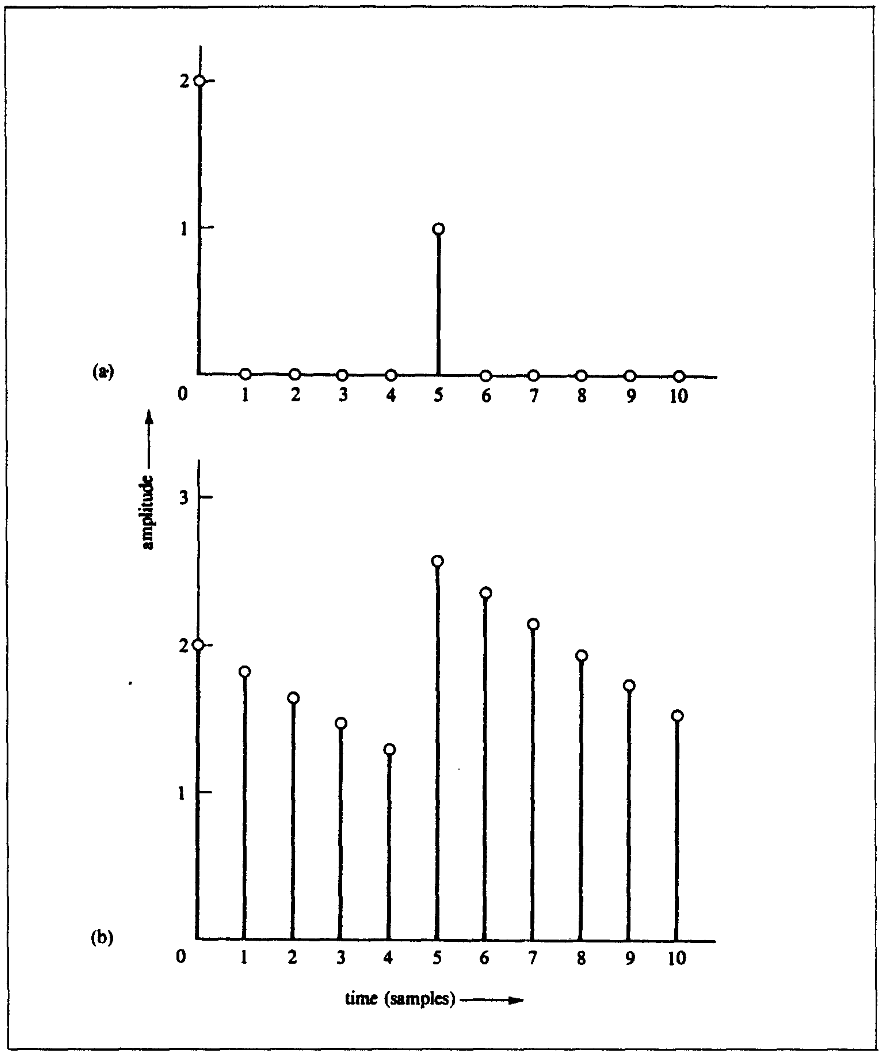

Figure 10. Input impulse pair and corresponding output for the filter $y(n) = x(n) + 0.9y(n-1)$.

a) Input: impulse of amplitude 2 plus delayed-impulse $2\delta(n) + \delta(n-5)$.

b) Output: $2h(n) + h(n-5)$.

the infinite summation [7]. However, all practically useful filters of the form of (2.4) satisfy these restrictions. If in addition to being linear, the filter is time-invariant, then $h(n,i) = h(n-i)$, which allows us to write

$$y(n) = \sum_{i=-\infty}^{\infty} x(i)h(n-i)\,. \tag{2.8}$$

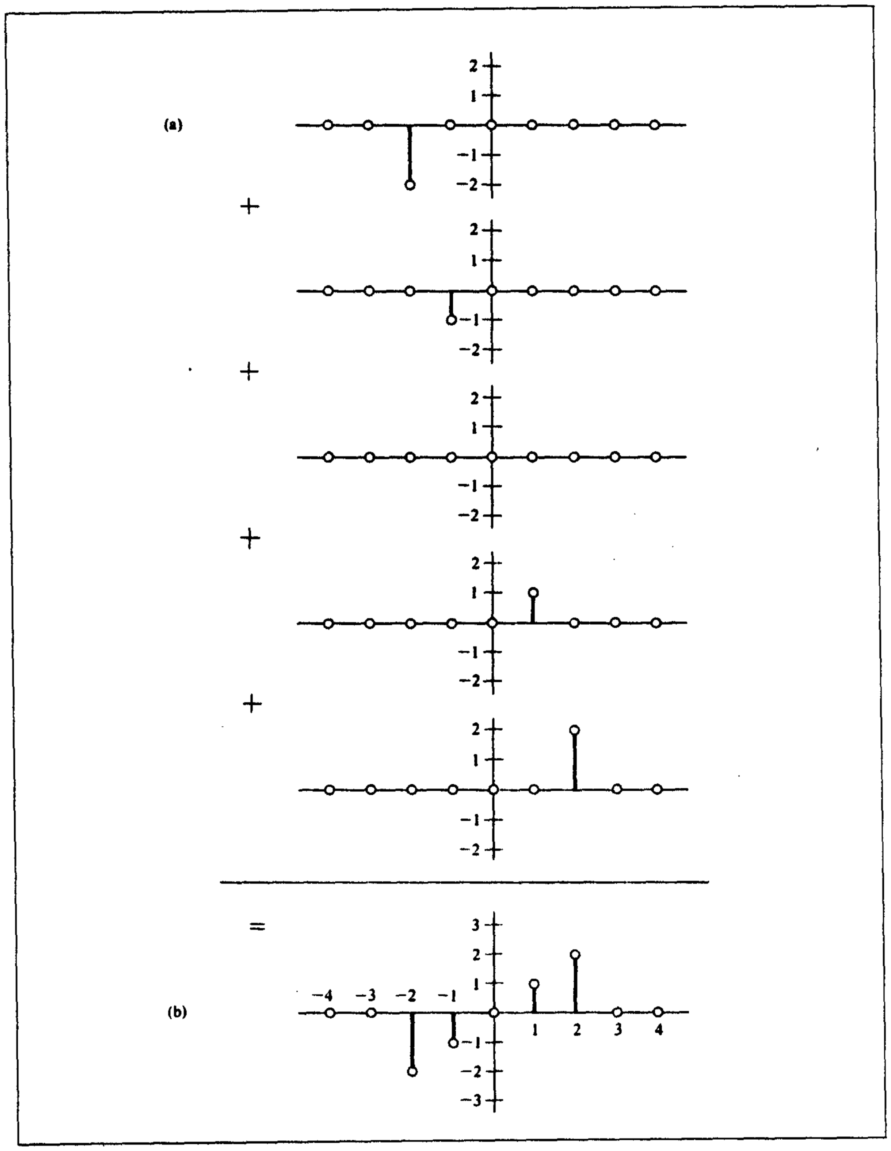

Figure 2.4. Any signal may be considered a sum of impulses.
a) Family of impulses at various amplitudes and time shifts.
b) Addition of impulses in a), giving part of a ramp.

This states that the filter output $y(n)$ is the *convolution* of the input $x(n)$ with the filter impulse response $h(n)$. You may object to the infinite sums in (2.6) through (2.8) on the grounds that your Pascal compiler complains when you say "**for** $I := -\infty$ **to** ∞ **do**." Well, there is a way to circumnavigate such deficiencies in Pascal. By choosing time 0 as the beginning of the signal, we may define $x(n)$ to be 0 for $n < 0$ so that $(-\infty)$ above might as well be replaced by 0. Also, if the filter is causal, then we'll have $h(n) = 0$ for $n < 0$, so the upper summation limit can be written as n. This gets us down to the ultimate impulse response representation of a linear, time-invariant, causal digital filter, namely,

$$y(n) = \sum_{i=0}^{n} x(i)h(n-i) \triangleq x*h(n).$$

It is easy to verify (by making the change of variable $j = n - i$) that convolution is commutative, i.e., $x*h(n) = h*x(n)$. Therefore, we can rewrite the impulse-response representation as

$$y(n) = \sum_{j=0}^{n} h(j)x(n-j).$$

This latter form looks more like the general difference equation presented in (2.4). In this form one can see that $h(j)$ may be interpreted as the a_j's in (2.4). It is also evident that the filter operates by summing weighted echoes of the input signal together. At time n, the weight of the echo from j samples ago $[x(n-j)]$ is $h(j)$.

We have shown that the output y of any LTI filter may be calculated by convolving the input x with the impulse response h. It is instructive to compare this method of filter implementation to the use of difference equations, (2.4). If there is no feedback, then the difference equation and the convolution formula are *identical;* in this case, $h(i) = a_i$ and there are no b's in (2.4). For recursive filters, we can convert the difference equation into a convolution by calculating the filter impulse response. However, this can be rather tedious, since with nonzero feedback coefficients the impulse response generally lasts forever. Of course, for stable filters the response is infinite only in theory; in practice, one may simply truncate the response after an appropriate length of time, such as after it falls below the "noise" level. The difference equation is typically less expensive to implement than the convolution formula when the filter is recursive.

It is natural to ask which of the two filter representations we have met thus far is the more useful in representing physical systems. In the case of a reverberant room, the impulse response has an approximately exponential envelope, and exponential decays are most easily simulated recursively. Filtering associated with the resonant qualities of a musical instrument is typically due to elastic deformations in the walls of a cavity, and the impulse responses of such systems tend to be sums of exponentially decaying sinusoids. The impulse response of every recursive filter is also a sum of exponentially decaying sinusoids (except in rare degenerate cases); hence, the difference equation is more compact than the convolution formula in these cases. However, in most practical circumstances it is not possible to directly measure the feedback coefficients (the b_i in (2.4)), and the best that can be done is to measure the filter response to a particular signal such as an impulse or a noise burst.

Consider, for example, the measurement of reverberation in a concert hall. We can record the sound of a sharp "crack" made by a bursting balloon or spark-gap device, and this can be considered an approximation to the impulse response of the hall between the impulse source and the receiving microphone*. If one desires to simulate the effect of this hall on a particular sound, the measured impulse response can be convolved with the sound. However, this is an extremely expensive and time-consuming computation. It is more efficient to first convert the impulse response into recursive filter coefficients and then implement a difference equation to simulate the hall. Unfortunately, it is

* It is better in practice to use a long sample of digitally generated "white noise" than to approximate an impulse.

not easy to find the best recursive filter. The popular technique known as *linear prediction* [8] can be used to find the feedback coefficients, but it assumes that there are no *feedforward* coefficients (i.e., standard linear prediction procedures assume that $M = 0$ in (2.4)). Fitting both feedforward and feedback coefficients to a given response is a difficult problem that is still under active investigation. Unfortunately, little attention has been paid to the quality of approximation in *perceptual* terms.

This concludes the discussion of time-domain filter descriptions. For the most part you need them only when trying to implement a filter on a computer. For thinking about the effect that a filter has on your sound, it is usually more appropriate to consider a frequency-domain picture, to which we now turn.

2.2.3. Transfer Function

From the discussion in the preceding section, if y is the output of a linear time-invariant filter with input x and impulse response h, then

$$y(n) = h*x(n), \tag{2.9}$$

where, as before, '$*$' denotes convolution. We now wish to take the z transform of both sides of this equation. The z transform is discussed in [11], and we only give its definition here. The z transform of the discrete-time signal $x(n)$ is defined to be

$$X(z) \triangleq \sum_{n=-\infty}^{\infty} x(n)z^{-n},$$

where z is a complex variable. Since in this paper all signals are considered to begin at time $n = 0$, and since all filters are assumed to be causal, the lower summation limit given above may be written as 0 rather than $-\infty$. We sometimes express the above relation by writing $X(z) = Z\{x(n)\}$ or $X(z) \leftrightarrow x(n)$.

The *convolution theorem* [13] states that

$$x*y(n) \leftrightarrow X(z)Y(z). \tag{2.10}$$

(See also [11] for more on convolution.) In words, this says that *convolution in the time domain is multiplication in the frequency domain.* Taking the z transform of both sides of (2.9) and applying the convolution theorem gives

$$Y(z) = H(z)X(z) \tag{2.11}$$

where $H(z)$ is the z transform of the filter impulse response. This equation tells us that the z transform of the filter output is the z transform of the input times the z transform of the impulse response. We may divide (2.11) by $X(z)$ to obtain

$$H(z) = \frac{Y(z)}{X(z)}.$$

This equation gives a new perspective on the impulse response of a filter. It states that the z transform of the filter impulse response is the ratio of the filter output to filter input transforms. That is, for any given input $x(n)$, the output $y(n)$ will be such that the z transform ratio is always $H(z)$ (a fixed function determined by the filter). Since multiplying the input transform by $H(z)$

gives the output transform, $H(z)$ gives us the *transfer* characteristics of the filter. Accordingly, we make the following definition.

> The *transfer function* $H(z)$ of a linear time-invariant discrete-time filter is defined to be the z transform of the impulse response $h(n)$.

Since z transforming the convolution representation for digital filters was so profitable, we will apply it also to the general difference equation, (2.4). To do this requires two more properties of the z transform, linearity and the *shift theorem* .

The *shift theorem* [13] for z transforms states that

$$x(n-k) \leftrightarrow z^{-k}X(z),$$

where, as always, $x(n) \leftrightarrow X(z)$ by definition. This means that a signal $x(n-k)$, which is the waveform $x(n)$ delayed by k samples, has the z transform $z^{-k}X(z)$, which is the z transform of x multiplied by z^{-k}. Using these two properties we can write down the z transform of any difference equation by inspection.

Repeating the general difference equation ((2.4)) for LTI filters, we have

$$\begin{aligned} y(n) = a_0x(n) + a_1x(n-1) + \cdots + a_Mx(n-M) \\ - b_1y(n-1) - \cdots - b_Ny(n-N), \end{aligned}$$

We now take the z transform of both sides, denoting the transform by $Z\{\}$. Because $Z\{\}$ is a linear operator, it may be distributed through the terms on the right-hand side as follows:

$$\begin{aligned} Z\{y(n)\} &= Z\{a_0x(n) + a_1x(n-1) + \cdots + a_Mx(n-M) \\ &\quad - b_1y(n-1) - \cdots - b_Ny(n-N)\} \\ &= Z\{a_0x(n)\} + Z\{a_1x(n-1)\} + \cdots + Z\{a_Mx(n-M)\} \\ &\quad - Z\{b_1y(n-1)\} - \cdots - Z\{b_Ny(n-N)\} \\ &= a_0Z\{x(n)\} + a_1Z\{x(n-1)\} + \cdots + a_MZ\{x(n-M)\} \\ &\quad - b_1Z\{y(n-1)\} - \cdots - b_NZ\{y(n-N)\} \\ &= a_0Z\{x(n)\} + a_1z^{-1}Z\{x(n)\} + \cdots + a_Mz^{-M}Z\{x(n)\} \\ &\quad - b_1z^{-1}Z\{y(n)\} - \cdots - b_Nz^{-N}Z\{y(n)\}, \end{aligned}$$

where we used properties 2) and 1) of linearity given in (2.2), followed by use of the shift theorem, in that order. Replacing $Z\{y(n)\}$ by $Y(z)$ and $Z\{x(n)\}$ by $X(z)$ results in

$$\begin{aligned} Y(z) = a_0X(z) + a_1z^{-1}X(z) + \cdots + a_Mz^{-M}X(z) \\ -b_1z^{-1}Y(z) - \cdots - b_Nz^{-N}Y(z), \end{aligned} \qquad (2.12)$$

The terms in $Y(z)$ may be grouped together on the left-hand side to get

$$\begin{aligned} Y(z) + b_1z^{-1}Y(z) + \cdots + b_Nz^{-N}Y(z) \\ = a_0X(z) + a_1z^{-1}X(z) + \cdots + a_Mz^{-M}X(z). \end{aligned}$$

Factoring out the common terms $Y(z)$ and $X(z)$ gives

$$Y(z)\left[1+b_1z^{-1}+\cdots+b_Nz^{-N}\right]=X(z)\left[a_0+a_1z^{-1}+\cdots+a_Mz^{-M}\right].$$

Finally, solving for $Y(z)/X(z)$, which equals the transfer function $H(z)$, yields

$$H(z)=\frac{Y(z)}{X(z)}=\frac{a_0+a_1z^{-1}+\cdots+a_Mz^{-M}}{1+b_1z^{-1}+\cdots+b_Nz^{-N}}. \tag{2.13}$$

Thus, taking the z transform of the general difference equation led to a new formula for the transfer function in terms of the difference equation coefficients. (Now the minus signs for the feedback coefficients should no longer be mysterious.)

The primary benefit of representing a filter by its transfer function is the insight gleaned from poles and zeros, which will be discussed shortly. First, however, we specialize the transfer function to what is called the *frequency response.*

2.2.4. Frequency Response

Beginning with (2.11), we have

$$Y(z)=H(z)X(z),$$

where $X(z)$ is the z transform of the filter input, $Y(z)$ is the z transform of the output signal, and $H(z)$ is the filter transfer function. A most basic property of the z transform is that setting z to $e^{j\omega T}$ gives us the spectrum [11]. In other words, the spectrum equals the z transform evaluated on the *unit circle.* Applying this fact to the equation above gives

$$Y(e^{j\omega T})=H(e^{j\omega T})X(e^{j\omega T}). \tag{2.14}$$

Thus, the spectrum of the filter output is just the input spectrum times $H(e^{j\omega T})$. This motivates the following definition.

The *frequency response* of a linear time-invariant digital filter is defined to be the transfer function, $H(z)$, evaluated on the unit circle, that is, $H(e^{j\omega T})$.

Since e, j, and T are constants, the frequency response is really only a function of radian frequency ω; it is not essentially a function of a complex variable as is the transfer function. Thus, the frequency response may be considered a complex-valued function of a real variable. The response at frequency f Hz, for example, is $H(e^{j2\pi fT})$, where T is the sampling period in seconds. It would be more convenient to define new functions and write simply $Y'(\omega)=H'(\omega)X'(\omega)$ instead of writing $e^{j\omega T}$ so often, but doing so would add a lot of new functions to an already complicated scenario. Furthermore, carrying $e^{j\omega T}$ along makes explicit the connection between the transfer function and the frequency response.

Notice that the fundamental significance of $e^{j\omega T}$ in this context is to place the frequency "axis," as it were, on a circle. This way, adding multiples of the sampling frequency f_s to $\omega/2\pi$ corresponds to traversing whole cycles around the circle. Since the spectrum of every discrete-time signal repeats with "period" $f_s=1/T$, it may be considered to consist of one copy of the spectrum defined over a circle.

Because every complex number can be represented as a magnitude and angle, the frequency response may be decomposed into two real-valued functions, the *amplitude response* and the *phase response*. Formally, we define them as follows:

$$G(\omega) \triangleq |H(e^{j\omega T})|$$
$$\Theta(\omega) \triangleq \angle H(e^{j\omega T})$$

so that

$$H(e^{j\omega T}) = G(\omega)e^{j\Theta(\omega)}. \tag{2.15}$$

Thus, $G(\omega)$ is the magnitude (or complex modulus) of $H(e^{j\omega T})$, and $\Theta(\omega)$ is the phase (or complex angle) of $H(e^{j\omega T})$.

The real valued function $G(\omega)$ is called the filter *amplitude response*, and it specifies the amplitude *gain* that the filter provides at each frequency. The real function $\Theta(\omega)$ is the *phase response*, and it gives the phase shift in radians that each input component sinusoid will undergo. These interpretations may be expressed mathematically by writing the terms of (2.14) in polar form to get:

$$\left|Y(e^{j\omega T})\right|e^{j\angle Y(e^{j\omega T})} = G(\omega)\left|X(e^{j\omega T})\right|e^{j\Theta(\omega)+\angle X(e^{j\omega T})}$$

which implies

$$\left|Y(e^{j\omega T})\right| = G(\omega)\left|X(e^{j\omega T})\right|$$
$$\angle Y(e^{j\omega T}) = \Theta(\omega) + \angle X(e^{j\omega T}).$$

Equation (2.15) gives the frequency response in polar form. For completeness, we give the transformations between polar and rectangular forms (i.e. for converting real and imaginary parts to magnitude and angle and vice versa). The real part of a complex number z is written as $\text{Re}\{z\}$, and the imaginary part of z is denoted by $\text{Im}\{z\}$.

$$\begin{aligned} G(\omega) &\triangleq \left|H(e^{j\omega T})\right| = \sqrt{\text{Re}^2\{H(e^{j\omega T})\} + \text{Im}^2\{H(e^{j\omega T})\}} \\ \Theta(\omega) &\triangleq \angle H(e^{j\omega T}) = \tan^{-1}\left(\frac{\text{Im}\left\{H(e^{j\omega T})\right\}}{\text{Re}\{H(e^{j\omega T})\}}\right) \end{aligned} \tag{2.16}$$

Going the other way from polar to rectangular (using Euler's formula again),

$$\text{Re}\left\{H(e^{j\omega T})\right\} = G(\omega)\cos\left[\Theta(\omega)\right]$$
$$\text{Im}\{H(e^{j\omega T})\} = G(\omega)\sin\left[\Theta(\omega)\right].$$

The application of these formulae to some basic example filters will be covered in chapter 3.

From (2.13), we have that the transfer function of a recursive filter is a ratio of polynomials in z. This may be expressed symbolically by writing

$$H(z) = \frac{a(z)}{b(z)}.$$

By elementary properties of complex numbers, we have

$$\begin{aligned} G(\omega) &= \frac{|a(e^{j\omega T})|}{|b(e^{j\omega T})|} \\ \Theta(\omega) &= \angle a(e^{j\omega T}) - \angle b(e^{j\omega T}). \end{aligned}$$

These relations often simplify calculations.

Consider again the *sine-wave analysis* method used in chapter 1 for the simplest lowpass filter. We let the input signal be a complex sinusoid

$$x(n) = Ae^{j\omega_0 nT+\phi},$$

having amplitude A and phase ϕ. Recall that for complex sinusoidal inputs, the amplitude response is measurable as the amplitude of the output divided by the amplitude of the input, or $G(\omega_0) = |y(n)|/|x(n)|$. (This simple formula for $G(\omega_0)$ holds *only* for complex sinusoidal inputs, since only a complex sinusoid has a constant instantaneous amplitude.) Also, the phase response is given by the phase of the output sinusoid minus the phase of the input sinusoid, or $\Theta(\omega_0) = \angle y(n) - \angle x(n)$. Thus, the output $y(n)$ is

$$\begin{aligned} y(n) &= [G(\omega_0)A]e^{j(\omega_0 nT+\phi+\Theta(\omega_0))} \\ &= G(\omega_0)e^{j\Theta(\omega_0)}Ae^{j\omega_0 nT+\phi} \\ &= H(e^{j\omega_0 T})Ae^{j\omega_0 nT+\phi}. \end{aligned}$$

As a special case of this example (in view of Euler's identity), the response to

$$x(n) = A\cos(\omega_0 nT + \phi)$$

is

$$y(n) = AG(\omega_0)\cos[\omega_0 nT + \phi + \Theta(\omega_0)].$$

Thus, as stated before, for a sinusoidal input at radian frequency ω_0, the amplitude is multiplied by $G(\omega_0)$, and the phase is additively changed by $\Theta(\omega_0)$. By superposition, we may readily generalize the above equation to the case in which $x(n)$ is an arbitrary sum of input sinusoids. Doing this yields the inverse DFT of (2.14):

$$x(n) = \sum_k A_x(\omega_k)e^{j\omega_k nT+\phi_k}$$

which implies

$$y(n) = \sum_k G(\omega_k)A_x(\omega_k)e^{j\omega_k nT+\phi_k+\Theta(\omega_k)}.$$

Note that the specification of the ideal lowpass filter in 1.1 was actually only a specification of $G(\omega)$. Often, we only wish to modify the spectrum energy at certain frequencies to make them either more or less audible. In these cases a particular $G(\omega)$ is desired, and we do not care about $\Theta(\omega)$ as long as it does not make any audible difference. More on $\Theta(\omega)$ is given in the next section.

In contrast to the polar representation of frequency response, the real and imaginary parts do not have such intuitively appealing individual interpretations. Consequently, the polar form is usually more desirable for expressing filter response as a function of frequency.

2.3. Alternate Representations in the Frequency Domain

This section discusses *phase delay* and *group delay*, and discusses their interpretation as filter *time delay*. In addition, *poles and zeros* are defined, and a graphical method for evaluating the amplitude and phase response directly from the poles and zeros is described.

2.3.1. Phase Delay and Group Delay

The phase response of a filter $\Theta(\omega)$ gives the *radian* phase shift experienced by each sinusoidal component of the input signal. Sometimes it is more meaningful to consider *phase delay* [13] which we define by

$$P(\omega) \triangleq -\frac{\Theta(\omega)}{\omega}.$$

The phase delay gives the *time delay* in seconds experienced by each sinusoidal component of the input signal. For example, in the simple lowpass filter of (1.1), we found that the phase response was $\Theta(\omega) = -\omega T/2$, which corresponds to a phase delay $P(\omega) = T/2$, or one-half sample.

More generally, if the input to a filter with frequency response $H(e^{j\omega T}) = G(\omega)e^{j\Theta(\omega)}$ is

$$x(n) = \cos(\omega nT),$$

then the output is

$$\begin{aligned} y(n) &= G(\omega)\cos[\omega nT + \Theta(\omega)] \\ &= G(\omega)\cos\{\omega[nT - P(\omega)]\}, \end{aligned}$$

and it can be seen that the phase delay expresses phase response as time delay.

In working with phase delay, however, care must be taken to ensure that all appropriate multiples of 2π have been included in $\Theta(\omega)$. We defined $\Theta(\omega)$ simply as the complex angle of the frequency response $H(e^{j\omega T})$, and this is not sufficient for obtaining a phase response which can be converted to true time delay. If multiples of 2π are discarded, as is done in the definition of complex angle, the phase delay is modified by multiples of the sinusoidal period. Since LTI filter analysis is based on sinusoids without beginning or end, one cannot in principle distinguish between "true" phase delay and a phase delay with discarded sinusoidal periods. Nevertheless, it is convenient to define the filter phase response as a *continuous* function of frequency with the property that $\Theta(0) = 0$ (for real filters). This specifies a means of "unwrapping" the phase response to get a consistent phase delay curve.

A more commonly encountered representation of filter phase response is called the *group delay*, and it is defined by

$$D(\omega) \triangleq -\frac{d}{d\omega}\Theta(\omega).$$

For linear phase responses the group delay and the phase delay are identical, and each may be interpreted as time delay.

For any phase function the group delay $D(\omega)$ may be interpreted as the *time delay* of the *amplitude envelope* of a sinusoid at frequency ω [13]. The bandwidth of the amplitude envelope in this interpretation must be restricted to a frequency interval over which the phase response is approximately linear. While the proof will not be given here, it should seem reasonable when the process of amplitude envelope detection is considered. The narrow "bundle" of frequencies centered at the carrier frequency ω is translated to 0 Hz. At this point it is evident that the group delay at the carrier frequency gives the slope of the linear phase of the translated spectrum. But this is a constant phase delay, and therefore it has the interpretation of true time delay for the amplitude envelope.

For example, consider the *phase vocoder* [14] which provides a bank of bandpass filters which decomposes the input signal into narrow spectral "slices". This is the analysis step. For synthesis (often called *additive synthesis*), a bank of sinusoidal oscillators is provided, with amplitude and frequency control inputs. The oscillator frequencies are tuned to the analysis filter center-frequencies, and the amplitude controls are driven by the amplitude envelopes measured in the analysis. (Typically, some data reduction or envelope modification has taken place in the amplitude envelope set.) With these oscillators, the band-slices are independently regenerated and summed together to form the original (or facsimile) input signal. Suppose we excite only channel k of the phase vocoder with the input

$$A(nT)\cos(\omega_k nT),$$

where ω_k is the center-frequency of the channel and the bandwidth of $A(nT)$ is smaller than half the channel bandwidth. If the phase of the channel filter is linear within the passband, then the filter output will be

$$y(n) = A[nT - D(\omega_k)]\cos\{\omega_k[nT - P(\omega_k)]\}.$$

Thus, for the phase vocoder the phase delay gives the time delay of the sinusoidal "carrier," and the group delay gives the time delay of the amplitude envelope.

Phase responses which are linear in frequency correspond to constant phase delay and group delay. If the phase response is nonlinear, then the relative phase of the sinusoidal signal components is generally altered by the filter. This may cause a smearing of attack transients such as in percussive sounds. Another term for the effects of nonlinear phase response is *phase dispersion*.

When one wishes to modify only the spectral energy distribution of a signal and not the spectral phase, then a *linear-phase filter* is desirable.* Linear-phase filters are characterized by a symmetrical impulse response [15], i.e., $h(n) = h(-n)$ (symmetric) or $h(n) = -h(-n)$ (antisymmetric).

In computer music applications it is often possible to store the entire signal to be filtered on disk or magnetic tape. In these cases the effects of nonlinear phase can removed entirely by filtering the data first forward then backward in time. The resultant phase response is 0, and the amplitude response becomes $G^2(\omega)$ (see problem 10).

For many applications (such as lowpass, bandpass, or highpass filtering), the frequency region where most of the phase dispersion occurs is at the extreme edge of the desired range or "passband," i.e., in the vicinity of the cutoff frequencies. Only filters with no feedback can have exactly linear phase, and such filters generally need many more multiplies for a given specification on the amplitude response $G(\omega)$ [15].

Linear phase filters are far from *minimum phase*. Minimum-phase filters give the smallest amount of phase shift (delay) for a specified amplitude response. Consequently, linear phase filters introduce excess time delay. (The extra time delay can be considered that needed to delay all frequencies equally.)

Phase dispersion due to nonlinear phase near the cut-off frequency of a lowpass filter usually appears as "ringing" in the time domain. This ringing can sometimes be heard as little "chirps" in the impulse response, if the cut-off is in the audio band. To be conservative, the filter *roll-off* (slope

* The name *linear-phase response filter* would be more precise in this context. Note, however, that the phase response of a filter equals the phase of the spectrum of its impulse response. It is therefore reasonable to use the term *filter phase* interchangeably with *filter phase response*.

of log $[G(\omega)|)$ should be limited so that the phase dispersion is inaudible whether or not the filter is linear phase or delay equalized. Linearizing the phase with a delay equalizer does not eliminate "ringing," it merely shifts it in time. A good rule-of-thumb is to keep the total impulse-response duration below the time-discrimination threshold of hearing.

For musical purposes, $G(\omega)$ is usually of primary interest. Notable exceptions are echo and reverberation, in which delay characteristics are at least as important. As a rule of thumb, filter delays less than two or three cycles at any given frequency are not important in music applications.

2.3.2. Poles and Zeros

We can write the general transfer function for the recursive LTI digital filter as

$$H(z) = g\frac{1 + \alpha_1 z^{-1} + \cdots + \alpha_M z^{-M}}{1 + b_1 z^{-1} + \cdots + b_N z^{-N}},$$

which is the same as (2.13) except that we have factored out the leading coefficient a_0 in the numerator (assumed to be nonzero) and called it g. (Here $\alpha_i = a_i/a_0$.) In the same way that $z^2 + 3z + 2$ can be factored into $(z+1)(z+2)$, we can factor the numerator and denominator to obtain

$$H(z) = g\frac{(1 - q_1 z^{-1})(1 - q_2 z^{-1})\cdots(1 - q_M z^{-1})}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})\cdots(1 - p_N z^{-1})}. \tag{2.17}$$

Assume, for simplicity, that none of the factors cancel out. The (possibly complex) numbers $\{q_1, \ldots, q_M\}$ are the roots of the numerator polynomial. When z equals any of these values, the transfer function evaluates to 0. For this reason, the roots of the *numerator* polynomial of the filter transfer function are called the *zeros* of the filter. Similarly, when z approaches any root of the denominator polynomial, the magnitude of the transfer function becomes larger and larger, approaching infinity. Consequently, the *denominator* roots $\{p_1, \ldots, p_N\}$ are called the *poles* of the filter.

The term *pole* really makes sense when you plot the magnitude of $H(z)$ as a function of z. Since z is complex, it may be taken to lie in a plane (the *z plane*). The magnitude of $H(z)$ is real and therefore can be represented by distance above the z-plane. The plot appears as an infinitely thin surface spanning in all directions over the z-plane. The zeros are the points where the surface dips down to touch the z-plane. At high altitude, the poles look like thin rods that go straight up forever, getting thinner the higher they go.

One can visualize the magnitude transfer function of any filter as follows. Let the z-plane be represented by the floor. Place a tall thin rod upright at the location of each pole. Now cover the rods with a thin flexible rubber membrane and stretch it taut. Pin the material down wherever there is a zero and you have the "heuristic magnitude transfer function" whose value is given by the height of the membrane above the floor. We can find the frequency response by drawing a line with a crayon on the rubber membrane above the circle of radius 1 on the floor. Keeping the crayon exactly above the circle on the floor will make your hand follow the membrane up and down. This rising and falling motion is tracing out the filter's amplitude response; i.e., the height of the marked line from the floor gives the gain of the filter at each frequency. Of course, it is much more accurate (and perhaps even easier) to write a program which plots the magnitude of $H(z)$ over an arbitrary set of z values.

Notice that the M feedforward coefficients from the general difference equation, (2.4), give rise to M zeros. Similarly, the N feedback coefficients in (2.4) give rise to N poles. This illustrates the general fact that zeros are caused by adding a finite number of input samples together and poles are caused by feedback. Recall that the filter order is the maximum of N and M. If $a_0 \neq 0$ in (2.13), it then follows that the filter order is determined by the number of poles or zeros, whichever is greater.

Now consider what happens when we take the factored form of the general transfer function, (2.17), and set z to $e^{j\omega T}$ to get the frequency response.

$$H(e^{j\omega T}) = g\frac{(1-q_1e^{-j\omega T})(1-q_2e^{-j\omega T})\cdots(1-q_Me^{-j\omega T})}{(1-p_1e^{-j\omega T})(1-p_2e^{-j\omega T})\cdots(1-p_Ne^{-j\omega T})}$$

As usual, we prefer the polar form for this expression. First let's consider the amplitude response $G(\omega) \triangleq |H(e^{j\omega T})|$.

$$\begin{aligned}
G(\omega) &= g\frac{|1-q_1e^{-j\omega T}||1-q_2e^{-j\omega T}|\cdots|1-q_Me^{-j\omega T}|}{|1-p_1e^{-j\omega T}||1-p_2e^{-j\omega T}|\cdots|1-p_Ne^{-j\omega T}|}\\
&= g\frac{|e^{jM\omega T}|}{|e^{jN\omega T}|}\frac{|1-q_1e^{-j\omega T}||1-q_2e^{-j\omega T}|\cdots|1-q_Me^{-j\omega T}|}{|1-p_1e^{-j\omega T}||1-p_2e^{-j\omega T}|\cdots|1-p_Ne^{-j\omega T}|}\\
&= g\frac{|e^{j\omega T}-q_1||e^{j\omega T}-q_2|\cdots|e^{j\omega T}-q_M|}{|e^{j\omega T}-p_1||e^{j\omega T}-p_2|\cdots|e^{j\omega T}-p_N|}
\end{aligned} \tag{2.18}$$

In the complex plane, every number may be treated as a vector. A term of the form $u - v$, where u and v are complex, is simply a vector from the tip of v to the tip of u, as shown in Figure 2.5. Thus, the term $e^{j\omega T} - q_i$ may be drawn as an arrow from the i^{th} zero to the point $e^{j\omega T}$ on the unit circle, and $e^{j\omega T} - p_i$ is an arrow from the i^{th} pole. Therefore, each term in (2.18) is the length of a vector drawn from a pole or zero to a single point on the unit circle, as shown in Figure 2.6 for two poles and two zeros.

The frequency response magnitude (amplitude response) is given by the product of the lengths of vectors drawn from the zeros divided by the product of lengths of vectors drawn from the poles.

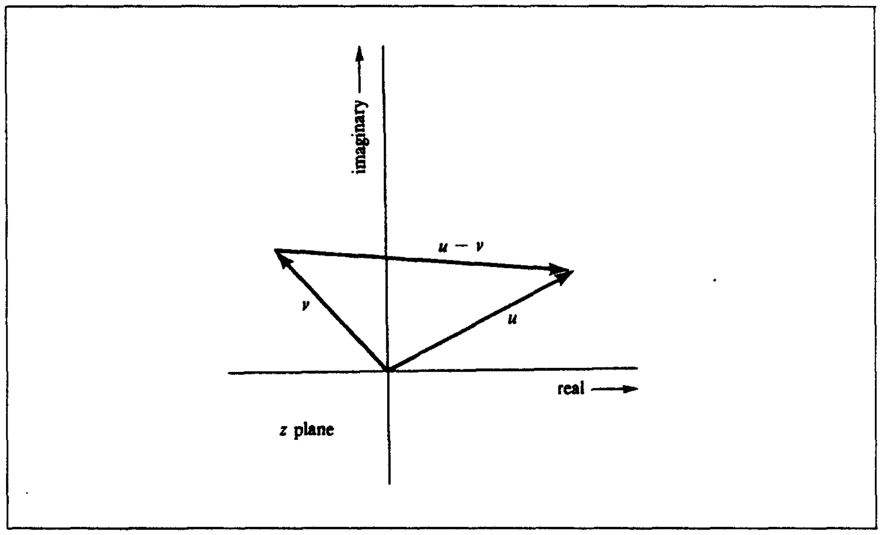

Figure 2.5. Treatment of complex numbers as vectors in a plane. In the complex plane, the number $z = x + jy$ may be plotted using the coordinates (x,y). The difference of two vectors $u = x_1 + jy_1$ and $v = x_2 + jy_2$ is $u - v = (x_1 - x_2) + j(y_1 - y_2)$. Translating the origin of the vector $u - v$ to the tip of v shows that $u - v$ is an arrow drawn from the tip of v to the tip of u. The length of a vector is unaffected by translation away from the origin. However, the angle of a translated vector must be measured relative to a translated copy of the real axis shown above as a dashed line.

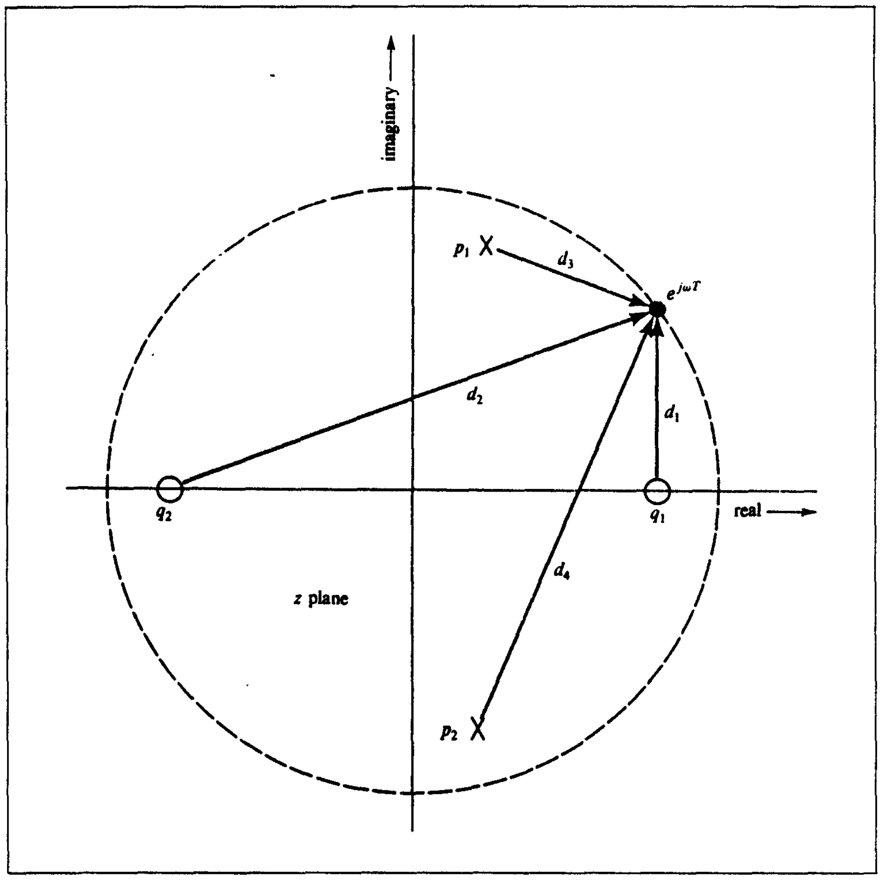

Figure 2.6. Measurement of amplitude response from a pole-zero diagram. A pole is represented in the complex plane by a small cross; a zero, by a small circle. Arrows are drawn from the poles and zeros to the point of the unit circle corresponding to the frequency at which the response is desired. To obtain the amplitude response, the product of the lengths of the arrows from the zeros ($d_1 d_2$) is divided by the product of the lengths of the arrows from the poles ($d_3 d_4$). Thus, at the frequency ω where the arrows join (which is slightly less than one-eighth the sampling rate) the gain of this two-pole two-zero filter is $G(\omega) = \frac{d_1 d_2}{(d_3 d_4)}$.

For example, the DC gain is obtained by multiplying the lengths of the lines drawn from all poles and zeros to the point $z = 1$. The filter gain at half the sampling rate is the product of the lengths of these lines when drawn to the point $z = -1$. For an arbitrary frequency f in Hz, we draw the arrows from the poles and zeros to the point $z = e^{j2\pi fT}$. 2.6 is drawn for a frequency in the vicinity of one-eighth the sampling rate. Figure 2.7 gives the complete amplitude response for the poles and zeros in 2.6.

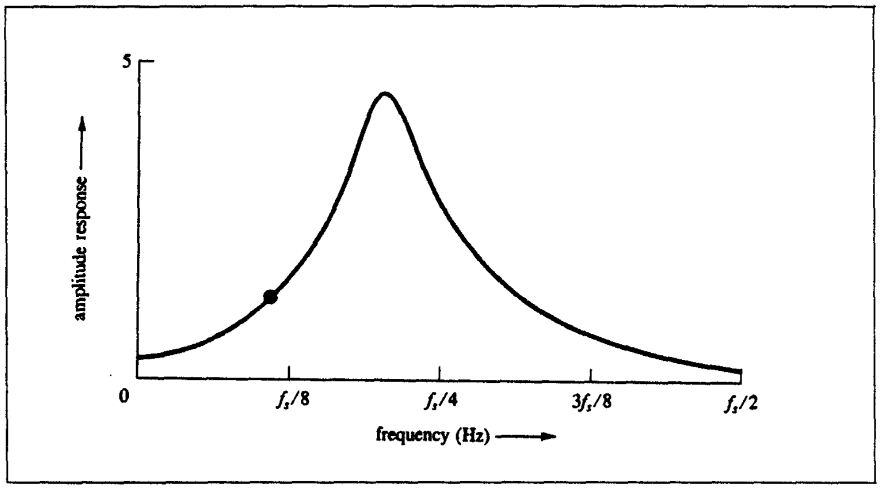

Figure 2.7. Amplitude response obtained from Fig. 2.6 by traversing the entire upper semicircle with $e^{j\omega T}$. The single point of the amplitude obtained in Fig. 2.6 is marked by a heavy dot. For real filters, precisely the same curve is obtained if the lower half of the unit circle is traversed, i.e., $G(\omega) = G(-\omega)$. Thus, plotting the response over positive frequencies only is sufficient.

The phase response is just as easy to evaluate graphically as is the amplitude response.

$$\begin{aligned}
\Theta(\omega) &\triangleq \angle g \frac{(1-q_1e^{-j\omega T})(1-q_2e^{-j\omega T})\cdots(1-q_Me^{-j\omega T})}{(1-p_1e^{-j\omega T})(1-p_2e^{-j\omega T})\cdots(1-p_Ne^{-j\omega T})} \\
&\triangleq \angle g e^{j(N-M)\omega T}\frac{(e^{j\omega T}-q_1)(e^{j\omega T}-q_2)\cdots(e^{j\omega T}-q_M)}{(e^{j\omega T}-p_1)(e^{j\omega T}-p_2)\cdots(e^{j\omega T}-p_N)} \\
&= (N-M)\omega T + \angle(e^{j\omega T}-q_1) + \angle(e^{j\omega T}-q_2) + \cdots + \angle(e^{j\omega T}-q_M) \\
&\quad - \angle(e^{j\omega T}-p_1) - \angle(e^{j\omega T}-p_2) - \cdots - \angle(e^{j\omega T}-p_N)
\end{aligned}$$

Since $e^{j\omega T} - z$ is a vector drawn from the point z to the point $e^{j\omega T}$, the angle of $e^{j\omega T} - z$ is the angle of the constructed vector (where a vector pointing horizontally to the right has an angle of 0). Therefore, the phase response at frequency f Hz is again obtained by drawing lines from all the poles and zeros to the point $e^{j2\pi fT}$ as shown in Figure 2.8. The angles of the lines from the zeros are added, and the angles of the lines from the poles are subtracted. Finally, $(N-M)2\pi fT$ appears when the number of poles and zeros is not equal. This factor comes from writing the transfer function as

$$H(z) = gz^{N-M}\frac{(z-q_1)(z-q_2)\cdots(z-q_M)}{(z-p_1)(z-p_2)\cdots(z-p_N)}$$

and may be thought of as arising from $N-M$ additional zeros at $z=0$ when $N > M$, or $M-N$ poles at $z=0$ for $M > N$.* Figure 2.9 gives the phase response for this two-pole two-zero example.

* One can take the point of view that every digital filter has an equal number of poles and zeros by counting those at $z = 0$ and $z = \infty$. It is customary, however, when discussing the number of poles and zeros a filter has, to neglect these since they correspond to pure delay and do not affect the amplitude response.

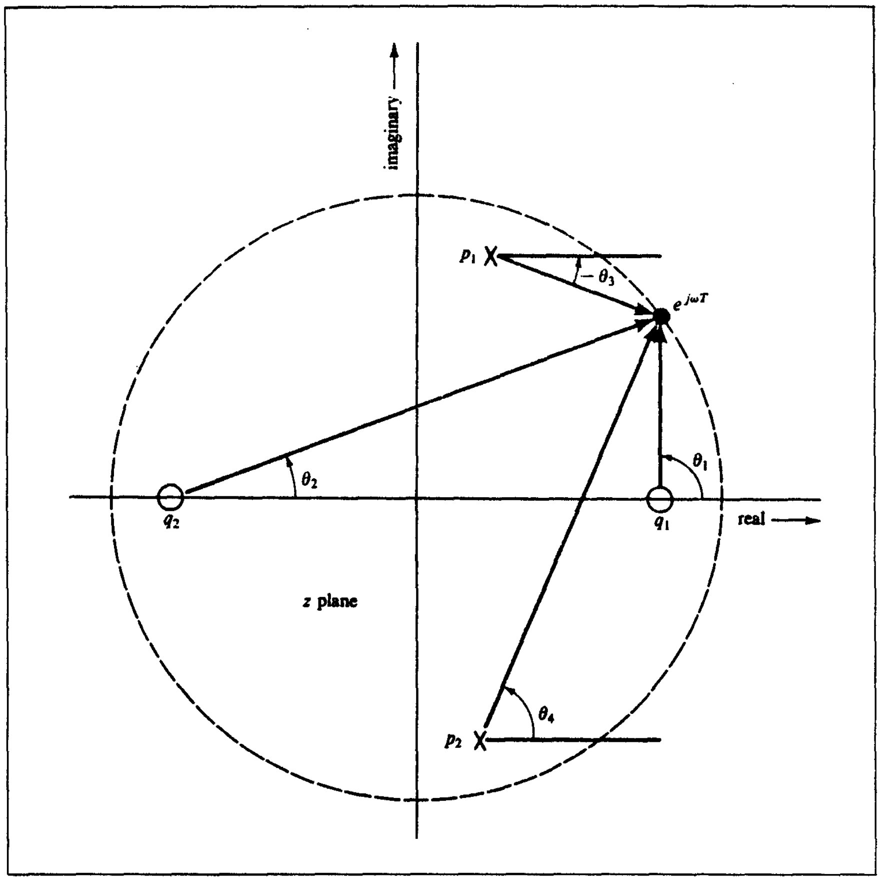

Figure 2.8. Measurement of phase response from a pole-zero diagram. Arrows are drawn from the poles and zeros to the point of the unit circle at which the response is desired. To obtain the phase response, the sum of the angles of the arrows from the poles $(\theta_3\theta_4)$ is subtracted from the sum of the angles of the arrows from the zeros $(\theta_1 + \theta_2)$. Thus, at the frequency ω the phase response of the two-pole two-zero filter is $\Theta(\omega) = \theta_1 + \theta_2 - \theta_3 - \theta_4$.

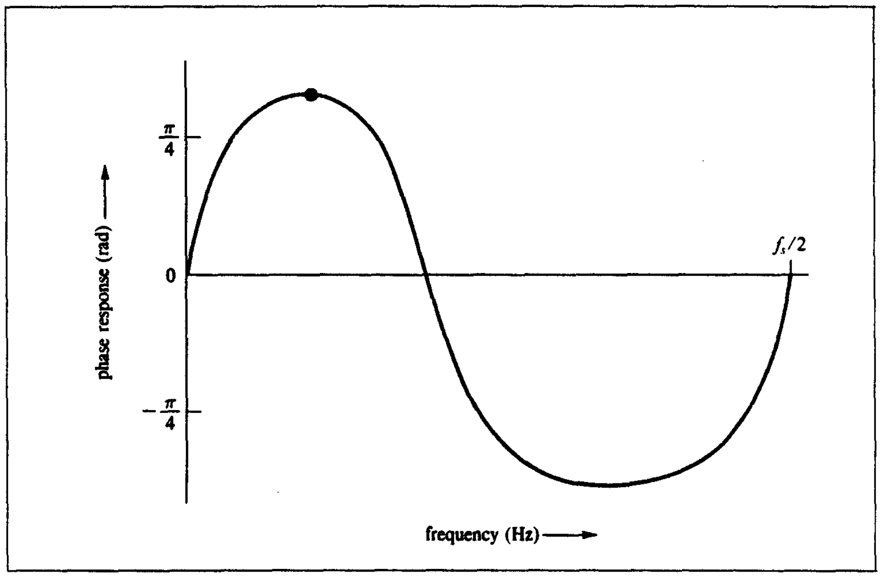

Figure 2.9. Phase response obtained from Fig. 2.8 for positive frequencies. The point of the phase response corresponding to the arrows in Fig. 2.8 is marked by a heavy dot. For real filters, $\Theta(-\omega) = -\Theta(\omega)$, so that the above curve is reflected about zero for negative frequencies.

Stability

A filter is said to be *stable* if its impulse response $h(n)$ decays to 0 as n goes to infinity. Unstable filters are to be avoided because they lead to overflow of the computer word, which, in turn, can make pom-pom's out of your speakers. The physical counterpart of an unstable filter is something that will never stop quivering after you hit it with a hammer (or, worse, will begin to shake more and more violently, like a washing machine with boots in it).

In terms of poles and zeros, a filter is stable if and only if all the poles are inside the unit circle in the z-plane. This is because the transfer function is the z transform of the impulse response, and if there is a pole outside the unit circle, then there is an exponentially increasing component of the impulse response. To see this, consider an impulse response of the form

$$h(n) = \begin{cases} R^n e^{j\omega nT}, & n \geq 0 \\ 0, & n < 0 \end{cases}.$$

This signal is a damped complex sinusoid when $0 < R < 1$. It oscillates with frequency ω and has an exponentially decaying amplitude envelope. If $R > 1$, the amplitude envelope increases exponentially as R^n. The signal $h(n)$ has the z transform

$$\begin{aligned} H(z) &= \sum_{n=0}^{\infty} R^n e^{j\omega nT} z^{-n} \\ &= \sum_{n=0}^{\infty} \left(Re^{j\omega T} z^{-1}\right)^n \\ &= \frac{1}{1 - Re^{j\omega T} z^{-1}} \end{aligned}$$

where the last step holds for $|Re^{j\omega T} z^{-1}| < 1$, which is true whenever $R < |z|$. Thus, the transfer function consists of a single pole at $z = Re^{j\omega T}$. Now consider what happens when we let R become greater than 1. The pole of $H(z)$ moves outside the unit circle, and the impulse response has an exponentially increasing amplitude. (Note $|h(n)| = R^n$.) Thus, the definition of stability is violated. Since the z transform exists only for $|z| > R$, we see that $R \geq 1$ implies that the z transform no longer exists on the unit circle, and hence the frequency response becomes undefined.

The argument given so far shows that a one-pole filter is stable if and only if its pole is inside the unit circle. In the case of an arbitrary transfer function, the behavior near any pole approaches that of a one-pole filter consisting of only that pole. Therefore, all poles must be inside the unit circle for stability. It is also the case that having all poles inside the unit circle is *sufficient* for ensuring the stability of any finite-order LTI filter (because any such filter can be expressed as a sum of complex one-pole filters).

Chapter 3

Examples and Important Special Cases

As a review of the analysis techniques discussed above, we will find the frequency response of four basic filters. These examples are the one-zero, one-pole, two-pole, and two-zero filters. They are important because every finite-order LTI filter expressible as difference equation such as (2.4) may be factored into a cascade of such sections. (This follows directly from the fact that every polynomial with real coefficients can be factored into first- and second-order polynomials with real coefficients.)

3.1. One-Zero Filter

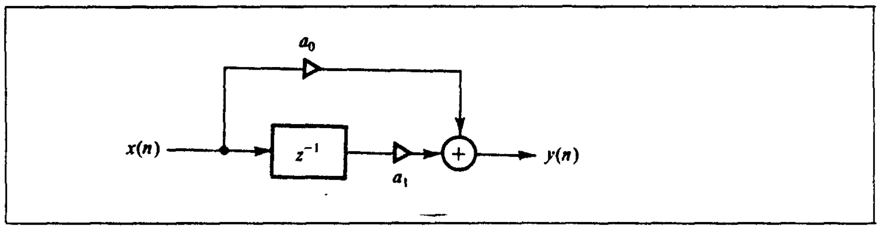

Figure 3.1. System diagram for the general one-zero filter $y(n) = a_0x(n) + a_1x(n-1)$.

Figure 3.1 gives the system diagram for the general one-zero filter. The frequency response for the one-zero filter may be found by the following steps:

Difference equation:	$y(n) = a_0x(n) + a_1x(n-1)$
***Z* transform:**	$Y(z) = a_0X(z) + a_1z^{-1}X(z)$
Transfer function:	$H(z) = \frac{Y(z)}{X(z)} = a_0 + a_1z^{-1}$
Frequency response:	$H(e^{j\omega T}) = a_0 + a_1e^{-j\omega T}$

By factoring out $e^{-j\omega T/2}$ from the frequency response, to balance the exponents of e, we can get this closer to polar form as follows:

$$\begin{aligned}
H(e^{j\omega T}) &= a_0 + a_1e^{-j\omega T}\\
&= (a_0 - a_1) + a_1 + a_1e^{-j\omega T}\\
&= (a_0 - a_1) + e^{-j\frac{\omega T}{2}}\left(a_1e^{j\frac{\omega T}{2}} + a_1e^{-j\frac{\omega T}{2}}\right)\\
&= (a_0 - a_1) + e^{-j\frac{\omega T}{2}}2a_1\cos(\frac{\omega T}{2})\\
&= (a_0 - a_1) + e^{-j\pi fT}2a_1\cos(\pi fT).
\end{aligned}$$

The term $a_0 - a_1$ may be interpreted as an order 0 filter section ("volume knob") with gain $(a_0 - a_1)$. This gain control is in parallel (i.e. summed) with a first-order section $a_1(1+z^{-1})$ having an amplitude response that varies sinusoidally from 1 to 0 as frequency goes from 0 to half the sampling rate (which is precisely the behavior of our simplest lowpass filter example, (1.1)). Figure 3.2 illustrates this interpretation of the general one-zero filter. Thus, the general one-zero filter can be interpreted as a a digital volume knob in parallel with the series combination of another gain control and the simplest lowpass filter of (1.1).

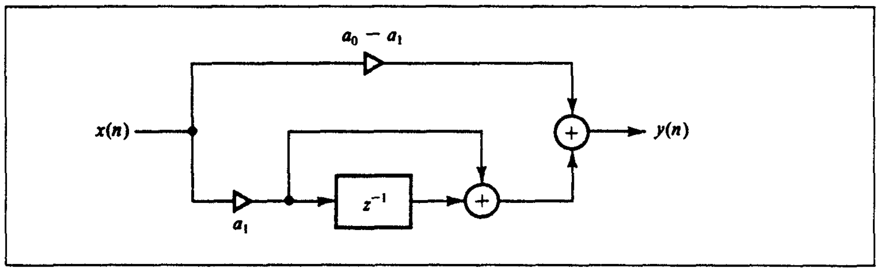

Figure 3.2. System diagram for the general one-zero filter written as $y(n) = (a_0 - a_1)x(n) + a_1[x(n) + x(n-1)]$. This is precisely the same filter as in Fig. 3.1.

Representing the general one-zero filter as a volume control in parallel with a scaled two-point average is useful for visualizing the possible range of frequency responses. For completeness, however, we now apply the general equations given in (2.16), for filter gain $G(\omega)$ and filter phase $\Theta(\omega)$ as a function of frequency:

$$\begin{aligned} H(e^{j\omega T}) &= a_0 + a_1 e^{-j\omega T} \\ &= a_0 + a_1 \cos(\omega T) - j a_1 \sin(\omega T) \\ G(\omega) &= \sqrt{[a_0 + a_1 \cos(\omega T)]^2 + [-a_1 \sin(\omega T)]^2} \\ &= \sqrt{a_0^2 + a_1^2 + 2a_0 a_1 \cos(\omega T)} \\ \Theta(\omega) &= \tan^{-1}\left(\frac{-a_1 \sin(\omega T)}{a_0 + a_1 \cos(\omega T)}\right). \end{aligned}$$

A plot of $G(\omega)$ and $\Theta(\omega)$ for $a_0 = 1$ and various values of a_1 is given in Fig. 3.3. The filter has a zero at $z = -a_1/a_0 = -a_1$ in the z-plane, which is always on the real axis. When a point on the unit circle comes close to the zero of the transfer function, the filter gain at that frequency is low. Notice that one zero can basically make either a highpass $(a_1/a_0 > 0)$ or a lowpass filter $(a_1/a_0 < 0)$. (This answers problem 2 at the end of chapter 1.) For the phase response calculation using the graphical method, it is necessary to include the pole at $z = 0$.

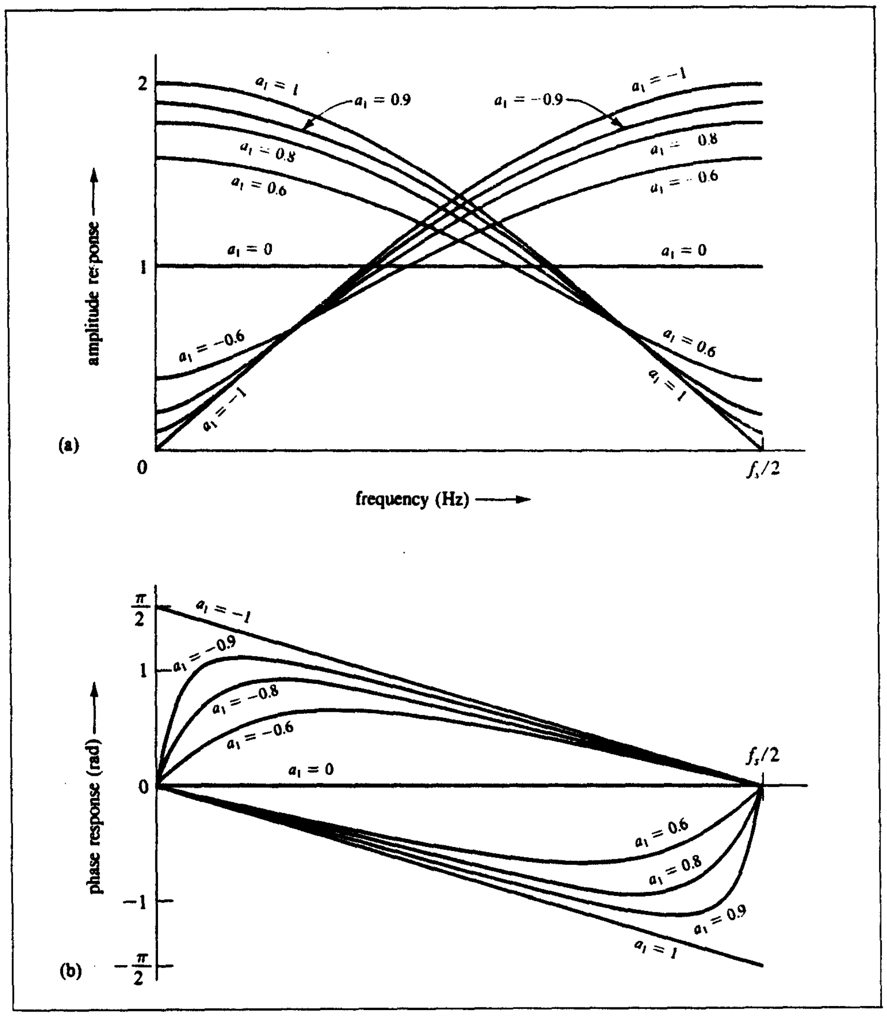

Figure 3.3. Family of frequency responses of the one-zero filter $y(n) = x(n) + a_1x(n-1)$ for various values of a_1.

a) Amplitude response.

b) Phase response.

3.2. One-Pole Filter

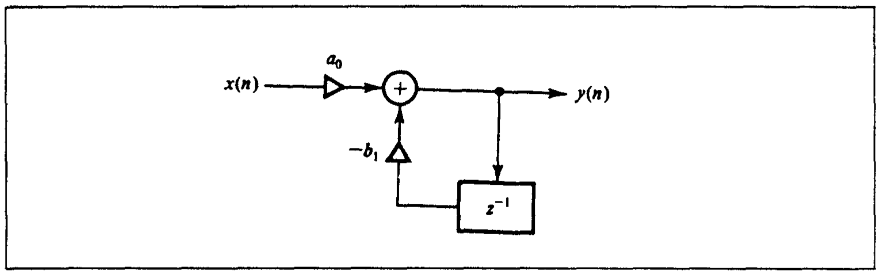

Figure 3.4. System diagram for the general one-pole filter $y(n) = a_0x(n) - b_1y(n-1)$.

Figure 3.4 gives the system diagram for the general one-pole filter. The road to the frequency response goes as follows:

Difference equation: $y(n) = a_0x(n) - b_1y(n-1)$

Z **transform:** $Y(z) = a_0X(z) - b_1z^{-1}Y(z)$

Transfer function: $H(z) = \frac{Y(z)}{X(z)} = \frac{a_0}{1 + b_1z^{-1}}$

Frequency response: $H(e^{j\omega T}) = \frac{a_0}{1 + b_1e^{-j\omega T}}$

The one-pole filter has a transfer function (hence frequency response) which is the inverse of that of a one-zero. The analysis is therefore quite analogous. A plot of the frequency response in polar form,

$$\begin{aligned} G(\omega) &= \frac{a_0}{\sqrt{[1 + b_1\cos(\omega T)]^2 + [-b_1\sin(\omega T)]^2}} \\ &= \frac{a_0}{\sqrt{1 + b_1^2 + 2b_1\cos(\omega T)}} \\ \Theta(\omega) &= -\tan^{-1}\left(\frac{-b_1\sin(\omega T)}{1 + b_1\cos(\omega T)}\right) \end{aligned}$$

for $a_0 = 1$ and various values of b_1 is given in Figure 3.5.

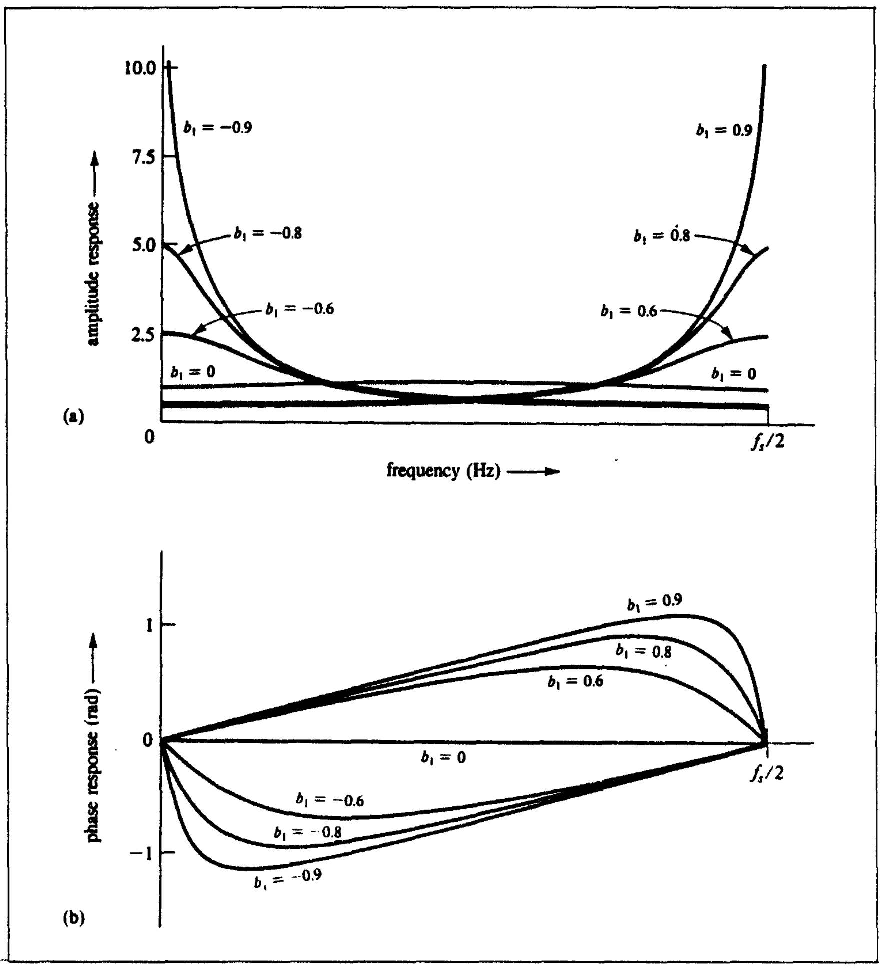

Figure 3.5. Family of frequency responses of the one-pole filter $y(n) = x(n) - b_1 y(n-1)$ for various values of b_1.

a) Amplitude response.

b) Phase response.

The filter has a pole, which is always real, at $z = -b_1$ in the z-plane (and a zero at $z = 0$). Notice that the one-pole exhibits either a lowpass or a highpass frequency response, like the one-zero. The lowpass character occurs when the pole is near the point $z = 1$ (low-frequency), which happens when b_1 approaches -1. Conversely, the highpass nature occurs when b_1 is positive.

The one-pole filter section can achieve much more drastic differences between the gain at high frequencies and the gain at low frequencies than can the one-zero filter. This difference is achieved in the one-pole by gain *boost* in the passband rather than *attenuation* in the stopband; thus, it is usually desirable when using a one-pole filter to set a_0 to a small value, such as $1 - |b_1|$, so that the peak gain is 1 or so. When the peak gain is 1, then the filter is unlikely to overflow. Finally, we note that the one-pole filter is stable if and only if $|b_1| < 1$.

3.3. Two-Pole Filter

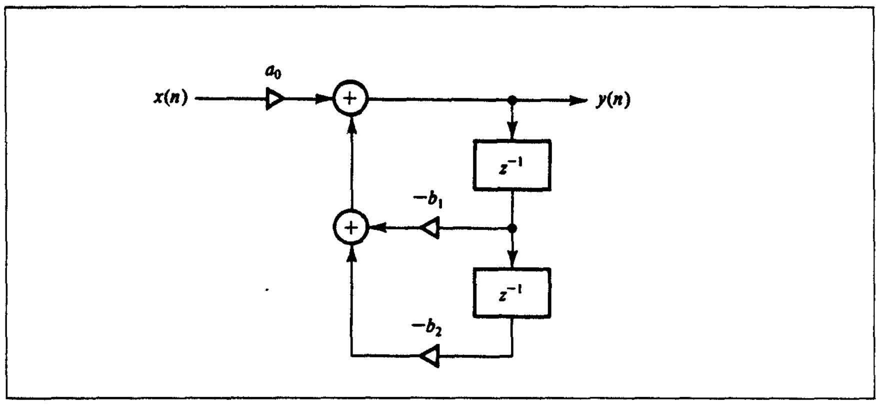

Figure 3.6. System diagram for the general two-pole filter $y(n) = a_0x(n) - b_1y(n-1) - b_2y(n-2)$.

The system diagram for the general two-pole filter is given in Fig. 3.6. We proceed as usual with the general analysis steps to obtain the following:

Difference equation:	$y(n) = a_0x(n) - b_1y(n-1) - b_2y(n-2)$
Z transform:	$Y(z) = a_0X(z) - b_1z^{-1}Y(z) - b_2z^{-2}Y(z)$
Transfer function:	$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0}{1 + b_1z^{-1} + b_2z^{-2}}$
Frequency response:	$H(e^{j\omega}) = \frac{a_0}{1 + b_1e^{-j\omega T} + b_2e^{-j2\omega T}}$

The numerator of $H(z)$ is a constant, so there are no zeros other than at the origin of the z-plane. Since there are two poles (denominator roots of $H(z)$), and the coefficients b_1 and b_2 are real, then the poles must be either real or a complex conjugate pair. When both poles are real, the two-pole is simply the cascade of two one-pole sections as analyzed in the previous section. That is, you can multiply pointwise two plots such as in Fig. 3.5 which correspond to each pole separately to get the two-pole amplitude response for two real poles.

Consider the case when the poles are not real but form a complex conjugate pair. Then we may express the poles in polar form (no pun desired) as $Re^{j\theta_c}$ and $Re^{-j\theta_c}$, where $R \geq 0$ is the common radius of both poles, and $\theta_c, -\theta_c$ are the two pole angles. In this case the filter exhibits a *resonance* at the radian frequency $\omega_c = \theta_c/T$. (A *resonance* may be loosely defined as a local peak in the amplitude response caused by a nearby pole.)

With this new notation for the filter poles, the two-pole transfer function must also be expressible as

$$\begin{aligned} H(z) &= \frac{a_0}{(1 - Re^{j\theta_c}z^{-1})(1 - Re^{-j\theta_c}z^{-1})} \\ &= \frac{a_0}{1 - 2R\cos(\theta_c)z^{-1} + R^2z^{-2}} \end{aligned} \tag{3.1}$$

from which we may identify $b_1 = -2R\cos(\theta_c)$ and $b_2 = R^2$. The difference equation may be rewritten as

$$y(n) = a_0 x(n) + [2R\cos(\theta_c)]y(n-1) - R^2 y(n-2) \tag{3.2}$$

The gain at the resonance frequency ω_c is found by substituting $z = e^{j\theta_c} = e^{j\omega_c T}$ into (3.1), which gives an upper bound on the peak gain of $1/(1-R)^2$.

Note that, for stability, we must have $R < 1$. The closer R is to 1, the higher the gain at the resonance frequency ω_c. If R is 0, the filter degenerates to $H(z) = a_0$, which is like a volume knob (constant gain versus frequency). The resonance frequency in Hz is of course given by the formula $f_c = \omega_c/2\pi$ Hz.

There is an approximate relation between bandwidth and R, for R close to 1, namely,

$$\begin{aligned} R &\approx e^{-\pi BT} \\ B &\approx -\frac{\ln(R)}{\pi T}, \end{aligned} \tag{3.3}$$

where B is the approximate bandwidth in Hz.

Thus, expressing the coefficients as in (3.2) directly relates the resonance frequency and the bandwidth of the resonator to the coefficients of the difference equation. Note that to sweep the resonance frequency around requires changing only the coefficient b_1, while changing the bandwidth (a function of R) requires both b_1 and b_2 to change.

Figure 3.7 shows a family of frequency responses for the two-pole resonator obtained by setting a_0 to 1 and varying R. The value of θ_c in all cases is $\pi/4$ (corresponding to $f_c = f_s/8$). The analytic expressions for amplitude and phase response are

$$\begin{aligned} G(\omega) &= \frac{a_0}{\sqrt{[1 + b_1\cos(\omega T) + b_2\cos(2\omega T)]^2 + [-b_1\sin(\omega T) - b_2\sin(2\omega T)]^2}} \\ \Theta(\omega) &= -\tan^{-1}\left(\frac{-b_1\sin(\omega T) - b_2\sin(2\omega T)}{1 + b_1\cos(\omega T) + b_2\cos(2\omega T)}\right), \end{aligned}$$

where $b_1 = -2R\cos(\theta_c)$ and $b_2 = R^2$.

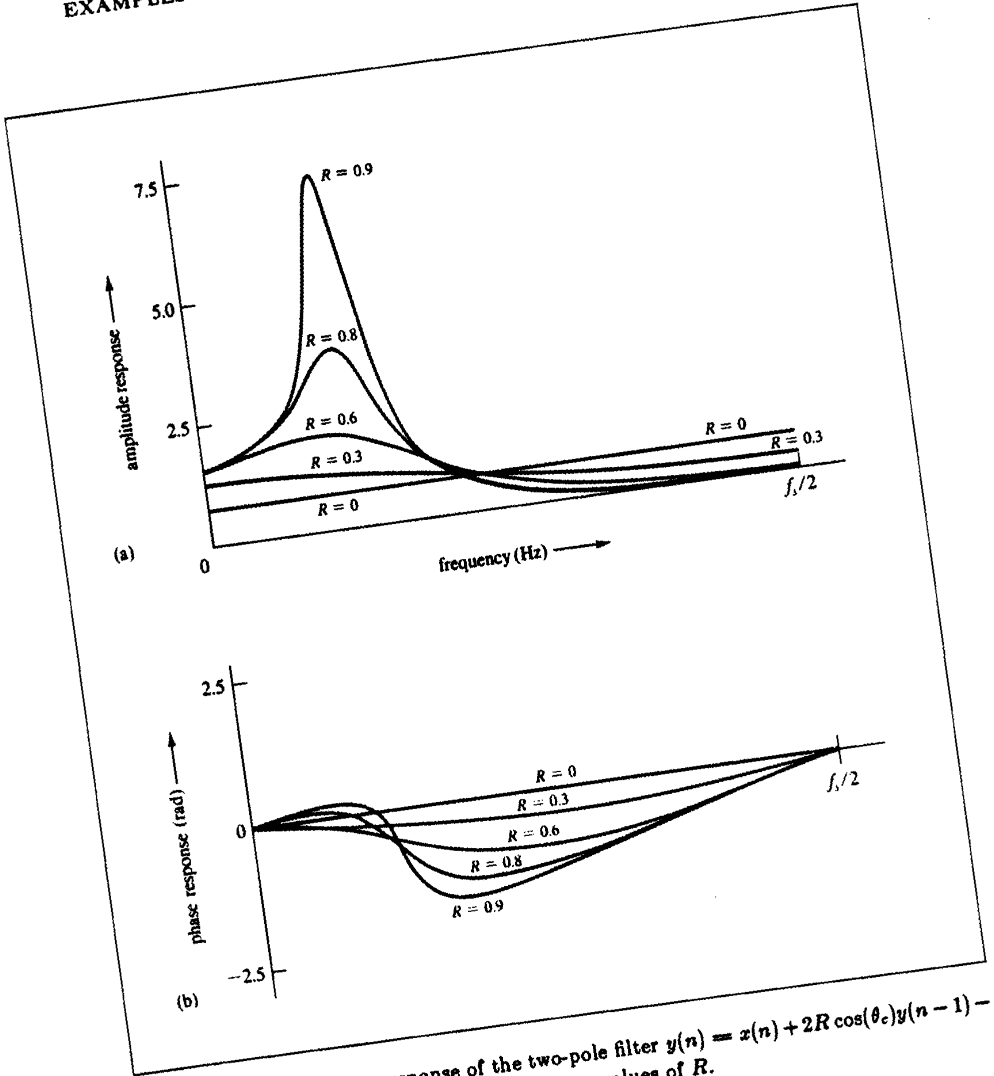

Figure 3.7. Frequency response of the two-pole filter $y(n) = x(n) + 2R\cos(\theta_c)y(n-1) - R^2y(n-2)$ with θ_c fixed at $\pi/4$ and for various values of R.
a) Amplitude response.
b) Phase response.

3.4. Two-Zero Filter

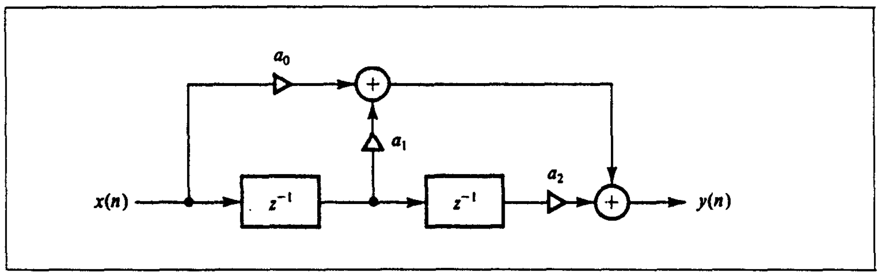

Figure 3.8. System diagram for the general two-zero filter $y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2)$.

The system diagram for the general two-zero filter is given in Fig. 3.8, and the derivation of frequency response is as follows:

Difference equation:	$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2)$
Z transform:	$Y(z) = a_0X(z) + a_1z^{-1}X(z) + a_2z^{-2}X(z)$
Transfer function:	$H(z) = \frac{Y(z)}{X(z)} = a_0 + a_1z^{-1} + a_2z^{-2}$
Frequency response:	$H(e^{j\omega}) = a_0 + a_1e^{-j\omega T} + a_2e^{-j2\omega T}$
Amplitude response:	$G(\omega) = \sqrt{[a_0 + a_1\cos(\omega T) + a_2\cos(2\omega T)]^2 + [-a_1\sin(\omega T) - a_2\sin(2\omega T)]^2}$
Phase response:	$\Theta(\omega) = \tan^{-1}\left(\frac{-a_1\sin(\omega T) - a_2\sin(2\omega T)}{a_0 + a_1\cos(\omega T) + a_2\cos(2\omega T)}\right)$

As before, if the roots of $a_0z^2 + a_1z + a_2$ are real, then the two-zero case reduces to our earlier analysis for the one-zero. Assuming the zeros to be complex, we may express the zeros in polar form as $Re^{j\theta_c}$ and $Re^{-j\theta_c}$, where $\theta_c = \omega_cT = 2\pi f_cT$.

Forming a general two-zero transfer function from this form leads to

$$\begin{aligned} H(z) &= a_0\left(1 - Re^{j\theta_c}z^{-1}\right)\left(1 - Re^{-j\theta_c}z^{-1}\right) \\ &= a_0\left[1 - 2R\cos(\theta_c)z^{-1} + R^2z^{-2}\right] \end{aligned}$$

from which we identify $a_1 = -2Ra_0\cos(\theta_c)$ and $a_2 = a_0R^2$, so that

$$y(n) = a_0\{x(n) - [2R\cos(\theta_c)]x(n-1) + R^2x(n-2)\}$$

is again the difference equation of the general two-zero filter with complex zeros. The frequency ω_c is now viewed as a notch frequency, or antiresonance frequency. The closer R is to 1, the more incisive is the notch at ω_c.

The approximate relation (3.3) between bandwidth and R given for the two-pole resonator now applies to the notch width.

Fig. 3.9 gives some two-zero frequency responses obtained by setting a_0 to 1 and varying R. The value of θ_c is again $\pi/4$. Note that the response is exactly analogous to the two-pole resonator with notches replacing the resonance peaks. Since the plots are on a linear magnitude scale, the two-zero amplitude response appears as the reciprocal of a two-pole response. On a dB scale, the two-zero response is an upside-down two-pole response.

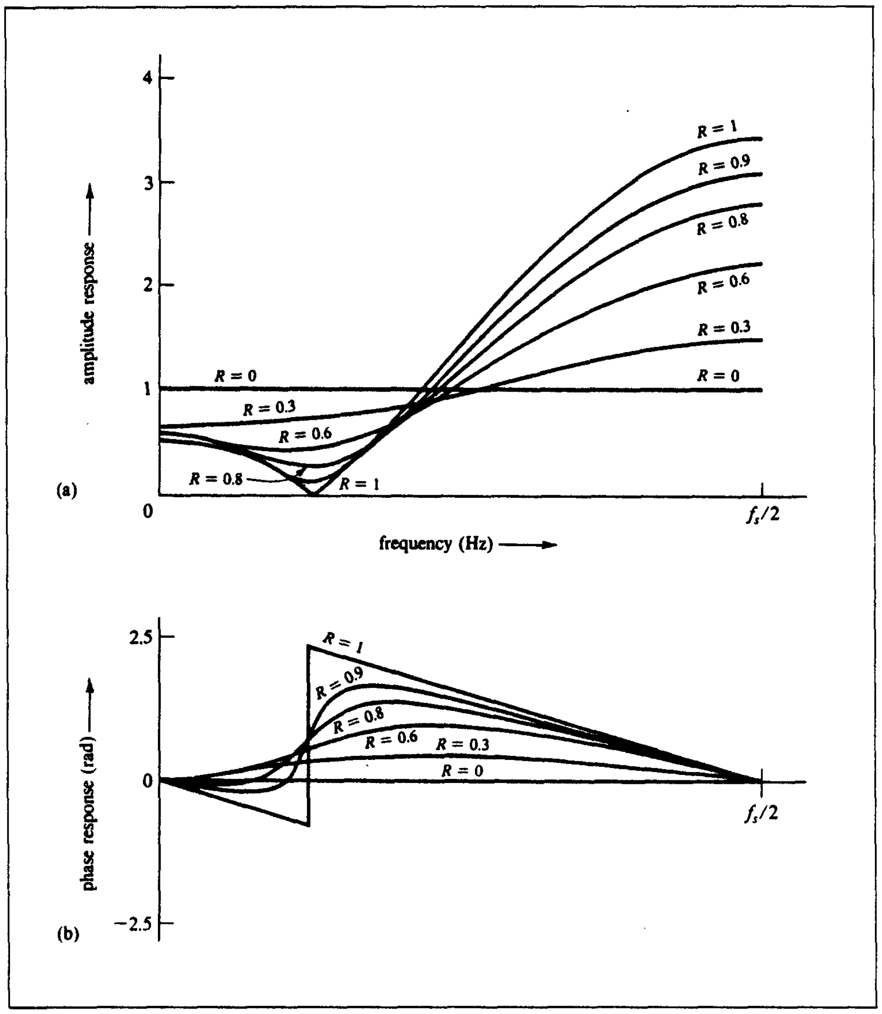

Figure 3.9. Frequency response of the two-zero filter $y(n) = x(n) - 2R\cos(\theta_c)x(n-1) + R^2x(n-2)$ with θ_c fixed at $\pi/4$ and for various values of R.

a) Amplitude response.

b) Phase response.

3.5. Conclusion

The basic tools for the analysis of digital filters have been presented. The main utility of these methods is in ascertaining how a given filter will affect the spectrum of a signal passing through it. Some of the concepts introduced include linearity, time-invariance, filter impulse response, difference equations, transfer functions, amplitude response, phase response, group delay, poles and zeros, filter stability, and the general use of complex numbers to represents signals and spectra.

However, this is still only the beginning. With these foundations, there are an unlimited number of avenues of investigation into reverberation, musical acoustics, time-varying spectral modifications, speech modeling by means of pulse-driven filters, musical instrument simulation, and so on. Given the immense range of naturally occurring filters in the domain of music, it is reasonable to suppose that filter theory will continue to provide valuable tools for the analysis and simulation of acoustical systems.

Acknowledgments

I would like to express appreciation to John Strawn for his very thorough editing assistance, and to Andy Schloss, David Jaffe, Andy Moorer, Janet Coursey, and Ken Shoemake for their helpful proofreading and suggestions. Special thanks are due also to Prof. C. S. Burrus of Rice University for his skill and patience in teaching the fundamentals of digital filter theory. The figures in this version of the paper were reprinted from [20] by permission of William Kaufmann Inc. Finally, the support of the Hertz Foundation during the time of the writing is gratefully acknowledged.

3.6. Problems

6. *Linearity*

Show that part 2) of (2.2) implies part 1) for rational g. (Recall that every rational number can be expressed as a ratio of integers.)

7. *Order*

From the definition of order given in the discussion after (2.4), determine the order of the following filter:

$$y(n) = x(n-1) + x(n-2) + y(n-1)$$

What is the order of the filter below?

$$y(n) = x(n) + x(n-1) + y(n-1)$$

Derive the frequency response for each filter. In what ways are these filters identical? How are they different?

8. *Hum Cancellation*

Design a second-order filter which will utterly reject 60 Hz hum, given a sampling rate of 40 KHz. How is the frequency response not ideal. What is a simple way to obtain a better frequency response at the price of higher filter order? (Hint: consider $\lim_{M\to\infty} |\cos(\theta)|^M$.)

9. *Allzero Form of a Recursive Filter*

Consider the difference equation for the recursive integrator, $y(n) = x(n)+y(n-1)$. Find its impulse response and write down its convolution representation. Why can you not program this form? How will the recursive form develop trouble even though you can program it? What's going on in the frequency domain?

10. *Canceling Nonlinear Phase*

A recording has been made from a condensor microphone and digitized into a disk file. The microphone introduced a wandering DC bias (a wandering average value that should be 0) which you removed with a high order recursive highpass filter. The producer has found out and threatens to wring your neck because he is fashionably afraid of nonlinear phase for reasons he does not understand. You claim that you can perfectly undo the nonlinear phase effects while making your highpassfiltering job twice as good in terms of the amplitude response (reverse-time filtering).

a) Prove that your method works by deriving the frequency response for a general filter used in the same way as your highpass filter.

b) Derive another way to accomplish the same thing without having to filter the data backward in time. (Hint: Apply the substitution $z \leftarrow z^{-1}$.) Is there anything "fishy" about this method? Pick an example recursive filter (such as (2.5)), and compute the impulse response for both the original filter and the transformed filter.

11. *Interchanging Sections*

Since the transfer functions of series-connected filter stages just multiply, the transfer function is unaffected if we permute the order of the sections. Use this fact to show that 2.1a is equivalent to 2.1b. (Hint: find a way to share delays.)

12. *The Complex Resonator*

There is nothing mandatory about real coefficients $\{a_i, b_i\}$ in (2.4). Write a Fortran program to implement a filter which has the frequency response,

$$H(e^{j\omega T}) = \frac{1}{1-(Re^{j\theta_c})e^{-j\omega T}}.$$

13. *A Graphic Equalizer*

Use the identity

$$\cos^2(\theta) + \sin^2(\theta) = 1$$

to design a two-band graphic equalizer which is transparent ($G(\omega) = 1$) when the two slide pots are even with each other. Is there any phase distortion caused by this equalizer?

14. *Two Types of Comb Filter*

Find the frequency response of

$$y(n) = a_0 x(n) + a_1 x(n-M)$$

where $M > 1$ is some integer. (If M varies with time, you get a popular device known as a *flanger.*) For $a_0 = a_1 = a$, find a general formula for the number of notches and the notch frequencies as a function of M. Next try the following filter:

$$y(n) = b_0 x(n) + b_1 y(n-M)$$

For $b_0 = b_1 = b$, find an expression for the number of peaks and the peak frequencies as a function of M. How does this allpole comb filter compare to the allzero comb filter above? How do the peak gains compare when $a = b$? (The peak gain is the maximum value assumed by $G(\omega)$.)

15. *The Allpass Filter*

Give the conditions necessary on $\{a_0, a_1, b_0, b_1\}$ in the previous problem such that the cascade of an allpole and allzero comb filter will have an amplitude response $G(\omega) = 1$, but a nondegenerate phase response. Where are the poles and zeros for this case? Is there a simple way to remember the relation between the a's and the b's? Check your rule by applying it to the numerator coefficients versus the denominator coefficients of a general transfer function and seeing if the transfer function is allpass. Can you also find the general allpass rule which specifies the zeros given the poles or vice versa?

Appendix A. Signal Representation

Units

All time is in *seconds* (sec). The continuous real-valued time variable is normally written t and the discrete integer-valued time variable is denoted n, with $t = nT$, where T is the sampling interval in seconds. Frequencies are either in cycles-per-second or radians-per-second. The name for cycles-per-second is *Hertz* (Hz). One cycle $= 2\pi$ radians $= 360$ degrees, and therefore f Hz is the same frequency as $2\pi f$ radians per second. It is easy to confuse the two because both radians and cycles are pure numbers, so that both types of frequency are in physical units of inverse seconds (sec^{-1}). As an example, a sine wave with period τ (greek "tau") seconds has frequency $f = (1/\tau)$ Hz, and radian frequency $\omega = 2\pi/\tau$ *rad/sec.* The sampling rate f_s is the reciprocal of the sampling period T. Note that since the sampling period is in seconds, the sampling rate $f_s = 1/T$ is in Hz. One might often find it helpful, however, to think "seconds-per-sample" and "samples-per-second," where "samples" is a dimensionless quantity (pure number) thrown in for clarity. The phase of a signal is written as ϕ as often as possible and is in radian units. The amplitude of a signal may be thought of in any arbitrary units such as volts, sound pressure, and the like.

Time-Waveform

We represent the time-waveform of a signal as a real-valued function of an integer, corresponding to continuous amplitude versus discrete time. This means that for any sample (numbered n), the amplitude may take on any real value. The time-waveform may appear in your mind as a kind of bar graph that extends forever to the left and right.

By convention, $x(n)$ denotes the filter input amplitude at sample n, and the output is called $y(n)$. This notation should be comfortable for anyone who has written a computer subroutine for processing samples of digitized sound. It could even be Fortran, except that lowercase is significant here.

The term *sinusoid* is used to mean a waveform of the type $A\cos(2\pi fnT + \phi) = A\cos(\omega nT + \phi)$, i.e. a cosine at amplitude A, frequency f, and phase ϕ.

Spectrum

In this paper, we think of filters primarily in terms of their effect on the spectrum of a signal. This is appropriate because the ear (to a first approximation) converts the time-waveform from the eardrum into a neurologically encoded spectrum. Mathematically, the spectrum of a signal is the Fourier transform* of its time-waveform. Equivalently, the spectrum is the z transform evaluated on the unit circle $z = e^{j\omega T}$ [13, 11]. We denote both the spectrum and the z transform of a signal by uppercase letters. For example, if the time-waveform is denoted $x(n)$, its z transform is called $X(z)$ and its spectrum is therefore $X(e^{j\omega T})$. The time-waveform $x(n)$ is said to "correspond" to its

* More precisely, the discrete-time Fourier transform. The more commonly heard term, "Discrete Fourier Transform" (DFT), refers to the one which is discrete in both time and frequency [11]. We do not encounter the DFT nor its high-speed implementation, the FFT.

z transform $X(z)$, meaning they are transform pairs. This correspondence is often denoted $x(n) \leftrightarrow X(z)$, or $x(n) \leftrightarrow X(e^{j\omega T})$. Both the z transform and its special case, the (discrete-time) Fourier transform, are said to transform from the *time domain* to the *frequency domain.*

We deal most often with discrete time nT (or simply n) but with continuous frequency f. This is because the computer can represent only digital signals, and digital time-waveforms are discrete in time but may have energy at any frequency. On the other hand, if we were going to talk about FFTs (Fast Fourier Transforms), then we would have to discretize the frequency variable in order to represent spectra inside the computer. We use spectra only for insight into the perceptual effects of digital filtering, and therefore we avoid discrete frequency as it only makes things seem more complicated than they already are.

When we wish to consider an entire signal as a "thing in itself," we write $x(\cdot)$ meaning the whole time-waveform $\{x(n), n = -\infty, \ldots, -1, 0, 1, \ldots, \infty\}$, or $X(\cdot)$ to mean the entire spectrum taken as a whole. Imagine for example that we have plotted $x(n)$ on a strip of paper that is infinitely long. Then $x(\cdot)$ refers to the complete picture while $x(n)$ refers to the n^{th} sample point on the plot.

Appendix B. Formulae Useful in Digital Filter Design

The symbol $\triangleq$ means "is defined as"; z stands for a complex number; and r, θ, x, and y are real numbers.

Complex Number Identities

$j \triangleq \sqrt{-1}$	$z \triangleq x + jy \triangleq re^{j\theta}$
$x = r\cos(\theta)$	$y = r\sin(\theta)$
$\lvert z_1 z_2\rvert = \lvert z_1\rvert\lvert z_2\rvert$	$\left\lvert\frac{z_1}{z_2}\right\rvert = \frac{\lvert z_1\rvert}{\lvert z_2\rvert}$
$\angle z_1 z_2 = \angle z_1 + \angle z_2$	$\angle\frac{z_1}{z_2} = \angle z_1 - \angle z_2$
$r = \lvert z\rvert = \sqrt{x^2+y^2}$	$\theta = \angle z = \tan^{-1}\left(\frac{y}{x}\right)$
$e^x \triangleq \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n = \sum_{n=0}^{\infty}\frac{x^n}{n!}$	$e = 2.7\,1828\,1828\,4\ldots$
$\lvert e^{j\theta}\rvert = 1$	$\angle r = 0$
$\lvert z\rvert = \lvert r\rvert\lvert e^{j\theta}\rvert = r$	$\angle z = \angle r + \angle e^{j\theta} = \theta$
$z_1z_2 = (x_1+jy_1)(x_2+jy_2)$ $= (x_1x_2 - y_1y_2) + j(x_1y_2 + x_2y_1)$	$z_1z_2 = \left(r_1e^{j\theta_1}\right)\left(r_2e^{j\theta_2}\right)$ $= (r_1r_2)e^{j(\theta_1+\theta_2)}$
$\overline{z} \triangleq x - jy = re^{-j\theta}$	$z\overline{z} = \lvert z\rvert^2 = x^2+y^2 = r^2$

Trigonometric Identities

$e^{j\theta} = \cos(\theta) + j\sin(\theta)$	$e^{jn\theta} = \cos(n\theta) + j\sin(n\theta)$
$\sqrt{e^{j\theta}} = e^{j\theta/2}$	$\cos^2\theta + \sin^2\theta = 1$
$\sin(\theta) = \frac{e^{j\theta}-e^{-j\theta}}{2j}$	$\cos(\theta) = \frac{e^{j\theta}+e^{-j\theta}}{2}$
$\sin(-\theta) = -\sin(\theta)$	$\cos(-\theta) = \cos(\theta)$
$\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$	$\sin(A+B) = \sin(A)\cos(B) + \cos(A)\sin(B)$
$\cos^2(\theta) = \frac{1}{2}[1+\cos(2\theta)]$	$\sin^2(\theta) = \frac{1}{2}[1-\cos(2\theta)]$
$\cos(A)\cos(B) = \frac{1}{2}[\cos(A+B)+\cos(A-B)]$	$\sin(A)\sin(B) = \frac{1}{2}[\cos(A-B)-\cos(A+B)]$
$\sin(A)\cos(B) = \frac{1}{2}[\sin(A+B)+\sin(A-B)]$	$\cos(A)\sin(B) = \frac{1}{2}[\sin(A+B)-\sin(A-B)]$
$\cos(A)+\cos(B) = 2\cos\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right)$	$\sin(A)+\sin(B) = 2\sin\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right)$
$\cos(A)-\cos(B) = -2\sin\left(\frac{A+B}{2}\right)\sin\left(\frac{A-B}{2}\right)$	$\sin(A)-\sin(B) = 2\cos\left(\frac{A+B}{2}\right)\sin\left(\frac{A-B}{2}\right)$
$\cos^2(A)-\cos^2(B) = -\sin(A+B)\sin(A-B)$	$\sin^2(A)-\sin^2(B) = \sin(A+B)\sin(A-B)$
$\cos^2(A)-\sin^2(B) = \cos(A+B)\cos(A-B)$	$\tan(A)+\tan(B) = \frac{\sin(A+B)}{\cos(A)\cos(B)}$

$\tan(\theta) \triangleq \frac{\sin(\theta)}{\cos(\theta)}$

$\tan\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{\sin(\theta)}$

$\cos(\theta) = \frac{1-t^2}{1+t^2}$

$\tan(\theta) = \frac{2t}{1-t^2}$

$\tan^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{1+\cos(\theta)}$

$\tan\left(\frac{\theta}{2}\right) = \frac{\sin(\theta)}{1+\cos(\theta)}$

$\sin(\theta) = \frac{2t}{1+t^2}, \qquad [t \triangleq \tan\left(\frac{\theta}{2}\right)]$

$\tan(A+B) = \frac{\tan(A)+\tan(B)}{1-\tan(A)\tan(B)}$

B.1. References

[1] Abramowitz, M., and I. A. Stegun, Ed., *Handbook of Mathematical Functions*, Dover, New York, 1965.

[2] Bers, L., *Calculus,* Holt, Rinehart, and Winston, New York, NY, 1969.

[3] Bracewell, R., *The Fourier Transform and its Applications*, McGraw-Hill, New York, 1965.

[4] Churchill, R. V., *Complex Variables and Applications*, McGraw-Hill, New York, 1960.

[5] Digital Signal Processing Committee, ed., *Programs for Digital Signal Processing,* IEEE Press, New York, 1979.

[6] Gold, B. and C. M. Rader, *Digital Processing of Signals,* McGraw-Hill, New York, 1969.

[7] Kailath, T., *Linear Systems,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1980.

[8] Markel, J. D. and B. Gold, *Linear Prediction of Speech,* Springer-Verlag, New York, 1976.

[9] Mathews, M. V., *The Technology of Computer Music,* MIT Press, Cambridge MA, 1969.

[10] Moore, F. R., "An Introduction to the Mathematics of Digital Signal Processing, Part I," Computer Music Journal vol. 2, no. 1, pp. 38–47, 1978.

[11] Moore, F. R., "An Introduction to the Mathematics of Digital Signal Processing, Part II," Computer Music Journal vol. 2, no. 2, pp. 38–60, 1978.

[12] Oppenheim, A. V. and R. W. Schafer, *Digital Signal Processing,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1975.

[13] Papoulis, A., *Signal Analysis,* McGraw-Hill, New York, 1977.

[14] Portnoff, M. R., "Implementation of the Digital Phase Vocoder Using the Fast Fourier Transform," *IEEE Trans. on Acoust., Speech, and Signal Proc.,* vol. ASSP–25, pp. 235–238, June 1977.

[15] Rabiner, L. R. and B. Gold, *Theory and Application of Digital Signal Processing,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1975.

[16] Rabiner, L. R., *Digital Processing of Speech Signals,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1978.

[17] Roads, C., and J. Strawn, eds. *Computer Music Tutorial,* MIT Press, Cambridge MA, 1985.

[18] Rosenbach, J. B. and E. A. Whitman, *College Algebra,* Ginn, New York, NY, 1949.

[19] Steiglitz, K., *An Introduction to Discrete Systems,* John Wiley and Sons, Inc., New York, 1974.

[20] Strawn, J., ed., *Digital Audio Signal Processing: An Anthology,* William Kaufmann, Inc., Los Altos, California, 1985.