CHAPTER **6**

Comb and String Filters

1 Comb filters

In this chapter we're going to explore some filters that are very useful for generating and transforming sound. By the end of the chapter we will have built an astonishly simple and efficient digital plucked-string instrument that is used all the time in computer music. In the process you'll get some practice using feedback filters, and also build up some intuition relating the behavior of comb filters to the standing waves on strings and in columns of air that we studied in Chapter 2.

In Chapter 4 we looked at inverse comb filters, but I haven't said yet where that name comes from. The inverse comb is described by the filter equation

$$y_t = x_t - R^L x_{t-L} \tag{1.1}$$

where x_t and y_t are the input and output signals, as usual. This filter uses only past inputs — no past outputs — so it's a feedforward filter. Suppose instead we consider the feedback filter that uses the past output y_{t-L} in place of $-x_{t-L}$:

$$y_t = x_t + R^L y_{t-L} \tag{1.2}$$

The reason we changed the sign of the delayed output signal will become clear shortly. To find the transfer function and frequency response, write Eq. 1.2 symbolically:

$$Y = X + R^L z^{-L} Y \tag{1.3}$$

Solving for the transfer function Y/X gives

$$\mathcal{H}(z) = \frac{1}{1 - R^L z^{-L}} \tag{1.4}$$

We'll call this feedback filter a *comb filter*.

A DSP Primer

with Applications to
Digital Audio
and
Computer Music

Ken Steiglitz

Department of Computer Science

Princeton University

Addison-Wesley Publishing Company, Inc.

Menlo Park, California • Reading, Massachusetts
New York • Don Mills, Ontario • Harlow, U.K. • Amsterdam
Bonn • Paris • Milan • Madrid • Sydney • Singapore • Tokyo
Seoul • Taipei • Mexico City • San Juan, Puerto Rico

Notice that the transfer function of this feedback filter is precisely the reciprocal of the transfer function of the feedforward filter called an inverse comb in Section 8 of Chapter 4. From this it follows that the magnitude frequency response of the comb filter is the reciprocal of that of the inverse comb. This explains the terminology. In fact, if we follow one filter with the other, the net result is to restore the original input signal; the two filters cancel each other out. Let's check this for the case of an inverse comb followed by a comb. As above, call the input to the inverse comb x and its output y. The signal y then becomes the input to the comb; call its output w. The equations for the two filters then become

$$\begin{aligned} y_t &= x_t - R^L x_{t-L} \\ w_t &= y_t + R^L w_{t-L} \end{aligned} \tag{1.5}$$

Solve the second equation for y_t and substitute in the first, yielding

$$x_t - R^L x_{t-L} = w_t - R^L w_{t-L} \tag{1.6}$$

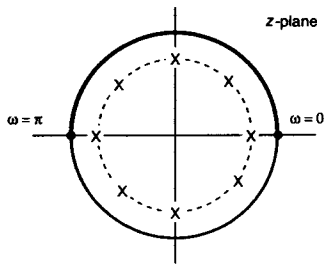

Fig. 1.1 The location of the poles of the comb filter in Eq. 1.2. The plot shown corresponds to a loop delay of $L = 8$ samples, and hence shows 8 poles.

Our goal is to show that the signals x and w are identical. Before we rush to conclude that this is implied by Eq. 1.6, there's a detail I've glossed over. We have to say something about how the filters get started, and whether the input x has been going on for all time. Let's make the simple assumption here that the signal x doesn't get started until $t = 0$ — in other words, that x_t is zero for $t < 0$. (I'll leave the fine point of what happens otherwise for Problem 1.) If this is so then Eq. 1.6 implies that $x_t = w_t$ for $t = 0, 1, \ldots, L-1$, because the delayed terms x_{t-L} and w_{t-L} are zero in that range. This implies, by the same reasoning, that $x_t = w_t$ for $t = L, L+1, \ldots, 2L-1$. We can continue the argument to infinity, block of L by block of L. In fancier terminology, this is a proof by induction on blocks of signal of length L.

Figure 1.1 shows the poles of the comb filter described by Eq. 1.2. They're exactly where the zeros of the inverse comb are (see Fig. 8.1 in Chapter 4) — at the zeros of the denominator $1 - R^L z^{-L}$, equally spaced around the circle of radius R.

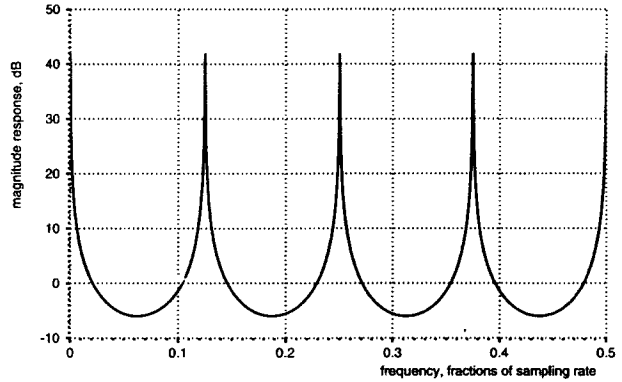

Fig. 1.2 Frequency response of the comb filter described in the previous figure. The case shown is for $R = 0.999$.

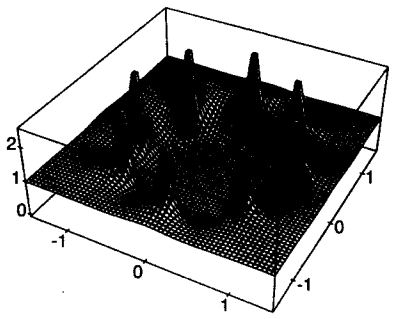

Fig. 1.3 Magnitude of the transfer function of the comb of Figs. 1.1 and 1.2, shown as a contour plot above the z-plane.

You can view the canceling of the comb and inverse comb filters as simply the poles of the comb transfer function canceling the zeros of the inverse comb transfer function. Figure 1.2 shows the magnitude frequency response of the comb for the case of a loop delay $L = 8$. It is of course just the reciprocal of the magnitude response of the inverse comb. In decibels, the reciprocal of a number becomes its negative, so one magnitude plot can be obtained by turning the other upside down, as hinted at the end of Chapter 4. It should be clear now why we call these "comb" filters.

Finally, a bird's-eye view of the magnitude response as a contour plot above the z-plane is shown in Fig. 1.3. It looks just like the pole pairs of three resons, plus another pair of poles at ± 1.

nalogy to standing waves

A comb filter works by adding, at each sample time, a delayed and attenuated version of the past output. You can think of this in physical terms: The delayed and attenuated output signal can be thought of as a returned *traveling wave*. This kind of analogy has been used in interesting ways recently by Julius Smith and Perry Cook to model musical instruments, and I've given some references to their work in the Notes at the end of this chapter. I want to introduce just the flavor of the idea here, and I'll return to this theme later when we discuss reverberation in Chapter 14.

Figure 2.1 shows the signal flowgraph for a comb filter, with some suggestion of a traveling-wave interpretation. An output wave travels around the feedback loop and returns after a delay of L samples; the return wave is added to the input, but only after it is attenuated by the factor R^L, which we'll assume is less than one in magnitude. You can think of the parameter R as the signal attenuation per sample. For example, the wave may be traveling through air, which absorbs a fraction of the wave's energy every T_s seconds, where T_s is the sampling interval. The delay L is the *round-trip* time in samples.

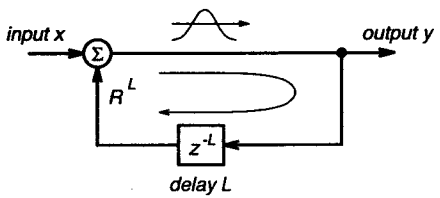

Fig. 2.1 The signal flowgraph of a comb filter with a hint of its traveling-wave interpretation.

In vibrating columns of air, waves are reflected at boundaries in two ways. Reflection at a closed end of a tube inverts the sign of a wave; reflection at an open end doesn't. What matters in Fig. 2.1 is the net effect of a round-trip, which we want to be no change in sign, being that we've chosen the sign of the feedback term to be

positive. Therefore the appropriate analogy to the comb filter is a tube of length $L/2$ (so the round-trip time is L), either closed at both ends or open at both ends. As for strings, we can't really imagine a vibrating string that is not tied down at its ends, so the analogy must be a string fixed at both ends, also of length $L/2$.

A string tied down at both ends (or a tube open or closed at both ends) has the natural resonant frequencies $k2\pi/L$ radians per sec, as shown in Eq. 5.15 in Chapter 2, where k is any integer. (That equation actually has frequencies $k\pi/L$; the factor of two is explained by the fact that here the string is of length $L/2$ instead of L.) These are precisely the angles where the poles of the comb filter occur in the z-plane. This checks our interpretation and gives us an intuitive way to understand the resonances as standing waves. The resonant frequencies are the ones that fit in the feedback loop an integer number of times, just as the standing waves are the waves that fit perfectly on the string or in the tube.

What happens if the sign of the wave is inverted in the course of a round-trip of the feedback loop? This corresponds to replacing the plus sign in Eq. 1.2 by a minus:

$$y_t = x_t - R^L y_{t-L} \tag{2.1}$$

Physically, this corresponds to the vibration of air in a tube that is closed at one end and open at the other. Recall from our work with tubes that the fundamental resonant frequency is now half of what it was with both ends closed or open, and that only odd harmonics are possible (see Eq. 7.12 in Chapter 2). Algebraically these frequencies are $k2\pi/(2L) = k\pi/L$, where k is an odd integer. (Again, there is a factor of two because now the round-trip length is L instead of $2L$.) The physical picture has sinusoids with maxima or minima at one end and zeros at the other (Fig. 7.1 in Chapter 2).

Let's check the comb filter with a sign inversion against the tube closed at one end and open at the other. The transfer function corresponding to Eq. 2.1 is

$$\mathcal{H}(z) = \frac{1}{1 + R^L z^{-L}} \tag{2.2}$$

and the poles are at roots of the equation

$$z^L = -1 \tag{2.3}$$

Since

$$-1 = e^{j\pi} \tag{2.4}$$

the roots are all shifted by an angle π/L with respect to the case of a comb filter without sign inversion, as shown in Fig. 2.2. These pole angles are in fact odd harmonics of the fundamental frequency π/L. I hope by this point you can anticipate the frequency response, shown in Fig. 2.3, from the pole pattern. Each pole near the unit circle causes a resonance peak — and the resonant frequencies of the comb are exactly the same as those of the analogous resonant tube.

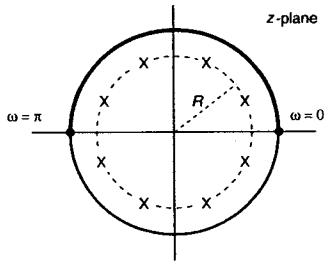

Fig. 2.2 Pole locations of an 8-pole comb filter with sign inversion around the feedback loop.

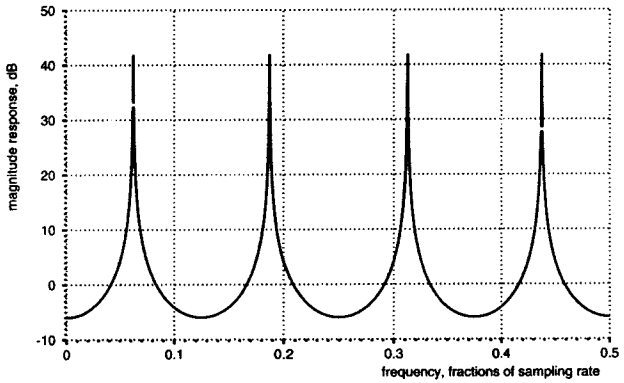

Fig. 2.3 Frequency response of the 8-pole comb filter with sign inversion. The case shown is for $R = 0.999$.

Plucked-string filters

Comb filters are versatile building blocks for making sounds of all sorts. Variations of them can be used to simulate reverberation, to transform the character of any input sound, or to construct a very striking and efficient instrument that sounds like a plucked string.

A comb filter gets you a lot for a little — a simple delay line holding 50 samples results in a filter with 25 pole-pairs, and therefore 25 resonant frequencies. The filter's frequency response is complicated, but its implementation entails only one multiplication and one addition per sample. In a sense the comb's success stems from the fact that it models a basic physical phenomenon: the return of an echo.

Next, I want to show how the physical interpretation of a comb filter can be exploited to derive the plucked-string filter. Suppose we apply a unit impulse to the comb filter in Eq. 1.2. What is the resulting impulse response? Well, the unit impulse returns L samples later, and is multiplied by the coefficient R^L. There is no other input to the delay line, so nothing else happens between time 0 and time L. The pulse of height R^L then enters the delay line and returns L samples later, and so forth. The impulse response is therefore

$$h_t = \begin{cases} R^t & \text{if } t = 0 \bmod L \\ 0 & \text{elsewhere} \end{cases} \tag{3.1}$$

That is, $h_t = R^t$ for $t = 0, L, 2L, \ldots$, and zero otherwise. This is a periodic sequence of pulses at the fundamental frequency f_s/L Hz, an integer fraction of the sampling rate — except that it decays at a rate determined by the parameter R. The closer R is to one, the slower it decays (and the closer the poles are to the unit circle). If you listen to the impulse response of a comb filter, that's exactly what you hear: a buzzy sound with pitch f_s/L that dies away. Not very exciting.

Remember that a string tied down at both ends supports standing waves because traveling waves are reflected from both those ends. The behavior of a comb filter is very similar, as noted in the previous section. We might therefore expect the impulse response of a comb filter to sound like a string that is suddenly set in motion — but it doesn't. Why not? Because the sound of a string changes in a certain way over the course of a note. This is an example of a recurrent theme in computer music and in psychoacoustics in general: sounds are not interesting unless they change their frequency content with time. Perfectly periodic waveforms are boring.

But the behavior of the comb filter does reflect the fact that a plucked string does not go on vibrating forever. Its energy is gradually dissipated to the rest of the world. Some energy is radiated in the form of sound to the air, and some is transmitted to the supports at the ends of the string, where it contributes to the radiation of sound by other parts of the instrument, like the bridge and sounding board. This decay in energy is captured by the poles of the comb filter being inside the unit circle at radius R, causing the waveform amplitude to decay by the factor R every sample (R^L in L samples).

We're still missing something critical: the insight that the different frequency components produced by a vibrating string decay *at different rates* [Karplus and Strong, 1983]. The high frequencies die away much faster than the low frequencies. This is illustrated in Fig. 3.1, which shows the spectrogram of a real plucked string, an acoustic guitar note. Notice how the high-frequency components present at the attack die away faster than the fundamental components.

Karplus and Strong suggest a very clever modification of the comb filter to take this effect into account. The idea is to insert a lowpass filter in the feedback loop so that every time the past output signal returns, its high-frequency components are diminished relative to its low-frequency components. This works like a charm. In fact it works so well it seems almost like magic.

What's more, the following very simple lowpass filter works well:

$$y_t = \tfrac{1}{2}[x_t + x_{t-1}] \tag{3.2}$$

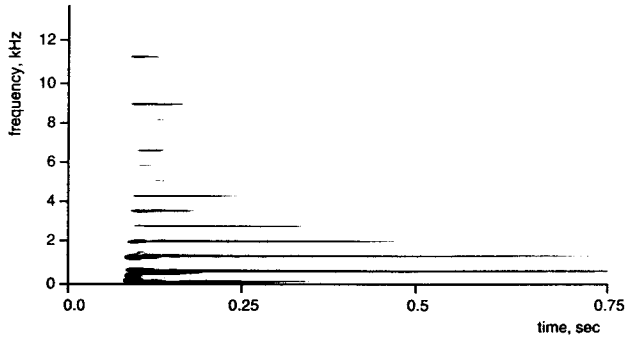

Fig. 3.1 Spectrogram of a real acoustic guitar note, the F# in the octave above middle C's octave, which corresponds to a pitch of 740 Hz. The abscissa is time in sec, and the ordinate is frequency in kHz. The horizontal bars show the harmonics of the fundamental frequency.

Except for the factor of two, we looked at the same filter in Section 7 of Chapter 4; it has the transfer function

$$\mathcal{H}(z) = \tfrac{1}{2}[1 + z^{-1}] \tag{3.3}$$

with a zero at the point in the z-plane that corresponds to the Nyquist frequency, $z = -1$. Its complex frequency response can be put in the form

$$H(\omega) = \cos(\omega/2)\, e^{j\omega(t - \frac{1}{2})} \tag{3.4}$$

The magnitude response $|H(\omega)| = |\cos(\omega/2)|$ starts at unity for zero frequency and slopes down to zero at the Nyquist frequency, as shown in Fig. 3.2. This is a modest lowpass filter, but it does the job nicely. The intuition is that the filter operates on the signal every time it executes a round-trip around the feedback loop of the comb. After m round-trips, the signal has been processed by the lowpass filter m times, so its frequency content has been multiplied by $|H(\omega)|^m$. It is therefore a good thing that the filter has a gently sloping frequency response; otherwise, the high-frequency components of the signal would get wiped out too fast.

Figure 3.3 shows the spectrogram of a note produced by the plucked-string filter just described. The abscissa shows time in sec, and the ordinate shows the frequency content for a time segment around that time. The faster decay of the higher frequencies is just what we planned.

We now come to an interesting point about phase response. In many cases, the phase response of a filter is not critical, but when the filter is in a feedback loop, as it is now, its effect on phase *can* be critical. Equation 3.4 shows that the lowpass filter has linear phase, so its delay is the same for all frequencies. In fact, the complex

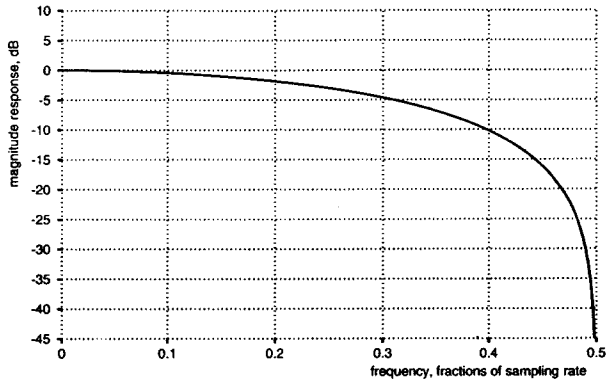

Fig. 3.2 The magnitude response of the simple lowpass feedforward filter used in the plucked-string filter.

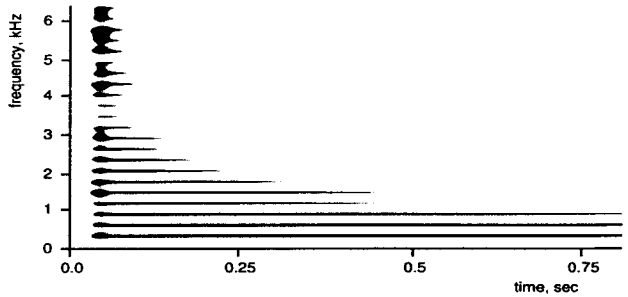

Fig. 3.3 Spectrogram of a note produced by the plucked-string filter. The abscissa shows time in sec, and the ordinate shows frequency in kHz. The parameters are $R = 0.9995$, $L = 75$, and $f_s = 22{,}050$. One hundred random numbers were used as initial input.

exponential factor is precisely equivalent to a delay of one-half sample. The loop delay is therefore $L + \frac{1}{2}$ samples, *not* L samples, and the fundamental frequency generated is $f_s/(L + \frac{1}{2})$. This is not a trivial matter; when $L = 50$, for example, the difference in frequency caused by the lowpass filter is about 1 percent — easily discernible by ear.

The plucked-string filter we have now is so nice to listen to, and so efficient, that it is one of the most commonly used computer instruments for real-time applications. Because it is so widely used, there has been a fair amount of work in tuning it up (literally and figuratively), and extending it to other kinds of sounds. We will discuss some of these ideas here, and for more information you should see [Karplus and Strong, 1983] and [Jaffe and Smith, 1983].

The filter we've constructed is a little more intricate than the simple feedforward or feedback filters we've seen so far: It consists of a feedforward loop within a feedback loop. In the next section we'll describe the filter's implementation and then take a look at its frequency response.

nplementing plucked-string filters

Figure 4.1 shows a signal flowgraph of the plucked-string filter, with its feedforward loop within its feedback loop.

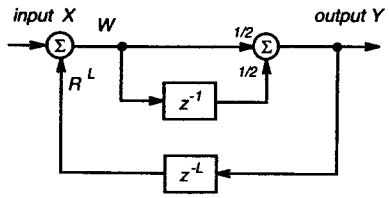

Fig. 4.1 Signal flowgraph for the plucked-string filter. Note the intermediate signal *w*.

To write the update equations for implementing the filter, it's convenient to introduce the intermediate signal w, which appears immediately after the closing of the feedback loop. The signal w is determined by the input x and the delayed and weighted output y, as follows:

$$w_t = x_t + R^L y_{t-L} \tag{4.1}$$

The output at time t is determined by the feedforward filter with input w, so

$$y_t = \tfrac{1}{2} w_t + \tfrac{1}{2} w_{t-1} \tag{4.2}$$

This is a little different from the situations we've seen up to now. The determination of the next value of the output is determined by two equations instead of one. But this presents no new difficulty. At each value of the sample number t, we first find w_t from x_t and y_{t-L}, using Eq. 4.1; then we find the output value y_t from w_t and w_{t-1}, using Eq. 4.2. Of course, just as in the simple feedforward or feedback filter, we need to save signal values for future use. In this case we need to save the past output values back to y_{t-L}, as well as the value of w at the previous sample, w_{t-1}.

You should be a little worried at this point about the possible side effects of what we did. We inserted a lowpass filter in the feedback loop of a comb to attenuate the high frequencies as they circulate around the loop. The magnitude response of the lowpass filter does have the desired effect, as we've seen from the spectrogram in Fig. 3.3. The phase response of the lowpass filter introduces an additional half-sample delay, and we argued that this makes the loop delay $L + \frac{1}{2}$ samples instead of L. But how do we know where the resonances of the altered filter really are? Are they at multiples of the fundamental frequency $f_s/(L + \frac{1}{2})$? The resonances of a filter with feedback are determined by its poles, and in the case of the simple comb, the poles are at the Lth roots of unity — equally spaced in frequency. But now the algebraic determination of the pole locations is very difficult (I don't know if it's even possible), and we are forced to look directly at the frequency response.

5 Resonances of the plucked-string filter

To look at the frequency response of the plucked-string filter we need to derive its transfer function. This is not very hard, using the same symbolic approach we used for simpler filters. Recall that a delay of one sample is represented by the operator z^{-1}; a delay of L samples by z^{-L}. In terms of these operators, Eqs. 4.1 and 4.2 become

$$W = X + R^L z^{-L} Y \tag{5.1}$$

and

$$Y = \tfrac{1}{2}[1 + z^{-1}]W \tag{5.2}$$

It is now a matter of a little algebra to solve for the ratio Y/X, the transfer function of the filter from input X to output Y. First substitute the expression for W in Eq. 5.1 into Eq. 5.2, getting an equation involving only Y and X. Then solve for Y in terms of X, yielding

$$\mathcal{H}(z) = Y/X = \frac{\tfrac{1}{2}[1 + z^{-1}]}{1 - R^L z^{-L} \tfrac{1}{2}[1 + z^{-1}]} \tag{5.3}$$

We're most interested in the magnitude response corresponding to this transfer function. This is not really hard to compute, but I want to take a little time to explain some details of the program I wrote to do it. It will be a good review of the previous two chapters. First, I multiplied the numerator and denominator of Eq. 5.3 by $2z^{L+1}$, to get the transfer function in the less confusing and more conventional form of a ratio of polynomials:

$$\mathcal{H}(z) = \frac{z^{L+1} + z^L}{2z^{L+1} - R^L z - R^L} \tag{5.4}$$

We want to evaluate this for z on the unit circle, so I then replaced z and its powers using Euler's formula:

$$\begin{aligned} z &= \cos\omega + j\sin\omega \\ z^L &= \cos(L\omega) + j\sin(L\omega) \\ z^{L+1} &= \cos((L+1)\omega) + j\sin((L+1)\omega) \end{aligned} \tag{5.5}$$

It's then easy to write out explicitly the real and imaginary parts of the numerator and denominator:

$$\begin{aligned} \mathit{Real}\,\{\text{numerator}\} &= \cos((L+1)\omega) + \cos(L\omega) \\ \mathit{Imag}\,\{\text{numerator}\} &= \sin((L+1)\omega) + \sin(L\omega) \\ \mathit{Real}\,\{\text{denominator}\} &= 2\cos((L+1)\omega) - R^L\cos\omega - R^L \\ \mathit{Imag}\,\{\text{denominator}\} &= 2\sin((L+1)\omega) - R^L\sin\omega \end{aligned} \tag{5.6}$$

where *Real* and *Imag* denote the real and imaginary parts, respectively. I assigned temporary variables for these four components, the real and imaginary parts of the numerator and denominator. The magnitude response is the magnitude of this as a complex function, or

$$|H(\omega)| = \left[\frac{[\mathit{Real}\,\{\text{numerator}\}]^2 + [\mathit{Imag}\,\{\text{numerator}\}]^2}{[\mathit{Real}\,\{\text{denominator}\}]^2 + [\mathit{Imag}\,\{\text{denominator}\}]^2}\right]^{1/2} \tag{5.7}$$

I then just evaluated this for ω on a grid in the range from 0 to π radians per sample.

Figure 5.1 shows the result when $L = 32$ and the coefficient $R = 0.999$. Since the round-trip delay of the feedback loop is 32.5 samples, we expect the resonances to occur at integer multiples of $f_s/32.5$, and these frequencies are marked by triangles on the graph.

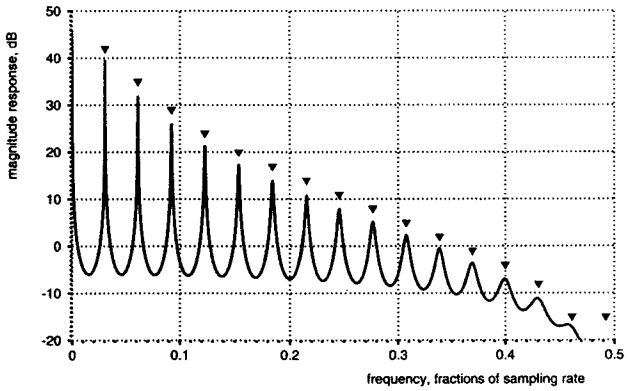

Fig. 5.1 Magnitude response of a plucked-string filter, for the case of a loop length $L = 32$ and pole radius $R = 0.999$. The expected resonant frequencies, integer multiples of $f_s/32.5$, are indicated by triangles.

The first interesting thing to notice in Fig. 5.1 is that the resonance peaks increase in width as frequency increases. This is exactly what we should expect, since the lowpass filter inserted in the loop causes higher frequencies to decay faster. Wider resonance peaks correspond to poles farther from the unit circle, and to signal components that decay faster.

Second, the peaks in the magnitude response line up very closely with the predicted integer multiples of $f_s/32.5$. The peaks are not precisely at the expected points, and they also are not at exact integer multiples of the frequency of the first peak. This deviation of the overtone series from a simple harmonic progression is smaller when the harmonic numbers are lower, and also when the loop delay L is larger (corresponding to a lower fundamental frequency). But the deviations, especially at the lower harmonics, are very tiny. Tinier than we usually need to worry about. For example, in our example with a loop delay of 32 samples, the tenth harmonic is off by only 0.027 percent. The deviations for lower harmonics and lower-pitched filters are even smaller.

A third noticeable difference in the magnitude response of the plucked-string filter, compared to a comb filter, is its general lowpass shape. The peaks decrease in amplitude with increasing frequency, whereas the peaks of the comb filter are all of equal height. This is not surprising, since we inserted a lowpass filter in the path between input and output.

The plucked-string filter is so useful for musical purposes that we will want to be able to tune its pitch very finely. That leads us to the first-order allpass filter, a useful and interesting filter in its own right.

6 The first-order allpass filter

At this point we have only crude control over the pitch of a plucked-string filter. We can choose the integer loop length L, yielding a fundamental pitch $f_s/(L + \frac{1}{2})$, but that integer L is all we have to work with. To see just how crude this control of pitch is, let's see what happens when $L = 10$. This is a perfectly reasonable example, by the way; if the sampling rate is 22,050 Hz, a loop length of 10 corresponds to a pitch of 22,050/10.5 = 2100 Hz, which is very close to the C three octaves above middle C. Now suppose we decrease L by one. This increases the pitch to 22,050/9.5 = 2321.05 Hz, which is almost up to the following D, a jump of almost a full step in the scale. We appear to be in real trouble if we want to produce the C# between the two. Smooth glissandos seem to be out of the question. Getting better control over the pitch of the plucked-string filter presents an interesting problem, which we'll now address.

Intuitively, the fundamental resonant frequency of the plucked-string filter is determined by the total delay around the feedback loop. If the total delay is D samples, or DT_s sec, the first resonant frequency is $1/(DT_s) = f_s/D$ Hz. We haven't said anything about D being an integer number of samples. In fact, in the plucked-string filter we have so far, D is the sum of the integer buffer length, L, plus one-half sample due to the lowpass filter, so D is *not* an integer. What we would like is a way to introduce additional delays of *fractions* of a sample period in the feedback loop. That would enable us to fine-tune the delay D and hence the pitch.

In fact, what we'd like is a filter that introduces, or comes close to introducing, an arbitrary fractional delay, but has no effect on the magnitude of the frequency

response around the feedback loop. We already have a loop with the lowpass characteristic we want for the plucked-string sound, and we don't want to tamper with a good thing. The idea is to try to construct a filter that has no effect on the magnitude of phasors, no matter what their frequency. Suppose we start with a pole at $z = \rho$, where ρ is some real number. Maybe we can add a zero to the filter transfer function so that the effect of the pole on the magnitude response will be canceled. Where should we put the zero? One answer is: the same place — that will cancel the effect of the pole perfectly. But, of course, that accomplishes nothing; it gets us back to a unity transfer function, and has no effect on the phase response.

Putting the zero at $-\rho$ doesn't do the trick. If ρ is positive, for example, the pole will have a lowpass effect, and a zero at $-\rho$ will have the same effect. The result will be to exaggerate rather than cancel the effect of the pole.

There aren't many other places to try. How about putting the zero at $z = 1/\rho$? That does put the zero closer to the lower than the higher frequency points on the unit circle, so its effect will be highpass — opposite that of the pole. This sounds promising. Let's look at the magnitude response, using Fig. 6.1. The vector from the pole to an arbitrary point on the unit circle is labeled with length B, and the corresponding vector from the zero is labeled with length C. The point on the unit circle is at frequency θ radians per sample.

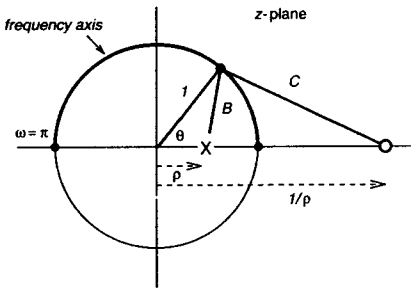

Fig. 6.1 Pole-zero diagram and some geometry for the first-order allpass section.

Recall that the magnitude response at frequency θ due to a zero is the length of the vector from the zero to the point on the unit circle at angle θ (Section 6 of Chapter 4). Similarly, the magnitude response due to the pole is the reciprocal of the length of the vector from the pole to the point on the unit circle. Therefore the magnitude response of the filter with both the zero and pole is the ratio of these lengths, C/B. Let's try to put this in terms of θ and the constant ρ.

We can express B in terms of ρ and θ using the law of cosines:

$$B^2 = 1 + \rho^2 - 2\rho\cos\theta \tag{6.1}$$

Similarly, we can write C in terms of ρ and θ using the same law:

$$C^2 = 1 + 1/\rho^2 - 2(1/\rho)\cos\theta \tag{6.2}$$

Multiplying Eq. 6.2 by ρ^2 yields the right-hand side of Eq. 6.1, so

$$\rho^2 C^2 = B^2 \tag{6.3}$$

or, forming the square of the magnitude response C/B:

$$C^2/B^2 = 1/\rho^2 \tag{6.4}$$

This is even better than we could have hoped for: The magnitude of the frequency response is perfectly independent of frequency! This sounds almost magical, but it is correct: if you place a zero at the reciprocal of the real pole position, the filter has a magnitude response that is absolutely constant with respect to frequency. All frequencies are passed with equal weight. We call such filters *allpass* filters.

Before we go on, remember that it is perfectly acceptable to have a zero outside the unit circle. A pole outside the unit circle causes instability, as noted in Section 2 of Chapter 5. But zeros are tamer creatures, and we can put them anywhere in the z-plane. This makes the allpass construction feasible.

We want to look at the phase response of our single-pole, single-zero allpass filter, but first let's construct the transfer function corresponding to the pole and zero in Fig. 6.1:

$$\mathcal{H}(z) = K\,\frac{z + 1/a}{z + a} \tag{6.5}$$

where K is any constant, and we've used the parameter a to avoid minus signs; the pole is at the point $z = -a$. We have the freedom to choose the constant factor K any way we want; it is convenient to choose it to force the transfer function to have the value one at zero frequency, the point $z = 1$. Setting $\mathcal{H}(1) = 1$ gives us $K = a$, and hence the transfer function is

$$\mathcal{H}(z) = \frac{z^{-1} + a}{1 + az^{-1}} \tag{6.6}$$

As usual, we've written the transfer function in terms of z^{-1}, the delay operator.

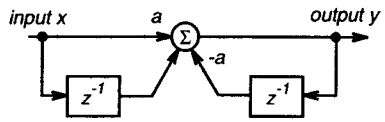

Fig. 6.2 Signal flowgraph for a first-order allpass filter.

The allpass filter we've just derived is a combination feedforward and feedback filter. If we choose to implement the feedforward part before the feedback part, we get the signal flowgraph shown in Fig. 6.2, corresponding to the filter equation

$$y_t = ax_t + x_{t-1} - ay_{t-1} \tag{6.7}$$

See Problem 5 for a more efficient implementation.

Allpass phase response

We finally get to the phase response of the allpass filter, the reason we started looking at it in the first place. When we get its phase response ϕ, it will tell us how much a phasor of frequency ω applied to the filter will be delayed. To be specific, suppose we apply the phasor $e^{j\omega t}$ as input. The output signal will be the phasor of unit magnitude with its phase shifted by $\phi(\omega)$:

$$e^{j\omega t}e^{j\phi(\omega)} = e^{j\omega(t + \phi(\omega)/\omega)} \tag{7.1}$$

The right-hand side of Eq. 7.1 shows that the phasor is shifted by $\phi(\omega)/\omega$ samples. The phase response $\phi(\omega)$ is usually negative, so $-\phi(\omega)/\omega$ represents a delay, called the *phase delay*.[†] In general, this phase delay is a function of the frequency ω. All this checks with the discussion in Section 7 of Chapter 4, where we pointed out that *exactly* linear phase in a feedforward filter results in a constant delay. What we're looking for in the allpass filter is a phase response that is at least approximately linear. Remembering that, let's find $\phi(\omega)$ for the allpass filter.

The way to start calculating either the magnitude or the phase response is to replace z by $e^{j\omega}$ in the transfer function, Eq. 6.6, to get the frequency response

$$H(\omega) = \frac{e^{-j\omega} + a}{1 + ae^{-j\omega}} \tag{7.2}$$

We could now find the phase response $\phi(\omega)$ by finding the real and imaginary parts of the numerator and denominator, and using the arctangent function as follows

$$\phi(\omega) = \arctan\left[\frac{\mathit{Imag}\,\{\text{numerator}\}}{\mathit{Real}\,\{\text{numerator}\}}\right] - \arctan\left[\frac{\mathit{Imag}\,\{\text{denominator}\}}{\mathit{Real}\,\{\text{denominator}\}}\right] \tag{7.3}$$

in analogy to the magnitude calculation we did in Section 5. Instead, I'm going to be a little tricky, in order to get the result in a particularly convenient form.

The idea is to try to introduce some symmetry in Eq. 7.2 by multiplying the numerator and denominator by $e^{j\omega/2}$:

$$H(\omega) = \frac{e^{-j\omega/2} + ae^{j\omega/2}}{e^{j\omega/2} + ae^{-j\omega/2}} \tag{7.4}$$

A good thing has now happened: We've succeeded in making the denominator very similar to the numerator. In fact, the only difference between them is that one is the complex conjugate of the other. If we set the numerator to $re^{j\psi(\omega)}$, the denominator is $re^{-j\psi(\omega)}$, and the ratio can be written

$$H(\omega) = \frac{re^{j\psi(\omega)}}{re^{-j\psi(\omega)}} = e^{2j\psi(\omega)} \tag{7.5}$$

[†] The term *phase delay* is used to distinguish this from *group delay*. See the Notes at the end of this chapter.

This shows that the phase response $\phi(\omega)$ is simply $2\psi(\omega)$, twice the phase angle of the numerator in Eq. 7.4. (As a side effect, it also confirms that the magnitude of the transfer function is unity.) The numerator can be written

$$(a + 1)\cos(\omega/2) + j(a - 1)\sin(\omega/2) \tag{7.6}$$

so the phase response of the allpass, finally, is

$$\phi(\omega) = -2\arctan\left[\frac{1 - a}{1 + a}\tan(\omega/2)\right] \tag{7.7}$$

This form for the phase is compact and pretty, but it's also particularly illuminating if we focus our attention on low frequencies. When x is small, $\tan x \approx x$, and this gives us the following low-frequency approximation

$$\phi(\omega) \approx -\frac{1 - a}{1 + a}\omega \approx -\delta\omega \tag{7.8}$$

where we've defined

$$\delta = \frac{1 - a}{1 + a} \tag{7.9}$$

The variable δ is an approximation to the phase delay $-\phi(\omega)/\omega$. From our discussion at the beginning of this section, the phase delay of the allpass filter is approximately equal to δ for low frequencies. We can also solve for a in terms of δ:

$$a = \frac{1 - \delta}{1 + \delta} \tag{7.10}$$

which is a handy formula if we specify the phase delay.

In practice a must always be less than one (why?), so δ is always positive. Furthermore, there is not much point in trying to approximate delays greater than one sample with the allpass, because we can always take care of the integer part by absorbing it into the buffer used to implement the loop delay, the integer L. We can therefore restrict δ to the range between 0 and 1, which is equivalent to restricting a to the same range.

Figure 7.1 shows plots of the phase response of the allpass filter for the ten values of a corresponding to $\delta = 0.1, 0.2, \ldots, 1.0$ samples. As predicted, the phase looks linear at low frequencies, with slope approximately equal to $-\delta$. The phase delay $-\phi(\omega)/\omega$ gives us a better idea of the quality of the approximation, and is plotted in Fig. 7.2 for the same range of δ. We see that the allpass delivers close to the desired delay at low frequencies. The errors are quite small for frequencies below $0.05f_s$. At the frequency $0.05f_s$, for example, which is 1102.5 Hz at a sampling rate of 22,050, the error is only 0.0031 samples for $\delta = 0.5$ samples. At the higher frequency of $0.2f_s$, the error is up to 0.0546 samples at the same δ.

Notice also from Fig. 7.2 that the allpass filter's approximation to constant delay is better for values of delay near 0 or 1 sample than it is near 0.5 samples. Think of it this way: a delay of a fraction of a sampling interval actually interpolates the signal between sample values. Interpolating midway is most difficult, because that point is

farthest from known sample values. Still, the one-zero, one-pole allpass filter does a reasonably good job at all delays for low frequencies.

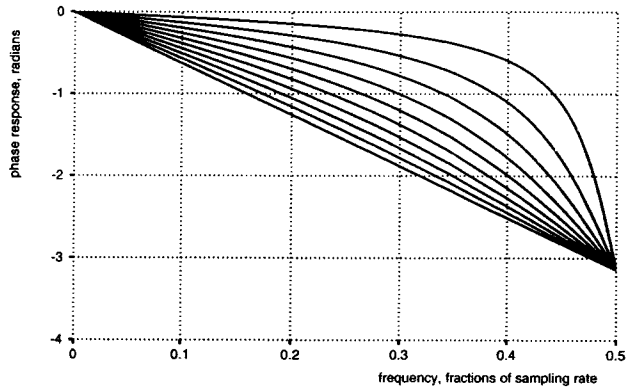

Fig. 7.1 Phase response for the first-order allpass filter; from the top, the prescribed delays δ are 0.1, 0.2, . . . ,1.0 samples.

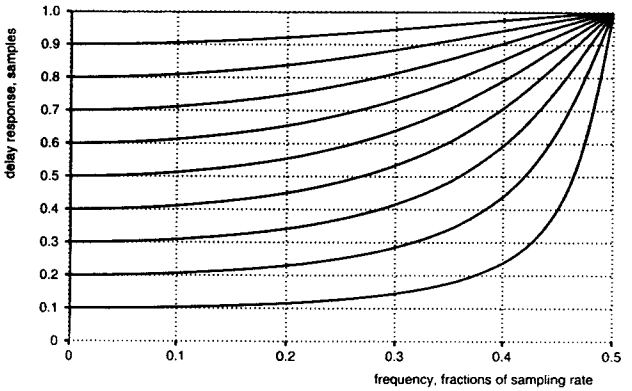

Fig. 7.2 Delay response for the first-order allpass filter; from the bottom, the prescribed delays δ are 0.1, 0.2, . . . ,1.0 samples.

All our work on the allpass filter has paid off. We've shown that it provides an effective and efficient way to tune the delay in a feedback loop. Now let's put the plucked-string instrument together.

8 Tuning plucked-string filters

Figure 8.1 shows the finished plucked-string filter. We have two main points left to discuss: the final details of tuning and the selection of the input signal.

The tuning of all the harmonics simultaneously with the allpass filter is not possible; from the plot of phase delay we see that the upper harmonics will have greater relative delay than the fundamental. That is, the upper partials will be flat. Jaffe and Smith [1983] suggest that this is not such a bad thing perceptually, and recommend tuning the filter so that the fundamental frequency is exactly correct. Let's run through an example to see how they do this.

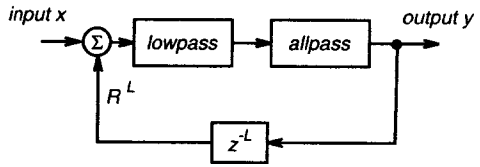

Fig. 8.1 Tunable plucked-string filter.

Suppose we want the lowest resonance of the plucked-string filter to occur at precisely 1000 Hz, using a sampling rate of 22,050 Hz. This corresponds to a loop delay of 22,050/1000 = 22.05 samples. Remember that the loop delay due to the buffer and lowpass filter is $L + \frac{1}{2}$ samples, so we should choose $L = 21$ to keep the δ of our allpass filter in the range between zero and one sample. We then wind up with a desired phase delay of $\delta = 22.05 - 21.5 = 0.55$ samples.

At this point we could use the approximate formula for the allpass filter parameter a in terms of specified phase delay, Eq. 7.10. But it would be best if we could get the frequency of the fundamental resonance exactly right. To do this, we stipulate that the negative of the exact phase response at the frequency ω_0 (from Eq. 7.7), divided by the frequency ω_0, be equal to the desired phase delay δ:

$$\delta = \frac{2}{\omega_0}\arctan\left[\frac{1-a}{1+a}\tan(\omega_0/2)\right] \tag{8.1}$$

By a stroke of luck, we can solve this exactly for the allpass filter parameter a in terms of δ [Jaffe and Smith, 1983]:

$$a = \frac{\sin((1-\delta)\omega_0/2)}{\sin((1+\delta)\omega_0/2)} \tag{8.2}$$

Notice that for small ω_0, this reduces to the approximate formula $(1 - \delta)/(1 + \delta)$, as we would expect. Our example, with $\omega_0 = 2\pi \cdot 1000/22{,}050$ radians and $\delta = 0.55$ samples, results in $a = 0.292495$. The approximate formula for a, Eq. 7.10, yields a phase delay of 0.552607 samples instead of the target 0.55 samples, about 0.5 percent high. Of course, the relative error in terms of the total loop delay of 22.05 samples is much smaller.

There are quite a few twists on the basic plucked-string filter idea, many mentioned in [Karplus and Strong, 1983] and [Jaffe and Smith, 1983]. I'll mention some of them in the Problems. But we've skipped a basic one, which I'll mention here: The initial excitation of the filter should be chosen to provide lots of high frequencies. This lends verisimilitude to the resulting note, ostensibly because the initial vibration of a real string has a healthy dose of high-frequency energy. The usual way to accomplish this is to start with an initial burst of random numbers, which I did to produce Fig. 3.3.

The output of the plucked-string filter is almost, but not quite, periodic. In some sense its musical quality depends both on its being close to periodic (so that it has a pitch), and on its not being periodic (so that it's interesting). In the next chapter we're going to develop the mathematical tools we need to understand perfectly periodic signals, paving the way for dealing with completely general signals.

lotes

Julius Smith has pioneered work on applying waveguide analogies to computer music. The following is a useful general article, with lots of references to related work:

> J. O. Smith, "Physical Modeling using Digital Waveguides, *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, Winter 1992.

Perry Cook has developed some striking applications based on these ideas. See, for example:

> P. R. Cook, "Tbone: An Interactive WaveGuide Brass Instrument Synthesis Workbench for the NeXT Machine," *Proc. International Computer Music Conf*, San Francisco, International Computer Music Association, pp. 297–300, 1991.

> P. R. Cook, "SPASM, a Real-Time Vocal Tract Physical Model Controller; and Singer, the Companion Software Synthesis System," *Computer Music Journal*, vol. 17, no. 1, pp. 30–44, Spring 1993.

The following back-to-back articles are a rich source of interesting ideas for extending and improving the basic plucked-string filter.

> [Karplus and Strong, 1983] K. Karplus and A. Strong, "Digital Synthesis of Plucked-String and Drum Timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43–55, Summer 1983.

> [Jaffe and Smith, 1983] D. A. Jaffe and J. O. Smith, "Extensions of the Karplus-Strong Plucked-String Algorithm," *Computer Music Journal*, vol. 7, no. 2, pp. 56–69, Summer 1983.

Paul Lansky's pieces "Night Traffic" and "Sound of Two Hands" are intriguing examples of using comb filters in computer music. His "Now and Then" uses plucked-string filters. These and other pieces are on his CD *HomeBrew*, Compact Disc BCD 9035, Bridge Records, 1992. Lansky uses digital signal processing and algorithmic techniques in much of his music. He comments in the liner notes to this disc that these pieces ". . . are attempts to view the mundane, everyday noises of daily life through a personal musical filter."

Add nonlinear distortion and feedback to the plucked-string filter and you get a versatile digital version of a rock guitar. Charles Sullivan shows how in the following paper, which makes ingenious use of many of the ideas we've studied up to now:

> C. R. Sullivan, "Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback," *Computer Music Journal*, vol. 14, no. 3, pp. 26–37, Fall 1990.

The family of allpass filters mentioned in Problem 9 is derived in

> [Fettweis, 1972] A. Fettweis, "A Simple Design of Maximally Flat Delay Digital Filters," *IEEE Trans. on Audio and Electroacoustics*, vol. AU-20, pp. 112–114, June 1972.

That paper actually provides a simple derivation of earlier results of J.-P. Thiran; for example

> J.-P. Thiran, "Recursive Digital Filters with Maximally Flat Group Delay," *IEEE Trans. on Circuit Theory*, vol. CT-18, pp. 659–664, Nov. 1971.

The distinction between phase and group delay is discussed in

> A. Papoulis, *The Fourier Integral and its Applications*, McGraw-Hill, New York, N.Y., 1962.

Problems

1. An inverse comb filter is followed by a comb filter with the same parameter, as in Eq. 1.5. Construct an input signal x for which the output w of the comb filter is different from x. Hint: From the discussion x must be nonzero for arbitrarily negative t.

2. Here's a project for video gamesters. Write an interactive flight simulator whose landscape is the magnitude response of a feedback filter over the z-plane. Don't try to go over a pole!

3. [Karplus and Strong, 1983], [Jaffe and Smith, 1983] Is the plucked-string filter stable if we use the value $R = 1$? (If you want a hint, peek at the next problem.)