Chapter 12

Multirate Signal Processing

In our study of discrete-time signals and systems, we have assumed that all signals in a given system have the same sampling rate. We have interpreted the time index n as an indicator of the physical time nT, where T is the sampling interval. A *multirate system* is characterized by the property that signals at different sampling rates are present. An example that you may be familiar with, although perhaps not aware of its meaning, is the audio compact-disc player. Today's CD players often carry the label "8X oversampling" (or a different number). Thus, the digital signal read from the CD, whose sampling rate is 44.1 kHz, is converted to a signal whose sampling rate is 8 times higher, that is, 352.8 kHz. We shall get back to this example in due course and explain the reason for this conversion.

Multirate systems have gained popularity since the early 1980s, and their uses have increased steadily since. Such systems are used for audio and video processing, communication systems, general digital filtering, transform analysis, and more. In certain applications, multirate systems are used out of necessity; in others, out of convenience. One compelling reason for considering multirate implementation for a given digital signal processing task is computational efficiency. A second reason is improved performance.

The two basic operations in a multirate system are decreasing and increasing the sampling rate of a signal. The former is called decimation, or down-sampling. The latter is called expansion, or up-sampling. A more general operation is sampling-rate conversion, which involves both decimation and expansion. These are the first topics discussed in this chapter: first in the time domain and then in transform domains.

Proper sampling-rate conversion always requires filtering. Linear filters used for sampling-rate conversion can be implemented efficiently. It turns out that sampling-rate conversion is sometimes advantageous in digital filtering even when the input and the output of the filter are needed at the same rate. This happens when the filter in question has a small bandwidth compared with the input sampling rate. Linear filtering in multirate systems is the next topic discussed in this chapter.

The second major topic of the chapter is *filter banks.* A filter bank is an aggregate of filters designed to work together and perform a common task. A typical filter bank has either a single input and many outputs, or many inputs and a single output. In the former case it is called an analysis bank; in the latter, a synthesis bank. Analysis banks are used for applications such as splitting a signal into several frequency bands. Synthesis banks are most commonly used for combining signals previously split by an analysis bank. The simplest form of a filter bank is the two-channel bank, which splits a signal into two, or combines two signals into one. We present several types of two-channel filter bank and discuss their properties and applications. Finally, we extend our discussion to more general filter banks. The subject of filter banks is rich, and our treatment of it serves only as a brief introduction.

12.1 Decimation and Expansion

Decimation can be regarded as the discrete-time counterpart of sampling. Whereas in sampling we start with a continuous-time signal $x(t)$ and convert it to a sequence of samples $x[n]$, in decimation we start with a discrete-time signal $x[n]$ and convert it to another discrete-time signal $y[n]$, which consists of *subsamples* of $x[n]$. Thus, the formal definition of M-fold *decimation*, or *down-sampling*, is

$$y[n] = x[nM], \quad n \in \mathbb{Z}. \tag{12.1}$$

Figure 12.1 illustrates 3-fold decimation. Note that the samples corresponding to $n = \dots, -2, 1, 4, \dots$ and $n = \dots, -1, 2, 5, \dots$ are lost in the decimation. In general, the samples of $x[n]$ corresponding to $n \neq kM$ are lost in M-fold decimation. This is similar to point sampling of a continuous-time signal $x(t)$, in which values corresponding to $t \neq nT$ are lost.

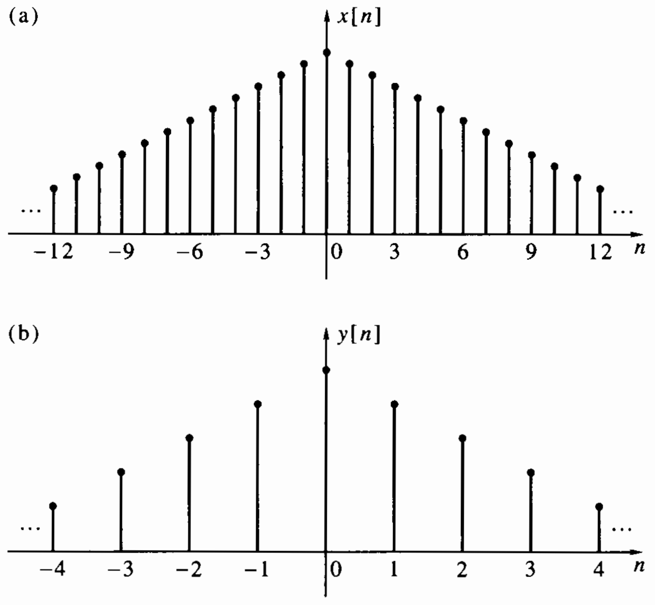

Figure 12.1 Decimation of a discrete-time signal by a factor of 3: (a) the original signal; (b) the decimated signal.

Figure 12.1 shows the samples of the decimated signal $y[n]$ spaced three times wider than the samples of $x[n]$. This is not a coincidence. In real time, the decimated signal indeed appears at a rate slower than that of the original signal by a factor M. If the sampling interval of $x[n]$ is T, then that of $y[n]$ is MT.

Expansion is another operation on a discrete-time signal that yields a discrete-time signal. It is related to reconstruction of a discrete-time signal, an operation that yields a continuous-time signal. However, the relation is less obvious than in the case of sampling and decimation. The mathematical definition of L-fold *expansion*, or *up-sampling*, is

$$y[n] = \begin{cases} x[n/L], & \text{if } n/L \text{ is an integer,} \\ 0, & \text{if } n/L \text{ is noninteger.} \end{cases} \tag{12.2}$$

Figure 12.2 illustrates 3-fold expansion. Expansion is an information-preserving operation: All samples of $x[n]$ are present in the expanded signal $y[n]$. The samples of the expanded signal appear at a rate faster than that of the original signal by a factor L. If the sampling interval of $x[n]$ is T, then that of $y[n]$ is T/L.

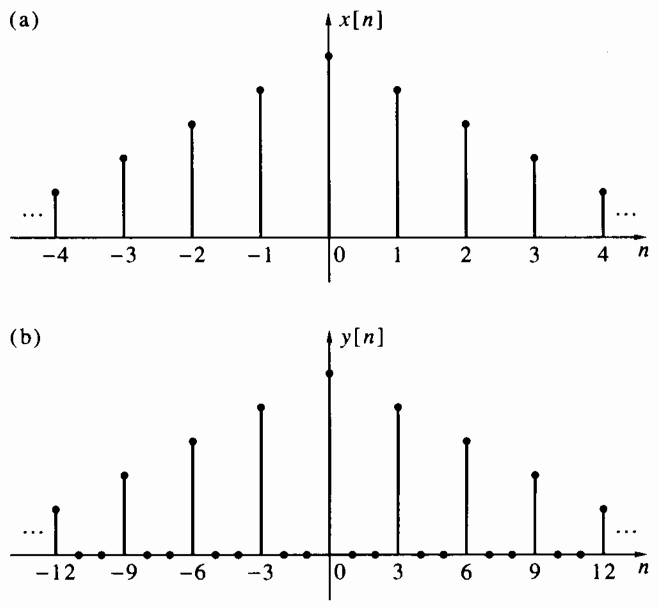

Figure 12.2 Expansion of a discrete-time signal by a factor of 3: (a) the original signal; (b) the expanded signal.

Expansion is also called *interpolation* in the digital signal processing literature. However, this term does not well describe the operation depicted in Figure 12.2, since the inserted values are identically zero and are not related to the signal values $x[n]$, which they purport to interpolate. This is in contrast with reconstruction by a zero-order hold, in which the reconstructed signal between sampling points has the value of the most recent sample. The term *expansion* better describes this operation; we reserve the term *interpolation* for another operation, to be described later.

We shall use the following notations for decimation and expansion:

$$\text{Decimation:} \quad y[n] = x_{(\downarrow M)}[n],$$
$$\text{Expansion:} \quad y[n] = x_{(\uparrow L)}[n].$$

Figure 12.3 shows common block-diagram descriptions of an M-fold decimator and an L-fold expander.

A MATLAB expression for decimation is

```
y = x(1:M:length(x));
```

There are a number of ways to implement expansion in MATLAB; for example,

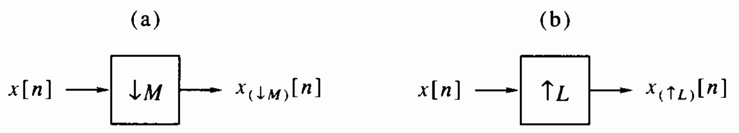

Figure 12.3 Block-diagram notations for (a) M-fold decimation and (b) L-fold expansion.

```
y = reshape([x; zeros(L-1,length(x))],1,L*length(x));
```

Decimation and expansion are both linear operations (Problem 12.1). However, neither is time invariant. To show this, consider the following counterexample. Let $x[n]$ be the sequence

$$x[n] = \begin{cases} 1, & n \text{ even}, \\ 0, & n \text{ odd}, \end{cases}$$

and let $v[n] = x[n-1]$, that is, the right shift of $x[n]$ by 1. Then the 2-fold decimation of $x[n]$ is the sequence $x_{(\downarrow 2)}[n] = 1$, whereas the 2-fold decimation of $v[n]$ is the sequence $v_{(\downarrow 2)}[n] = 0$. The latter is obviously not a shift by 1 of the former. Also, the 2-fold expansion of $x[n]$ is

$$x_{(\uparrow 2)}[n] = \begin{cases} 1, & n \text{ is divisible by } 4, \\ 0, & n \text{ otherwise}, \end{cases}$$

whereas the 2-fold expansion of $v[n]$ is

$$v_{(\uparrow 2)}[n] = \begin{cases} 1, & n-2 \text{ is divisible by } 4, \\ 0, & n \text{ otherwise}. \end{cases}$$

Here the latter sequence is the right shift by 2 of the former, instead of right shift by 1.

Decimation and expansion do not commute in general. For example, if we expand $x[n]$ by a factor L and then decimate the result by the same factor, we get $x[n]$ back. However, if we decimate first and then expand, we get the sequence

$$y[n] = \begin{cases} x[n], & \text{if } n/L \text{ is an integer}, \\ 0, & \text{if } n/L \text{ is noninteger}. \end{cases}$$

In one special case, as the next theorem shows, expansion and decimation do commute.

Theorem 12.1 If M and L are relatively prime, then decimation by M and expansion by L are commutative operations, that is,

$$\{x_{(\downarrow M)}\}_{(\uparrow L)}[n] = \{x_{(\uparrow L)}\}_{(\downarrow M)}[n]. \tag{12.3}$$

Proof We have, by the definitions of decimation and expansion,

$$\{x_{(\downarrow M)}\}_{(\uparrow L)}[n] = \begin{cases} x[nM/L], & nM \text{ is divisible by } L, \\ 0, & \text{otherwise}, \end{cases} \tag{12.4}$$

$$\{x_{(\uparrow L)}\}_{(\downarrow M)}[n] = \begin{cases} x[nM/L], & n \text{ is divisible by } L, \\ 0, & \text{otherwise}. \end{cases} \tag{12.5}$$

When M and L are coprime, nM is divisible by L if and only if n itself is divisible by L, hence equality of the two expressions follows. □

Example 12.1 Let $M = 3$ and $L = 2$. Then both sequences in (12.3) are

$$x[0], 0, x[3], 0, x[6], 0, \ldots .$$

On the other hand, let $M = 6$ and $L = 4$. Then,

$$\{x_{(\downarrow 6)}\}_{(\uparrow 4)}[n] = x[0], 0, 0, 0, x[6], 0, 0, 0, x[12], 0, \ldots,$$

but

$$\{x_{(\uparrow 4)}\}_{(\downarrow 6)}[n] = x[0], 0, x[3], 0, x[6], 0, \ldots .$$

□

12.2 Transforms of Decimated and Expanded Sequences

The analysis and understanding of decimation and expansion are greatly facilitated by their transforms. We shall use the following notations for the z-transforms and Fourier transforms of a decimated sequence:

$$\{\mathcal{Z}x_{(\downarrow M)}\}(z) = X^{\mathrm{z}}_{(\downarrow M)}(z) = \{X^{\mathrm{z}}(z)\}_{(\downarrow M)}, \tag{12.6a}$$

$$\{\mathcal{F}x_{(\downarrow M)}\}(\theta) = X^{\mathrm{f}}_{(\downarrow M)}(\theta) = \{X^{\mathrm{f}}(\theta)\}_{(\downarrow M)}. \tag{12.6b}$$

For the transforms of expanded sequences we shall use

$$\{\mathcal{Z}x_{(\uparrow L)}\}(z) = X^{\mathrm{z}}_{(\uparrow L)}(z) = \{X^{\mathrm{z}}(z)\}_{(\uparrow L)}, \tag{12.7a}$$

$$\{\mathcal{F}x_{(\uparrow L)}\}(\theta) = X^{\mathrm{f}}_{(\uparrow L)}(\theta) = \{X^{\mathrm{f}}(\theta)\}_{(\uparrow L)}. \tag{12.7b}$$

In each case, the third notation is inaccurate, but sometimes it is more convenient than the other two, especially when we want to apply decimation or expansion to a product of transforms.

As we have seen, decimation and expansion are linear, but not time-invariant operations. Therefore, we do not expect them to be described by transfer functions. In other words, we do not expect to get expressions such as

$$X^{\mathrm{z}}_{(\downarrow M)}(z) = H^{\mathrm{z}}(z)X(z), \quad \text{or} \quad X^{\mathrm{z}}_{(\uparrow L)}(z) = H^{\mathrm{z}}(z)X^{\mathrm{z}}(z).$$

However, these two transforms can be expressed in terms of $X(z)$, as we now show.

Consider decimation first. Let $\{c_M[n]\}$ be the *comb* signal

$$c_M[n] = \sum_{k=-\infty}^{\infty} \delta[n - kM] \tag{12.8}$$

(see Figure 12.4). The comb signal is the digital counterpart of the impulse train (cf. Figure 2.4). By (4.6), we can express $c_M[n]$ as

$$c_M[n] = \frac{1}{M}\sum_{m=0}^{M-1} W_M^{mn}, \quad \text{where} \quad W_M = e^{j2\pi/M}. \tag{12.9}$$

Note the similarity of this equality to Poisson's formula (2.48).

Using the comb signal, we can express the decimated signal as

$$x_{(\downarrow M)}[n] = x[nM]c_M[nM], \tag{12.10}$$

and its z-transform as

$$X^{\mathrm{z}}_{(\downarrow M)}(z) = \sum_{n=-\infty}^{\infty} x[nM]c_M[nM]z^{-n} = \sum_{n=-\infty}^{\infty} x[n]c_M[n]z^{-n/M}. \tag{12.11}$$

The last equality follows because $c_M[n]$ is identically zero for n that is not an integer multiple of M, so the sum on the right of (12.11) includes only indices that are integer

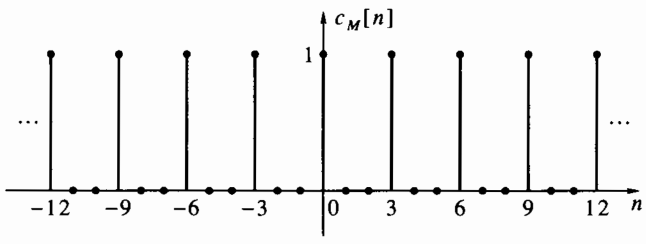

Figure 12.4 The comb signal $c_M[n]$ (shown for $M = 3$).

multiples of M. Now, substitution of (12.9) in (12.11) gives

$$X^{\mathrm{z}}_{(\downarrow M)}(z) = \frac{1}{M}\sum_{n=-\infty}^{\infty}\sum_{m=0}^{M-1} x[n]W_M^{mn}z^{-n/M} = \frac{1}{M}\sum_{m=0}^{M-1}\left[\sum_{n=-\infty}^{\infty} x[n](z^{1/M}W_M^{-m})^{-n}\right]$$
$$= \frac{1}{M}\sum_{m=0}^{M-1} X^{\mathrm{z}}(z^{1/M}W_M^{-m}). \tag{12.12}$$

Example 12.2 Let $x[n] = \alpha^n$, $n \ge 0$. Then $x_{(\downarrow M)}[n] = \alpha^{Mn}$, $n \ge 0$. We have

$$X^{\mathrm{z}}(z) = \frac{z}{z-\alpha}, \quad X^{\mathrm{z}}_{(\downarrow M)}(z) = \frac{z}{z-\alpha^M}.$$

Therefore, we get from (12.12) the nontrivial identity

$$\frac{z}{z-\alpha^M} = \frac{1}{M}\sum_{m=0}^{M-1}\frac{z^{1/M}W_M^{-m}}{z^{1/M}W_M^{-m}-\alpha}. \tag{12.13}$$

□

The meaning of formula (12.12) is perhaps not obvious, but it will become clear when we pass from the z-transform to the Fourier transform of $x_{(\downarrow M)}[n]$. We do this by substituting $z = e^{j\theta}$ in (12.12), obtaining

$$X^{\mathrm{f}}_{(\downarrow M)}(\theta) = \frac{1}{M}\sum_{m=0}^{M-1} X^{\mathrm{f}}\left(\frac{\theta - 2\pi m}{M}\right). \tag{12.14}$$

This result bears an obvious resemblance to the sampling theorem [see (3.9)]. We can therefore think of (12.14) as the discrete-time sampling theorem. The derivation of (12.12), from which we obtained (12.14), is similar to the proof of Theorem 3.1.

The implications of (12.14) on aliasing caused by decimation are similar to those in the case of sampling of a continuous-time signal. In general, if $X^{\mathrm{f}}(\theta)$ occupies the entire bandwidth from $-\pi$ to π, then $X^{\mathrm{f}}_{(\downarrow M)}(\theta)$ will be aliased due to the superposition of the M shifted and frequency-scaled transforms in (12.14). This is depicted in Figure 12.5, which illustrates the aliasing phenomenon for $M = 3$. Part a of the figure shows the Fourier transform of the original signal. Part b shows the M shifted and frequency-scaled replicas (three in this case). Part c shows the superposition of the three replicas, which yields the transform of the decimated signal. As we see in part c, the Fourier transform of the decimated signal has a shape different from that of the original signal.

If the signal $x[n]$ is band limited to $\theta \in [-\pi/M, \pi/M]$, the decimated signal $x_{(\downarrow M)}[n]$ is alias free, as depicted in Figure 12.6. As we see, the Fourier transform of the decimated signal occupies the entire bandwidth, because of the M-fold expansion of the frequency scale, but is not aliased. We can regard the condition that the Fourier

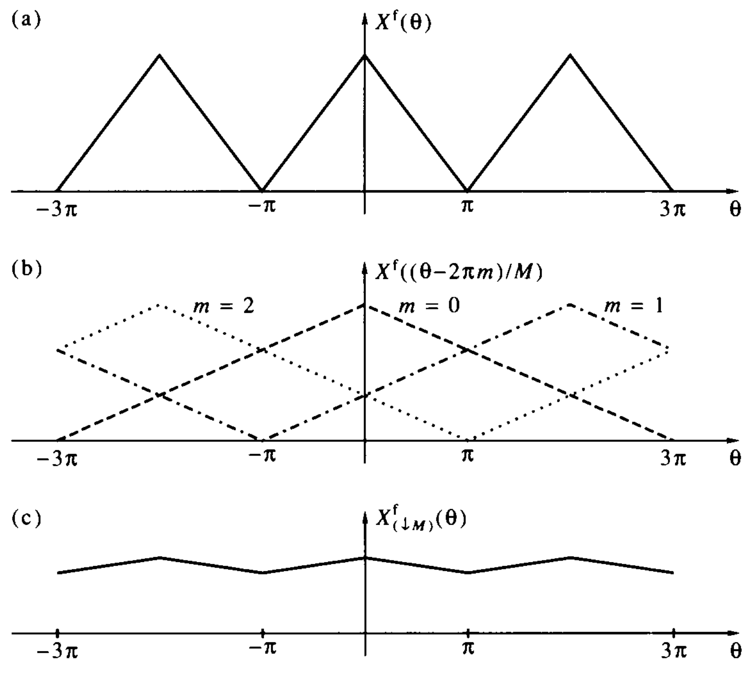

Figure 12.5 Aliasing caused by decimation: (a) Fourier transform of the original signal; (b) shifted and frequency-scaled replicas of part a; (c) Fourier transform of the decimated signal.

transform is nonzero only for $\theta \in [-\pi/M, \pi/M]$ as Nyquist's no-aliasing condition for decimation of discrete-time signals.

The preceding is not the only case in which the decimated signal is not aliased. Another important case is that of a complex band-pass signal whose Fourier transform is nonzero only in the range $\theta \in [(2k-1)\pi/M, (2k+1)\pi/M]$ for some integer k. Figure 12.7 illustrates such a case, for $M = 3$ and $k = 1$. The Fourier transform has an asymmetrical shape in this example, to emphasize that it belongs to a complex signal, rather than a real signal. As we see, the Fourier transform of the decimated signal occupies the entire bandwidth, and it is not apparent from looking at it that the original signal was band pass. Nonetheless, there is no aliasing, so no information about the original signal was lost in the decimation process.

Computation of the transforms of expanded signals is much easier than that of decimated signals. The z-transform of an expanded signal is given by

$$X^z_{(\uparrow L)}(z) = \sum_{n=-\infty}^{\infty} x_{(\uparrow L)}[nL]z^{-nL} = \sum_{n=-\infty}^{\infty} x[n]z^{-nL} = X^z(z^L). \qquad (12.15)$$

and the Fourier transform by

$$X^f_{(\uparrow L)}(\theta) = X^f(L\theta). \qquad (12.16)$$

Figure 12.8 illustrates the expansion operation in the frequency domain. Part a shows the Fourier transform of the original signal and part b shows the transform of the expanded signal. As we see, the shape of the Fourier transform is contracted by a factor L in the frequency axis and is repeated L times in the range $[-\pi, \pi]$. Other than the contraction, the shape of the transform is preserved, confirming our conclusion that expansion does not lead to aliasing.

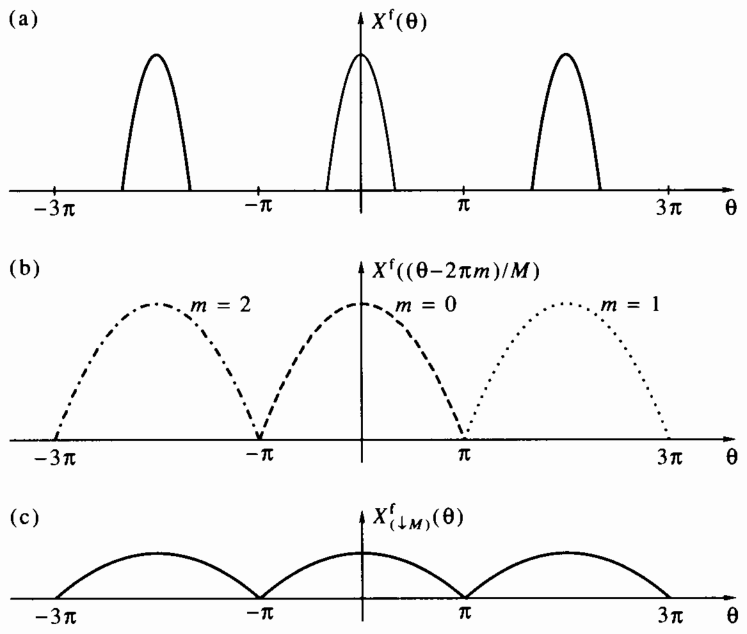

Figure 12.6 Alias-free decimation of a band-limited signal: (a) Fourier transform of the original signal; (b) shifted and frequency-scaled replicas of part a; (c) Fourier transform of the decimated signal.

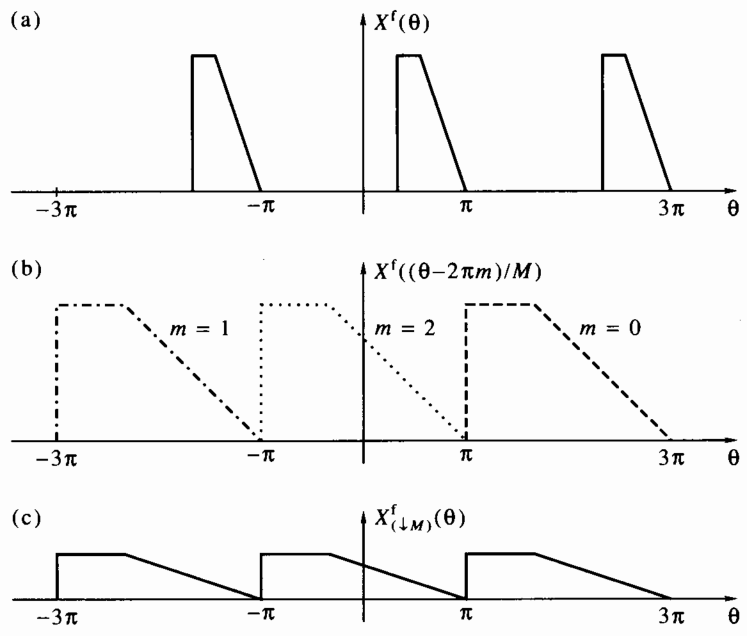

Figure 12.7 Alias-free decimation of a complex band-pass signal: (a) Fourier transform of the original signal; (b) shifted and frequency-scaled replicas of part a; (c) Fourier transform of the decimated signal.

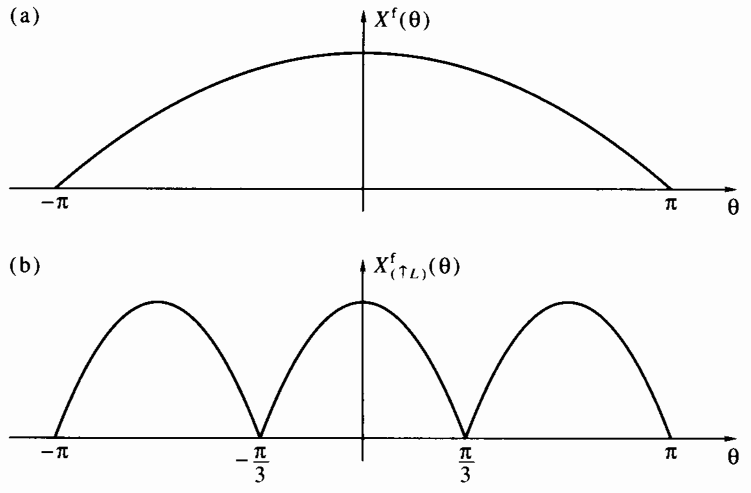

Figure 12.8 Expansion in the frequency domain: Fourier transform of the original signal (a) and the expanded signal (b).

We have seen how the spectra of decimated and expanded signals are related to those of the corresponding original signals. When showing the spectra as a function of the variable θ, they appear stretched by a factor M in the case of decimation, and compressed by a factor L in the case of expansion. However, we recall that the variable θ is related to the physical frequency ω by the formula $\omega = \theta/T$, where T is the sampling interval. We also recall that the sampling interval of a decimated signal is M times larger than that of the original signal. Similarly, the sampling interval of an expanded signal is L times smaller than that of the original signal. The conclusion is that the frequency range of the spectra, expressed in terms of ω, is not affected by either decimation or expansion. For example, suppose that the signal $x[n]$ was obtained from a continuous-time signal having bandwidth of ± 100 Hz by sampling at $T = 0.001$ second. Then $x[n]$ occupies the range $\pm 0.2\pi$ in the θ domain. Now suppose that $y[n]$ is obtained from $x[n]$ by 5-fold decimation. Then the spectrum of $y[n]$ is alias free and it occupies the range $\pm\pi$ in the θ domain. The sampling rate of $y[n]$ is 0.005 second, so its physical bandwidth is ± 100 Hz, same as that of the original continuous-time signal. If we now expand $y[n]$ by a factor of 5 to get the signal $z[n]$, the spectrum of $z[n]$ will exhibit five replicas of the basic shape, and one of those will occupy the range $\pm 0.2\pi$ in the θ domain. Since the sampling interval of $z[n]$ is 0.001 second, the corresponding physical frequency is again ± 100 Hz.

12.3 Linear Filtering with Decimation and Expansion

12.3.1 Decimation

Since decimation, like sampling, leads to potential aliasing, it is desirable to precede the decimator with an antialiasing filter. Unlike sampling, here the input signal is already in discrete time, so we use a digital antialiasing filter. The antialiasing filter, also called the *decimation filter*, should approximate an ideal low-pass filter with cutoff frequency π/M. This is illustrated in Figure 12.9, for $M = 4$. In this figure, the input signal $x[n]$ has bandwidth slightly over $\pm\pi/4$. The decimation filter $H^f(\theta)$ eliminates the spectral components outside the frequency range $[-\pi/4, \pi/4]$, resulting in the signal $v[n]$. This signal is decimated by 4, yielding the output signal $y[n]$, whose spectrum occupies the entire bandwidth. Note that $Y^{\mathrm{f}}(\theta)$ is faithful to the part of $X^{\mathrm{f}}(\theta)$ in the range $[-\pi/4, \pi/4]$, but the information outside this range is lost.

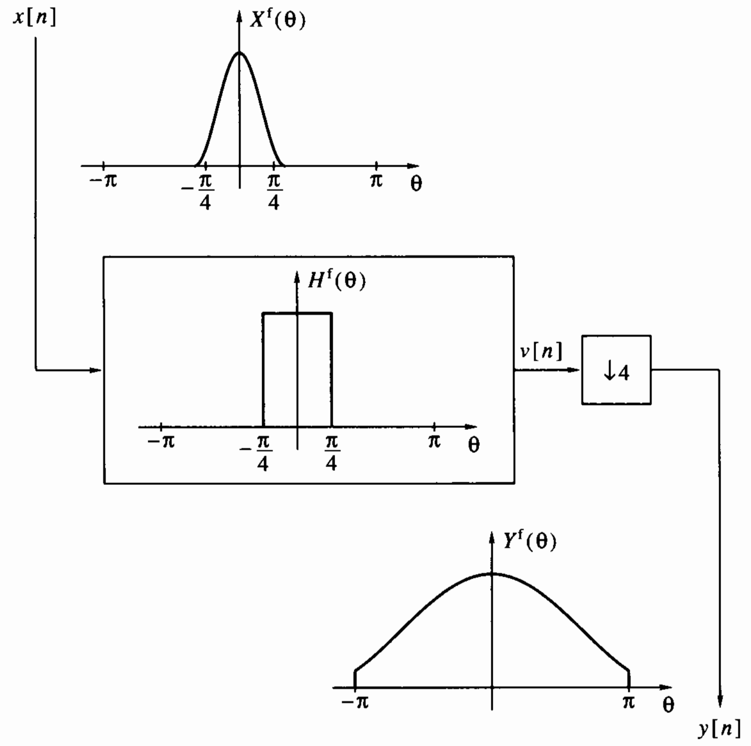

Figure 12.9 Passing a signal through an antialiasing filter prior to decimation.

Linear filtering followed by M-fold decimation is described mathematically by

$$y[n] = \sum_i h[i]x[nM - i] = \sum_i x[i]h[nM - i]. \tag{12.17}$$

Since we need only one out of every M outputs of the filter, we have a potential for M-fold computational saving. When the filter $h[n]$ is IIR, it is difficult to take advantage of this potential. However, when $h[n]$ is FIR, computational saving can be realized easily. For an Nth-order FIR filter, the necessary number of multiply-add operations is about N/M per input data point. We defer a discussion on programming of decimation with FIR filtering until the next section.

The following point concerning operation count in FIR decimation filters is worth noting. It is often reasonable to design the filter such that its transition bandwidth $\Delta\theta$ is a fixed fraction of the cutoff frequency, say $\Delta\theta = \mu\pi/M$, where μ depends on the specific application. As we know from FIR filter design theory, the order N of the filter is approximately inversely proportional to the transition bandwidth, the proportionality factor depending on the required stop-band attenuation and pass-band ripple [see (9.56c) or (9.79)]. Thus, N is approximately proportional to M. Therefore, the number of operations per input data point, N/M, is approximately independent of the decimation ratio M and is determined only by the relative steepness of the transition region, the stop-band attenuation, and the pass-band ripple.

Example 12.3 With $\Delta\theta = \mu\pi/M$, we get from (9.56c) and (9.79),

$$\frac{N}{M} \approx \frac{-20\log_{10}(\min\{\delta_p, \delta_s\}) - 7.95}{2.285\mu\pi} \quad \text{(Kaiser window design)}, \tag{12.18a}$$

$$\frac{N}{M} \approx \frac{-20\log_{10}\sqrt{\delta_p\delta_s} - 13}{2.32\mu\pi} \quad \text{(equiripple design)}. \tag{12.18b}$$

For example, if $\delta_p = 0.01$, $\delta_s = 0.001$, and $\mu = 0.1$, the required number of operations per input data point is about 73 in a Kaiser window filter, or about 51 in an equiripple filter. This example illustrates that the computational cost of decimation can be quite heavy in practice. □

Example 12.4 In Section 3.5 we studied physical A/D converters. There we saw that the two major parameters by which an A/D converter is judged are the number of bits and the speed. In applications such as audio or biomedical signals, speed requirements are easy to meet, since the signal bandwidth is low (e.g., up to 20 kHz for audio signals). On the other hand, a large number of bits may be required, resulting in complex and expensive hardware. We now show how an A/D with a relatively high speed and a relatively low number of bits can be made to give a higher number of bits at a reduced speed.

Suppose that the bandwidth of the input signal is f_m and, instead of sampling at the Nyquist rate $2f_m$ we sample at a higher rate f_{sam}, such that $M = 0.5f_{\text{sam}}/f_m$ is an integer. As we saw in Section 11.7.4, the variance of the quantization noise of the A/D is $2^{-2(B-1)}/12$, where B is the number of bits. The noise is white, so its power spectral density is constant over the frequency range $[-0.5f_{\text{sam}}, 0.5f_{\text{sam}}]$. Now, since the signal bandwidth is f_m, we can pass the output signal of the A/D through a low-pass filter with cutoff frequency f_m. Assuming that the filter is ideal, it would reduce the noise variance by a factor M without harming the signal. After doing so, we can decimate the signal by M. The discrete-time signal thus obtained will be sampled at the Nyquist rate, but the variance of the noise will only be $M^{-1} \cdot 2^{-2(B-1)}/12$, and the noise will still be white at the new sampling rate. Thus, we have effectively gained $0.5\log_2 M$ bits. Each doubling of the sampling rate provides an extra half-bit, or each quadrupling of the rate provides an extra bit. In reality the effective gain is less than half a bit, since the decimation filter is not ideal, so it does not completely eliminate the noise in the stop band. In Section 14.7 we shall use this idea again; see page 586. □

12.3.2 Expansion

Expansion does not cause aliasing, but it yields undesired replicas of the signal's spectrum in the range $[-\pi, \pi]$, as we saw in Figure 12.8. These replicas can be eliminated by a digital low-pass filter following the expander, as shown in Figure 12.10 (for $L = 3$). In this figure, the input signal $x[n]$ occupies the entire bandwidth. The expanded signal $v[n]$ has three replicas in the frequency domain, each occupying a bandwidth of $2\pi/3$. The filter approximates an ideal low-pass filter with cutoff frequency $\pi/3$ (or π/L in general), and retains only the base-band replica. The spectrum of the output signal $y[n]$ is band limited to $\pm\pi/3$ (or $\pm\pi/L$ in general).

The combination of expansion followed by low-pass filtering is called *interpolation*, and the low-pass filter is called an *interpolation filter*, or an *anti-imaging filter*. Whereas decimation may or may not be preceded by a decimation filter, expansion is almost always followed by an interpolation filter, because without such a filter the expanded signal has little use.

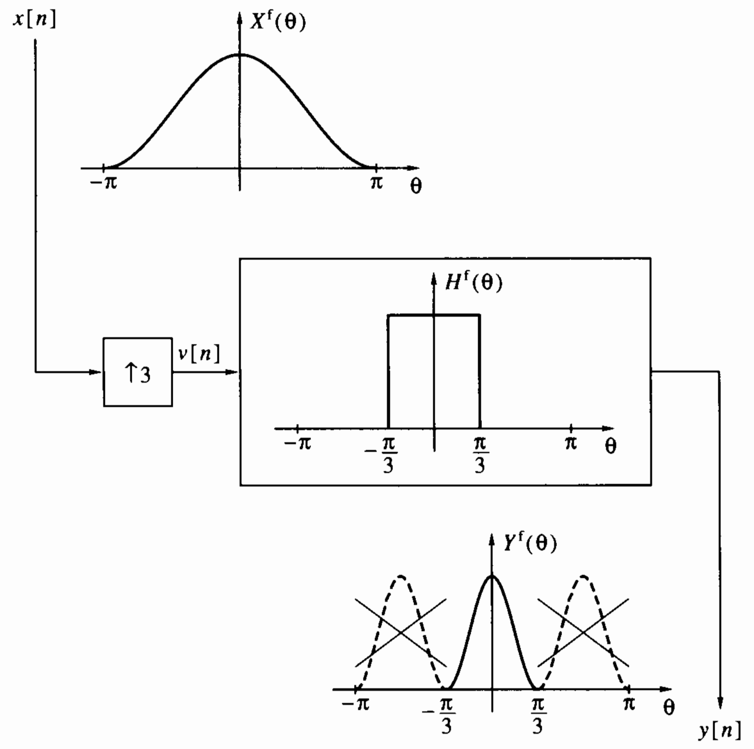

Figure 12.10 Passing an expanded signal through an interpolation filter. The dashed lines in $Y^{\mathrm{f}}(\theta)$ indicate parts of the spectrum that are eliminated by the filter.

L-fold expansion followed by linear filtering is described mathematically by

$$y[n] = \sum_i x[i]h[n - Li]. \tag{12.19}$$

As in the case of decimation, we can reduce the amount of computation by a factor L, compared with regular convolution, in case the filter $h[n]$ is FIR. The required number of multiply-add operations is about N/L per output data point. We shall discuss the programming of interpolation with FIR filtering in the next section.

The comments we have made regarding the operation count apply to expansion as well. If the transition bandwidth is set to a fixed fraction of the cutoff frequency, then the number of operations per output data point is approximately independent of the interpolation ratio and is determined only by the relative steepness of the transition region and the required attenuation and ripple tolerances.

Let us now discuss interpolation in more detail. Consider an interpolation filter $h[n]$ that is an ideal low-pass filter with cutoff frequency π/L. Such a filter is IIR and not causal, so it cannot be implemented, but we can still explore its theoretical behavior. The impulse response of the filter is

$$h[n] = \frac{1}{2\pi}\int_{-\pi/L}^{\pi/L} Le^{j\theta n}d\theta = \operatorname{sinc}\left(\frac{n}{L}\right). \tag{12.20}$$

Substitution in (12.19) gives

$$y[n] = \sum_{i=-\infty}^{\infty} x[i]\operatorname{sinc}\left(\frac{n - iL}{L}\right). \tag{12.21}$$

This result is similar to Shannon's interpolation formula (3.23). Indeed, ideal interpolation of a discrete-time signal can be viewed as ideal reconstruction of that signal (i.e., its conversion to a band-limited, continuous-time signal), followed by resampling at a rate L times higher than the original rate. If we apply this operation to (3.23), we will get precisely (12.21). Practical interpolation approximates the ideal sinc filter by a causal filter, usually chosen to be FIR.

Example 12.5 Music signals on compact discs are sampled at 44.1 kHz. When the signal is converted to analog, faithful reconstruction is required up to 20 kHz, with only little distortion. As we saw in Section 3.4, this is extremely difficult to do with analog hardware, because of the sinc-shaped frequency response of the zero-order hold and the small margin available for extra filtering (from 20 to 22.05 kHz). A common technique for overcoming this problem is called *oversampling*, and it essentially consists of the following steps:

1. The digital signal is expanded by a certain factor, typically 8, followed by an interpolation filter. The sampling rate of the resulting signal is now 352.8 kHz, but its bandwidth is still only 22.05 kHz.
2. The interpolated signal is input to a zero-order hold. The frequency response of the ZOH is sinc shaped, and its bandwidth (to the first zero crossing) is 352.8 kHz.
3. The output of the ZOH is low-pass filtered to a bandwidth of 20 kHz by an analog filter. Such a filter is relatively easy to design and implement, since we have a large margin (between 20 kHz and 352.8 kHz) over which the frequency response can decrease gradually. The bandwidth of the digital signal is limited to 22.05 kHz, so the analog filter will little affect it.

Figure 12.11 illustrates this procedure. Part a shows a 20 kHz sinusoidal signal sampled at 44.1 kHz, denoted by $x[n]$. The five samples represent a little over two periods of the signal. Such a discrete-time signal would be extremely hard to reconstruct faithfully by means of a zero-order hold followed by an analog filter. Part b shows an 8-fold interpolation of $x[n]$, denoted by $y[n]$. Part c shows the signal $\hat{y}(t)$, reconstructed from $y[n]$ by a zero-order hold. Finally, part d shows the output signal $z(t)$, obtained by passing $\hat{y}(t)$ through a low-pass filter with cutoff frequency $f_p = 20$ kHz. For simplicity, we have not shown the delays introduced by the interpolation filter and the analog low-pass filter. The oversampling technique is implemented nowadays in all CD players and in most digital processing systems of music signals. □

12.3.3 Sampling-Rate Conversion

A common use of multirate signal processing is for sampling-rate conversion. Suppose we are given a digital signal $x[n]$ sampled at interval T_1, and we wish to obtain from it a signal $y[n]$ sampled at interval T_2. Ideally, $y[n]$ should be spectrally identical to $x[n]$. The techniques of decimation and interpolation enable this operation, provided the ratio T_1/T_2 is a rational number, say L/M. We distinguish between two possibilities:

1. $T_1 > T_2$, meaning that the sampling rate should be increased. This is always possible without aliasing.
2. $T_1 < T_2$, meaning that the sampling rate should be decreased. This is possible without aliasing only if $x[n]$ is band limited to a frequency range not higher than $\pm\pi T_1/T_2$. If $x[n]$ does not fulfill this condition, a part of its frequency contents must be eliminated to avoid aliasing.

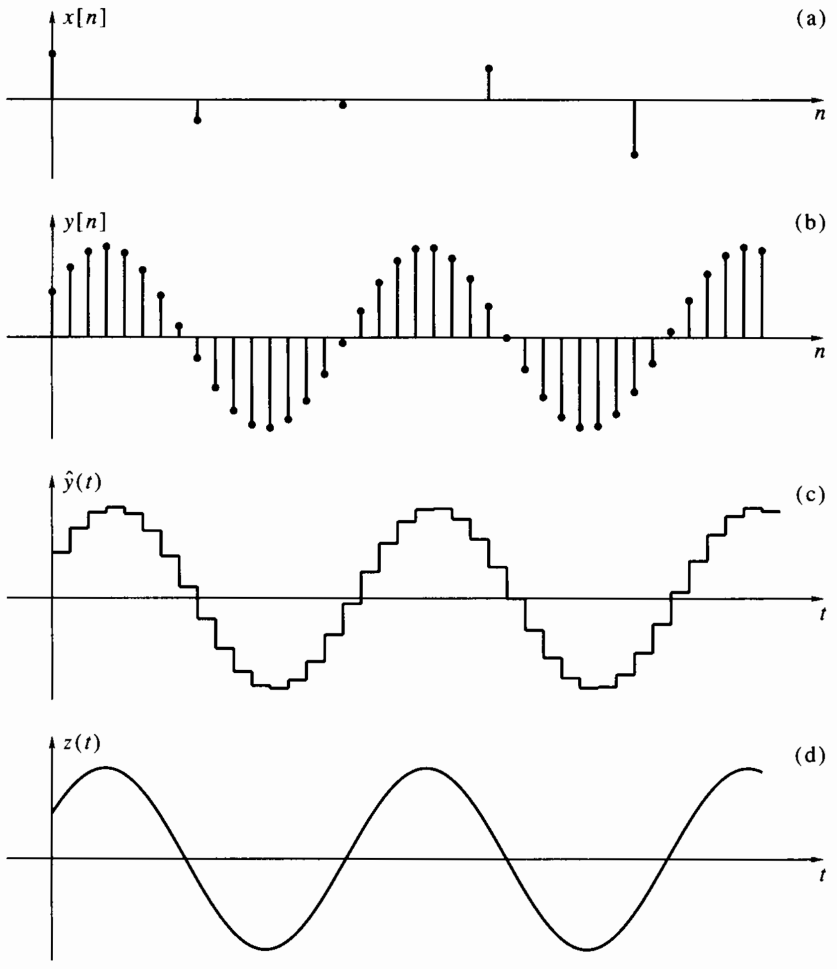

Figure 12.11 Demonstration of oversampling in compact-disc music signal reconstruction: (a) a 20 kHz sinusoidal signal sampled at 44.1 kHz; (b) 8-fold interpolation of the sampled signal; (c) zero-order hold reconstruction of the interpolated signal; (d) analog low-pass filtering of the reconstructed signal.

Sampling-rate conversion can be accomplished by L-fold expansion, followed by low-pass filtering and M-fold decimation. Figure 12.12 depicts this sequence of operations. The low-pass filter performs both interpolation of the expanded signal and antialiasing. If the sampling rate is to be increased, then $L > M$. The low-pass filter should then have a cutoff frequency π/L. If the sampling rate is to be decreased, then $L < M$. The low-pass filter should then have a cutoff frequency π/M. In this case, the filter will eliminate a part of the signal's frequency contents if its original bandwidth is higher than $\pi L/M$. Thus, the sampling-rate conversion filter should always have a cutoff frequency $\pi/\max\{L, M\}$.

Sampling-rate conversion is described mathematically by

$$y[n] = \sum_i x[i]h[Mn - Li]. \tag{12.22}$$

The number of multiply-add operations can be estimated as follows. Assume we have P input points, so the number of output points is PL/M, and for each output point we

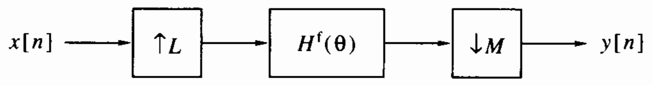

Figure 12.12 Sampling-rate conversion by expansion, filtering, and decimation.

need about N/L operations. Thus, the number of operations is about N/M per input point, or N/L per output point. The length of the filter is proportional to $\max\{L, M\}$. We get that if $L > M$ (sampling rate increase), the number of operations per output point is approximately constant, whereas if $L < M$ (sampling rate decrease), the number of operations per input point is approximately constant.

12.4 Polyphase Filters

As we saw in Section 12.3, linear filtering with decimation, interpolation, and sampling-rate conversion allows potential computational savings. However, it is not always obvious how to take advantage of this potential. *Polyphase filters* is a name given to certain realizations of multirate filtering operations, which facilitate computational savings in both software and hardware implementations. As we shall see, polyphase implementations are useful for FIR filters, but not for IIR.

12.4.1 The Multirate Identities

In preparation for our discussion of polyphase filters, we now derive two useful identities, known as the *multirate identities.*

The decimation identity Suppose we decimate a discrete-time signal $x[n]$ by a factor M and then pass it through a filter $H^{z}(z)$, as shown in Figure 12.13(a). The decimation identity asserts that this operation is equivalent to expanding the impulse response of $H^{z}(z)$ by a factor M, then feeding $x[n]$ to the new filter, and finally decimating by a factor M, as shown in Figure 12.13(b).

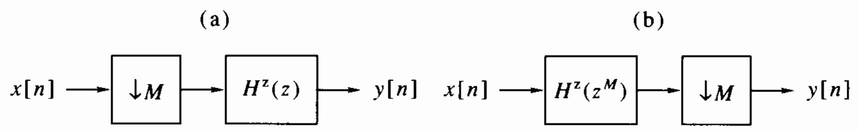

Figure 12.13 The decimation identity.

The decimation identity is easy to understand in the time domain. Both operations are expressed by the time-domain formula

$$y[n] = \sum_{k} h[k]x[(n-k)M]. \tag{12.23}$$

A z-domain proof is given as follows. The output in Figure 12.13(b) has a z-transform

$$\{X^{z}(z)H^{z}(z^{M})\}_{(\downarrow M)} = \frac{1}{M}\sum_{m=0}^{M-1} X^{z}(z^{1/M}W_{M}^{-m})H^{z}((z^{1/M}W_{M}^{-m})^{M})$$

$$= \frac{1}{M}\sum_{m=0}^{M-1} X^{z}(z^{1/M}W_{M}^{-m})H^{z}(z) = H^{z}(z)\{X^{z}(z)\}_{(\downarrow M)}. \tag{12.24}$$

The right side is precisely the z-domain description of Figure 12.13(a).

The expansion identity Suppose we pass a discrete-time signal $x[n]$ through a filter $H^z(z)$ and then expand it by a factor L, as shown in Figure 12.14(a). The expansion identity asserts that this operation is equivalent to expanding both the signal and the impulse response of $H^z(z)$ by a factor L, then feeding the expanded signal to the new filter, as shown in Figure 12.14(b).

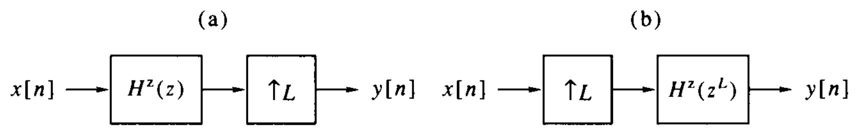

Figure 12.14 The expansion identity.

The expansion identity is easy to understand in the z domain. Both operations are expressed by the z-domain formula

$$Y^z(z) = H^z(z^L)X^z(z^L). \tag{12.25}$$

A time-domain proof is given as follows. The output in Figure 12.14(b) is given by

$$y[n] = \sum_k h_{(\uparrow L)}[k]x_{(\uparrow L)}[n-k] = \begin{cases} \sum_i h[i]x\left[\frac{n}{L} - i\right], & n \text{ is divisible by } L, \\ 0, & \text{otherwise.} \end{cases} \tag{12.26}$$

The right side is precisely the time-domain description of Figure 12.14(a).

12.4.2 Polyphase Representation of Decimation

Consider again the operation (12.17), which represents filtering followed by decimation, and express i as

$$i = i'M + m, \quad 0 \le m \le M-1.$$

Then we get

$$\begin{aligned} y[n] &= \sum_{i=0}^{N} h[i]x[nM-i] = \sum_{m=0}^{M-1}\sum_{i'} h[i'M+m]x[nM-i'M-m] \\ &= \sum_{m=0}^{M-1}\sum_{i'} h[i'M+m]x[(n-i')M-m]. \end{aligned} \tag{12.27}$$

For each $0 \le m \le M-1$, define the sequence $p_m[i']$ as

$$p_m[i'] = h[i'M+m], \tag{12.28}$$

and the sequence $u_m[n]$ as

$$u_m[n] = x[nM-m]. \tag{12.29}$$

Substitution of these definitions in (12.27) then gives

$$y[n] = \sum_{m=0}^{M-1} \{p_m * u_m\}[n]. \tag{12.30}$$

Let us interpret this result:

1. For each m, the sequence $p_m[i']$ is obtained by advancing (i.e., left shifting) the sequence $h[n]$ by m, then decimating by M.
2. For each m, the sequence $u_m[n]$ is obtained by delaying (i.e., right shifting) the sequence $x[n]$ by m, then decimating by M.

3. The output $y[n]$ is obtained by performing M convolutions of the sequences $u_m[n]$ with the corresponding FIR filters $p_m[i']$, then adding the results.

The M FIR filters $p_m[i']$ are called the *polyphase* components of $h[n]$. Their corresponding transfer functions are

$$P_m^z(z) = \sum_{i'} p_m[i']z^{-i'} = \sum_{i'} h[i'M + m]z^{-i'}. \tag{12.31}$$

The transfer function $H^z(z)$ can be expressed in terms of the polyphase components as

$$H^z(z) = \sum_{m=0}^{M-1} z^{-m}P_m^z(z^M). \tag{12.32}$$

Therefore, the z-domain description of filtering followed by decimation is

$$Y^z(z) = \sum_{m=0}^{M-1} \{P_m^z(z^M)z^{-m}X^z(z)\}_{(\downarrow M)}. \tag{12.33}$$

By the decimation identity, this can be written as

$$Y^z(z) = \sum_{m=0}^{M-1} P_m^z(z)\{z^{-m}X^z(z)\}_{(\downarrow M)}. \tag{12.34}$$

However, by our definition of the sequences $u_m[n]$ we have

$$U_m^z(z) = \{z^{-m}X^z(z)\}_{(\downarrow M)}. \tag{12.35}$$

Therefore,

$$Y^z(z) = \sum_{m=0}^{M-1} P_m^z(z)U_m^z(z), \tag{12.36}$$

which is precisely the z-domain description of (12.30).

Figure 12.15 illustrates how the polyphase components are obtained from a given FIR sequence. Figure 12.16 illustrates the operation (12.30), or its z-domain equivalent (12.36). The block diagram in this figure is called the *polyphase decomposition* of the filtering and decimation operation. The procedure ppdec in Program 12.1 implements filtering and decimation by the polyphase decomposition.

An alternative way of implementing the polyphase decomposition is to replace the delay chain by a *commutator*, that is, an M-position switch that routes the input data to the polyphase filters in succession. This implementation is shown in Figure 12.17. Note the order in which the commutator arm is switched: At time $n = 0$ it is at the top position, at time $n = 1$ it is at the bottom position, at time $n = 2$ at the prior-to-bottom position, and so on. In other words, the rotation is counterclockwise, starting at the top position. In software implementation, the commutator is conceptual, rather than physical, and switching is done by appropriate program statements.

We finally remark the notations $p_m[n]$ and $P_m^z(z)$ for the polyphase components are ambiguous, since they depend not only on $h[n]$ and on m, but also on the decimation factor M. To avoid cumbersome notation, we shall continue to omit the dependence on M, but keep in mind that M has to be specified for the polyphase components to be uniquely defined.

12.4.3 Polyphase Representation of Expansion

Consider again the operation (12.19), and express n as

$$n = n'L + L - 1 - l, \quad 0 \le l \le L - 1.$$

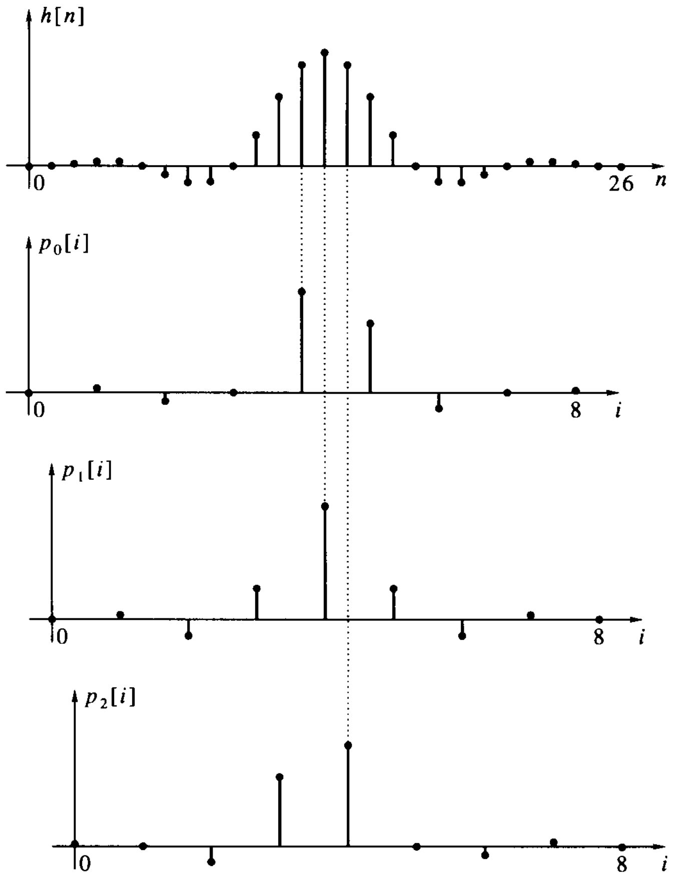

Figure 12.15 The polyphase components of an FIR filter, illustrated for $N = 26$ and $M = 3$.

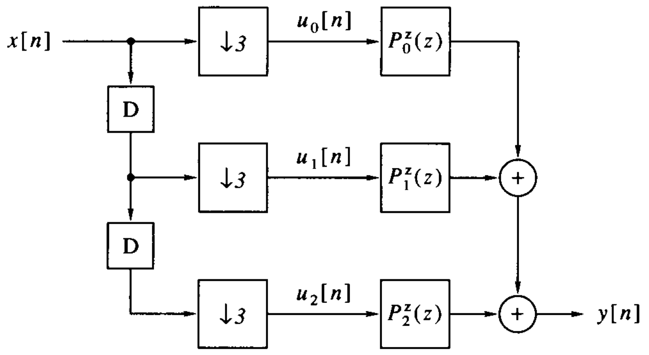

Figure 12.16 Polyphase decomposition of filtering and M-fold decimation (shown for $M = 3$).

Then we get

$$y[n'L + L - 1 - l] = \sum_i x[i]h[(n' - i)L + L - 1 - l]. \tag{12.37}$$

For each $0 \le l \le L - 1$, define the sequence $q_l[n']$ as

$$q_l[n'] = h[n'L + L - 1 - l], \tag{12.38}$$

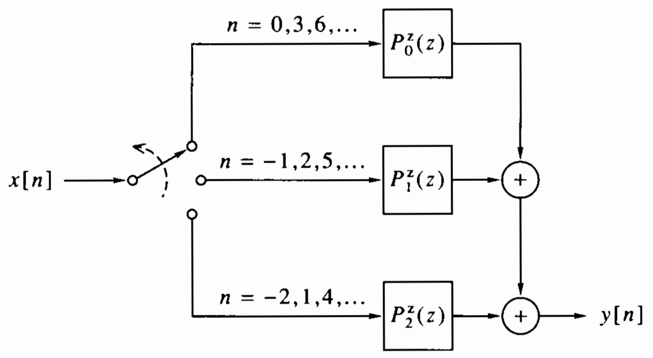

Figure 12.17 Polyphase decomposition of filtering and M-fold decimation with a commutator instead of delays and decimation (shown for $M = 3$).

and the sequence $v_l[n']$ as

$$v_l[n'] = y[n'L + L - 1 - l]. \tag{12.39}$$

Substitution of these definitions in (12.37) then gives

$$v_l[n'] = \{x * q_l\}[n']. \tag{12.40}$$

We also observe that $y[n]$ can be expressed in terms of the $v_l[n']$ as

$$y[n] = \sum_{l=0}^{L-1} \{v_l\}_{(\uparrow L)}[n - L + 1 + l]. \tag{12.41}$$

This formula is verified as follows. By definition of L-fold expansion, the value $\{v_l\}_{(\uparrow L)}[n - L + 1 + l]$ is equal to $v_l[(n - L + 1 + l)/L]$ when $(n - L + 1 + l)/L$ is integer, and is equal to 0 otherwise. By (12.39), this is equal to $y[n]$. For each n, exactly one of the $\{(n - L + 1 + l)/L,\ 0 \le l \le L - 1\}$ is integer, so only one term in the sum (12.41) is nonzero.

As we see, $q_l[n']$ can be regarded as the polyphase components of $h[n]$ corresponding to a decimation factor L, indexed in reversed order. The sequence of operations for computing $y[n]$ is therefore as follows:

1. Convolve the input sequence $x[n]$ with each of the polyphase components $q_l[n']$ to obtain the sequences $v_l[n']$.
2. Expand each sequence $v_l[n']$ by a factor L.
3. Delay the expanded sequences $\{v_l\}_{(\uparrow L)}[n]$ by $L - l - 1$ time units and add them. Note that the addition is fictitious, since at each time point only one of the sequences has nonzero value.

The transfer function $H^z(z)$ can be expressed in terms of the polyphase components as

$$H^z(z) = \sum_{l=0}^{L-1} z^{-(L-1-l)} Q_l^z(z^L). \tag{12.42}$$

Therefore, the z-domain description of expansion followed by filtering is

$$Y^z(z) = \sum_{l=0}^{L-1} z^{-(L-1-l)} Q_l^z(z^L) X_{(\uparrow L)}^z(z). \tag{12.43}$$

By the expansion identity, this can be written as

$$Y^z(z) = \sum_{l=0}^{L-1} z^{-(L-1-l)} \{Q_l^z(z)X^z(z)\}_{(\uparrow L)}. \tag{12.44}$$

However, by our definition of the sequences $v_l[n]$ we have

$$V_l^z(z) = Q_l^z(z)X^z(z). \tag{12.45}$$

Therefore,

$$Y^z(z) = \sum_{l=0}^{L-1} z^{-(L-1-l)} \{V_l^z(z)\}_{(\uparrow L)}, \tag{12.46}$$

which is precisely the z-domain description of (12.41). Figure 12.18 illustrates this sequence of operations. The block diagram in this figure is called the *polyphase decomposition* of the expansion and filtering operation. The procedure ppint in Program 12.2 implements expansion and filtering by the polyphase decomposition.

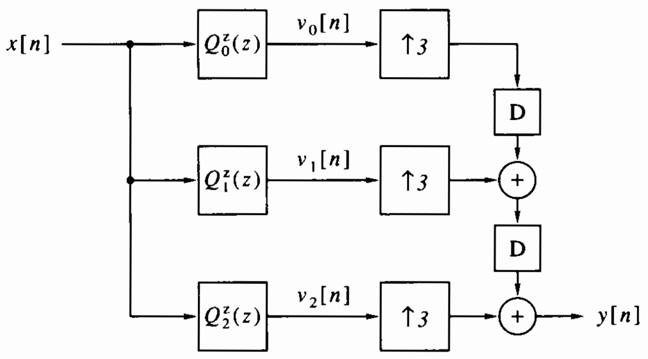

Figure 12.18 Polyphase decomposition of L-fold expansion and filtering (shown for $L = 3$).

An alternative way of implementing the polyphase decomposition is to replace the delay chain by a commutator, as we did in the case of decimation. This implementation is shown in Figure 12.19.

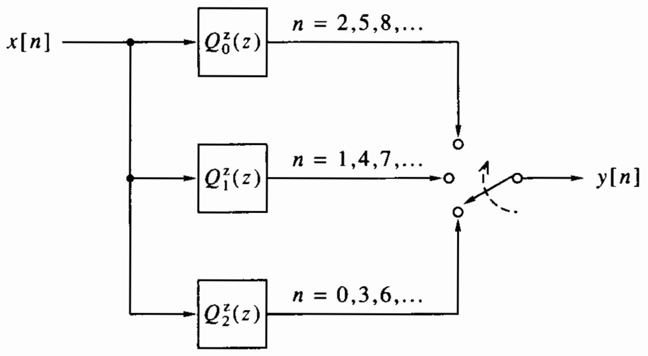

Figure 12.19 Polyphase decomposition of L-fold expansion and filtering with a commutator instead of expansion and delays (shown for $L = 3$).

12.4.4 Polyphase Representation of Sampling-Rate Conversion

Sampling-rate conversion can be done either by efficient interpolation followed by brute-force decimation or by brute-force interpolation followed by efficient decimation. Both approaches are wasteful, however, since neither achieves the minimum possible rate of computation. To get the maximum possible computational efficiency, we need to attack the problem from the root.

Consider again the operation (12.22) and express n and i as

$$n = n'L + l, \quad 0 \le l \le L - 1,$$
$$i = i'M + m, \quad 0 \le m \le M - 1.$$

Then we get

$$y[n'L + l] = \sum_{m=0}^{M-1} \sum_{i'} x[i'M + m]h[(n' - i')ML + lM - mL]. \tag{12.47}$$

It is convenient, in this case, to break the sum over m into two parts. Define $m_0 = \lfloor lM/L \rfloor$, so that

$$\begin{aligned} y[n'L + l] = &\sum_{m=0}^{m_0} \sum_{i'} x[i'M + m]h[(n' - i')ML + lM - mL] \\ &+ \sum_{m=m_0+1}^{M-1} \sum_{i'} x[i'M + m]h[(n' - 1 - i')ML + lM + (M - m)L]. \end{aligned} \tag{12.48}$$

For each $0 \le l \le L - 1$ and $0 \le m \le M - 1$, define the following sequences:

$$r_{l,m}[i'] = \begin{cases} h[i'ML + lM - mL], & m \le m_0, \\ h[i'ML + lM + (M - m)L], & m_0 + 1 \le m \le M - 1. \end{cases} \tag{12.49}$$

$$u_m[i'] = x[i'M + m], \tag{12.50}$$

$$v_l[n'] = y[n'L + l]. \tag{12.51}$$

Substitution of these definitions in (12.48) then gives

$$v_l[n'] = \sum_{m=0}^{m_0} \{u_m * r_{l,m}\}[n'] + \sum_{m=m_0+1}^{M-1} \{u_m * r_{l,m}\}[n' - 1]. \tag{12.52}$$

Finally we have, as in (12.41),

$$y[n] = \sum_{l=0}^{L-1} \{v_l\}_{(\uparrow L)}[n - l]. \tag{12.53}$$

The filters $r_{l,m}[i']$ are the polyphase components for sampling-rate conversion. There is a total of ML components. Note the slight irregularity in their definition, because of the need to distinguish between m below and above m_0. The sequences $u_m[i']$ are shifts and M-fold decimations of the input sequence, whereas $v_l[n']$ are shifts and L-fold decimations of the output sequence.

The procedure ppsrc in Program 12.3 is a MATLAB implementation of sampling-rate conversion by polyphase decomposition. As a special case, it can perform pure decimation (use $L = 1$), or pure interpolation (use $M = 1$). The program is perhaps not as transparent as the programs for polyphase decimation and interpolation, but it follows equations (12.49) through (12.53) closely.

12.5 Multistage Schemes*

When decimation by a large factor is required, it is often advantageous to divide the process into several stages. Such a division is possible if the decimation factor M is a composite number. Suppose that $M = M_1M_2$; then we can perform M-fold decimation as illustrated in Figure 12.20. We first filter the input signal $x[n]$ by a low-pass filter $H_1^z(z)$, then decimate by M_1, filter the decimated signal $u[n]$ by a low-pass filter $H_2^z(z)$, and finally decimate by M_2. The output signal $y[n]$ is decimated by M with respect to $x[n]$. The filters $H_1^z(z)$, $H_2^z(z)$ should be designed such that aliasing will be below a prescribed level and overall pass-band and stop-band tolerances will be met. This two-stage scheme often leads to computational savings, as we shall illustrate.

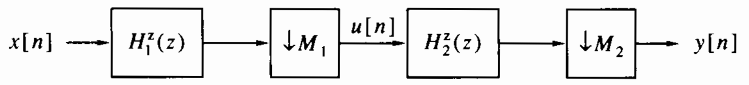

Figure 12.20 Multistage decimation.

Let us discuss the specifications of the two low-pass filters in two-stage decimation. We assume that the stop-band lower frequency of the overall system is $\theta_s = \pi/M = \pi/M_1M_2$, and the pass-band upper frequency is $\theta_p < \theta_s$. We also assume that tolerances δ_p, δ_s are given for the overall system. Figure 12.21(a) shows the desired frequency response of a low-pass filter in single-stage decimation. In this figure we take $M_1 = 3$, $M_2 = 2$, $M = 6$.

Let $\theta_{p,1}$, $\theta_{s,1}$ be the band-edge frequencies of the first filter. Since we are going to decimate by a factor M_1, it appears as though we need $\theta_{s,1} = \pi/M_1$ to avoid aliasing in the first decimation. Further thinking reveals that we can do with a higher stop-band frequency. After M_1-fold decimation, the first replica on the right of the zeroth replica is going to be centered around $\theta = 2\pi/M_1$, and stretched to the left down to $\theta = 2\pi/M_1 - \theta_{s,1}$. We can therefore take

$$\frac{2\pi}{M_1} - \theta_{s,1} = \frac{\pi}{M}, \quad \text{or} \quad \theta_{s,1} = \frac{2\pi}{M_1} - \frac{\pi}{M} = \frac{(2M_2 - 1)\pi}{M},$$

and still avoid aliasing in the frequency range up to π/M. This is illustrated in Figure 12.21(b). There will be aliasing at frequencies higher than π/M, but this aliasing is of no concern, since it is going to be eliminated at the next stage.

The pass-band upper frequency $\theta_{p,1}$ should be identical to θ_p of the overall system. It can be higher, but there is no advantage in this. The pass-band tolerance should be smaller than δ_p, since pass-band ripple will be present at the next stage as well. It is reasonable to require pass-band tolerance $0.5\delta_p$ at each of the two stages. The stop-band tolerance should be δ_s, the same as for the overall system.

Figure 12.21(c) shows the desired frequency response of the second filter. Now the sampling rate has already been reduced M_1-fold, so the band-edge frequencies of the second filter should be $\theta_{p,2} = M_1\theta_p$ and $\theta_{s,2} = M_1\theta_s = \pi/M_2$ in the new θ domain. The pass-band and stop-band tolerances should be $0.5\delta_p$ and δ_s, respectively. The following example illustrates multistage decimation.

Example 12.6 We are given a signal sampled at 96 kHz. The useful information in the signal is in the frequency band 0 to 3 kHz. It is decided to decimate the signal by $M = 12$. The required tolerances are $\delta_p = 0.01$, $\delta_s = 0.001$. A single-stage decimation filter must have $\theta_p = \pi/16$, $\theta_s = \pi/12$. Suppose we use equiripple design for the decimation filter. According to (9.79), the necessary order is $N = 244$. Since the

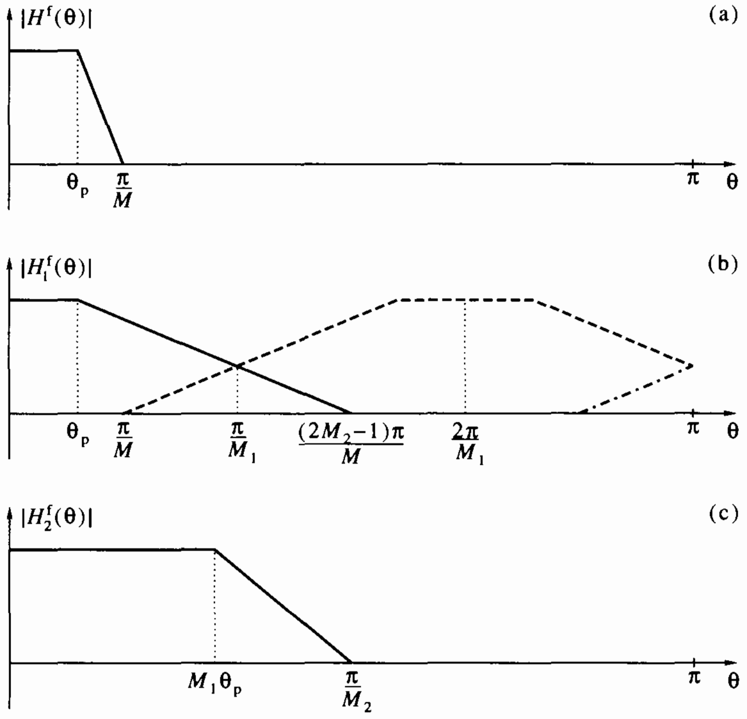

Figure 12.21 Frequency responses of filters for multistage decimation: (a) response of a single-state decimation filter; (b) response of the first-stage filter, dashed and dot-dashed lines show replicas generated by decimation; (c) response of the second-stage filter.

decimation ratio is 12, the number of operations per input data point is about 20.3. Now consider two-stage decimation, with $M_1 = 3$, $M_2 = 4$. The specifications of the first filter are

$$\theta_{p,1} = \pi/16, \quad \theta_{s,1} = 7\pi/12, \quad \delta_{p,1} = 0.005, \quad \delta_{s,1} = 0.001.$$

The corresponding order of an equiripple filter is $N_1 = 11$. Since the decimation ratio is 3, the number of operations per input data point is about 3.7. The specifications of the second filter are

$$\theta_{p,2} = 3\pi/16, \quad \theta_{s,2} = \pi/4, \quad \delta_{p,2} = 0.005, \quad \delta_{s,2} = 0.001.$$

The corresponding order of an equiripple filter is $N_2 = 88$. Since the decimation ratio is now 12, the number of operations per input data point is about 7.3. In summary, two-stage implementation requires about 11 operations per input point, or about half the number in single-stage implementation. You are asked to repeat these calculations when the roles of M_1 and M_2 are reversed (Problem 12.13). □

Multistage schemes can be used for interpolation equally well. Suppose that the interpolation factor is composite, say $L = L_1L_2$. Then we can perform L-fold interpolation as illustrated in Figure 12.22. We first expand the input signal $x[n]$ by L_1 and pass it through a low-pass filter $H_1^z(z)$, then expand by L_2, and finally pass through a low-pass filter $H_2^z(z)$. The output signal $y[n]$ is interpolated by L with respect to $x[n]$. The filters $H_1^z(z)$, $H_2^z(z)$ should be designed such that images will be suppressed below a prescribed level and overall pass-band and stop-band tolerances will be met. This two-stage scheme often leads to computational savings, as we shall illustrate.

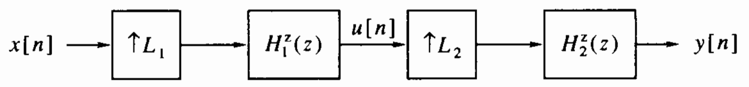

Figure 12.22 Multistage interpolation.

Let us discuss the specifications of the two low-pass filters in two-stage interpolation. Figure 12.23(a) shows the Fourier transform of the input signal. We assume that the frequency response decays to zero at the high frequency range, to allow for finite transition band of the interpolation filter. If this is not the case, the interpolation filter will inevitably attenuate a part of the high-frequency components. Figure 12.23(b) shows the Fourier transform of the signal expanded by L_1, before and after the filter. The frequency response of $H_1^z(z)$ in this figure is assumed to be equal to $U^f(\theta)$ in magnitude. Clearly $\theta_{s,1}$ must be equal to π/L_1. Figure 12.23(c) shows the Fourier transform of the signal expanded by L_2, before and after the filter, and the magnitude response of $H_2^z(z)$. As we see, $\theta_{s,2}$ can be as large as $(2L_1 - 1)\pi/L$ and still suppress all images. The pass-band tolerances $\delta_{p,1}$, $\delta_{p,2}$ should be taken as $0.5\delta_p$ each, as in the case of decimation. The stop-band tolerances $\delta_{s,1}$, $\delta_{s,2}$ should be taken according to the desired level of image suppression.

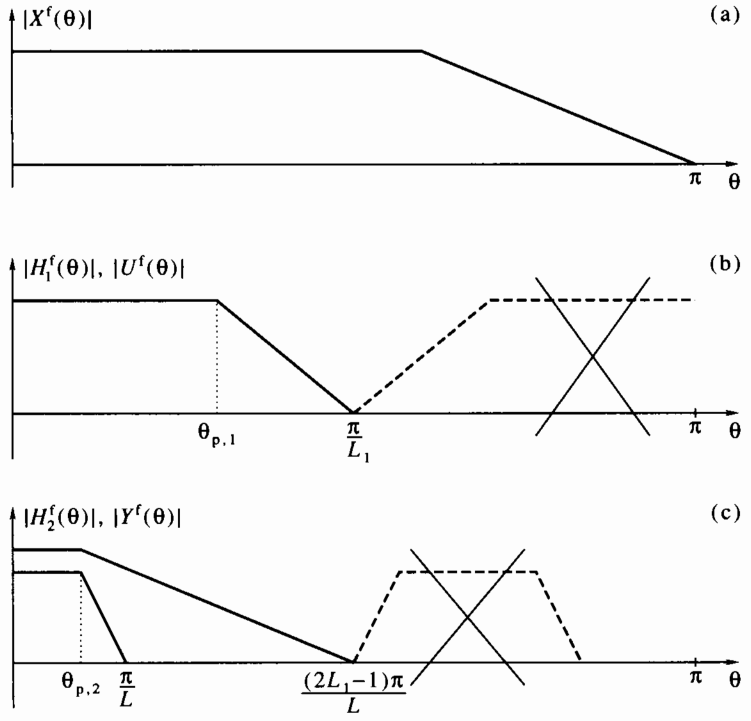

Figure 12.23 Frequency responses of signals and filters for multistage interpolation: (a) response of the input signal; (b) response of the first-stage filter, dashed line shows image suppressed by the filter; (c) response of the second-stage filter and of the output signal.

It is clear from the preceding description that multistage interpolation is dual to multistage decimation. For example, if we reverse the procedure described in Example 12.6 (with $L_1 = 4$, $L_2 = 3$), we will find that the filters have the same orders, and the same computational saving can be achieved.

Multistage schemes are also useful for sampling-rate conversion systems. As an example, consider the problem of transferring digitally recorded music from digital tape to compact disc. Digital audiotapes use sampling rate 48 kHz and compact discs use 44.1 kHz. The required ratio L/M is therefore $147/160$. Single-stage sampling-rate conversion would require expanding the input signal to over 7 MHz. Factoring L and M into primes, we find that $147 = 7 \times 7 \times 3$, and $160 = 2^5 \times 5$. We can therefore build a three-stage system: one stage with $L_1/M_2 = 7/10$, a second with $L_2/M_2 = 7/8$, and a third with $L_3/M_3 = 3/2$. Such a system would be much more efficient than a single-stage system and would obviate the need for signals in the megahertz range.

12.6 Filter Banks*

12.6.1 Subband Processing

An important application of multirate signal processing is *subband processing*. Subband processing is based on splitting the frequency range into M segments (subbands), which together encompass the entire range. Each subband is processed independently, as called for by the specific application. If necessary, the subbands are recombined, after processing, to form an output signal whose bandwidth occupies the entire frequency range.

Example 12.7 Consider a multiple-user communication system in which several (say M) users share a single carrier channel, whose bandwidth is sufficient to accommodate all M users. One method of sharing the channel between users is by *frequency division multiplexing*. In this method, a subband of width $1/M$ of the entire channel is allocated to each user. The receiver of such a system has to split the received signal into subbands, then examine the contents of each separately. Traditional systems of this kind used analog band-pass filtering for this purpose. However, modern systems rely more and more on digital filtering. In this application, there is usually no need to reconstruct the full bandwidth signal again after processing. □

The division of a signal into subbands is usually accomplished by a *filter bank*. As the name implies, a filter bank is a collection of band-pass filters, all processing the same input signal. Figure 12.24 illustrates the structure of a filter bank. In this figure, the transfer functions of the band-splitting filters are $\{G_m^z(z),\ 0 \le m \le M-1\}$. These filters are also known as the *analysis bank*. The outputs of the analysis filters, the signals $u_m[n]$, are fed into the subband processing system, resulting in the processed subband signals $v_m[n]$. If there is a need to reconstruct the signal after processing, this is usually accomplished by a second bank of filters, called the *synthesis bank*. In Figure 12.24, the transfer functions of the synthesis filters are $\{H_m^z(z),\ 0 \le m \le M-1\}$.

The output $y[n]$ of the filter bank will practically always be different from the input signal $x[n]$, mainly because of the block performing the subband processing (e.g., the compression method described in Example 12.8; see page 493). However, suppose we eliminate the subband processing block and short-circuit each signal $u_m[n]$ to the corresponding $v_m[n]$. It is highly desirable to design the filter bank such that, in this case, $y[n]$ will be equal to $x[n]$, except for constant delay. A filter bank for which (in the absence of subband processing) $y[n]$ differs from $x[n]$ only by a constant delay and a constant scale factor, is said to have the *perfect reconstruction* property. Whether a

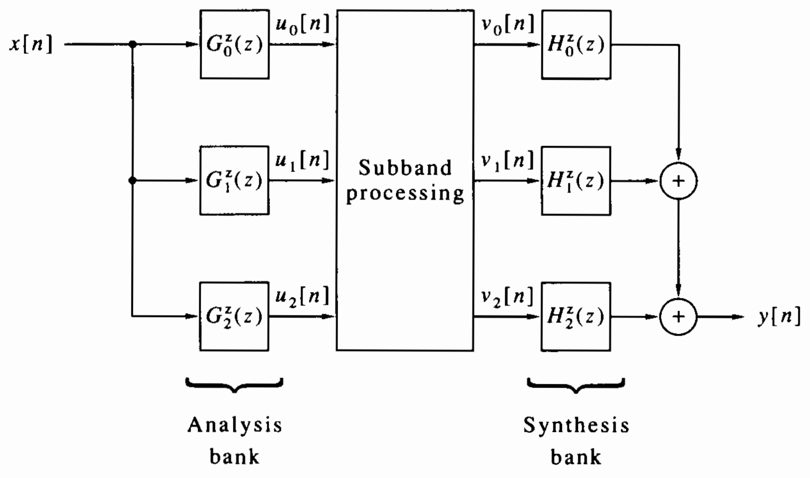

Figure 12.24 Subband processing with a filter bank.

given filter bank has this property depends on the design of the analysis and synthesis filters. For now, we say only that perfect reconstruction is highly desirable, but not at all easy to achieve.

Figure 12.25 shows a typical division of the frequency range into subbands. In this figure, all subbands have the same width. However, this is by no means a necessity. In certain applications it is of advantage to use subbands of different widths. When all subbands have the same width, the filter bank is called *uniform*. The following points are worth noting:

1. The zeroth subband usually occupies the range $[-\pi/M, \pi/M]$. The filter $g_0[n]$, which is responsible for this subband, has real coefficients.
2. All the other subbands are *nonsymmetric* as a function of the frequency θ, implying that their coefficients are complex.
3. The organization of the subbands for odd M slightly differs from that for even M. In the case of even M, band number $M/2$ is split evenly between positive and negative frequencies.
4. It is common for the subbands to overlap slightly, as shown in Figure 12.25. This overlap is necessary to prevent gaps in the spectrum, since physical filters have finite transition bands.

12.6.2 Decimated Filter Banks

The signals $u_m[n]$ shown in Figure 12.24 are band limited. If the filter bank is uniform, each has a bandwidth of about $2\pi/M$. As we saw in Section 12.2, such a signal can be decimated by a factor up to M without being aliased. This is true whether the signal is low pass (cf. Figure 12.6) or band pass (cf. Figure 12.7). In either case, the decimated signal occupies the entire frequency band (if decimated M-fold). The filter bank shown in Figure 12.24 is wasteful, since it does not use this property of the subband signals.

Figure 12.26 shows a modified filter bank, in which each subband signal is decimated by a factor K. The filters in this bank are complex in general (except when $K = 2$). The decimation factor can be either equal to M or smaller (but not larger, since this will lead to aliasing in general). This is called a *decimated filter bank.* If $K = M$, it is

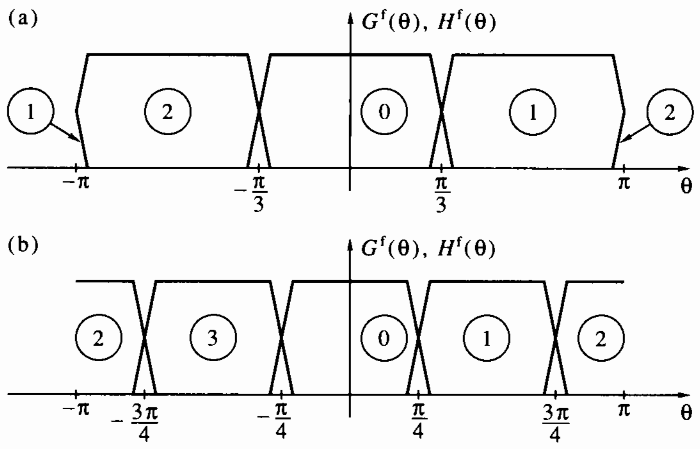

Figure 12.25 Division of the frequency range into bands in subband processing: (a) odd number (shown for $M = 3$); (b) even number (shown for $M = 4$).

called a *maximally decimated filter bank.* Note that we are using this terminology only for uniform filter banks. Before recombination, the processed subband signals should be expanded by the same factor K (unless sampling-rate conversion is required), then interpolated by the narrow-band filters $h_m[n]$, and finally combined to form $y[n]$.

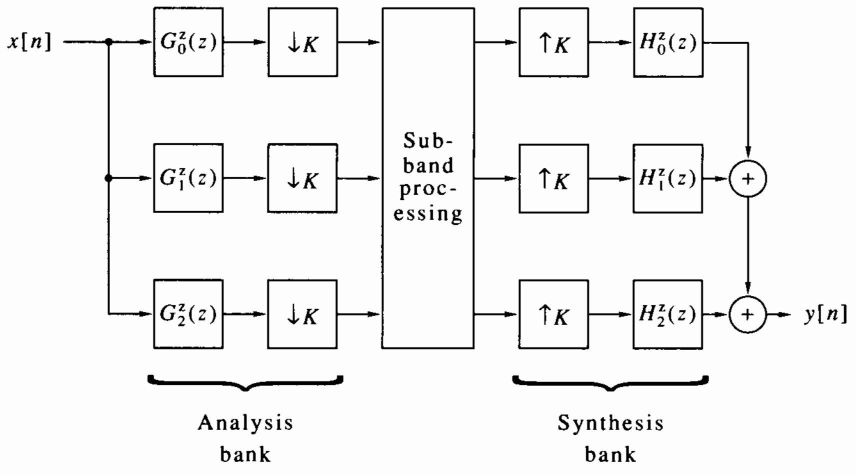

Figure 12.26 A decimated filter bank.

Maximally decimated filter banks achieve the maximum possible computational efficiency, because the subband signals have the lowest possible rate. However, the decimation and interpolation filters $g_m[n]$ and $h_m[n]$ are typically required to meet tight specifications. For this and other reasons, it is sometimes preferable to decimate by a factor smaller than the highest possible. We continue to discuss only maximally decimated filter banks.

12.7 Two-Channel Filter Banks*

The simplest filter bank separates the input signal into two bands, therefore it is called a *two-channel filter bank.* Figure 12.27 shows a maximally decimated, two-channel filter bank. The left side shows the analysis part, and the right side—the synthesis part. The analysis filter $G_0^z(z)$ is low pass, and its pass band should be approximately $[0, 0.5\pi]$. The analysis filter $G_1^z(z)$ is high pass, and its pass band should be approximately $[0.5\pi, \pi]$. The same holds for the synthesis filters $H_0^z(z)$ and $H_1^z(z)$, respectively. The outputs of the analysis filters, $u_0[n]$ and $u_1[n]$, are decimated by 2. The figure does not show the subband processing part, but assumes that the decimated signals are immediately expanded by 2, to yield the signals $v_0[n]$ and $v_1[n]$. These are passed through the synthesis filters and combined to form the output signal $y[n]$. The filters in a two-channel filter bank have real coefficients. They can be either FIR or IIR; here we limit ourselves to FIR filters.[1]

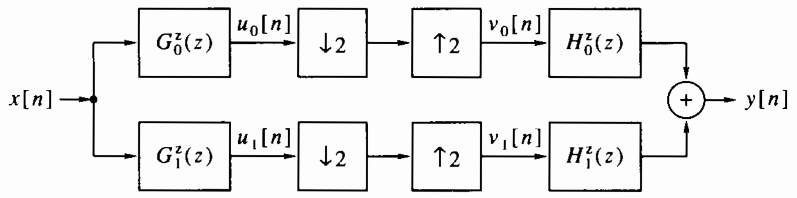

Figure 12.27 A maximally decimated, two-channel filter bank.

12.7.1 Properties of Two-Channel Filter Banks

Before we discuss the design of a two-channel filter bank, we analyze its properties and, in particular, explain what errors are likely to arise and how they can be eliminated or minimized. We do this by computing the z-transform of the output signal as a function of the z-transform of the input and the transfer functions of the four filters. The intermediate signals $u_0[n]$ and $u_1[n]$ have the z-transforms

$$U_0^z(z) = G_0^z(z)X^z(z), \quad U_1^z(z) = G_1^z(z)X^z(z). \tag{12.54}$$

Using Problem 12.4, we can express the z-transforms of $v_0[n]$ and $v_1[n]$ as

$$V_0^z(z) = 0.5U_0^z(z) + 0.5U_0^z(-z), \tag{12.55a}$$
$$V_1^z(z) = 0.5U_1^z(z) + 0.5U_1^z(-z). \tag{12.55b}$$

Substituting (12.54) in (12.55b) gives

$$V_0^z(z) = 0.5G_0^z(z)X^z(z) + 0.5G_0^z(-z)X^z(-z), \tag{12.56a}$$
$$V_1^z(z) = 0.5G_1^z(z)X^z(z) + 0.5G_1^z(-z)X^z(-z). \tag{12.56b}$$

Therefore,

$$Y^z(z) = H_0^z(z)V_0^z(z) + H_1^z(z)V_1^z(z) = 0.5[G_0^z(z)H_0^z(z) + G_1^z(z)H_1^z(z)]X^z(z) + 0.5[G_0^z(-z)H_0^z(z) + G_1^z(-z)H_1^z(z)]X^z(-z). \tag{12.57}$$

Formula (12.57) expresses the output signal as a sum of two components. The first is the response to the input signal $X^z(z)$, and the second is the response to the alias component $X^z(-z)$ (see Problem 12.5). To eliminate the alias component from the output signal, it is necessary and sufficient to have

$$G_0^z(-z)H_0^z(z) + G_1^z(-z)H_1^z(z) = 0. \tag{12.58}$$

There are many ways of satisfying this condition. A convenient one is to take the synthesis filters as

$$H_0^z(z) = 2G_1^z(-z), \quad H_1^z(z) = -2G_0^z(-z). \tag{12.59}$$

This choice not only eliminates the alias component, but also satisfies the basic requirement from the synthesis filters, namely that $H_0^z(z)$ be low pass and $H_1^z(z)$ be high pass (Problem 12.15).

Note that elimination of aliasing in the output signal does not imply its elimination in any of the channels separately. The decimated signals $u_0[2n]$ and $u_1[2n]$ are aliased in general. This aliasing is exactly canceled by the summing operation at the output of the filter bank, provided (12.58) is satisfied. It would not be desirable to eliminate the aliasing in $u_0[2n]$ and $u_1[2n]$ individually, because this would inevitably lead to considerable distortions in the frequency range around 0.5π, due to the finite transition bands of the analysis filters.

When condition (12.59) is used, the output formula (12.57) becomes

$$Y^z(z) = [G_0^z(z)G_1^z(-z) - G_0^z(-z)G_1^z(z)]X^z(z). \tag{12.60}$$

In this case, the filter bank is linear time invariant and its transfer function is

$$F^z(z) = G_0^z(z)G_1^z(-z) - G_0^z(-z)G_1^z(z). \tag{12.61}$$

An ideal filter bank would satisfy $F^z(z) = 1$, therefore $y[n] = x[n]$. Such a filter is not realizable, however, since all four filters must be causal. Next best is to have

$$F^z(z) = cz^{-l} \tag{12.62}$$

for a constant c and a nonnegative integer l. In this case $y[n] = cx[n-l]$, so $y[n]$ is a distortion-free reconstruction of $x[n]$. The filter bank that satisfies (12.62) is said to have the *perfect reconstruction* property.

A filter bank that does not satisfy (12.62) leads to distortions. In general,

$$F^z(z) = A(\theta)e^{j\phi(\theta)}.$$

If $A(\theta) \not\equiv c$, the filter bank has *amplitude distortion.* If $\phi(\theta)$ is not linear, it has *phase distortion.* A filter bank that does not have the perfect reconstruction property can have either kind of distortion, or both.

12.7.2 Quadrature Mirror Filter Banks

An analysis filter pair satisfying

$$G_1^z(z) = G_0^z(-z), \quad \text{or} \quad G_1^f(\theta) = G_0^f(\theta - \pi) \tag{12.63}$$

is called a *quadrature mirror filter bank,* or QMF bank for short [Esteban and Galand, 1977]. Note that (12.63) can also be expressed in the form

$$G_1^f(0.5\pi + \theta) = \bar{G}_0^f(0.5\pi - \theta). \tag{12.64}$$

We see that $G_1^f(\cdot)$ is the mirror image of $G_0^f(\cdot)$ around the point $\theta = 0.5\pi$: The magnitude is symmetric, and the phase is antisymmetric with respect to this point. This is the reason for the name *quadrature mirror filter.* We also see that $G_1^f(\theta)$ is high pass if $G_0^f(\theta)$ is low pass.

The transfer function of a QMF bank is obtained by substituting (12.64) in (12.61). This gives

$$F^z(z) = [G_0^z(z)]^2 - [G_0^z(-z)]^2. \tag{12.65}$$

Problem 12.16 explores the limitations on FIR filter banks resulting from the quadrature mirror restriction. As shown there, such a bank can have perfect reconstruction only if $G_0^z(z)$ contains two nonzero coefficients at most. An FIR filter having only two nonzero coefficients is too simple to be of any practical use. Therefore, the use of FIR QMF banks behooves us to forfeit the perfect reconstruction property.

A QMF bank whose prototype filter $G_0^z(z)$ has linear phase is free of phase distortion. To show this, let

$$G_0^f(\theta) = A(\theta)e^{-j0.5\theta N}, \quad G_0^f(\theta-\pi) = A(\theta-\pi)e^{-j0.5(\theta-\pi)N}, \tag{12.66}$$

where N is the order of $G_0^z(z)$ and $A(\theta)$ is its amplitude function. We get from (12.65),

$$F^f(\theta) = [A^2(\theta) - (-1)^N A^2(\theta-\pi)]e^{-j\theta N}. \tag{12.67}$$

Since $A(\theta)$ is either symmetric or antisymmetric,

$$|F^f(0.5\pi)| = |A^2(0.5\pi) - (-1)^N A^2(0.5\pi)|. \tag{12.68}$$

Therefore, if we choose an even-order N, we will get $F^f(0.5\pi) = 0$. This implies that N must be odd, for otherwise we get an unacceptable amplitude distortion at $\theta = 0.5\pi$. Assuming that N is odd, we get from (12.67),

$$F^f(\theta) = [A^2(\theta) + A^2(\theta-\pi)]e^{-j\theta N} = [|G_0^f(\theta)|^2 + |G_0^f(\theta-\pi)|^2]e^{-j\theta N}. \tag{12.69}$$

The amplitude distortion of a linear-phase QMF bank is

$$|F^f(\theta)| - 1 = |G_0^f(\theta)|^2 + |G_0^f(\theta-\pi)|^2 - 1. \tag{12.70}$$

In practice, the amplitude distortion can be made very small by a careful design of $G_0^z(z)$. Distortions of 0.01 and smaller can be achieved, using numerical optimization techniques. Such techniques have been derived by Johnston [1980] and Jain and Crochiere [1983]; we do not discuss them here.

12.7.3 Perfect Reconstruction Filter Banks

To get perfect reconstruction FIR filter banks, we must abandon the quadrature mirror property (12.63). Smith and Barnwell [1984], and Mintzer [1985] developed FIR filter banks having the perfect reconstruction property. They replaced (12.63) by the condition

$$G_1^z(z) = (-z)^{-N} G_0^z(-z^{-1}). \tag{12.71}$$

Filters satisfying (12.71) are called *conjugate quadrature filters*, or CQF. The coefficients of $G_1^z(z)$ are related to those of $G_0^z(z)$ by (Problem 12.21)

$$g_1[n] = (-1)^n g_0[N-n]. \tag{12.72}$$

Substituting (12.71) in (12.61), we get

$$F^z(z) = z^{-N} G_0^z(z) G_0^z(z^{-1}) - (-z)^{-N} G_0^z(-z) G_0^z(-z^{-1}). \tag{12.73}$$

Assuming further that N is chosen to be odd, we get

$$F^z(z) = z^{-N}[G_0^z(z) G_0^z(z^{-1}) + G_0^z(-z) G_0^z(-z^{-1})]. \tag{12.74}$$

A sufficient condition for (12.74) to be of the form (12.62) is that

$$G_0^z(z) G_0^z(z^{-1}) + G_0^z(-z) G_0^z(-z^{-1}) = 1. \tag{12.75}$$

Comparing this with (8.73), we see that (12.75) means that $G_0^z(z)G_0^z(z^{-1})$ is a zero-phase, half-band filter. We also see that

$$G_0^f(\theta)\bar{G}_0^f(\theta) + G_0^f(\theta-\pi)\bar{G}_0^f(\theta-\pi) = |G_0^f(\theta)|^2 + |G_0^f(\theta-\pi)|^2 = 1. \tag{12.76}$$

A filter $G_0^z(z)$ satisfying (12.76) is said to be *power symmetric*. Thus, a filter $G_0^z(z)$ is power symmetric if $G_0^z(z)G_0^z(z^{-1})$ is a zero-phase half band. The design procedure of a CQF bank therefore concentrates on finding an odd-order, power-symmetric filter whose pass band is approximately $[0, 0.5\pi]$, and such that required pass-band and stop-band tolerances are met. Once such a filter has been found, the filter bank design is completed by using (12.71) and (12.59).

The design of a filter $G_0^z(z)$ that meets the aforementioned requirements is not straightforward. The first step is to design a zero-phase, half-band filter $R^z(z)$. Then this filter must be factored as

$$R^z(z) = G_0^z(z)G_0^z(z^{-1}). \tag{12.77}$$

Such a factorization is called *spectral factorization*, and is possible if and only if

$$R^f(\theta) \geq 0, \quad \text{for all} \quad -\pi \leq \theta \leq \pi. \tag{12.78}$$

Necessity of this condition is obvious, since $R^f(\theta) = |G_0^f(\theta)|^2$. Sufficiency will follow from the procedure described next. We are therefore faced with two tasks: (1) to design a zero-phase, half-band filter that meets (12.78), and (2) to carry out the spectral factorization (12.77). We now describe methods for performing these two tasks.

Nonnegative half-band filter design We describe two methods for designing a half-band filter that satisfies (12.78), one based on windows and one based on the Parks-McClellan algorithm. In the first method, we start by choosing a window $w[n]$ of even length $2M + 2$. We convolve this window with itself to get a window $\{w * w\}[n]$ of length $4M + 3$, and normalize the new window so that its middle coefficient will be 1. The new window has a nonnegative kernel function $[W^f(\theta)]^2$. We then apply this window to the ideal impulse response of a zero-phase, half-band filter, which is

$$h_d[n] = 0.5\text{sinc}(0.5n). \tag{12.79}$$

The resulting impulse response is half band of order $4M + 2$, since $h_d[n]$ is half band (see Problem 8.11). Furthermore, the convolution of $H_d(\theta)$ with $[W^f(\theta)]^2$ is nonnegative for all θ, being the integral of two nonnegative functions. This method is simple to implement but requires experimentation with M and the window type, for meeting the tolerances. Also, as with all window-based methods, it does not yield a filter of the smallest possible order.

The second method is based on the idea developed in Problem 9.35 [Vaidyanathan and Nguyen, 1987]. There it is shown how to design an equiripple half-band filter, starting from a one-band filter designed by the Parks-McClellan algorithm. Suppose the half-band filter thus designed, $R_1^z(z)$, has an order $4M + 2$ and tolerances δ_p, δ_s. We assume that the coefficients of $R_1^z(z)$ are in the range $-(2M + 1) \leq n \leq 2M + 1$, so this filter is zero phase. Since the filter is equiripple, $R_1^f(\theta)$ is bounded from below by $-\delta_s$. Therefore, if we take

$$R^z(z) = \frac{0.5}{0.5 + \delta_s}[R_1^z(z) + \delta_s], \tag{12.80}$$

then we will have $R^f(\theta) \geq 0$, as required. The coefficients of $R^z(z)$ are obtained from those of $R_1^z(z)$ by

$$r[n] = \frac{0.5}{0.5 + \delta_s}(r_1[n] + \delta_s\delta[n]). \tag{12.81}$$

The factor $0.5/(0.5 + \delta_s)$ is necessary, because a zero-phase, half-band filter must satisfy $R^f(0.5\pi) = 0.5$ (see Problem 8.11). Since $R_1^f(0.5\pi) = 0.5$, this is exactly the factor that meets this requirement.

The frequency response $R^{\mathrm{f}}(\theta)$ obtained by the aforementioned procedure will be exactly zero at the points of local minima in the stop band of $R_1^{\mathrm{f}}(\theta)$. The spectral factorization procedure described next is numerically sensitive to these zeros. The accuracy of spectral factorization is greatly improved if $R^{\mathrm{f}}(\theta)$ is strictly greater than zero at all frequencies, as in the case of window design. This can be achieved by a slight change in the procedure: We replace δ_{s} by δ_{s}', slightly larger than δ_{s} (for example, $\delta_{\mathrm{s}}' = \mu\delta_{\mathrm{s}}$, where μ is slightly larger than 1). You are asked to compute the pass-band and stop-band tolerances of $R^{\mathrm{z}}(z)$ as a function of δ_{p}, δ_{s}, and δ_{s}' (Problem 12.22).

Spectral factorization Having obtained a filter $R^{\mathrm{z}}(z)$ of order $4M + 2$, we are looking for a filter $G_0^{\mathrm{z}}(z)$ of order $2M + 1$ that will satisfy (12.77). We observe that, if β is a zero of $G_0^{\mathrm{z}}(z)$, then β^{-1} is a zero of $G_0^{\mathrm{z}}(z^{-1})$, hence both β and β^{-1} are zeros of $R^{\mathrm{z}}(z)$. Since we have designed $R^{\mathrm{z}}(z)$ such that $R^{\mathrm{f}}(\theta) > 0$ for all θ, $R^{\mathrm{z}}(z)$ has no zeros on the unit circle.[2] Therefore, the $4M + 2$ zeros of $R^{\mathrm{z}}(z)$ can be divided into two sets: the set $\{\beta_k,\ 1 \le k \le 2M + 1\}$ of zeros inside the unit circle, and the set of their reciprocals $\{\beta_k^{-1},\ 1 \le k \le 2M + 1\}$. We can now construct $G_0^{\mathrm{z}}(z)$ as

$$G_0^{\mathrm{z}}(z) = C \prod_{k=1}^{2M+1} (1 - \tilde{\beta}_k z^{-1}), \tag{12.82}$$

where each of the $\tilde{\beta}_k$ can be taken as either β_k or β_k^{-1}. We thus have $2^{(2M+1)}$ different choices for $G_0^{\mathrm{z}}(z)$, each satisfying (12.77). In particular, choosing $\tilde{\beta}_k = \beta_k$ for all k makes $G_0^{\mathrm{z}}(z)$ a minimum-phase filter. In general, no choice can make $G_0^{\mathrm{z}}(z)$ a linear-phase filter.

The constant C in (12.82) is computed as follows. We know that

$$r[0] = \sum_{n=0}^{2M+1} (g_0[n])^2 = 0.5, \tag{12.83}$$

because of the half-band property of $R^{\mathrm{z}}(z)$. We therefore compute the coefficients of $G_0^{\mathrm{z}}(z)$ by expanding (12.82) first with $C = 1$. We then take C^2 as half the reciprocal of the sum of squares of the coefficients, and multiply the vector of coefficients by C. The new coefficients now satisfy (12.83).

The procedure cqfw in Program 12.4 designs a CQF bank using the window-based method and the spectral factorization method we have described. It accepts an even-length window as input and provides the four filters (analysis and synthesis) as outputs. The program first convolves the window with itself, normalizes the result, and calls `firdes` to design a half-band filter with this window. The filter thus obtained is factored to its roots, and the set of roots inside the unit circle is selected. These are expanded to form the polynomial $G_0^{\mathrm{z}}(z)$, which is then normalized to satisfy (12.83). Finally, $G_1^{\mathrm{z}}(z)$ is obtained according to (12.71), and $H_0^{\mathrm{z}}(z)$, $H_1^{\mathrm{z}}(z)$ according to (12.59).

12.7.4 Tree-Structured Filter Banks

Splitting of a signal into two bands has limited usefulness. However, splitting into 2^L channels is possible for any integer L using a tree-structured filter bank, as shown in Figure 12.28 (using $L = 2$ as an example). Each of the two outputs of the first level is passed to a two-channel bank at a second level. This process can be continued according to the desired number of subbands. The filter pairs shown in Figure 12.28 are all equal. This, however, is not necessary; different pairs can be used at different levels.

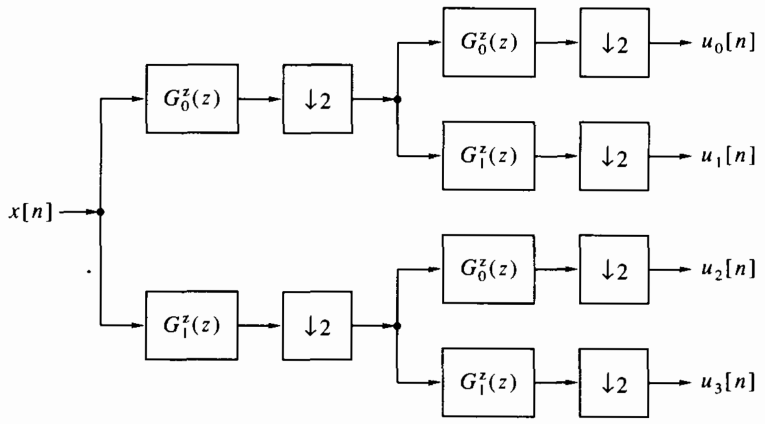

Figure 12.28 A tree-structured analysis filter bank.

The synthesis tree-structured filter bank corresponding to the bank in Figure 12.28 is shown in Figure 12.29. If the outputs of the analysis bank are fed to the synthesis bank, the complete system will have perfect reconstruction provided each two-channel bank has perfect reconstruction.

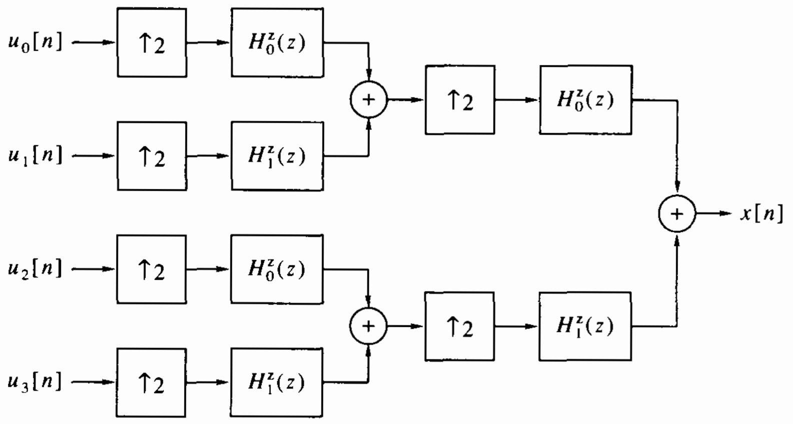

Figure 12.29 A tree-structured synthesis filter bank.

A full-blown tree, such as the those shown in Figures 12.28 and 12.29, is not always needed. A tree can be pruned by eliminating parts of certain levels. For example, the two bottom filters at the second level in the tree shown in Figure 12.28 can be eliminated. Then the filter bank will have only three outputs, two decimated by 4 and one decimated by 2. The synthesis bank must be constructed in a dual manner, to preserve the perfect reconstruction property.

Example 12.8 Compression of speech signals is highly desirable in all applications requiring transmission or storage of speech. Examples include commercial telephony, voice communication by radio, and storage of speech for multimedia applications. When speech is compressed and later reconstructed, the resulting quality usually undergoes a certain degradation. A common measure of speech quality is the mean opinion score (MOS). The MOS of a compressed speech is obtained as follows. An ensemble of people is recruited and listens to a sample of the reconstructed speech. Each person is asked to give a score between 1 to 5, where:

- 5 is for an excellent quality with imperceptible degradation;
- 4 is for a good quality with perceptible, but not annoying, degradation;
- 3 is for a fair quality with perceptible and slightly annoying degradation;
- 2 is for a poor quality with annoying, but not objectionable, degradation;
- 1 is for a bad quality with objectionable degradation.

The individual scores are averaged to give the MOS. For example, MOS of 4.5 and above is usually regarded as *toll quality*, meaning that it can be used for commercial telephony. MOS of 3 to 4 is regarded as *communications quality*, and is acceptable for many specialized applications. MOS lower than 3 is regarded as *synthetic quality*.

Speech signals are typically converted to a digital form by sampling at a rate 8 kHz or higher (up to about 11 kHz). A speech signal sampled at 8 kHz and quantized to 8 bits per sample has MOS of about 4.5. The corresponding rate is 64,000 bits per second. The *compression ratio* is defined as the number of bits per second before compression divided by that after compression.

One of the first speech compression techniques put into use was *subband coding*. In subband coding, the frequency range of the sampled speech is split into a number subbands by an analysis filter bank. The spectrum of a speech signal is decidedly nonuniform over the frequency range, so its subbands have different energies. This makes it possible to quantize each subband with a different number of bits. Reconstruction of the compressed speech consists of decoding each subband separately, then combining them to a full bandwidth signal using a synthesis filter bank.

The following scheme, due to Crochiere [1981], is typical of subband coding. The signal is sampled at 8 kHz, and split into four subbands, as shown in Figure 12.28. Each of these bands thus has a bandwidth of 1 kHz. Next, the band 0 to 1 kHz is split further into two bands. We therefore get a pruned tree with five bands: 0–0.5, 0.5–1, 1–2, 2–3, and 3–4 kHz. Three quantization schemes have been proposed:

1. Quantization to 5 bits per sample in the first two bands, 4 bits per sample in the third and fourth bands, and 3 bits per sample in the fifth band. The bit rate thus obtained is
$$(2 \times 5 \times 1000) + (2 \times 4 \times 2000) + 3 \times 2000 = 32,000.$$
This scheme achieves MOS of 4.3 [Daumer, 1982].
2. Quantization to 5 bits per sample in the first two bands, 4 bits per sample in the third band, 3 bits per sample in the fourth band, and 0 bits per sample in the fifth band (i.e., no use of this band). The bit rate thus obtained is
$$(2 \times 5 \times 1000) + 4 \times 2000 + 3 \times 2000 = 24,000.$$
This scheme achieves MOS of 3.9 [Daumer, 1982].
3. Quantization to 4 bits per sample in the first two bands, 2 bits per sample in the third and fourth bands, and 0 bits per sample in the fifth band. The bit rate thus obtained is
$$(2 \times 4 \times 1000) + (2 \times 2 \times 2000) = 16,000.$$
This scheme achieves MOS of 3.1 [Daumer, 1982]

Subband coding has been superseded by more efficient techniques and is not considered a state-of-art speech compression technique any more. However, it has recently gained popularity in compression of high-fidelity audio. A typical raw bit rate for high-fidelity audio is about 0.7 megabit per second per channel. The MPEG[3] standard for audio compression defines 32 subbands and allows compression down to 128 kilobits per second per channel. This represents a compression ratio of about 5.5 and gives almost CD-like music quality. □

12.7.5 Octave-Band Filter Banks

An *octave-band filter bank* is a special kind of a pruned-tree filter bank, constructed according to the following rule: At each level, the high-pass output is pruned and the low-pass output is forwarded to the next level. Figure 12.30 shows a three-level octave-band filter bank. The left half forms the analysis bank, and the right half the synthesis bank. In this figure we have used a self-explanatory concise depiction of filtering-decimation and expansion-filtering.

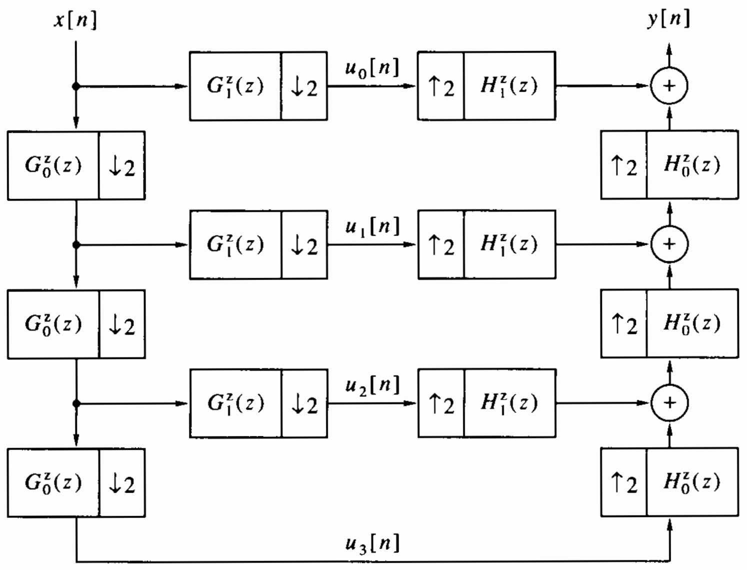

Figure 12.30 An octave-band filter bank.

The bandwidths of the output signals of the analysis bank are in octaves. For example, in Figure 12.30, the bandwidth of $u_0[n]$ is $[0.5\pi, \pi]$, that of $u_1[n]$ is $[0.25\pi, 0.5\pi]$, that of $u_2[n]$ is $[0.125\pi, 0.25\pi]$, and that of $u_3[n]$ is $[0, 0.125\pi]$. An L-level bank divides the full bandwidth into L octaves and yields $L+1$ decimated signals. The last two signals have the same bandwidth, namely $2^{-L}\pi$. As in the case of a general pruned-tree filter bank, perfect reconstruction holds for the complete bank if it holds for the two-channel bank.

Octave-band filter banks are closely related to *wavelets*. The subject of wavelets has received enormous popularity in recent years, but is not discussed in this book; see Vetterli and Kovacevic [1995].

12.8 Uniform DFT Filter Banks*

12.8.1 Filter Bank Interpretation of the DFT

Uniform DFT filter banks offer a simple, yet powerful way of implementing a maximally decimated filter bank. As a motivation for this topic, consider a simple example. Suppose we pass the signal $x[n]$ through a chain of $M-1$ delays and denote the output of the mth delay by

$$x_m[n] = x[n-m]. \tag{12.84}$$

Now compute, at each time point n, the M-point conjugate DFT of the vector

$$\{x_0[n], x_1[n], \dots, x_{M-1}[n]\}$$

(the reason for conjugation will be explained soon). Denote by $u_m[n]$ the mth component of the output vector at time n. Then

$$u_m[n] = \sum_{i=0}^{M-1} x_i[n] W_M^{im} = \sum_{i=0}^{M-1} x[n-i] W_M^{im}. \tag{12.85}$$

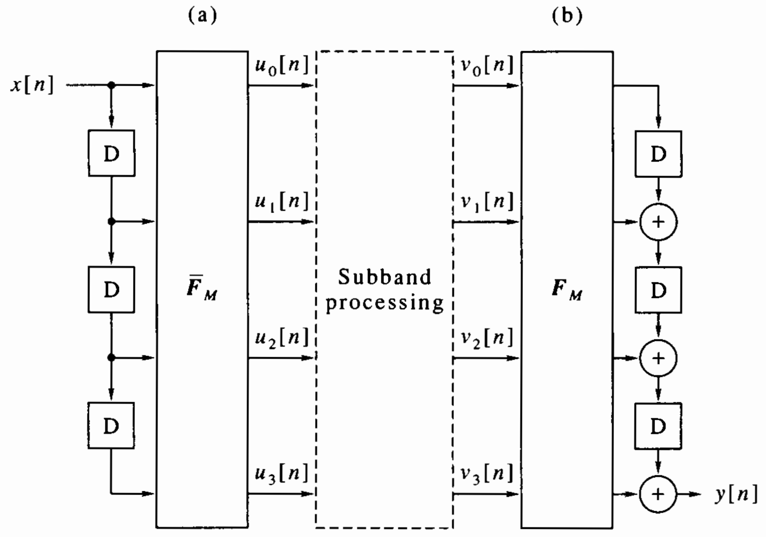

Figure 12.31 A simple uniform DFT filter bank ($M = 4$): (a) analysis bank; (b) synthesis bank.

The preceding operations are shown schematically in Figure 12.31(a), for $M = 4$. Recall that F_M is the DFT matrix [as defined in (4.19)] and $\bar{F}_M$ is its conjugate. As we see, equation (12.85) represents a linear convolution between the (potentially infinite) sequence $x[n]$ and the length-M sequence

$$g_m[n] = W_M^{nm}, \quad 0 \le n \le M-1. \tag{12.86}$$

Therefore, this sequence can be regarded as the impulse response of a length-M FIR filter, and (12.85) can be regarded as the result of passing $x[n]$ through this filter. Since there are M such filters, we get a filter bank. Let us compute the transfer functions and frequency responses of these filters. We have

$$G_m^z(z) = \sum_{n=0}^{M-1} W_M^{nm} z^{-n} = \frac{1-(zW_M^{-m})^{-M}}{1-(zW_M^{-m})^{-1}}, \tag{12.87}$$

and

$$G_m^{\mathrm{f}}(\theta) = \frac{1 - e^{-jM(\theta - 2\pi m/M)}}{1 - e^{-j(\theta - 2\pi m/M)}} = D\left(\theta - \frac{2\pi m}{M}, M\right) e^{-j0.5(\theta - 2\pi m/M)(M-1)}, \tag{12.88}$$

where $D(\cdot, \cdot)$ is the Dirichlet kernel, defined in (6.7). Consider in particular the filter corresponding to $m = 0$. The transfer function and the frequency response of this filter are given by

$$G_0^{\mathrm{z}}(z) = 1 + z^{-1} + \cdots + z^{-(M-1)} = \frac{1 - z^{-M}}{1 - z^{-1}}, \tag{12.89}$$

$$G_0^{\mathrm{f}}(\theta) = D(\theta, M) e^{-j0.5\theta(M-1)}. \tag{12.90}$$

Then we see from (12.87) and (12.89) that

$$G_m^{\mathrm{z}}(z) = G_0^{\mathrm{z}}(zW_M^{-m}). \tag{12.91}$$

Therefore, all M filters in the bank are obtained from the single prototype filter $G_0^{\mathrm{z}}(z)$ through the relationship (12.91). Similarly,

$$G_m^{\mathrm{f}}(\theta) = G_0^{\mathrm{f}}(\theta - 2\pi m/M), \tag{12.92}$$

so the frequency responses of the M filters are obtained from that of $G_0(\theta)$ by right shifts of $2\pi m/M$ in the frequency domain.

We summarize the properties of the DFT filter bank shown in Figure 12.31(a) as follows:

1. All filters in the bank have length M, equal to the size of the bank (which is also equal to the size of the DFT).
2. All filters are obtained from a single prototype filter $G_0^{\mathrm{z}}(z)$ by the operation (12.91), which represents shifting in the frequency domain. The shifting is to the right because we used conjugate DFT in (12.85). If we were to use direct DFT, $G_m^{\mathrm{f}}(\theta)$ would be related to $G_0^{\mathrm{f}}(\theta)$ by shifting to the left.
3. The frequency response of the filter $G_0^{\mathrm{z}}(z)$ has the shape of a Dirichlet kernel. It is therefore low pass, but its stop-band attenuation is only 13.5 dB and its pass-band response is mediocre. It must therefore be considered as an unsatisfactory filter for most applications.
4. The filters $G_m^{\mathrm{z}}(z)$, $m \geq 1$ are band pass, having the same pass-band and stop-band properties as $G_0^{\mathrm{z}}(z)$. Note that their coefficients are complex in general.
5. The entire filter bank can be implemented with $M - 1$ delays, which is the number of delays needed for a single filter.

Figure 12.31(b) shows how the filter bank outputs can be processed to get back the original signal $x[n]$. In the absence of subband processing, $v_m[n] = u_m[n]$ for all n and m. We first pass the vector

$$\{v_0[n], v_1[n], \ldots, v_{M-1}[n]\}$$

through a direct DFT matrix F_M. By the properties of the DFT, this yields the vector

$$\{Mx[n], Mx[n-1], \ldots, Mx[n-M+1]\}.$$

Observing how the chain of delays and adders is arranged in Figure 12.31(b), we conclude that the output $y[n]$ is equal to $M^2x[n-M+1]$. Therefore, the uniform DFT filter bank has the perfect reconstruction property: Its output is equal to the input up to a constant delay and a constant scale factor.

The uniform DFT filter bank shown in Figure 12.31 is not decimated. All signals have the same rate as the input signal $x[n]$. Further examination of this figure reveals that it is highly redundant. We will not lose any information if we decimate the delay outputs by M before feeding them to the matrix $\bar{F}_M$. Then, at time n, the input vector to this matrix will be

$$\{x[nM], x[nM-1], \dots, x[nM-M+1]\}.$$

Consecutive vectors are now made of consecutive nonoverlapping segments of length M of the input signal. Now the DFT is performed once every M points, instead of at each time point. The synthesis bank is treated in a similar manner: the DFT output vector is expanded M-fold and passed through a delay chain. Now the output $y[n]$ is equal to $Mx[n-M+1]$, so the perfect reconstruction property is preserved. Figure 12.32 illustrates the analysis and synthesis DFT filter banks in their decimated forms. In this figure we have omitted the subband processing block for simplicity.

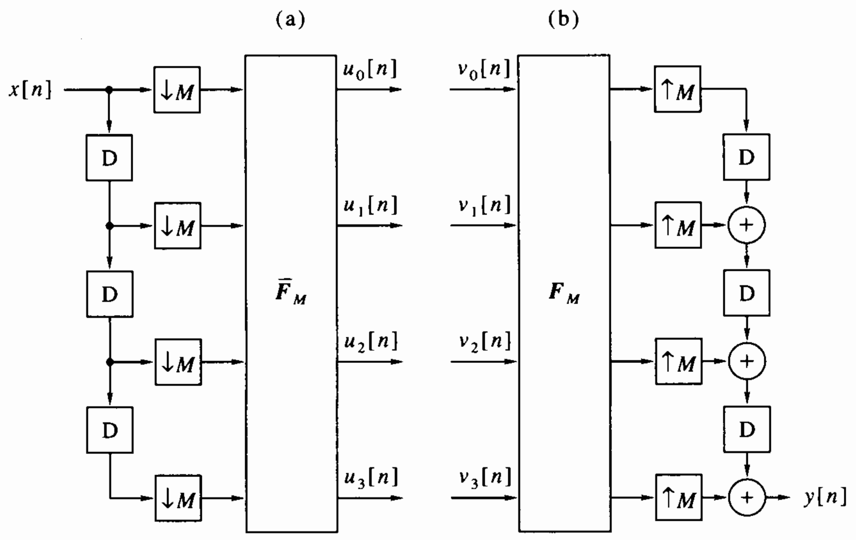

Figure 12.32 A maximally decimated version of the uniform DFT filter bank shown in Figure 12.31: (a) analysis bank; (b) synthesis bank.

The maximally decimated uniform DFT filter bank shown in Figure 12.32 can also be implemented with commutators at the input and at the output instead of the delay chains, as in Figures 12.17 and 12.19.

12.8.2 Windowed DFT Filter Banks

The DFT filter bank we have described suffers from a major drawback, which severely limits its practicality: the inferior frequency response characteristics of the filters $G_m^z(z)$. A simple modification partially corrects this problem: Multiply the input vector to the analysis DFT by a length-M window $\{w[m],\ 0 \le m \le M-1\}$. Then (12.85) changes to

$$u_m[n] = \sum_{i=0}^{M-1} x[n-i]w[i]W_M^{im}, \tag{12.93}$$

so the impulse responses of the filters become

$$g_m[n] = w[n]W_M^{nm}, \quad 0 \le n \le M-1. \tag{12.94}$$

The frequency response of the prototype filter becomes

$$G_0^{\mathrm{f}}(\theta) = \sum_{n=0}^{M-1} w[n]e^{-j\theta n}, \tag{12.95}$$

which is the kernel function of the window. Thus, we can control the stop-band attenuation of the filter by an appropriate choice of window, for example, by using a Kaiser window with α chosen according to the desired stop-band attenuation. The pass-band behavior cannot be controlled in this manner, however, and may remain unsatisfactory for certain applications.

The synthesis filter bank can be modified to achieve perfect reconstruction by multiplying the outputs of the synthesis DFT by the sequence $\{1/w[m],\ 0 \le m \le M-1\}$. This is possible if and only if none of the window coefficients is zero.

12.8.3 A Uniform DFT Filter Bank of Arbitrary Order

The reason for the unsatisfactory characteristics of the uniform DFT filter banks presented above is the limit on the order of the individual filters, that is, the restriction that it must be equal to $M-1$. We now show how this restriction can be removed, thus enabling the filters to meet any desired pass-band and stop-band specifications.

The defining property of a uniform DFT filter bank with an arbitrary prototype filter $G_0^{\mathrm{z}}(z)$ is

$$G_m^{\mathrm{z}}(z) = G_0^{\mathrm{z}}(zW_M^{-m}), \quad 1 \le m \le M-1. \tag{12.96}$$

In other words, all band-pass filters are obtained from the prototype filter by shifting the frequency response along the θ axis by an integer multiple of $2\pi/M$.

Suppose we have designed $G_0^{\mathrm{z}}(z)$ to meet certain transition bandwidth, pass-band ripple, and stop-band attenuation. Let

$$G_0^{\mathrm{z}}(z) = \sum_{i=0}^{M-1} z^{-i}P_i^{\mathrm{z}}(z^M) \tag{12.97}$$

be the polyphase decomposition of $G_0^{\mathrm{z}}(z)$. Then $G_m^{\mathrm{z}}(z)$ is given by

$$G_m^{\mathrm{z}}(z) = G_0^{\mathrm{z}}(zW_M^{-m}) = \sum_{i=0}^{M-1} (zW_M^{-m})^{-i}P_i^{\mathrm{z}}(z^M W_M^{-mM}) = \sum_{i=0}^{M-1} W_M^{im}z^{-i}P_i^{\mathrm{z}}(z^M). \tag{12.98}$$

Let $x[n]$ be the input sequence and $u_m[n]$ the output of the mth filter. Then

$$U_m^{\mathrm{z}}(z) = \sum_{i=0}^{M-1} W_M^{im}P_i^{\mathrm{z}}(z^M)z^{-i}X^{\mathrm{z}}(z). \tag{12.99}$$

By the decimation identity, we can write the z-transform of the decimated sequence $\{u_m\}_{(\downarrow M)}[n]$ as

$$\{U_m^{\mathrm{z}}\}_{(\downarrow M)}(z) = \sum_{i=0}^{M-1} W_M^{im}P_i^{\mathrm{z}}(z)\{z^{-i}X^{\mathrm{z}}(z)\}_{(\downarrow M)}. \tag{12.100}$$

Figure 12.33 shows an implementation of (12.100). As we see, the sequence of operations is as follows:

1. Delay and decimate the input signal $x[n]$ (a commutator can be used for this).
2. Pass each of the decimated sequences through the appropriate polyphase component of $G_0^{\mathrm{z}}(z)$.
3. Perform M-point conjugate DFT on the output vector of the polyphase filters at a rate $1/M$ of the input signal rate.

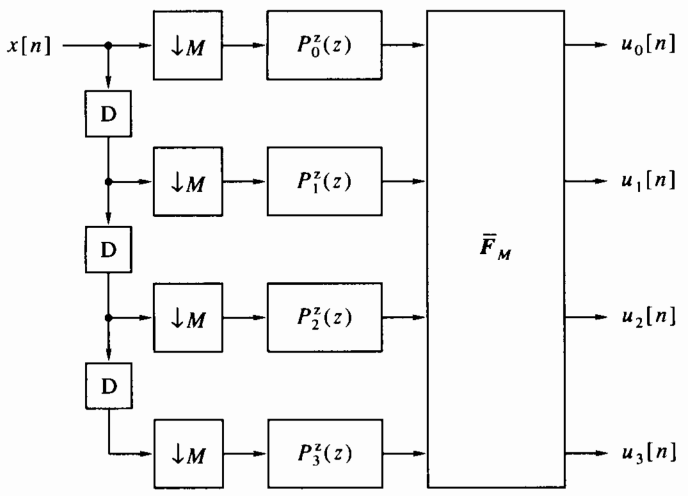

Figure 12.33 A maximally decimated uniform DFT analysis filter bank (shown for $M = 4$).

The structure in Figure 12.33 is called a *maximally decimated uniform DFT analysis filter bank.* The DFT filter bank shown in Figure 12.32(a) is a special case, where all the polyphase components are $P_m^z(z) = 1$.

The filter bank shown in Figure 12.33 can be implemented efficiently. Denote the order of the prototype filter by N. Each polyphase filter performs approximately N/M complex operations every M samples of the input signal. Together they perform N/M complex operations per sample of the input signal. Then, about $0.5M \log_2 M$ complex operations are needed for the DFT every M samples of the input signal, or $0.5 \log_2 M$ per sample. The total is about $N/M + 0.5 \log_2 M$ complex operations per sample of the input signal. At this cost we get M filtering-decimation operations, each of order N. Furthermore, we have the freedom of choosing any prototype filter according to the given specifications.

The procedure `udftanal` in Program 12.5 gives a MATLAB implementation of a maximally decimated uniform DFT analysis filter bank. It is similar to `ppdec`, except that the filtered decimated signals are not combined, but passed as a vector to the IFFT routine. Note also that the MATLAB function `ifft` operates on each column of the matrix `u` individually, and that it gives an additional scale factor $1/M$.

Figure 12.34 shows a maximally decimated uniform DFT synthesis filter bank. The filters $Q_m^z(z)$ are the polyphase components of the prototype synthesis filter $H_0^z(z)$, indexed in reversed order, according to the convention used for expansion [cf. (12.38)]. The sequence of operations carried out by the synthesis bank is as follows:

1. Perform M-point DFT on the input vector.
2. Pass each of the DFT outputs through the appropriate polyphase component.
3. Expand, delay, and sum the polyphase filters' outputs (a commutator can be used for this).

The procedure `udftsynt` in Program 12.6 gives a MATLAB implementation of a maximally decimated uniform DFT synthesis filter bank.

We recall that the simple uniform DFT filter bank shown in Figure 12.32 has the perfect reconstruction property. This property is not shared by a general uniform DFT filter bank. To see the source of the problem, assume we connect the filter banks in Figures 12.33 and 12.34 in tandem, so $v_m[n] = u_m[n]$. Then the conjugate DFT and the

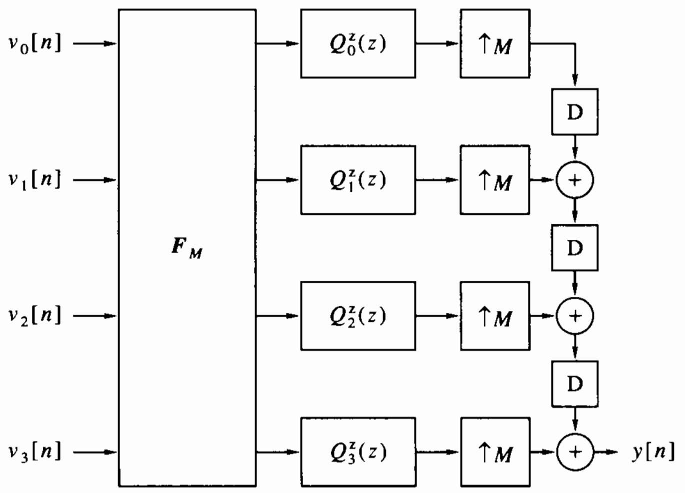

Figure 12.34 A maximally decimated uniform DFT synthesis filter bank (shown for $M = 4$).

DFT cancel each other, except for a scale factor M. The output $Y^z(z)$ can be expressed in terms of the input $X^z(z)$ as follows:

$$Y^z(z) = M \sum_{i=0}^{M-1} \{\{z^{-i}X^z(z)\}_{(\downarrow M)} P_i^z(z) Q_i^z(z)\}_{(\uparrow M)} z^{-(M-1-i)}. \tag{12.101}$$

Using (12.12) for the operation $(\downarrow M)$, we get

$$Y^z(z) = \sum_{i=0}^{M-1} \left\{ \left[\sum_{m=0}^{M-1} (z^{1/M}W_M^{-m})^{-i} X^z(z^{1/M}W_M^{-m}) \right] P_i^z(z) Q_i^z(z) \right\}_{(\uparrow M)} z^{-(M-1-i)}. \tag{12.102}$$

Using (12.15) for the operation $(\uparrow M)$, we get

$$\begin{aligned} Y^z(z) &= \sum_{i=0}^{M-1} \sum_{m=0}^{M-1} (zW_M^{-m})^{-i} X^z(zW_M^{-m}) P_i^z(z^M) Q_i^z(z^M) z^{-(M-1-i)} \\ &= z^{-(M-1)} \sum_{m=0}^{M-1} X^z(zW_M^{-m}) \sum_{i=0}^{M-1} W_M^{mi} P_i^z(z^M) Q_i^z(z^M). \end{aligned} \tag{12.103}$$

Hence we arrive at the following conclusion: A maximally decimated uniform DFT analysis-synthesis filter bank has the perfect reconstruction property if and only if the polyphase components of the prototype filters satisfy

$$\frac{1}{M} \sum_{i=0}^{M-1} W_M^{mi} P_i^z(z^M) Q_i^z(z^M) = cz^{-l}\delta[m] = \begin{cases} cz^{-l}, & m = 0, \\ 0, & m \neq 0, \end{cases} \tag{12.104}$$

for a nonzero constant c and an integer delay l. The left side of (12.104) is the inverse DFT of the sequence $\{P_i^z(z^M)Q_i^z(z^M),\ 0 \le i \le M-1\}$. Therefore, (12.104) holds if and only if

$$P_i^z(z^M) Q_i^z(z^M) = cz^{-l}, \quad 0 \le i \le M-1. \tag{12.105}$$

This condition has been derived by Portnoff [1980].

The simple uniform DFT filter bank has the perfect reconstruction property, since

$$P_i^z(z) = Q_i^z(z) = 1, \quad 0 \le i \le M-1.$$

Similarly, the windowed uniform DFT filter bank has the perfect reconstruction property, since

$$P_i^z(z) = w[i], \; Q_i^z(z) = 1/w[i], \quad 0 \le i \le M - 1.$$

Nontrivial solutions to (12.105), in which the prototype filters $G_0^z(z)$, $H_0^z(z)$ are FIR of order larger than M, have been found by Farkash and Raz, but we shall not discuss them here. Nontrivial solutions also exist if we allow IIR filters, but these are not of practical interest for this application.

We remark, in conclusion, that the lack of perfect reconstruction does not mean that the filter bank is not useful. DFT filter banks have many applications, especially when analysis alone is required, without synthesis.

12.9 Summary and Complements

12.9.1 Summary

This chapter has served as an introduction to the area of multirate signal processing, an area of increasing importance and popularity.

The basic multirate operations are decimation and expansion. Decimation is similar to sampling, in that it aliases the spectrum in general, unless the signal has sufficiently low bandwidth prior to decimation. Expansion does not lead to aliasing, but it generates images in the frequency domain. Therefore, decimation and expansion are almost always accompanied by filtering. A decimation filter acts like an antialiasing filter: It precedes the decimator and its purpose is to limit the signal bandwidth to $\pm\pi/M$, where M is the decimation ratio. An expansion filter acts to interpolate the expanded signal, so it is usually called an interpolation filter. It succeeds the expander and its purpose is to limit the signal bandwidth to $\pm\pi/L$, where L is the expansion ratio.

Sampling-rate conversion is a combined operation of expansion, filtering, and decimation. A sampling-rate converter changes the sampling rate of a signal by a rational factor L/M.

Filters used for decimation, interpolation, and sampling-rate conversion are usually FIR. Decimation and interpolation allow for considerable savings in the number of operations. Polyphase filter structures are particularly convenient for this purpose.

Filter banks are used for either separating a signal to several frequency bands (analysis banks) or for combining signals at different frequency bands to one signal (synthesis banks). To increase computational efficiency and to reduce the data rate, filter banks are usually decimated; that is, the signal rate at each band is made proportional to the bandwidth. A useful property that an analysis-synthesis filter bank pair may have is perfect reconstruction. Perfect reconstruction means that a signal passing through an analysis bank and through a synthesis bank is unchanged, except for a delay and a constant scale factor.

The simplest filter banks have two channels. The simplest of those are quadrature mirror filter banks. QMF banks do not have perfect reconstruction, but they can be made nearly so if designed properly. Conjugate quadrature filter banks, on the other hand, do have perfect reconstruction. These are, however, more difficult to design. Also, the individual filters do not have linear phase.

A filter bank having more than two channels can be built from two-channel filter banks, by connecting them in a tree structure. A tree-structured filter bank can be full or pruned. A special case of a pruned tree is the octave-band filter bank. In such a bank, the bandwidths of the signals occupy successive octave ranges.

When there is a need for a filter bank having more than two channels, a tree-structured, two-channel bank is not necessarily the most efficient. Of the many schemes for M-channel filter banks developed in recent years, we presented only the uniform DFT filter bank. A uniform DFT analysis filter bank uses a single prototype filter, whose polyphase components are implemented separately, and their outputs undergo length-M DFT. The synthesis filter inverts the DFT operation and reconstructs the signal by a prototype interpolation filter implemented by its polyphase components. A uniform DFT filter bank is easy to design and implement, but it does not possess the perfect reconstruction property in general (except for simple cases, whose usefulness is limited).

The first book completely devoted to multirate systems and filter banks is by Crochiere and Rabiner [1983]. Recent books on this subject are by Vaidyanathan [1993], Fliege [1994], and Vetterli and Kovacevic [1995].

12.9.2 Complements

1. [p. 488] Two-channel IIR filter banks are discussed in Vaidyanathan [1993, Sec. 5.3].
2. [p. 492] Spectral factorization by root computation is also possible if $R^{z}(z)$ has zeros on the unit circle. Each such zero must have multiplicity 2. One out of every two such zeros is included in the set of zeros of $G_0^{z}(z)$. However, root-finding programs are sensitive to double zeros. Even worse, a small error in δ_s (when equiripple design is used) may cause the condition $R^{f}(\theta) \geq 0$ to be violated. In this case, the double zero splits into two simple zeros on the unit circle, rendering the zero selection procedure impossible.
3. [p. 495] MPEG is an acronym for Motion Picture Experts Group, a standard adopted by the International Standards Organization (ISO) for compression and coding of motion picture video and audio. The definitive documents for this standard are ISO/IEC JTC1 CD 11172, *Coding of Moving Pictures and Associated Audio for Digital Storage Media up to 1.5 Mbits/s*, 1992, and ISO/IEC JTC1 CD 13818, *Generic Coding of Moving Pictures and Associated Audio*, 1994.

12.10 MATLAB Programs

Program 12.1 Filtering and decimation by polyphase decomposition.

```
function y = ppdec(x,h,M);
% Synopsis: y = ppdec(x,h,M).
% Convolution and M-fold decimation, by polyphase decomposition.
% Input parameters:
% x: the input sequence
% h: the FIR filter coefficients
% M: the decimation factor.
% Output parameters:
% y: the output sequence.

lh = length(h); lp = floor((lh-1)/M) + 1;
p = reshape([reshape(h,1,lh),zeros(1,lp*M-lh)],M,lp);
lx = length(x); ly = floor((lx+lh-2)/M) + 1;
lu = floor((lx+M-2)/M) + 1; % length of decimated sequences
u = [zeros(1,M-1),reshape(x,1,lx),zeros(1,M*lu-lx-M+1)];
u = flipud(reshape(u,M,lu)); % the decimated sequences
y = zeros(1,lu+lp-1);
for m = 1:M, y = y + conv(u(m,:),p(m,:)); end
y = y(1:ly);
```

Program 12.2 Expansion and filtering by polyphase decomposition.

```
function y = ppint(x,h,L);
% Synopsis: y = ppint(x,h,L).
% L-fold expansion and convolution, by polyphase decomposition.
% Input parameters:
% x: the input sequence
% h: the FIR filter coefficients
% L: the expansion factor.
% Output parameters:
% y: the output sequence.

lh = length(h); lq = floor((lh-1)/L) + 1;
q = flipud(reshape([reshape(h,1,lh),zeros(1,lq*L-lh)],L,lq));
lx = length(x); ly = lx*L+lh-1;
lv =lx + lq; % length of interpolated sequences
v = zeros(L,lv);
for l = 1:L, v(l,1:lv-1) = conv(x,q(l,:)); end
y = reshape(flipud(v),1,L*lv);
y = y(1:ly);
```

Program 12.3 Sampling-rate conversion by polyphase decomposition.

```
function y = ppsrc(x,h,L,M);
% Synopsis:  y = ppsrc(x,h,L,M).
% Sampling-rate conversion by polyphase filters.
% Input parameters:
% x: the input sequence
% h: the conversion filter
% L, M: the interpolation and decimation factors.
% Output parameters:
% y: the output sequence

ML = M*L; lh = length(h); lx = length(x);
ly = floor((L*lx+lh-2)/M)+1; % length of the result
K = floor((lh-1)/ML)+1; % max length of polyphase components
r = zeros(ML,K); % storage for polyphase components
for l = 0:L-1, % build polyphase components
for m = 0:M-1,
   temp = h(rem(l*M+(M-m)*L,ML)+1:ML:lh);
   if (length(temp) > 0),
   r(M*l+m+1,1:length(temp)) = temp; end
end, end
x = [reshape(x,1,lx),zeros(1,M)]; % needed for the 1 delay
lx = lx + M;
lu = floor((lx-1)/M) + 1; % length of the sequences u_m
x = [x, zeros(1,M*lu-lx)];
x = reshape(x,M,lu); % now the rows of x are the u_m
y = zeros(L,K+lu-1);
for l = 0:L-1, % loop on sequences v_l
for m = 0:M-1, % loop on sequences u_m
   if (m <= floor(l*M/L)),
      temp = x(m+1,:);
   else
      temp = [0,x(m+1,1:lu-1)];
   end
   y(l+1,:) = y(l+1,:) + conv(r(M*l+m+1,:),temp);
end, end
y = reshape(y,1,L*(K+lu-1));
y = y(1:ly);
```

Program 12.4 Conjugate quadrature filter design by windowing.

```
function [g0,g1,h0,h1] = cqfw(w);
% Synopsis: [g0,g1,h0,h1] = cqfw(w).
% Designs a Smith-Barnwell CQF filter bank by windowing.
% Input parameters:
% w: the window; must have even length, which will also be
%    the length of the filters.
% Output parameters:
% g0, g1: the analysis filters
% h0, h1: the synthesis filters

N = length(w)-1; % order of the output filters
w = conv(w,w); w = (1/w(N+1))*w;
r = firdes(length(w)-1,[0,0.5*pi,1],w);
rr = roots(r);
g0 = real(poly(rr(find(abs(rr) < 1))));
g0 = reshape(g0/sqrt(2*sum(g0.^2)),1,N+1);
h1 = (-1).^(0:N).*g0; g1 = fliplr(h1);
h0 = 2*fliplr(g0); h1 = 2*h1;
```

Program 12.5 A maximally decimated uniform DFT analysis filter bank, implemented by polyphase filters.

```
function u = udftanal(x,g,M);
% Synopsis: u = udftanal(x,g,M).
% Maximally decimated uniform DFT analysis filter bank.
% Input parameters:
% x: the input sequence
% g: the FIR filter coefficients
% M: the decimation factor.
% Output parameters:
% u: a matrix whose rows are the output sequences.

lg = length(g); lp = floor((lg-1)/M) + 1;
p = reshape([reshape(g,1,lg),zeros(1,lp*M-lg)],M,lp);
lx = length(x); lu = floor((lx+M-2)/M) + 1;
x = [zeros(1,M-1),reshape(x,1,lx),zeros(1,M*lu-lx-M+1)];
x = flipud(reshape(x,M,lu)); % the decimated sequences
u = [];
for m = 1:M, u = [u; conv(x(m,:),p(m,:))]; end
u = ifft(u);
```

Program 12.6 A maximally decimated uniform DFT synthesis filter bank, implemented by polyphase filters.

```
function y = udftsynt(v,h,M);
% Synopsis: y = udftsynt(v,h,M).
% Maximally decimated uniform DFT synthesis filter bank.
% Input parameters:
% v: a matrix whose rows are the input sequences
% h: the FIR filter coefficients
% M: the expansion factor.
% Output parameters:
% y: the output sequence

lh = length(h); lq = floor((lh-1)/M) + 1;
q = flipud(reshape([reshape(h,1,lh),zeros(1,lq*M-lh)],M,lq));
v = fft(v);
y = [];
for m = 1:M, y = [conv(v(m,:),q(m,:)); y]; end
y = y(:).';
```

12.11 Problems

12.1 Prove that both decimation and expansion are linear operations.

12.2 Let $x[n] = \cos(\theta_0 n + \phi_0)$. Express $x_{(\downarrow M)}[n]$ in the time and frequency domains. Discuss possible aliasing of the decimated signal as a function of the parameters θ_0 and M.

12.3 Repeat Problem 12.2 for $x_{(\uparrow L)}[n]$.

12.4 A signal $x[n]$ is decimated by M, and the result is expanded by M to yield a signal $y[n]$. Express the z-transform of $y[n]$ in terms of the z-transform of $x[n]$.

12.5 The z-transform of a signal obtained by M-fold decimation followed by M-fold expansion contains M components, as you have obtained in Problem 12.4. Express each of these components in the time domain as a function of $x[n]$, and interpret the result. Discuss the special case $M = 2$. The last $M - 1$ terms in the sum are called the *alias components* of $x[n]$.

12.6 We wish to perform linear interpolation of a signal $x[n]$ by a factor L. Linear interpolation is defined by

$$y[nL + i] = \frac{L-i}{L}x[n] + \frac{i}{L}x[n+1], \quad -\infty < n < \infty,\ 0 \le i \le L-1.$$

Show that linear interpolation can be performed by the standard expansion/filtering scheme (12.19), and find the impulse response of the interpolation filter.

12.7 Consider the moving-average filter introduced in Problem 11.4. Suppose the output is to be decimated by N.

(a) Derive an expression for the Fourier transform of the decimated output as a function of the Fourier transform of the input.

(b) Suggest an efficient implementation of the filter-decimator.

12.8 This problem examines the possibility of representing IIR filters by polyphase components.

(a) Find the polyphase components $P_m^z(z)$ of the first-order IIR filter

$$H^z(z) = \frac{1}{1 - \alpha z^{-1}}.$$

What is the order of each polyphase component?

(b) Use part a for finding the polyphase components $P_m^z(z)$ for the second-order IIR filter (where A and α are complex)

$$H^z(z) = \frac{A}{1 - \alpha z^{-1}} + \frac{\bar{A}}{1 - \bar{\alpha} z^{-1}}.$$

What is the order of each component?

(c) Use parts a and b for finding the polyphase components of any IIR filter whose poles are simple from its partial fraction decomposition (11.11). What is the order of each component?

(d) Suppose we implement IIR filtering followed by decimation as in Figure 12.16, using the polyphase components derived in part c. Would that lead to computational saving? If so, by what factor?

12.9 This problem shows how to implement a *fractional delay* using interpolation and decimation. The ideal frequency response of fractional delay is

$$H^{\mathrm{f}}(\theta) = e^{-j\theta\tau},$$

where τ is noninteger. We assume that τ is rational, that is, $\tau = l/M$, where l, M are coprime positive integers. The idea is to interpolate the input signal $x[n]$ by a factor M, then delay the interpolated signal by l time units, and finally decimate the result by a factor M.

(a) Explain why this method works in principle, why it only approximates the ideal frequency response, and what errors are introduced in the process.

(b) Show that only one polyphase component of the interpolation filter is actually used in this procedure.

12.10 A discrete-time signal $x[n]$ is reconstructed using a causal reconstructor $h(t)$ [cf. (3.31)]:

$$\hat{x}(t) = \sum_{m=-\infty}^{\lfloor t/T \rfloor} x[m]h(t - mT).$$

The reconstructed signal is sampled at interval T/L, where L is an integer, to yield a discrete time signal $y[n] = \hat{x}(nT/L)$. Write an expression for $y[n]$ in terms of $x[n]$, and relate the result to the material in Sections 10.7 and sec:decexplinfil.

12.11 Let $h[n]$ be the impulse response of a linear-phase FIR filter. Are the polyphase components of the filter linear phase in general? What about the special case $M = 2$? Distinguish between even and odd N.

12.12* Suppose we need to design a low-pass digital FIR filter with $\theta_{\mathrm{s}} \ll \pi$. The tolerance parameters are δ_{p}, δ_{s}. The output rate must be equal to the input rate. The standard approach is to design the filter by one of the methods studied in Chapter 9. Figure 12.35 shows a multirate alternative to the standard approach. The signal is low-pass filtered by $H_1^z(z)$, whose specification parameters are $\theta_{\mathrm{p}}, \theta_{\mathrm{s}}, \tilde{\delta}_{\mathrm{p},1}, \tilde{\delta}_{\mathrm{s},1}$. Then the output is decimated by a factor $M = \lfloor \pi/\theta_{\mathrm{s}} \rfloor$. The decimated signal $u[n]$ is immediately expanded by a factor M to yield the signal $v[n]$. Finally, $v[n]$ is low-pass filtered by $H_2^z(z)$, whose specification parameters are $\theta_{\mathrm{p}}, \theta_{\mathrm{s}}, \tilde{\delta}_{\mathrm{p},2}, \tilde{\delta}_{\mathrm{s},2}$.

$$\begin{tikzcd} x[n] \arrow[r] & \boxed{H_1^z(z)} \arrow[r] & \boxed{\downarrow M} \arrow[r, "u[n]"] & \boxed{\uparrow M} \arrow[r, "v[n]"] & \boxed{H_2^z(z)} \arrow[r] & y[n] \end{tikzcd}$$

Figure 12.35 Pertaining to Problem 12.12.

The main idea behind the multirate scheme is that the two filters will require only N_1/M and N_2/M operations per point, respectively (where N_1, N_2 are the respective orders). Therefore, we may hope to achieve a lower operation count than in common unirate filtering. The possibility of achieving this is not guaranteed, however, because each of the two filters must meet tighter tolerances than the unirate filter. In addition, decimation leads to aliasing, so $\tilde{\delta}_{\mathrm{s},1}$ must be set independently of the other parameters to keep the aliasing level below a prescribed limit.

The aim of this problem is to determine the tolerance parameters of the two low-pass filters, to estimate their orders, and to determine the conditions under which the multirate scheme is more efficient than a unirate filter. We assume that equiripple design is used for all filters, so the order formula (9.79) is applicable.

(a) Show that the total pass-band ripple of the output signal $y[n]$ can be up to $\tilde{\delta}_{\mathrm{p},1} + \tilde{\delta}_{\mathrm{p},2}$.

(b) Show that the total stop-band attenuation of the output signal is equal to $\tilde{\delta}_{\mathrm{s},2}$.

(c) Find $\tilde{\delta}_{\mathrm{p},1}$, $\tilde{\delta}_{\mathrm{p},2}$ that will minimize $N_1 + N_2$ subject to the constraint

$$\tilde{\delta}_{\mathrm{p},1} + \tilde{\delta}_{\mathrm{p},2} = \delta_{\mathrm{p}},$$

where δ_{p} is fixed. Hint: Use (9.79) and note that $\tilde{\delta}_{\mathrm{s},1}$, $\tilde{\delta}_{\mathrm{s},2}$, M, and $|\theta_{\mathrm{p}} - \theta_{\mathrm{s}}|$ are fixed.

(d) For $\tilde{\delta}_{\mathrm{p},1}$, $\tilde{\delta}_{\mathrm{p},2}$ found in part c, find a condition on M as a function of δ_{p}, $\delta_{\mathrm{s},1}$, $\delta_{\mathrm{s},2}$ such that

$$\frac{N_1 + N_2}{M} < N,$$

where N is the order of a unirate filter that meets the specifications. This condition can be used for deciding when multirate low-pass filtering is preferable to unirate filtering.

(e) Consider a low-pass filter required to meet the following specifications:

$$\theta_{\mathrm{p}} = 0.02\pi,\ \theta_{\mathrm{s}} = 0.04\pi,\ \delta_{\mathrm{p}} = 0.05,\ \delta_{\mathrm{s}} = 0.001.$$

Assume that $\tilde{\delta}_{\mathrm{s},1} = 0.001$. Compute the number of operations per data point in unirate filtering and in multirate filtering.

12.13* Repeat Example 12.6 for $M_1 = 4$, $M_2 = 3$. Compare the efficiency of this scheme to the one presented in Example 12.6.

12.14* Design a three-stage up-sampling system for compact disc, as described in Example 12.5. The input and output signal rates are 44.1 and 352.8 kHz. Each stage should up-sample the signal by 2. Assume that the input signal contains no energy at frequencies higher than 20 kHz. The required tolerances are $\delta_{\mathrm{p}} = 0.001$, $\delta_{\mathrm{s}} = 0.0001$. As an answer, give the specifications and the orders of the three filters (there is no need to give the coefficients).

12.15* Show that $H_0^z(z)$ and $H_1^z(z)$ defined in (12.59) are low pass and high pass, respectively.

12.16* Consider a QMF filter bank constructed from FIR filters $G_0^z(z)$, $G_1^z(z)$, satisfying (12.63). Show that the only case in which the transfer function $F^z(z)$ can be of the form (12.65) is when $G_0^z(z)$ has two nonzero coefficients at most. Hint: Express $F^z(z)$ as a product of two FIR transfer functions. The conclusion from this result is that QMF FIR filter banks constructed according to the condition (12.63) cannot have the perfect reconstruction property, except when $G_0^z(z)$ has a particularly simple form, which is of little use. Therefore, to preserve both FIR and perfect reconstruction properties, the restriction (12.63) must be removed.

12.17* Let the analysis filter $G_0^z(z)$ of a QMF bank be expressed in terms of its two polyphase components $P_{0,0}^z(z)$, $P_{0,1}^z(z)$.

(a) Express each of the three other filters in the bank, $G_1^z(z)$, $H_0^z(z)$, $H_1^z(z)$, in terms of $P_{0,0}^z(z)$ and $P_{0,1}^z(z)$.

(b) Construct a realization of the complete bank (analysis and synthesis) in terms of the polyphase components, using the polyphase identities.

12.18* A *two-channel time division multiplexer* (TDM) is a device that accepts two signals $x_0[n]$, $x_1[n]$ and outputs the signal

$$y[n] = \begin{cases} x_0[m], & n = 2m, \\ x_1[m], & n = 2m+1. \end{cases}$$

(a) Show how to implement a two-channel TDM using two expanders and a delay element.

(b) Show how to split the signal $y[n]$ back into its components using two decimators and a delay element.

12.19* A *two-channel frequency division multiplexer* (FDM) is a device that accepts two signals $x_0[n]$, $x_1[n]$ and outputs a signal $y[n]$ such that $y[n]$ has twice the rate (hence twice the bandwidth) of each of the $x_i[n]$, and the spectrum of $y[n]$ in the band $[0, 0.5\pi]$ replicates the full spectrum of $x_0[n]$, whereas the spectrum of $y[n]$ in the band $[0.5\pi, \pi]$ replicates the full spectrum of $x_1[n]$.

(a) Show how to implement a two-channel FDM using two expanders and two filters (assume that the filters are ideal).

(b) Show how to split the signal $y[n]$ back into its components using two decimators and two filters (assume that the filters are ideal).

12.20* The scheme shown in Figure 12.36 is called a *two-channel transmultiplexer.*

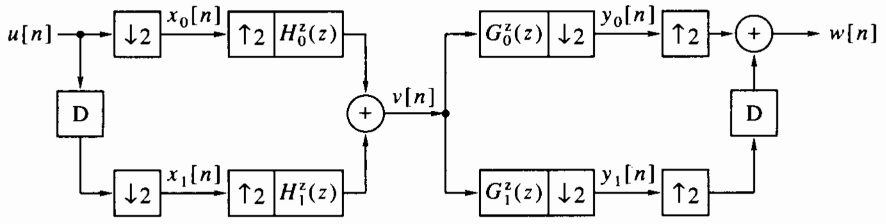

Figure 12.36 A two-channel transmultiplexer (pertaining to Problem 12.20).

(a) Explain what a transmultiplexer performs. Hint: Interpret the relationships between $\{x_0[n], x_1[n]\}$ and $u[n]$, between $v[n]$ and $\{x_0[n], x_1[n]\}$, between $\{y_0[n], y_1[n]\}$ and $w[n]$, and between $w[n]$ and $\{y_0[n], y_1[n]\}$. Use your solutions to Problems 12.18 and 12.19.

(b) Derive the z-transform relationships between the pair $\{Y_0^z(z^2), Y_1^z(z^2)\}$ and the pair $\{X_0^z(z^2), X_1^z(z^2)\}$.

(c) Let the synthesis filters be related to the analysis filters according to

$$H_0^z(z) = 2z^{-1}G_1^z(-z), \quad H_1^z(z) = -2z^{-1}G_0^z(-z)$$

[this is the same as (12.59) except for the additional delay]. Show that this makes $Y_0^z(z^2)$ independent of $X_1^z(z^2)$ and $Y_1^z(z^2)$ independent of $X_0^z(z^2)$. Such a dependence is called *cross-talk.* The aforementioned choice of synthesis filters eliminates cross-talk, in the same way that aliasing is eliminated in a two-channel filter bank.

(d) Show that taking $G_0^z(z)$ and $G_1^z(z)$ to be conjugate quadrature filters leads to perfect reconstruction from $X_0^z(z^2)$ to $Y_0^z(z^2)$ and from $X_1^z(z^2)$ to $Y_1^z(z^2)$.

12.21* Derive (12.72) from (12.71).

12.22* Refer to the design method of nonnegative half-band filters (12.80), but with $\delta_s' = \mu\delta_s$ instead of δ_s. Express the pass-band and stop-band tolerances $\widetilde{\delta_p}$, $\widetilde{\delta_s}$ of $R^z(z)$ in terms of δ_p, δ_s, and μ. Then invert these relationships and find δ_p, δ_s necessary to obtain prescribed values of $\widetilde{\delta_p}$, $\widetilde{\delta_s}$ (assuming μ is fixed).

12.23* Let each of the filters in an L-level, two-channel, tree-structured analysis filter bank have order N. Estimate the total number of operations per input data point.

12.24* Let each of the filters in an L-octave analysis filter bank (such as the one shown in Figure 12.30) have order N. Estimate the total number of operations per input data point.

12.25* Write a MATLAB program that implements an octave-band analysis filter bank. The inputs to the program are the two analysis filters (assumed to be FIR), the number of octaves L, and the input signal. The program outputs are the $L + 1$ output signals.