


14

Subband Coding

14.1 Overview

n this chapter we present the second of three approaches to compression in which the source output is decomposed into constituent parts. Each constituent part is encoded using one or more of the methods that have been described previously. The approach described in this chapter, known as subband coding, relies on separating the source output into different bands of frequencies using digital filters. We provide a general description of the subband coding system and, for those readers with some knowledge of Z-transforms, a more mathematical analysis of the system. The sections containing the mathematical analysis are not essential to understanding the rest of the chapter and are marked with a *. If you are not interested in the mathematical analysis, you should skip these sections. This is followed by a description of a popular approach to bit allocation. We conclude the chapter with applications to audio and image compression.

14.2 Introduction

In previous chapters we looked at a number of different compression schemes. Each of these schemes is most efficient when the data have certain characteristics. A vector quantization scheme is most effective if blocks of the source output show a high degree of clustering. A differential encoding scheme is most effective when the sample-to-sample difference is small. If the source output is truly random, it is best to use scalar quantization or lattice vector quantization. Thus, if a source exhibited certain well-defined characteristics, we could choose a compression scheme most suited to that characteristic. Unfortunately, most source outputs exhibit a combination of characteristics, which makes it difficult to select a compression scheme exactly suited to the source output.

In the last chapter we looked at techniques for decomposing the source output into different frequency bands using block transforms. The transform coefficients had differing statistics and differing perceptual importance. We made use of these differences in allocating bits for encoding the different coefficients. This variable bit allocation resulted in a decrease in the average number of bits required to encode the source output. One of the drawbacks of transform coding is the artificial division of the source output into blocks, which results in the generation of coding artifacts at the block edges, or blocking. One approach to avoiding this blocking is the lapped orthogonal transform (LOT) [192]. In this chapter we look at a popular approach to decomposing the image into different frequency bands without the imposition of an arbitrary block structure. After the input has been decomposed into its constituents, we can use the coding technique best suited to each constituent to improve compression performance. Furthermore, each component of the source output may have different perceptual characteristics. For example, quantization error that is perceptually objectionable in one component may be acceptable in a different component of the source output. Therefore, a coarser quantizer that uses fewer bits can be used to encode the component that is perceptually less important.

Consider the sequence $\{x_n\}$ plotted in Figure 14.1. We can see that, while there is a significant amount of sample-to-sample variation, there is also an underlying long-term trend shown by the dotted line that varies slowly.

One way to extract this trend is to average the sample values in a moving window. The averaging operation smooths out the rapid variations, making the slow variations more evident. Let's pick a window of size two and generate a new sequence $\{y_n\}$ by averaging neighboring values of x_n :

$$y_n = \frac{x_n + x_{n-1}}{2}. \quad (14.1)$$

The consecutive values of y_n will be closer to each other than the consecutive values of x_n . Therefore, the sequence $\{y_n\}$ can be coded more efficiently using differential encoding than we could encode the sequence $\{x_n\}$. However, we want to encode the sequence $\{x_n\}$, not the sequence $\{y_n\}$. Therefore, we follow the encoding of the averaged sequence $\{y_n\}$ by the difference sequence $\{z_n\}$:

$$z_n = x_n - y_n = x_n - \frac{x_n + x_{n-1}}{2} = \frac{x_n - x_{n-1}}{2}. \quad (14.2)$$

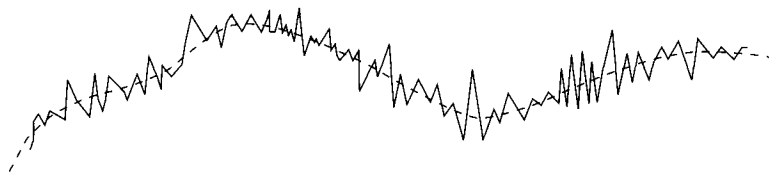


FIGURE 14.1 A rapidly changing source output that contains a long-term component with slow variations.

The sequences $\{y_n\}$ and $\{z_n\}$ can be coded independently of each other. This way we can use the compression schemes that are best suited for each sequence.

Example 14.2.1:

Suppose we want to encode the following sequence of values $\{x_n\}$:

$$10 \quad 14 \quad 10 \quad 12 \quad 14 \quad 8 \quad 14 \quad 12 \quad 10 \quad 8 \quad 10 \quad 12$$

There is a significant amount of sample-to-sample correlation, so we might consider using a DPCM scheme to compress this sequence. In order to get an idea of the requirements on the quantizer in a DPCM scheme, let us take a look at the sample-to-sample differences $x_n - x_{n-1}$:

$$10 \quad 4 \quad -4 \quad 2 \quad 2 \quad -6 \quad 6 \quad -2 \quad -2 \quad -2 \quad 2 \quad 2$$

Ignoring the first value, the dynamic range of the differences is from -6 to 6 . Suppose we want to quantize these values using m bits per sample. This means we could use a quantizer with $M = 2^m$ levels or reconstruction values. If we choose a uniform quantizer, the size of each quantization interval, Δ , is the range of possible input values divided by the total number of reconstruction values. Therefore,

$$\Delta = \frac{12}{M}$$

which would give us a maximum quantization error of $\frac{\Delta}{2}$ or $\frac{6}{M}$.

Now let's generate two new sequences $\{y_n\}$ and $\{z_n\}$ according to (14.1) and (14.2). All three sequences are plotted in Figure 14.2. Notice that given y_n and z_n , we can always recover x_n :

$$x_n = y_n + z_n. \quad (14.3)$$

Let's try to encode each of these sequences. The sequence $\{y_n\}$ is

$$10 \quad 12 \quad 12 \quad 11 \quad 13 \quad 11 \quad 11 \quad 13 \quad 11 \quad 10 \quad 9 \quad 11$$

Notice that the $\{y_n\}$ sequence is “smoother” than the $\{x_n\}$ sequence—the sample-to-sample variation is much smaller. This becomes evident when we look at the sample-to-sample differences:

$$10 \quad 2 \quad 0 \quad -1 \quad 2 \quad -2 \quad 0 \quad 2 \quad -2 \quad -1 \quad -1 \quad 2$$

The difference sequences $\{x_n - x_{n-1}\}$ and $\{y_n - y_{n-1}\}$ are plotted in Figure 14.3. Again, ignoring the first difference, the dynamic range of the differences $y_n - y_{n-1}$ is 4. If we take the dynamic range of these differences as a measure of the range of the quantizer, then for an M -level quantizer, the step size of the quantizer is $\frac{4}{M}$ and the maximum quantization

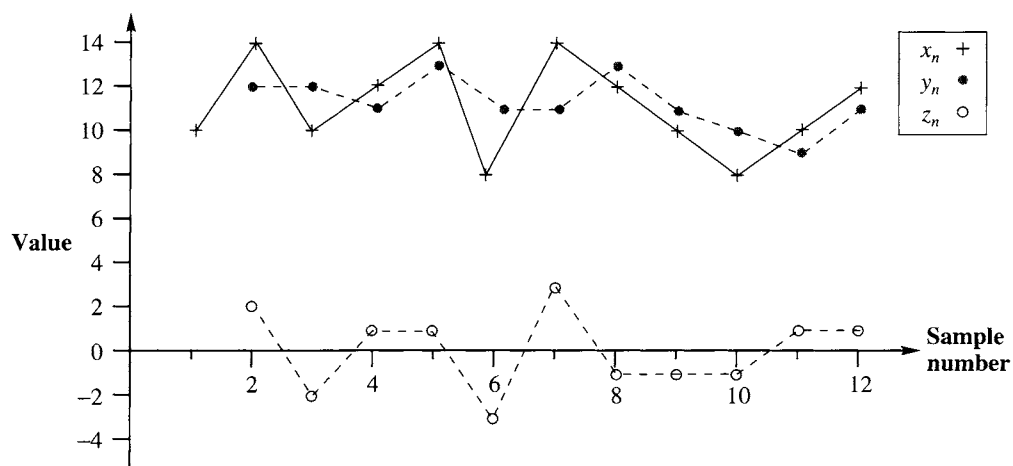


FIGURE 14.2 Original set of samples and the two components.

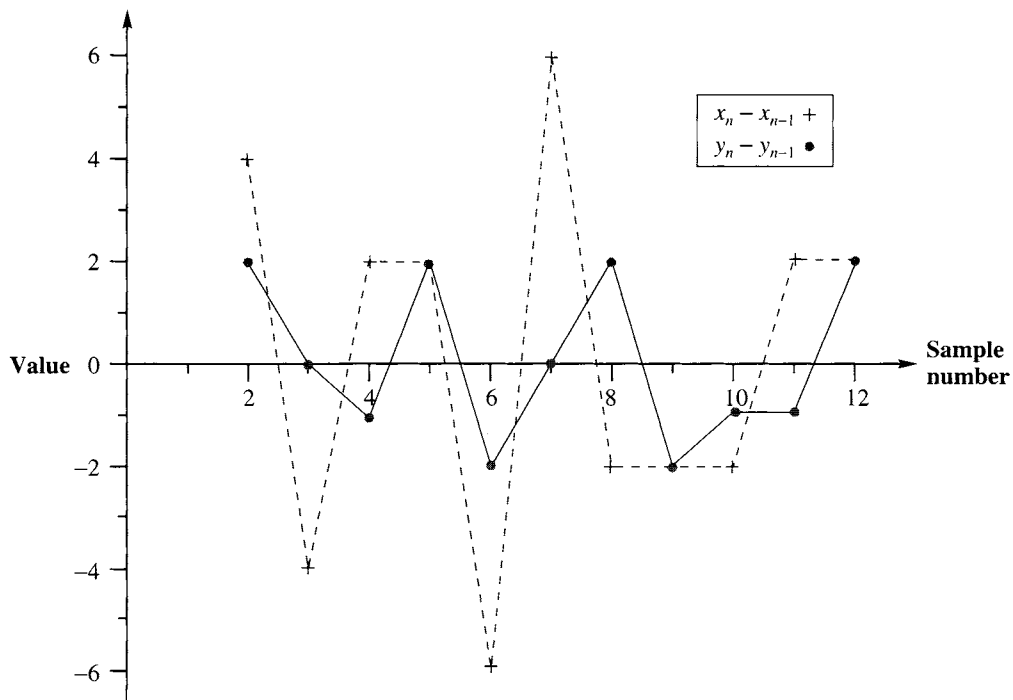


FIGURE 14.3 Difference sequences generated from the original and averaged sequences.

error is $\frac{2}{M}$. This maximum quantization error is one-third the maximum quantization error incurred when the $\{x_n\}$ sequence is quantized using an M -level quantizer. However, in order to reconstruct $\{x_n\}$, we also need to transmit $\{z_n\}$. The $\{z_n\}$ sequence is

$$0 \quad 2 \quad -2 \quad 1 \quad 1 \quad -3 \quad 3 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1$$

The dynamic range for z_n is 6, half the dynamic range of the difference sequence for $\{x_n\}$. (We could have inferred this directly from the definition of z_n .) The sample-to-sample difference varies more than the actual values. Therefore, instead of differentially encoding this sequence, we quantize each individual sample. For an M -level quantizer, the required step size would be $\frac{6}{M}$, giving a maximum quantization error of $\frac{3}{M}$.

For the same number of bits per sample, we can code both y_n and z_n and incur less distortion. At the receiver, we add y_n and z_n to get the original sequence x_n back. The maximum possible quantization error in the reconstructed sequence would be $\frac{5}{M}$, which is less than the maximum error we would incur if we encoded the $\{x_n\}$ sequence directly.

Although we use the same number of bits for each value of y_n and z_n , the number of elements in each of the $\{y_n\}$ and $\{z_n\}$ sequences is the same as the number of elements in the original $\{x_n\}$ sequence. Although we are using the same number of bits per sample, we are transmitting twice as many samples and, in effect, doubling the bit rate.

We can avoid this by sending every other value of y_n and z_n . Let's divide the sequence $\{y_n\}$ into subsequences $\{y_{2n}\}$ and $\{y_{2n-1}\}$ —that is, a subsequence containing only the odd-numbered elements $\{y_1, y_3, \dots\}$, and a subsequence containing only the even-numbered elements $\{y_2, y_4, \dots\}$. Similarly, we divide the $\{z_n\}$ sequence into subsequences $\{z_{2n}\}$ and $\{z_{2n-1}\}$. If we transmit either the even-numbered subsequences or the odd-numbered subsequences, we would transmit only as many elements as in the original sequence. To see how we recover the sequence $\{x_n\}$ from these subsequences, suppose we only transmitted the subsequences $\{y_{2n}\}$ and $\{z_{2n}\}$:

$$y_{2n} = \frac{x_{2n} + x_{2n-1}}{2}$$

$$z_{2n} = \frac{x_{2n} - x_{2n-1}}{2}.$$

To recover the even-numbered elements of the $\{x_n\}$ sequence, we add the two subsequences. In order to obtain the odd-numbered members of the $\{x_n\}$ sequence, we take the difference:

$$y_{2n} + z_{2n} = x_{2n} \tag{14.4}$$

$$y_{2n} - z_{2n} = x_{2n-1}. \tag{14.5}$$

Thus, we can recover the entire original sequence $\{x_n\}$, sending only as many bits as required to transmit the original sequence while incurring less distortion.

Is the last part of the previous statement still true? In our original scheme we proposed to transmit the sequence $\{y_n\}$ by transmitting the differences $y_n - y_{n-1}$. As we now need to transmit the subsequence $\{y_{2n}\}$, we will be transmitting the differences $y_{2n} - y_{2n-2}$ instead. In order for our original statement about reduction in distortion to hold, the dynamic range

of this new sequence of differences should be less than or equal to the dynamic range of the original difference. A quick check of the $\{y_n\}$ shows us that the dynamic range of the new differences is still 4, and our claim of incurring less distortion still holds. ♦

There are several things we can see from this example. First, the number of different values that we transmit is the same, whether we send the original sequence $\{x_n\}$ or the two subsequences $\{y_n\}$ and $\{z_n\}$. Decomposing the $\{x_n\}$ sequence into subsequences did not result in any increase in the number of values that we need to transmit. Second, the two subsequences had distinctly different characteristics, which led to our use of different techniques to encode the different sequences. If we had not split the $\{x_n\}$ sequence, we would have been using essentially the same approach to compress both subsequences. Finally, we could have used the same decomposition approach to decompose the two constituent sequences, which then could be decomposed further still.

While this example was specific to a particular set of values, we can see that decomposing a signal can lead to different ways of looking at the problem of compression. This added flexibility can lead to improved compression performance.

Before we leave this example let us formalize the process of decomposing or *analysis*, and recomposing or *synthesis*. In our example, we decomposed the input sequence $\{x_n\}$ into two subsequences $\{y_n\}$ and $\{z_n\}$ by the operations

$$y_n = \frac{x_n + x_{n-1}}{2} \quad (14.6)$$

$$z_n = \frac{x_n - x_{n-1}}{2}. \quad (14.7)$$

We can implement these operations using discrete time filters. We briefly considered discrete time filters in Chapter 12. We take a slightly more detailed look at filters in the next section.

14.3 Filters

A system that isolates certain frequency components is called a *filter*. The analogy here with mechanical filters such as coffee filters is obvious. A coffee filter or a filter in a water purification system blocks coarse particles and allows only the finer-grained components of the input to pass through. The analogy is not complete, however, because mechanical filters always block the coarser components of the input, while the filters we are discussing can selectively let through or block any range of frequencies. Filters that only let through components below a certain frequency f_0 are called low-pass filters; filters that block all frequency components below a certain value f_0 are called high-pass filters. The frequency f_0 is called the *cutoff frequency*. Filters that let through components that have frequency content above some frequency f_1 but below frequency f_2 are called band-pass filters.

One way to characterize filters is by their *magnitude transfer function*—the ratio of the magnitude of the input and output of the filter as a function of frequency. In Figure 14.4 we show the magnitude transfer function for an ideal low-pass filter and a more realistic low-pass filter, both with a cutoff frequency of f_0 . In the ideal case, all components of the input signal with frequencies below f_0 are unaffected except for a constant amount of

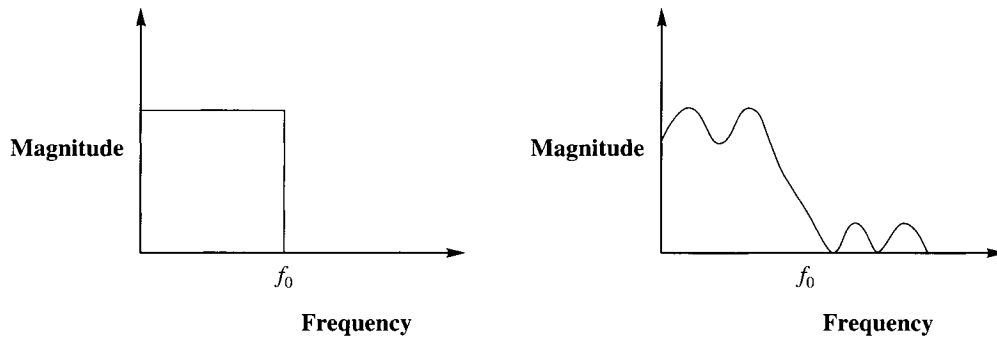


FIGURE 14.4 Ideal and realistic low-pass filter characteristics.

amplification. All frequencies above f_0 are blocked. In other words, the cutoff is sharp. In the case of the more realistic filter, the cutoff is more gradual. Also, the amplification for the components with frequency less than f_0 is not constant, and components with frequencies above f_0 are not totally blocked. This phenomenon is referred to as *ripple* in the passband and stopband.

The filters we will discuss are digital filters, which operate on a sequence of numbers that are usually samples of a continuously varying signal. We have discussed sampling in Chapter 12. For those of you who skipped that chapter, let us take a brief look at the sampling operation.

How often does a signal have to be sampled in order to reconstruct the signal from the samples? If one signal changes more rapidly than another, it is reasonable to assume that we would need to sample the more rapidly varying signal more often than the slowly varying signal in order to achieve an accurate representation. In fact, it can be shown mathematically that if the highest frequency component of a signal is f_0 , then we need to sample the signal at more than $2f_0$ times per second. This result is known as the *Nyquist theorem* or *Nyquist rule* after Harry Nyquist, a famous mathematician from Bell Laboratories. His pioneering work laid the groundwork for much of digital communication. The Nyquist rule can also be extended to signals that only have frequency components between two frequencies f_1 and f_2 . If f_1 and f_2 satisfy certain criteria, then we can show that in order to recover the signal exactly, we need to sample the signal at a rate of at least $2(f_2 - f_1)$ samples per second [123].

What would happen if we violated the Nyquist rule and sampled at less than twice the highest frequency? In Chapter 12 we showed that it would be impossible to recover the original signal from the sample. Components with frequencies higher than half the sampling rate show up at lower frequencies. This process is called *aliasing*. In order to prevent aliasing, most systems that require sampling will contain an “anti-aliasing filter” that restricts the input to the sampler to be less than half the sampling frequency. If the signal contains components at more than half the sampling frequency, we will introduce distortion by filtering out these components. However, the distortion due to aliasing is generally more severe than the distortion we introduce due to filtering.

Digital filtering involves taking a weighted sum of current and past inputs to the filter and, in some cases, the past outputs of the filter. The general form of the input-output relationships of the filter is given by

$$y_n = \sum_{i=0}^N a_i x_{n-i} + \sum_{i=1}^M b_i y_{n-i} \quad (14.8)$$

where the sequence $\{x_n\}$ is the input to the filter, the sequence $\{y_n\}$ is the output from the filter, and the values $\{a_i\}$ and $\{b_i\}$ are called the *filter coefficients*.

If the input sequence is a single 1 followed by all 0s, the output sequence is called the impulse response of the filter. Notice that if the b_i are all 0, then the impulse response will die out after N samples. These filters are called *finite impulse response* (FIR) filters. The number N is sometimes called the number of *taps* in the filter. If any of the b_i have nonzero values, the impulse response can, in theory, continue forever. Filters with nonzero values for some of the b_i are called *infinite impulse response* (IIR) filters.

Example 14.3.1:

Suppose we have a filter with $a_0 = 1.25$ and $a_1 = 0.5$. If the input sequence $\{x_n\}$ is given by

$$x_n = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0, \end{cases} \quad (14.9)$$

then the output is given by

$$\begin{aligned} y_0 &= a_0 x_0 + a_1 x_{-1} = 1.25 \\ y_1 &= a_0 x_1 + a_1 x_0 = 0.5 \\ y_n &= 0 \quad n < 0 \text{ or } n > 1. \end{aligned}$$

This output is called the impulse response of the filter. The impulse response sequence is usually represented by $\{h_n\}$. Therefore, for this filter we would say that

$$h_n = \begin{cases} 1.25 & n = 0 \\ 0.5 & n = 1 \\ 0 & \text{otherwise.} \end{cases} \quad (14.10)$$

Notice that if we know the impulse response we also know the values of a_i . Knowledge of the impulse response completely specifies the filter. Furthermore, because the impulse response goes to zero after a finite number of samples (two in this case), the filter is an FIR filter.

The filters we used in Example 14.2.1 are both two-tap FIR filters with impulse responses

$$h_n = \begin{cases} \frac{1}{2} & n = 0 \\ \frac{1}{2} & n = 1 \\ 0 & \text{otherwise} \end{cases} \quad (14.11)$$

for the “averaging” or low-pass filter, and

$$h_n = \begin{cases} \frac{1}{2} & n = 0 \\ -\frac{1}{2} & n = 1 \\ 0 & \text{otherwise} \end{cases} \quad (14.12)$$

for the “difference” or high-pass filter.

Now let’s consider a different filter with $a_0 = 1$ and $b_1 = 0.9$. For the same input as above, the output is given by

$$y_0 = a_0 x_0 + b_1 y_{-1} = 1(1) + 0.9(0) = 1 \quad (14.13)$$

$$y_1 = a_0 x_1 + b_1 y_0 = 1(0) + 0.9(1) = 0.9 \quad (14.14)$$

$$y_2 = a_0 x_2 + b_1 y_1 = 1(0) + 0.9(0.9) = 0.81 \quad (14.15)$$

$$\vdots \quad \vdots$$

$$y_n = (0.9)^n. \quad (14.16)$$

The impulse response can be written more compactly as

$$h_n = \begin{cases} 0 & n < 0 \\ (0.9)^n & n \geq 0. \end{cases} \quad (14.17)$$

Notice that the impulse response is nonzero for all $n \geq 0$, which makes this an IIR filter. ♦

Although it is not as clear in the IIR case as it was in the FIR case, the impulse response completely specifies the filter. Once we know the impulse response of the filter, we know the relationship between the input and output of the filter. If $\{x_n\}$ and $\{y_n\}$ are the input and output, respectively, of a filter with impulse response $\{h_n\}_{n=0}^M$, then $\{y_n\}$ can be obtained from $\{x_n\}$ and $\{h_n\}$ via the following relationship:

$$y_n = \sum_{k=0}^M h_k x_{n-k}, \quad (14.18)$$

where M is finite for an FIR filter and infinite for an IIR filter. The relationship, shown in (14.18), is known as *convolution* and can be easily obtained through the use of the properties of linearity and shift invariance (see Problem 1).

Because FIR filters are simply weighted averages, they are always stable. When we say a filter is stable we mean that as long as the input is bounded, the output will also be bounded. This is not true of IIR filters. Certain IIR filters can give an unbounded output even when the input is bounded.

Example 14.3.2:

Consider a filter with $a_0 = 1$ and $b_1 = 2$. Suppose the input sequence is a single 1 followed by 0s. Then the output is

$$y_0 = a_0 x_0 + b_1 y_{-1} = 1(1) + 2(0) = 1 \quad (14.19)$$

$$y_1 = a_0 x_0 + b_1 y_0 = 1(0) + 2(1) = 2 \quad (14.20)$$

$$y_2 = a_0 x_1 + b_1 y_1 = 1(0) + 2(2) = 4 \quad (14.21)$$

$$\vdots \quad \vdots$$

$$y_n = 2^n. \quad (14.22)$$

Even though the input contained a single 1, the output at time $n = 30$ is 2^{30} , or more than a billion! ♦

Although IIR filters can become unstable, they can also provide better performance, in terms of sharper cutoffs and less ripple in the passband and stopband for a fewer number of coefficients.

The study of design and analysis of digital filters is a fascinating and important subject. We provide some of the details in Sections 14.5–14.8. If you are not interested in these topics, you can take a more utilitarian approach and make use of the literature to select the necessary filters rather than design them. In the following section we briefly describe some of the families of filters used to generate the examples in this chapter. We also provide filter coefficients that you can use for experiment.

14.3.1 Some Filters Used in Subband Coding

The most frequently used filter banks in subband coding consist of a cascade of stages, where each stage consists of a low-pass filter and a high-pass filter, as shown in Figure 14.5. The most popular among these filters are the *quadrature mirror filters* (QMF), which were first proposed by Crosier, Esteban, and Galand [197]. These filters have the property that if the impulse response of the low-pass filter is given by $\{h_n\}$, then the high-pass impulse response is given by $\{(-1)^n h_{N-1-n}\}$. The QMF filters designed by Johnston [198] are widely used in a number of applications. The filter coefficients for 8-, 16-, and 32-tap filters are given in Tables 14.1–14.3. Notice that the filters are symmetric; that is,

$$h_{N-1-n} = h_n \quad n = 0, 1, \dots, \frac{N}{2} - 1. \quad (14.23)$$

As we shall see later, the filters with fewer taps are less efficient in their decomposition than the filters with more taps. However, from Equation (14.18) we can see that the number of taps dictates the number of multiply-add operations necessary to generate the filter outputs. Thus, if we want to obtain more efficient decompositions, we do so by increasing the amount of computation.

Another popular set of filters are the Smith-Barnwell filters [199], some of which are shown in Tables 14.4 and 14.5.

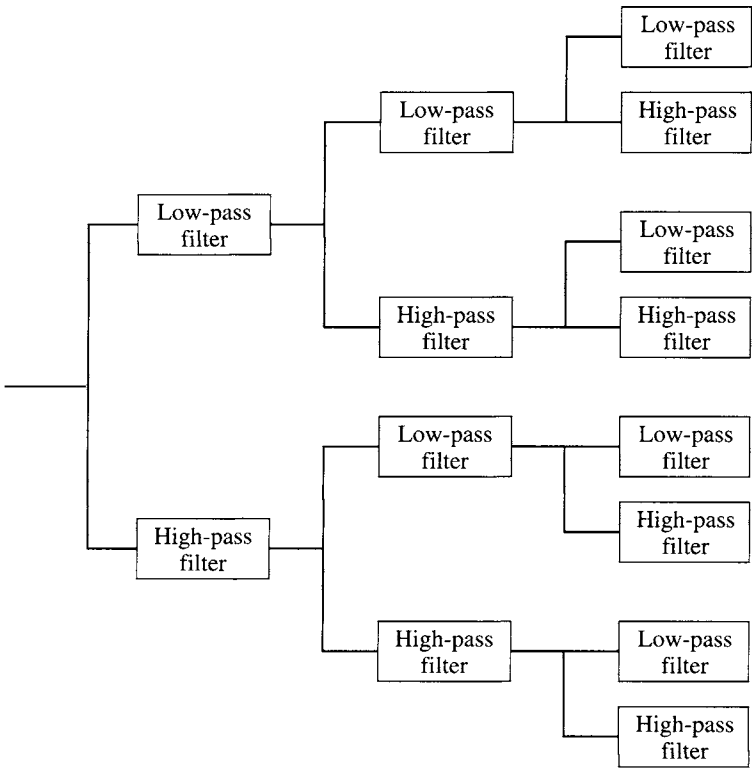


FIGURE 14. 5 An eight-band filter bank.

**TABLE 14 . 1 Coefficients for the 8-tap
Johnston low-pass filter.**

h_0, h_7	0.00938715
h_1, h_6	0.06942827
h_2, h_5	-0.07065183
h_3, h_4	0.48998080

**TABLE 14 . 2 Coefficients for the 16-tap
Johnston low-pass filter.**

h_0, h_{15}	0.002898163
h_1, h_{14}	-0.009972252
h_2, h_{13}	-0.001920936
h_3, h_{12}	0.03596853
h_4, h_{11}	-0.01611869
h_5, h_{10}	-0.09530234
h_6, h_9	0.1067987
h_7, h_8	0.4773469

TABLE 14.3 Coefficients for the 32-tap Johnston low-pass filter.

h_0, h_{31}	0.0022551390
h_1, h_{30}	-0.0039715520
h_2, h_{29}	-0.0019696720
h_3, h_{28}	0.0081819410
h_4, h_{27}	0.00084268330
h_5, h_{26}	-0.014228990
h_6, h_{25}	0.0020694700
h_7, h_{24}	0.022704150
h_8, h_{23}	-0.0079617310
h_9, h_{22}	-0.034964400
h_{10}, h_{21}	0.019472180
h_{11}, h_{20}	0.054812130
h_{12}, h_{19}	-0.044524230
h_{13}, h_{18}	-0.099338590
h_{14}, h_{17}	0.13297250
h_{15}, h_{16}	0.46367410

TABLE 14.4 Coefficients for the eight-tap Smith-Barnwell low-pass filter.

h_0	0.0348975582178515
h_1	-0.01098301946252854
h_2	-0.06286453934951963
h_3	0.223907720892568
h_4	0.556856993531445
h_5	0.357976304997285
h_6	-0.02390027056113145
h_7	-0.07594096379188282

TABLE 14.5 Coefficients for the 16-tap Smith-Barnwell low-pass filter.

h_0	0.02193598203004352
h_1	0.001578616497663704
h_2	-0.06025449102875281
h_3	-0.0118906596205391
h_4	0.137537915636625
h_5	0.05745450056390939
h_6	-0.321670296165893
h_7	-0.528720271545339
h_8	-0.295779674500919
h_9	0.0002043110845170894
h_{10}	0.02906699789446796
h_{11}	-0.03533486088708146
h_{12}	-0.006821045322743358
h_{13}	0.02606678468264118
h_{14}	0.001033363491944126
h_{15}	-0.01435930957477529

These families of filters differ in a number of ways. For example, consider the Johnston eight-tap filter and the Smith-Barnwell eight-tap filter. The magnitude transfer functions for these two filters are plotted in Figure 14.6. Notice that the cutoff for the Smith-Barnwell filter is much sharper than the cutoff for the Johnston filter. This means that the separation provided by the eight-tap Johnston filter is not as good as that provided by the eight-tap Smith-Barnwell filter. We will see the effect of this when we look at image compression later in this chapter.

These filters are examples of some of the more popular filters. Many more filters exist in the literature, and more are being discovered.

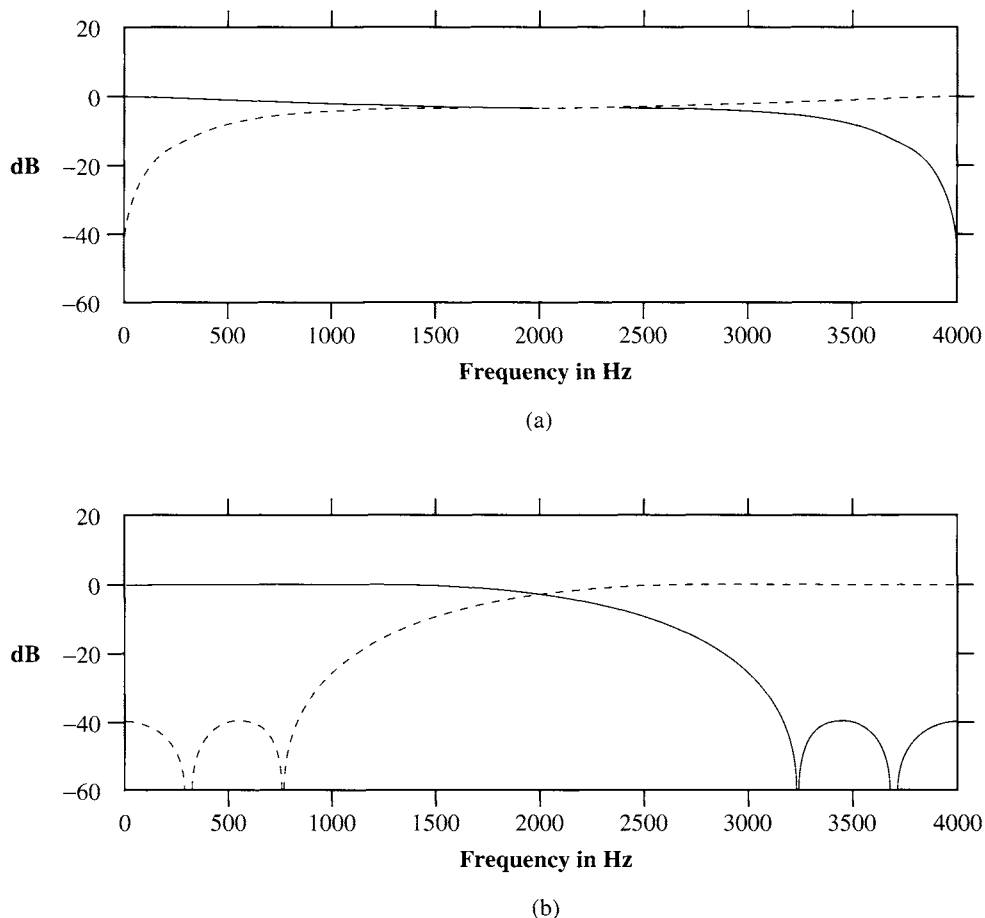


FIGURE 14. 6 Magnitude transfer functions of the (a) eight-tap Johnston and (b) eight-tap Smith-Barnwell filters.

14.4 The Basic Subband Coding Algorithm

The basic subband coding system is shown in Figure 14.7.

14.4.1 Analysis

The source output is passed through a bank of filters, called the analysis filter bank, which covers the range of frequencies that make up the source output. The passbands of the filters can be nonoverlapping or overlapping. Nonoverlapping and overlapping filter banks are shown in Figure 14.8. The outputs of the filters are then subsampled.

The justification for the subsampling is the Nyquist rule and its generalization, which tells us that we only need twice as many samples per second as the range of frequencies. This means that we can reduce the number of samples at the output of the filter because the range of frequencies at the output of the filter is less than the range of frequencies at the input to the filter. This process of reducing the number of samples is called *decimation*,¹ or *downsampling*. The amount of decimation depends on the ratio of the bandwidth of the filter output to the filter input. If the bandwidth at the output of the filter is $1/M$ of the bandwidth at the input to the filter, we would decimate the output by a factor of M by keeping every M th sample. The symbol $M \downarrow$ is used to denote this decimation.

Once the output of the filters has been decimated, the output is encoded using one of several encoding schemes, including ADPCM, PCM, and vector quantization.

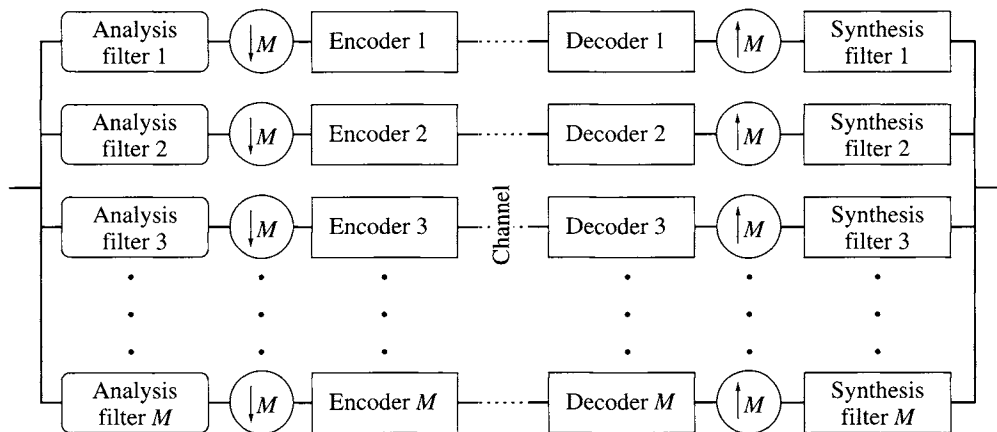


FIGURE 14.7 Block diagram of the subband coding system.

¹ The word *decimation* has a rather bloody origin. During the time of the Roman empire, if a legion broke ranks and ran during battle, its members were lined up and every tenth person was killed. This process was called decimation.

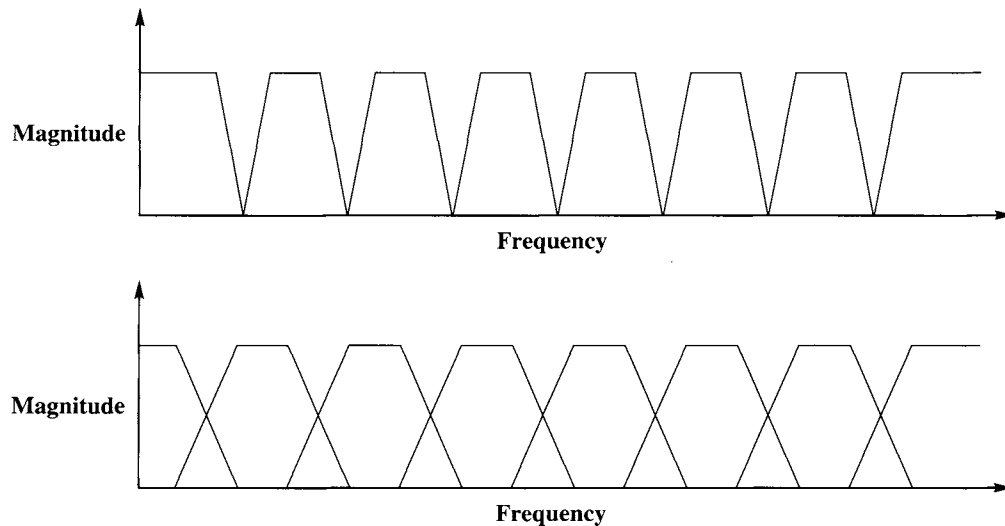


FIGURE 14.8 Nonoverlapping and overlapping filter banks.

14.4.2 Quantization and Coding

Along with the selection of the compression scheme, the allocation of bits between the subbands is an important design parameter. Different subbands contain differing amounts of information. Therefore, we need to allocate the available bits among the subbands according to some measure of the information content. There are a number of different ways we could distribute the available bits. For example, suppose we were decomposing the source output into four bands and we wanted a coding rate of 1 bit per sample. We could accomplish this by using 1 bit per sample for each of the four bands. On the other hand, we could simply discard the output of two of the bands and use 2 bits per sample for the two remaining bands. Or, we could discard the output of three of the four filters and use 4 bits per sample to encode the output of the remaining filter.

This *bit allocation* procedure can have a significant impact on the quality of the final reconstruction, especially when the information content of different bands is very different.

If we use the variance of the output of each filter as a measure of information, and assume that the compression scheme is scalar quantization, we can arrive at several simple bit allocation schemes (see Section 13.5). If we use a slightly more sophisticated model for the outputs of the filters, we can arrive at significantly better bit allocation procedures (see Section 14.9).

14.4.3 Synthesis

The quantized and coded coefficients are used to reconstruct a representation of the original signal at the decoder. First, the encoded samples from each subband are decoded at the receiver. These decoded values are then upsampled by inserting an appropriate number of

0s between samples. Once the number of samples per second has been brought back to the original rate, the upsampled signals are passed through a bank of reconstruction filters. The outputs of the reconstruction filters are added to give the final reconstructed outputs.

We can see that the basic subband system is simple. The three major components of this system are the *analysis and synthesis filters*, the *bit allocation* scheme, and the *encoding* scheme. A substantial amount of research has focused on each of these components. Various filter bank structures have been studied in order to find filters that are simple to implement and provide good separation between the frequency bands. In the next section we briefly look at some of the techniques used in the design of filter banks, but our descriptions are necessarily limited. For a (much) more detailed look, see the excellent book by P.P. Vaidyanathan [200].

The bit allocation procedures have also been extensively studied in the contexts of subband coding, wavelet-based coding, and transform coding. We have already described some bit allocation schemes in Section 13.5, and we describe a different approach in Section 14.9. There are also some bit allocation procedures that have been developed in the context of wavelets, which we describe in the next chapter.

The separation of the source output according to frequency also opens up the possibility for innovative ways to use compression algorithms. The decomposition of the source output in this manner provides inputs for the compression algorithms, each of which has more clearly defined characteristics than the original source output. We can use these characteristics to select separate compression schemes appropriate to each of the different inputs.

Human perception of audio and video inputs is frequency dependent. We can use this fact to design our compression schemes so that the frequency bands that are most important to perception are reconstructed most accurately. Whatever distortion there has to be is introduced in the frequency bands to which humans are least sensitive. We describe some applications to the coding of speech, audio, and images later in this chapter.

Before we proceed to bit allocation procedures and implementations, we provide a more mathematical analysis of the subband coding system. We also look at some approaches to the design of filter banks for subband coding. The analysis relies heavily on the Z-transform concepts introduced in Chapter 12 and will primarily be of interest to readers with an electrical engineering background. The material is not essential to understanding the rest of the chapter; if you are not interested in these details, you should skip these sections and go directly to Section 14.9.

14.5 Design of Filter Banks ★

In this and the following starred section we will take a closer look at the analysis, down-sampling, up-sampling, and synthesis operations. Our approach follows that of [201]. We assume familiarity with the Z-transform concepts of Chapter 12. We begin with some notation. Suppose we have a sequence x_0, x_1, x_2, \dots . We can divide this sequence into two subsequences: x_0, x_2, x_4, \dots and x_1, x_3, x_5, \dots using the scheme shown in Figure 14.9, where z^{-1} corresponds to a delay of one sample and $\downarrow M$ denotes a subsampling by a factor of M . This subsampling process is called *downsampling* or *decimation*.

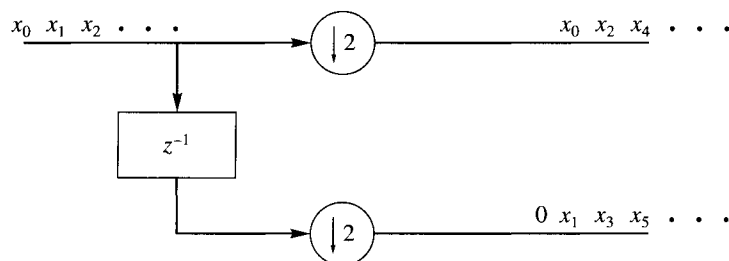


FIGURE 14. 9 Decomposition of an input sequence into its odd and even components.

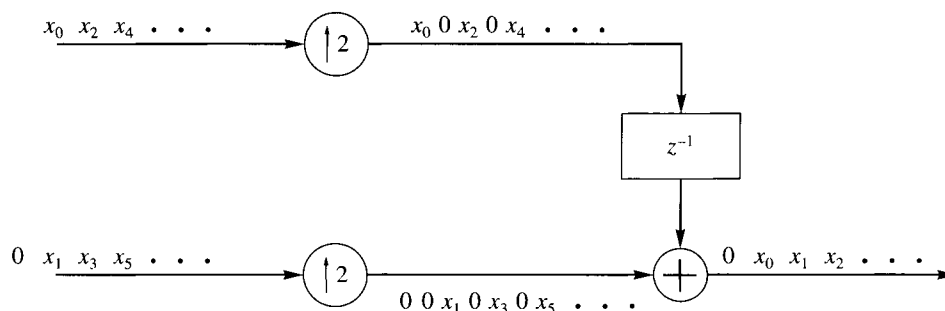


FIGURE 14. 10 Reconstructing the input sequence from its odd and even components.

The original sequence can be recovered from the two downsampled sequences by inserting 0s between consecutive samples of the subsequences, delaying the top branch by one sample and adding the two together. Adding 0s between consecutive samples is called *upsampling* and is denoted by $\uparrow M$. The reconstruction process is shown in Figure 14.10.

While we have decomposed the source output sequence into two subsequences, there is no reason for the statistical and spectral properties of these subsequences to be different. As our objective is to decompose the source output sequences into subsequences with differing characteristics, there is much more yet to be done.

Generalizing this, we obtain the system shown in Figure 14.11. The source output sequence is fed to an ideal low-pass filter and an ideal high-pass filter, each with a bandwidth of $\pi/2$. We assume that the source output sequence had a bandwidth of π . If the original source signal was sampled at the Nyquist rate, as the output of the two filters have bandwidths half that of the original sequence, the filter outputs are actually oversampled by a factor of two. We can, therefore, subsample these signals by a factor of two without any loss of information. The two bands now have different characteristics and can be encoded differently. For the moment let's assume that the encoding is performed in a lossless manner so that the reconstructed sequence exactly matches the source output sequence.

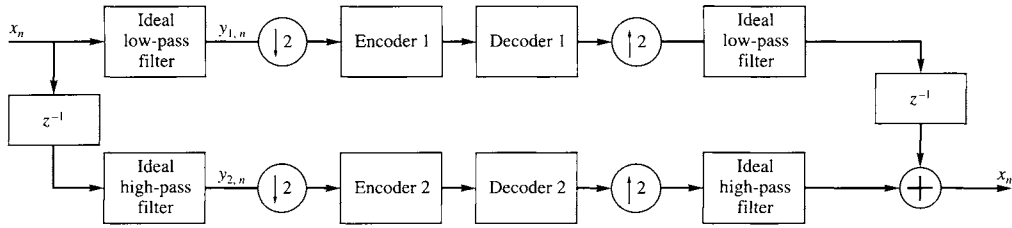


FIGURE 14. 11 Decomposition into two bands using ideal filters.

Let us look at how this system operates in the frequency domain. We begin by looking at the downsampling operation.

14.5.1 Downsampling ★

To see the effects of downsampling, we will obtain the Z-transform of the downsampled sequence in terms of the original source sequence. Because it is easier to understand what is going on if we can visualize the process, we will use the example of a source sequence that has the frequency profile shown in Figure 14.12. For this sequence the output of the ideal filters will have the shape shown in Figure 14.13.

Let's represent the downsampled sequence as $\{w_{i,n}\}$. The Z-transform $W_1(z)$ of the downsampled sequence $w_{1,n}$ is

$$W_1(z) = \sum w_{1,n} z^{-n}. \quad (14.24)$$

The downsampling operation means that

$$w_{1,n} = y_{1,2n}. \quad (14.25)$$

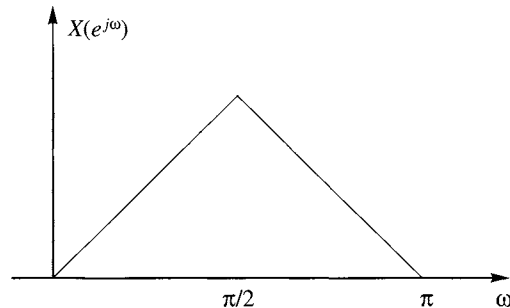


FIGURE 14. 12 Spectrum of the source output.

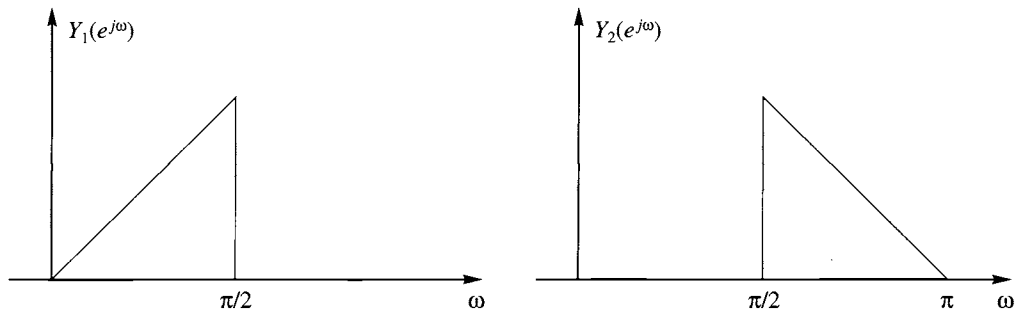


FIGURE 14.13 Spectrum of the outputs of the ideal filters.

In order to find the Z-transform of this sequence, we go through a two-step process. Define the sequence

$$y'_{1,n} = \frac{1}{2}(1 + e^{jn\pi})y_{1,n} \quad (14.26)$$

$$= \begin{cases} y_{1,n} & n \text{ even} \\ 0 & \text{otherwise.} \end{cases} \quad (14.27)$$

We could also have written Equation (14.26) as

$$y'_{1,n} = \frac{1}{2}(1 + (-1)^n)y_{1,n}$$

however, writing the relationship as in Equation (14.26) makes it easier to extend this development to the case where we divide the source output into more than two bands.

The Z-transform of $y'_{1,n}$ is given as

$$Y'_1(z) = \sum_{n=-\infty}^{\infty} \frac{1}{2}(1 + e^{jn\pi})y_{1,n}z^{-n}. \quad (14.28)$$

Assuming all summations converge,

$$Y'_1(z) = \frac{1}{2} \sum_{n=-\infty}^{\infty} y_{1,n}z^{-n} + \frac{1}{2} \sum_{n=-\infty}^{\infty} y_{1,n}(ze^{-j\pi})^{-n} \quad (14.29)$$

$$= \frac{1}{2}Y_1(z) + \frac{1}{2}Y_1(-z) \quad (14.30)$$

where we have used the fact that

$$e^{-j\pi} = \cos(\pi) - j \sin \pi = -1.$$

Noting that

$$w_{1,n} = y'_{1,2n} \quad (14.31)$$

$$W_1(z) = \sum_{n=-\infty}^{\infty} w_{1,n} z^{-n} = \sum_{n=-\infty}^{\infty} y'_{1,2n} z^{-n}. \quad (14.32)$$

Substituting $m = 2n$,

$$W_1(z) = \sum_{m=-\infty}^{\infty} y'_{1,m} z^{-\frac{m}{2}} \quad (14.33)$$

$$= Y_1'(z^{\frac{1}{2}}) \quad (14.34)$$

$$= \frac{1}{2} Y_1(z^{\frac{1}{2}}) + \frac{1}{2} Y_1(-z^{\frac{1}{2}}). \quad (14.35)$$

Why didn't we simply write the Z-transform of $w_{1,n}$ directly in terms of $y_{1,n}$ and use the substitution $m = 2n$? If we had, the equivalent equation to (14.33) would contain the odd indexed terms of $y_{1,n}$, which we know do not appear at the output of the downsampler. In Equation (14.33), we also get the odd indexed terms of $y'_{1,n}$; however, as these terms are all zero (see Equation (14.26)), they do not contribute to the Z-transform.

Substituting $z = e^{j\omega}$ we get

$$W_1(e^{j\omega}) = \frac{1}{2} Y_1(e^{j\frac{\omega}{2}}) + \frac{1}{2} Y_1(-e^{j\frac{\omega}{2}}). \quad (14.36)$$

Plotting this for the $Y_1(e^{j\omega})$ of Figure 14.13, we get the spectral shape shown in Figure 14.14; that is, the spectral shape of the downsampled signal is a stretched version of the spectral shape of the original signal. A similar situation exists for the downsampled signal $w_{2,n}$.

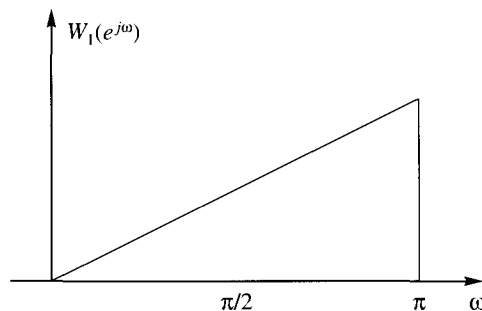


FIGURE 14. 14 Spectrum of the downsampled low-pass filter output.

14.5.2 Upsampling ★

Let's take a look now at what happens after the upsampling. The upsampled sequence $v_{1,n}$ can be written as

$$v_{1,n} = \begin{cases} w_{1,\frac{n}{2}} & n \text{ even} \\ 0 & n \text{ odd.} \end{cases} \quad (14.37)$$

The Z-transform $V_1(z)$ is thus

$$V_1(z) = \sum_{n=-\infty}^{\infty} v_{1,n} z^{-n} \quad (14.38)$$

$$= \sum_{n=-\infty}^{\infty} w_{1,\frac{n}{2}} z^{-n} \quad n \text{ even} \quad (14.39)$$

$$= \sum_{m=-\infty}^{\infty} w_{1,m} z^{-2m} \quad (14.40)$$

$$= W_1(z^2). \quad (14.41)$$

The spectrum is sketched in Figure 14.15. The “stretching” of the sequence in the time domain has led to a compression in the frequency domain. This compression has also resulted in a replication of the spectrum in the $[0, \pi]$ interval. This replication effect is called *imaging*. We remove the images by using an ideal low-pass filter in the top branch and an ideal high-pass filter in the bottom branch.

Because the use of the filters prior to sampling reduces the bandwidth, which in turn allows the downsampling operation to proceed without aliasing, these filters are called *anti-aliasing* filters. Because they decompose the source output into components, they are also called *analysis* filters. The filters after the upsampling operation are used to recombine the original signal; therefore, they are called *synthesis* filters. We can also view these filters as interpolating between nonzero values to recover the signal at the point that we have inserted zeros. Therefore, these filters are also called *interpolation* filters.

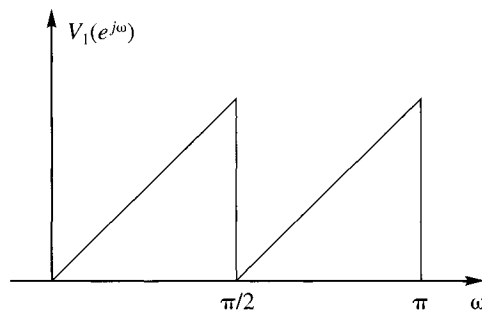


FIGURE 14.15 Spectrum of the upsampled signal.

Although the use of ideal filters would give us perfect reconstruction of the source output, in practice we do not have ideal filters available. When we use more realistic filters in place of the ideal filters, we end up introducing distortion. In the next section we look at this situation and discuss how we can reduce or remove this distortion.

14.6 Perfect Reconstruction Using Two-Channel Filter Banks ★

Suppose we replace the ideal low-pass filter in Figure 14.11 with a more realistic filter with the magnitude response shown in Figure 14.4. The spectrum of the output of the low-pass filter is shown in Figure 14.16. Notice that we now have nonzero values for frequencies above $\frac{\pi}{2}$. If we now subsample by two, we will end up sampling at *less* than twice the highest frequency, or in other words, we will be sampling at below the Nyquist rate. This will result in the introduction of aliasing distortion, which will show up in the reconstruction. A similar situation will occur when we replace the ideal high-pass filter with a realistic high-pass filter.

In order to get perfect reconstruction after synthesis, we need to somehow get rid of the aliasing and imaging effects. Let us look at the conditions we need to impose upon the filters $H_1(z)$, $H_2(z)$, $K_1(z)$, and $K_2(z)$ in order to accomplish this. These conditions are called *perfect reconstruction* (PR) conditions.

Consider Figure 14.17. Let's obtain an expression for $\hat{X}(z)$ in terms of $H_1(z)$, $H_2(z)$, $K_1(z)$, and $K_2(z)$. We start with the reconstruction:

$$\hat{X}(z) = U_1(z) + U_2(z) \quad (14.42)$$

$$= V_1(z)K_1(z) + V_2(z)K_2(z). \quad (14.43)$$

Therefore, we need to find $V_1(z)$ and $V_2(z)$. The sequence $v_{1,n}$ is obtained by upsampling $w_{1,n}$. Therefore, from Equation (14.41),

$$V_1(z) = W_1(z^2). \quad (14.44)$$

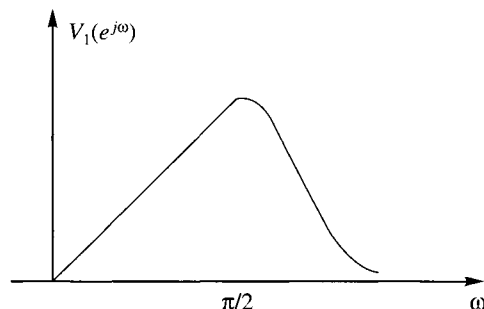


FIGURE 14.16 Output of the low-pass filter.

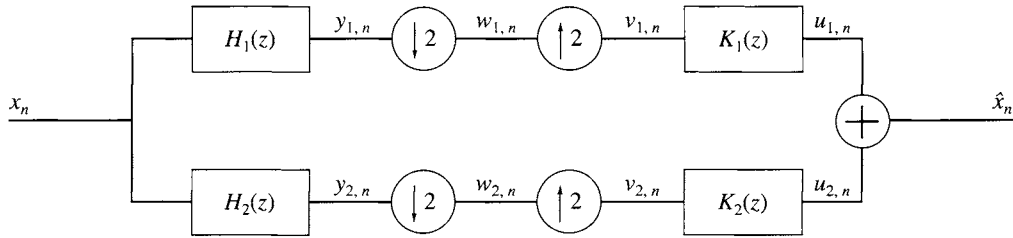


FIGURE 14.17 Two-channel subband decimation and interpolation.

The sequence $w_{1,n}$ is obtained by downsampling $y_{1,n}$,

$$Y_1(z) = X(z)H_1(z).$$

Therefore, from Equation (14.35),

$$W_1(z) = \frac{1}{2} \left[X(z^{\frac{1}{2}})H_1(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})H_1(-z^{\frac{1}{2}}) \right] \quad (14.45)$$

and

$$V_1(z) = \frac{1}{2} [X(z)H_1(z) + X(-z)H_1(-z)]. \quad (14.46)$$

Similarly, we can also show that

$$V_2(z) = \frac{1}{2} [X(z)H_2(z) + X(-z)H_2(-z)]. \quad (14.47)$$

Substituting the expressions for $V_1(z)$ and $V_2(z)$ into Equation (14.43) we obtain

$$\begin{aligned} \hat{X}(z) &= \frac{1}{2} [H_1(z)K_1(z) + H_2(z)K_2(z)] X(z) \\ &\quad + \frac{1}{2} [H_1(-z)K_1(z) + H_2(-z)K_2(z)] X(-z). \end{aligned} \quad (14.48)$$

For perfect reconstruction we would like $\hat{X}(z)$ to be a delayed and perhaps amplitude-scaled version of $X(z)$; that is,

$$\hat{X}(z) = cX(z)z^{-n_0}. \quad (14.49)$$

In order for this to be true, we need to impose conditions on $H_1(z)$, $H_2(z)$, $K_1(z)$, and $K_2(z)$. There are several ways we can do this, with each approach providing a different solution. One approach involves writing Equation (14.48) in matrix form as

$$\hat{X}(z) = \frac{1}{2} \begin{bmatrix} K_1(z) & K_2(z) \end{bmatrix} \begin{bmatrix} H_1(z) & H_1(-z) \\ H_2(z) & H_2(-z) \end{bmatrix} \begin{bmatrix} X(z) \\ X(-z) \end{bmatrix} \quad (14.50)$$

For perfect reconstruction, we need

$$\begin{bmatrix} K_1(z) & K_2(z) \end{bmatrix} \begin{bmatrix} H_1(z) & H_1(-z) \\ H_2(z) & H_2(-z) \end{bmatrix} = \begin{bmatrix} cz^{-n_0} & 0 \end{bmatrix} \quad (14.51)$$

where we have absorbed the factor of $\frac{1}{2}$ into the constant c . This means that the synthesis filters $K_1(z)$ and $K_2(z)$ satisfy

$$\begin{bmatrix} K_1(z) & K_2(z) \end{bmatrix} = \frac{cz^{-n_0}}{\det[\mathcal{H}(z)]} \begin{bmatrix} H_2(-z) & -H_1(-z) \end{bmatrix} \quad (14.52)$$

where

$$\mathcal{H}(z) = \begin{bmatrix} H_1(z) & H_1(-z) \\ H_2(z) & H_2(-z) \end{bmatrix} \quad (14.53)$$

If $H_1(z)$ and/or $H_2(z)$ are IIR filters, the reconstruction filters can become quite complex. Therefore, we would like to have both the analysis and synthesis filters be FIR filters. If we select the analysis filters to be FIR, then in order to guarantee that the synthesis filters are also FIR we need

$$\det[\mathcal{H}(z)] = \gamma z^{-n_1}$$

where γ is a constant. Examining $\det[\mathcal{H}(z)]$

$$\begin{aligned} \det[\mathcal{H}(z)] &= H_1(z)H_2(-z) - H_1(-z)H_2(z) \\ &= P(z) - P(-z) = \gamma z^{-n_1} \end{aligned} \quad (14.54)$$

where $P(z) = H_1(z)H_2(-z)$. If we examine Equation (14.54), we can see that n_1 has to be odd because all terms containing even powers of z in $P(z)$ will be canceled out by the corresponding terms in $P(-z)$. Thus, $P(z)$ can have an arbitrary number of even-indexed coefficients (as they will get canceled out), but there must be only one nonzero coefficient of an odd power of z . By choosing any valid factorization of the form

$$P(z) = P_1(z)P_2(z) \quad (14.55)$$

we can obtain many possible solutions of perfect reconstruction FIR filter banks with

$$H_1(z) = P_1(z) \quad (14.56)$$

and

$$H_2(z) = P_2(-z). \quad (14.57)$$

Although these filters are perfect reconstruction filters, for applications in data compression they suffer from one significant drawback. Because these filters may be of unequal bandwidth, the output of the larger bandwidth filter suffers from severe aliasing. If the output of both bands is available to the receiver, this is not a problem because the aliasing is canceled out in the reconstruction process. However, in many compression applications we discard the subband containing the least amount of energy, which will generally be the output of the filter with the smaller bandwidth. In this case the reconstruction will contain a large amount of aliasing distortion. In order to avoid this problem for compression applications, we generally wish to minimize the amount of aliasing in each subband. A class of filters that is useful in this situation is the *quadrature mirror filters* (QMF). We look at these filters in the next section.

14.6.1 Two-Channel PR Quadrature Mirror Filters ★

Before we introduce the quadrature mirror filters, let's rewrite Equation (14.48) as

$$\hat{X}(z) = T(z)X(z) + S(z)X(-z) \quad (14.58)$$

where

$$T(z) = \frac{1}{2} [H_1(z)K_1(z) + H_2(z)K_2(z)] \quad (14.59)$$

$$S(z) = \frac{1}{2} [H_1(-z)K_1(z) + H_2(-z)K_2(z)]. \quad (14.60)$$

In order for the reconstruction of the input sequence $\{x_n\}$ to be a delayed, and perhaps scaled, version of $\{x_n\}$, we need to get rid of the aliasing term $X(-z)$ and have $T(z)$ be a pure delay. To get rid of the aliasing term, we need

$$S(z) = 0, \quad \forall z.$$

From Equation (14.60), this will happen if

$$K_1(z) = H_2(-z) \quad (14.61)$$

$$K_2(z) = -H_1(-z). \quad (14.62)$$

After removing the aliasing distortion, a delayed version of the input will be available at the output if

$$T(z) = cz^{-n_0} \quad c \text{ is a constant.} \quad (14.63)$$

Replacing z by $e^{j\omega}$, this means that we want

$$|T(e^{j\omega})| = \text{constant} \quad (14.64)$$

$$\arg(T(e^{j\omega})) = K\omega \quad K \text{ constant.} \quad (14.65)$$

The first requirement eliminates amplitude distortion, while the second, the linear phase requirement, is necessary to eliminate phase distortion. If these requirements are satisfied,

$$\hat{x}(n) = cx(n - n_0). \quad (14.66)$$

That is, the reconstructed signal is a delayed version of input signal $x(n)$. However, meeting both requirements simultaneously is not a trivial task.

Consider the problem of designing $T(z)$ to have linear phase. Substituting (14.61) and (14.62) into Equation (14.59), we obtain

$$T(z) = \frac{1}{2} [H_1(z)H_2(-z) - H_1(-z)H_2(z)]. \quad (14.67)$$

Therefore, if we choose $H_1(z)$ and $H_2(z)$ to be linear phase FIR, $T(z)$ will also be a linear phase FIR filter. In the QMF approach, we first select the low-pass filter $H_1(z)$, then define the high-pass filter $H_2(z)$ to be a mirror image of the low-pass filter:

$$H_2(z) = H_1(-z). \quad (14.68)$$

This is referred to as a *mirror* condition and is the original reason for the name of the QMF filters [200]. We can see that this condition will force both filters to have equal bandwidth.

Given the mirror condition and $H_1(z)$, a linear phase FIR filter, we will have linear phase and

$$T(z) = \frac{1}{2}[H_1^2(z) - H_1^2(-z)]. \quad (14.69)$$

It is not clear that $|T(e^{j\omega})|$ is a constant. In fact, we will show in Section 14.8 that a linear phase two-channel FIR QMF bank with the filters chosen as in Equation (14.68) can have PR property if and only if $H_1(z)$ is in the simple two-tap form

$$H_1(z) = h_0 z^{-2k_0} + h_1 z^{-(2k_1+1)}. \quad (14.70)$$

Then, $T(z)$ is given by

$$T(z) = 2h_0 h_1 z^{-(2k_0+2k_1+1)} \quad (14.71)$$

which is of the desired form cz^{-n_0} . However, if we look at the magnitude characteristics of the two filters, we see that they have poor cutoff characteristics. The magnitude of the low-pass filter is given by

$$|H_1(e^{j\omega})|^2 = h_0^2 + h_1^2 + 2h_0 h_1 \cos(2k_0 - 2k_1 - 1)\omega \quad (14.72)$$

and the high-pass filter is given by

$$|H_2(e^{j\omega})|^2 = h_0^2 + h_1^2 - 2h_0 h_1 \cos(2k_0 - 2k_1 - 1)\omega. \quad (14.73)$$

For $h_0 = h_1 = k_0 = k_1 = 1$, the magnitude responses are plotted in Figure 14.18. Notice the poor cutoff characteristics of these two filters.

Thus, for perfect reconstruction with no aliasing and no amplitude or phase distortion, the mirror condition does not seem like such a good idea. However, if we slightly relax these rather strict conditions, we can obtain some very nice designs. For example, instead of attempting to eliminate all phase and amplitude distortion, we could elect to eliminate only the phase distortion and *minimize* the amplitude distortion. We can optimize the coefficients of $H_1(z)$ such that $|T(e^{j\omega})|$ is made as close to a constant as possible, while minimizing the stopband energy of $H_1(z)$ in order to have a good low-pass characteristic. Such an optimization has been suggested by Johnston [198] and Jain and Crochiere [202]. They construct the objective function

$$J = \alpha \int_{\omega_s}^{\pi} |H_1(e^{j\omega})|^2 d\omega + (1 - \alpha) \int_0^{\pi} (1 - |T(e^{j\omega})|^2) d\omega \quad (14.74)$$

which has to be minimized to obtain $H_1(z)$ and $T_1(z)$, where ω_s is the cutoff frequency of the filter.

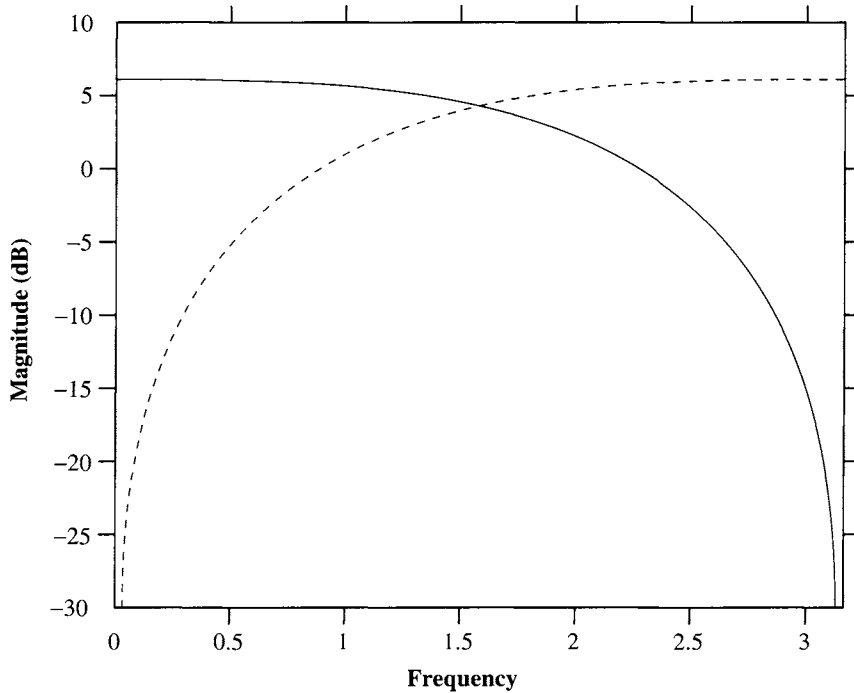


FIGURE 14.18 Magnitude characteristics of the two-tap PR filters.

We can also go the other way and eliminate the amplitude distortion, then attempt to minimize the phase distortion. A review of these approaches can be found in [201, 200].

14.6.2 Power Symmetric FIR Filters ★

Another approach, independently discovered by Smith and Barnwell [199] and Mintzer [203], can be used to design a two-channel filter bank in which aliasing, amplitude distortion, and phase distortion can be completely eliminated. As discussed earlier, choosing

$$\begin{aligned} K_1(z) &= -H_2(-z) \\ K_2(z) &= H_1(-z) \end{aligned} \quad (14.75)$$

eliminates aliasing. This leaves us with

$$T(z) = \frac{1}{2}[H_1(-z)H_2(z) - H_1(z)H_2(-z)].$$

In the approach due to Smith and Barnwell [199] and Mintzer [203], with N an odd integer, we select

$$H_2(z) = z^{-N}H_1(-z^{-1}) \quad (14.76)$$

so that

$$T(z) = \frac{1}{2}z^{-N}[H_1(z)H_1(z^{-1}) + H_1(-z)H_1(-z^{-1})]. \quad (14.77)$$

Therefore, the perfect reconstruction requirement reduces to finding a prototype low-pass filter $H(z) = H_1(z)$ such that

$$Q(z) = H(z)H(z^{-1}) + H(-z)H(-z^{-1}) = \text{constant}. \quad (14.78)$$

Defining

$$R(z) = H(z)H(z^{-1}), \quad (14.79)$$

the perfect reconstruction requirement becomes

$$Q(z) = R(z) + R(-z) = \text{constant}. \quad (14.80)$$

But $R(z)$ is simply the Z-transform of the autocorrelation sequence of $h(n)$. The autocorrelation sequence $\rho(n)$ is given by

$$\rho(n) = \sum_{k=0}^N h_k h_{k+n}. \quad (14.81)$$

The Z-transform of $\rho(n)$ is given by

$$R(z) = \mathcal{Z}[\rho(n)] = \mathcal{Z}\left[\sum_{k=0}^N h_k h_{k+n}\right]. \quad (14.82)$$

We can express the sum $\sum_{k=0}^N h_k h_{k+n}$ as a convolution:

$$h_n \otimes h_{-n} = \sum_{k=0}^N h_k h_{k+n}. \quad (14.83)$$

Using the fact that the Z-transform of a convolution of two sequences is the product of the Z-transforms of the individual sequences, we obtain

$$R(z) = \mathcal{Z}[h_n]\mathcal{Z}[h_{-n}] = H(z)H(z^{-1}). \quad (14.84)$$

Writing out $R(z)$ as the Z-transform of the sequence $\{\rho(n)\}$ we obtain

$$R(z) = \rho(N)z^N + \rho(N-1)z^{N-1} + \cdots + \rho(0) + \cdots + \rho(N-1)z^{-N-1} + \rho(N)z^{-N}. \quad (14.85)$$

Then $R(-z)$ is

$$R(-z) = -\rho(N)z^N + \rho(N-1)z^{N-1} - \cdots + \rho(0) - \cdots + \rho(N-1)z^{-N-1} - \rho(N)z^{-N}. \quad (14.86)$$

Adding $R(z)$ and $R(-z)$, we obtain $Q(z)$ as

$$Q(z) = 2\rho(N-1)z^{N-1} + 2\rho(N-1)z^{N-3} + \cdots + \rho(0) + \cdots + 2\rho(N-1)z^{-N-1}. \quad (14.87)$$

Notice that the terms containing the odd powers of z got canceled out. Thus, for $Q(z)$ to be a constant all we need is that for even values of the lag n (except for $n = 0$), $\rho(n)$ be zero. In other words

$$\rho(2n) = \sum_{k=0}^N h_k h_{k+2n} = 0, \quad n \neq 0. \quad (14.88)$$

Writing this requirement in terms of the impulse response:

$$\sum_{k=0}^N h_k h_{k+2n} = \begin{cases} 0 & n \neq 0 \\ \rho(0) & n = 0. \end{cases} \quad (14.89)$$

If we now normalize the impulse response,

$$\sum_{k=0}^N |h_k|^2 = 1 \quad (14.90)$$

we obtain the perfect reconstruction requirement

$$\sum_{k=0}^N h_k h_{k+2n} = \delta_n. \quad (14.91)$$

In other words, for perfect reconstruction, the impulse response of the prototype filter is orthogonal to the twice-shifted version of itself.

14.7 M-Band QMF Filter Banks ★

We have looked at how we can decompose an input signal into two bands. In many applications it is necessary to divide the input into multiple bands. We can do this by using a recursive two-band splitting as shown in Figure 14.19, or we can obtain banks of filters that directly split the input into multiple bands. Given that we have good filters that provide two-band splitting, it would seem that using a recursive splitting, as shown in Figure 14.19, would be an efficient way of obtaining an M -band split. Unfortunately, even when the spectral characteristics of the filters used for the two-band split are quite good, when we employ them in the tree structure shown in Figure 14.19, the spectral characteristics may not be very good. For example, consider the four-tap filter with filter coefficients shown in Table 14.6. In Figure 14.20 we show what happens to the spectral characteristics when we look at the two-band split (at point **A** in Figure 14.19), the four-band split (at point **B** in Figure 14.19), and the eight-band split (at point **C** in Figure 14.19). For a two-band split the magnitude characteristic is flat, with some aliasing. When we employ these same filters to obtain a four-band split from the two-band split, there is an increase in the aliasing. When we go one step further to obtain an eight-band split, the magnitude characteristics deteriorate substantially, as evidenced by Figure 14.20. The various bands are no longer clearly distinct. There is significant overlap between the bands, and hence there will be a significant amount of aliasing in each band.

In order to see why there is an increase in distortion, let us follow the top branch of the tree. The path followed by the signal is shown in Figure 14.21a. As we will show later

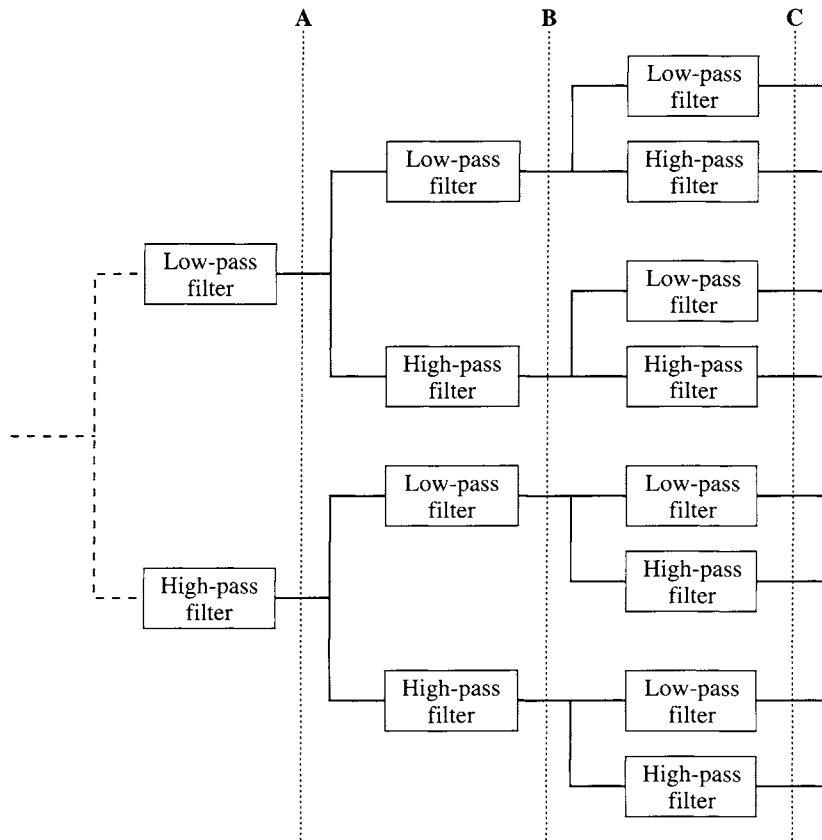


FIGURE 14. 19 Decomposition of an input sequence into multiple bands by recursively using a two-band split.

TABLE 14 . 6 Coefficients for the four-tap Daubechies low-pass filter.

h_0	0.4829629131445341
h_1	0.8365163037378079
h_2	0.2241438680420134
h_3	-0.1294095225512604

(Section 14.8), the three filters and downsamplers can be replaced by a single filter and downsampler as shown in Figure 14.21b, where

$$A(z) = H_L(z)H_L(z^2)H_L(z^4). \quad (14.92)$$

If $H_L(z)$ corresponds to a 4-tap filter, then $A(z)$ corresponds to a $3 \times 6 \times 12 = 216$ -tap filter! However, this is a severely constrained filter because it was generated using only

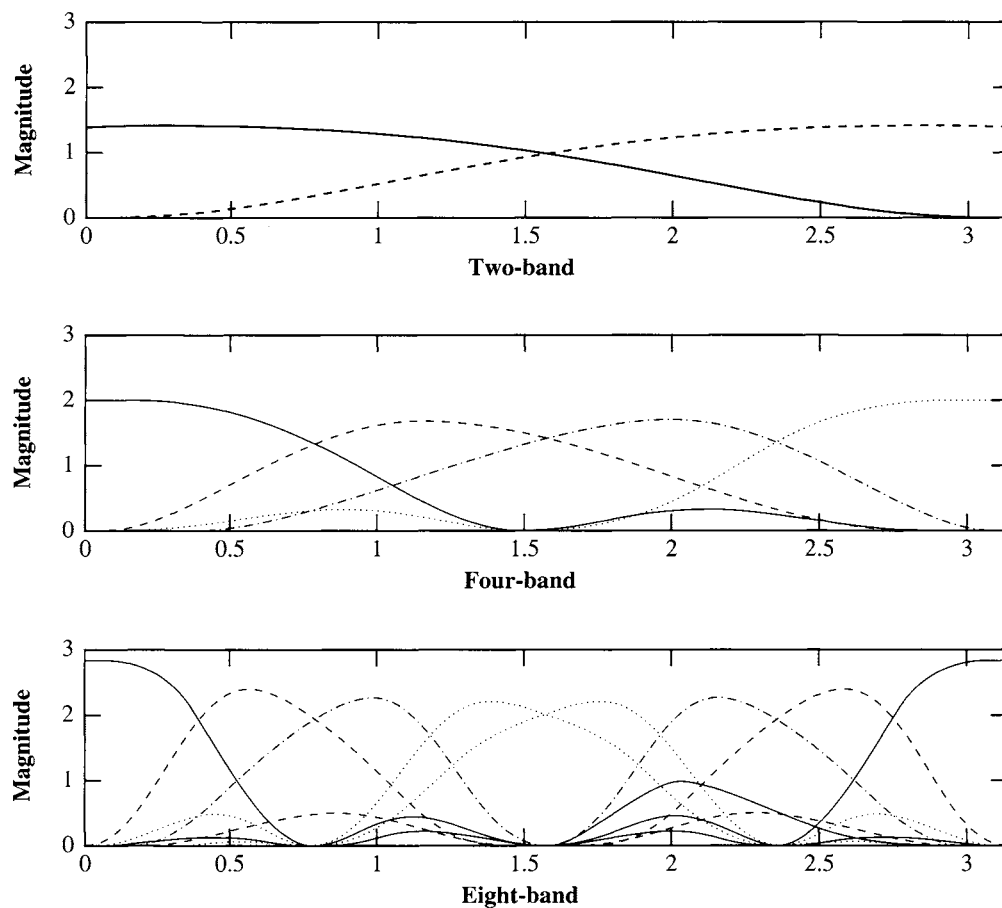


FIGURE 14.20 Spectral characteristics at points A, B, and C.

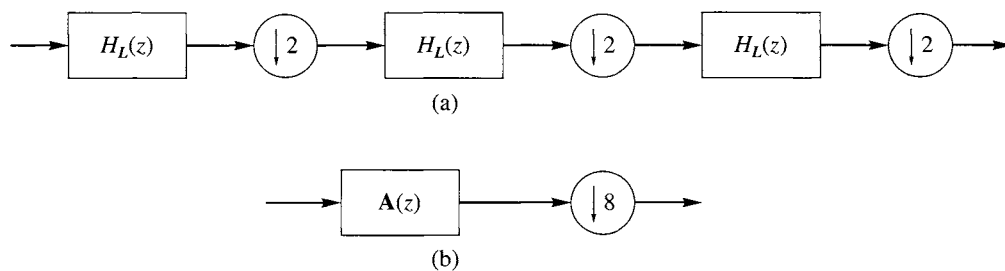


FIGURE 14.21 Equivalent structures for recursive filtering using a two-band split.

four coefficients. If we had set out to design a 216-tap filter from scratch, we would have had significantly more freedom in selecting the coefficients. This is a strong motivation for designing filters directly for the M -band case.

An M -band filter bank has two sets of filters that are arranged as shown in Figure 14.7. The input signal $x(n)$ is split into M frequency bands using an analysis bank of M filters of bandwidth π/M . The signal in any of these M channels is then downsampled by a factor L . This constitutes the analysis bank. The subband signals $y_k(n)$ are encoded and transmitted. At the synthesis stage the subband signals are then decoded, upsampled by a factor of L by interlacing adjacent samples with $L - 1$ zeros, and then passed through the synthesis or interpolation filters. The output of all these synthesis filters is added together to obtain the reconstructed signal. This constitutes the synthesis filter bank. Thus, the analysis and synthesis filter banks together take an input signal $x(n)$ and produce an output signal $\hat{x}(n)$. These filters could be any combination of FIR and IIR filters.

Depending on whether M is less than, equal to, or greater than L , the filter bank is called an *underdecimated*, *critically (maximally) decimated*, or *overdecimated* filter bank. For most practical applications, maximal decimation or “critical subsampling” is used.

A detailed study of M -band filters is beyond the scope of this chapter. Suffice it to say that in broad outline much of what we said about two-band filters can be generalized to M -band filters. (For more on this subject, see [200].)

14.8 The Polyphase Decomposition ★

A major problem with representing the combination of filters and downsamplers is the time-varying nature of the up- and downsamplers. An elegant way of solving this problem is with the use of *polyphase decomposition*. In order to demonstrate this concept, let us first consider the simple case of two-band splitting. We will first consider the analysis portion of the system shown in Figure 14.22. Suppose the analysis filter $H_1(z)$ is given by

$$H_1(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + \dots \quad (14.93)$$

By grouping the odd and even terms together, we can write this as

$$H_1(z) = (h_0 + h_2 z^{-2} + h_4 z^{-4} + \dots) + z^{-1}(h_1 + h_3 z^{-2} + h_5 z^{-4} + \dots). \quad (14.94)$$

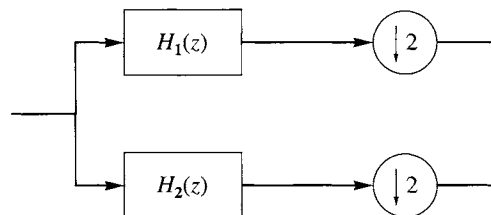


FIGURE 14.22 Analysis portion of a two-band subband coder.

Define

$$H_{10}(z) = h_0 + h_2 z^{-1} + h_4 z^{-2} + \dots \quad (14.95)$$

$$H_{11}(z) = h_1 + h_3 z^{-1} + h_5 z^{-2} + \dots \quad (14.96)$$

Then $H_1(z) = H_{10}(z^2) + z^{-1}H_{11}(z^2)$. Similarly, we can decompose the filter $H_2(z)$ into components $H_{20}(z)$ and $H_{21}(z)$, and we can represent the system of Figure 14.22 as shown in Figure 14.23. The filters $H_{10}(z)$, $H_{11}(z)$ and $H_{20}(z)$, $H_{21}(z)$ are called the polyphase components of $H_1(z)$ and $H_2(z)$.

Let's take the inverse Z-transform of the polyphase components of $H_1(z)$:

$$h_{10}(n) = h_{2n} \quad n = 0, 1, \dots \quad (14.97)$$

$$h_{11}(n) = h_{2n+1} \quad n = 0, 1, \dots \quad (14.98)$$

Thus, $h_{10}(n)$ and $h_{11}(n)$ are simply the impulse response h_n downsampled by two. Consider the output of the downsampler for a given input $X(z)$. The input to the downsampler is $X(z)H_1(z)$; thus, the output from Equation (14.35) is

$$Y_1(z) = \frac{1}{2} X\left(z^{\frac{1}{2}}\right) H_1\left(z^{\frac{1}{2}}\right) + \frac{1}{2} X\left(-z^{\frac{1}{2}}\right) H_1\left(-z^{\frac{1}{2}}\right). \quad (14.99)$$

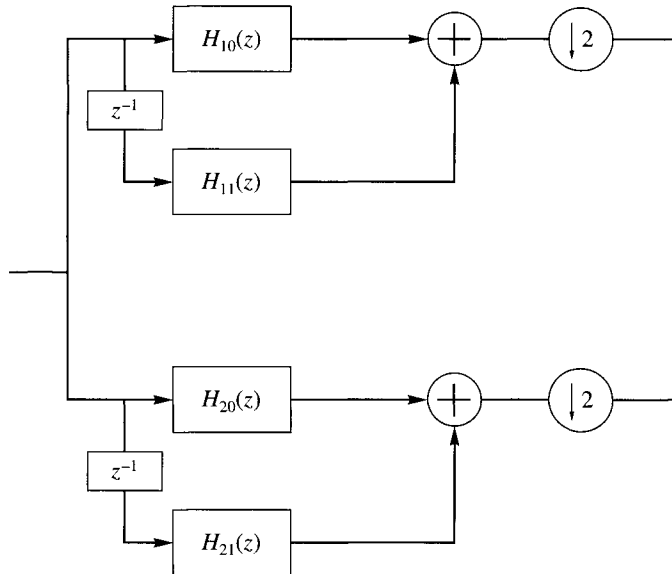


FIGURE 14. 23 Alternative representation of the analysis portion of a two-band subband coder.

Replacing $H_1(z)$ with its polyphase representation, we get

$$Y_1(z) = \frac{1}{2}X\left(z^{\frac{1}{2}}\right)\left[H_{10}(z) + z^{-\frac{1}{2}}H_{11}(z)\right] + \frac{1}{2}X\left(-z^{\frac{1}{2}}\right)\left[H_{10}(z) - z^{-\frac{1}{2}}H_{11}(z)\right] \quad (14.100)$$

$$= H_{10}(z)\left[\frac{1}{2}X\left(z^{\frac{1}{2}}\right) + \frac{1}{2}X\left(-z^{\frac{1}{2}}\right)\right] + H_{11}(z)\left[\frac{1}{2}z^{-\frac{1}{2}}X\left(z^{\frac{1}{2}}\right) - \frac{1}{2}z^{-\frac{1}{2}}X\left(-z^{\frac{1}{2}}\right)\right] \quad (14.101)$$

Note that the first expression in square brackets is the output of a downsampler whose input is $X(z)$, while the quantity in the second set of square brackets is the output of a downsampler whose input is $z^{-1}X(z)$. Therefore, we could implement this system as shown in Figure 14.24.

Now let us consider the synthesis portion of the two-band system shown in Figure 14.25. As in the case of the analysis portion, we can write the transfer functions in terms of their polyphase representation. Thus,

$$G_1(z) = G_{10}(z^2) + z^{-1}G_{11}(z^2) \quad (14.102)$$

$$G_2(z) = G_{20}(z^2) + z^{-1}G_{21}(z^2). \quad (14.103)$$

Consider the output of the synthesis filter $G_1(z)$ given an input $Y_1(z)$. From Equation (14.41), the output of the upsampler is

$$U_1(z) = Y_1(z^2) \quad (14.104)$$

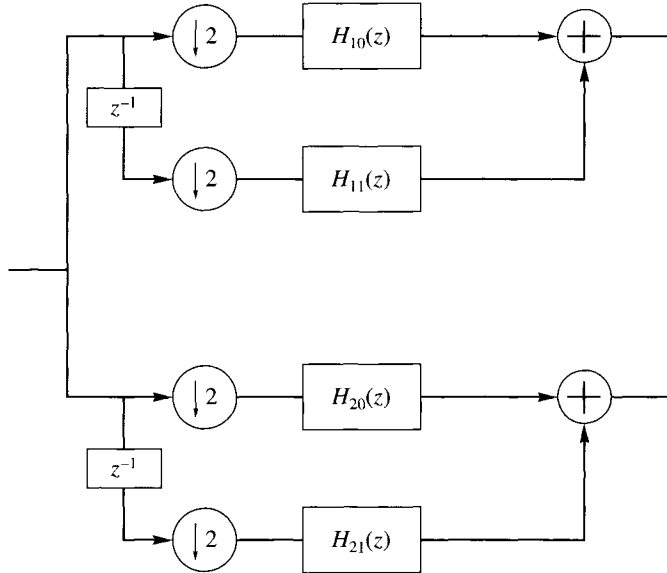


FIGURE 14. 24 Polyphase representation of the analysis portion of a two-band subband coder.

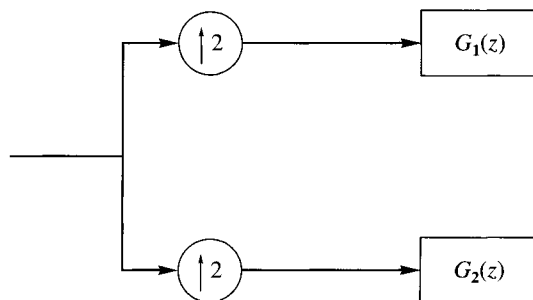


FIGURE 14.25 The synthesis portion of a two-band subband coder.

and the output of $G_1(z)$ is

$$V_1(z) = Y_1(z^2)G_1(z) \quad (14.105)$$

$$= Y_1(z^2)G_{10}(z^2) + z^{-1}Y_1(z^2)G_{11}(z^2). \quad (14.106)$$

The first term in the equation above is the output of an upsampler that *follows* a filter with transfer function $G_{10}(z)$ with input $Y(z)$. Similarly, $Y_1(z^2)G_{11}(z^2)$ is the output of an upsampler that *follows* a filter with transfer function $G_{11}(z)$ with input $Y(z)$. Thus, this system can be represented as shown in Figure 14.26.

Putting the polyphase representations of the analysis and synthesis portions together, we get the system shown in Figure 14.27. Looking at the portion in the dashed box, we can see that this is a completely linear time-invariant system.

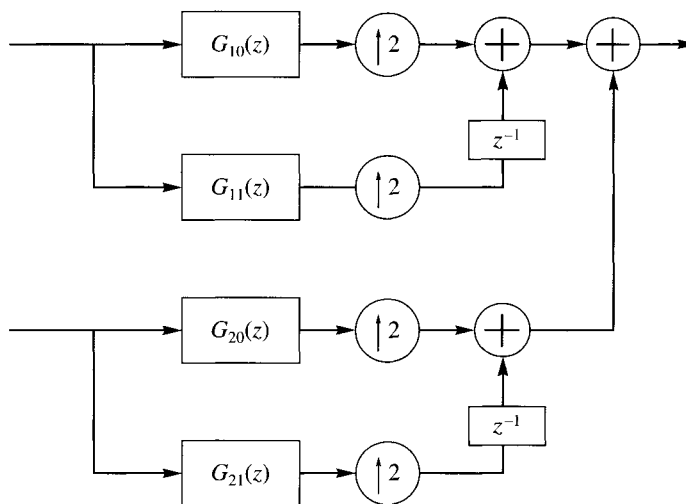


FIGURE 14.26 Polyphase representation of the synthesis portion of a two-band subband coder.

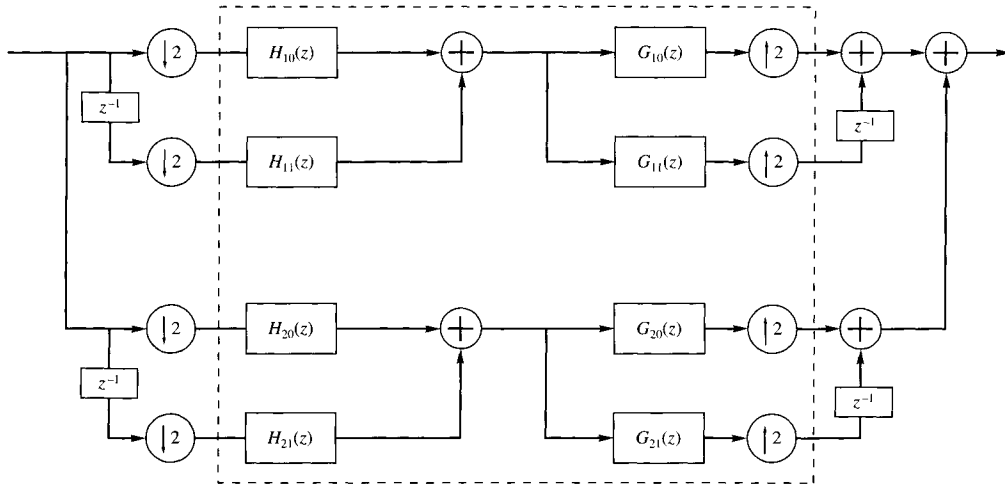


FIGURE 14.27 Polyphase representation of the two-band subband coder.

The polyphase representation can be a very useful tool for the design and analysis of filters. While many of its uses are beyond the scope of this chapter, we can use this representation to prove our statement about the two-band perfect reconstruction QMF filters.

Recall that we want

$$T(z) = \frac{1}{2} [H_1(z)H_2(-z) - H_1(-z)H_2(z)] = cz^{-n_0}.$$

If we impose the mirror condition $H_2(z) = H_1(-z)$, $T(z)$ becomes

$$T(z) = \frac{1}{2} [H_1^2(z) - H_1^2(-z)]. \quad (14.107)$$

The polyphase decomposition of $H_1(z)$ is

$$H_1(z) = H_{10}(z^2) + z^{-1}H_{11}(z^2).$$

Substituting this into Equation (14.107) for $H_1(z)$ and

$$H_1(-z) = H_{10}(z^2) - z^{-1}H_{11}(z^2)$$

for $H_1(-z)$, we obtain

$$T(z) = 2z^{-1}H_{10}(z^2)H_{11}(z^2). \quad (14.108)$$

Clearly, the only way $T(z)$ can have the form cz^{-n_0} is if both $H_{10}(z)$ and $H_{11}(z)$ are simple delays; that is,

$$H_{10}(z) = h_0z^{-k_0} \quad (14.109)$$

$$H_{11}(z) = h_1z^{-k_1}. \quad (14.110)$$

This results in

$$T(z) = 2h_0h_1z^{-(2k_0+2k_1+1)} \quad (14.111)$$

which is of the form cz^{-n_0} as desired. The resulting filters have the transfer functions

$$H_1(z) = h_0z^{-2k_0} + h_1z^{-(2k_1+1)} \quad (14.112)$$

$$H_2(z) = h_0z^{-2k_0} - h_1z^{-(2k_1+1)} \quad (14.113)$$

14.9 Bit Allocation

Once we have separated the source output into the constituent sequences, we need to decide how much of the coding resource should be used to encode the output of each synthesis filter. In other words, we need to allocate the available bits between the subband sequences. In the previous chapter we described a bit allocation procedure that uses the variances of the transform coefficient. In this section we describe a bit allocation approach that attempts to use as much information about the subbands as possible to distribute the bits.

Let's begin with some notation. We have a total of B_T bits that we need to distribute among M subbands. Suppose R corresponds to the average rate in bits per sample for the overall system, and R_k is the average rate for subband k . Let's begin with the case where the input is decomposed into M equal bands, each of which is decimated by a factor of M . Finally, let's assume that we know the rate distortion function for each band. (If you recall from Chapter 8, this is a rather strong assumption and we will relax it shortly.) We also assume that the distortion measure is such that the total distortion is the sum of the distortion contribution of each band.

We want to find the bit allocation R_k such that

$$R = \frac{1}{M} \sum_{k=1}^M R_k \quad (14.114)$$

and the reconstruction error is minimized. Each value of R_k corresponds to a point on the rate distortion curve. The question is where on the rate distortion curve for each subband should we operate to minimize the average distortion. There is a trade-off between rate and distortion. If we decrease the rate (that is, move down the rate distortion curve), we will increase the distortion. Similarly, if we want to move to the left on the rate distortion curve and minimize the distortion, we end up increasing the rate. We need a formulation that incorporates both rate and distortion and the trade-off involved. The formulation we use is based on a landmark paper in 1988 by Yaacov Shoham and Allen Gersho [204]. Let's define a functional J_k :

$$J_k = D_k + \lambda R_k \quad (14.115)$$

where D_k is the distortion contribution from the k th subband and λ is a Lagrangian parameter. This is the quantity we wish to minimize. In this expression the parameter λ in some sense specifies the trade-off. If we are primarily interested in minimizing the distortion, we can set λ to a small value. If our primary interest is in minimizing the rate, we keep the value of

λ large. We can show that the values of D_k and R_k that minimize J_k occur where the slope of the rate distortion curve is λ . Thus, given a value of λ and the rate distortion function, we can immediately identify the values of R_k and D_k . So what should the value of λ be, and how should it vary between subbands?

Let's take the second question first. We would like to allocate bits in such a way that any increase in any of the rates will have the same impact on the distortion. This will happen when we pick R_k in such a way that the slopes of the rate distortion functions for the different subbands are the same; that is, we want to use the same λ for each subband. Let's see what happens if we do not. Consider the two rate distortion functions shown in Figure 14.28. Suppose the points marked x on the rate distortion functions correspond to the selected rates. Obviously, the slopes, and hence the values of λ , are different in the two cases. Because of the differences in the slope, an increase by ΔR in the rate R_1 will result in a much larger decrease in the distortion than the increase in distortion if we decreased R_2 by ΔR . Because the total distortion is the sum of the individual distortions, we can therefore reduce the overall distortions by increasing R_1 and decreasing R_2 . We will be able to keep doing this until the slope corresponding to the rates are the same in both cases. Thus, the answer to our second question is that we want to use the same value of λ for all the subbands.

Given a set of rate distortion functions and a value of λ , we automatically get a set of rates R_k . We can then compute the average and check if it satisfies our constraint on the total number of bits we can spend. If it does not, we modify the value of λ until we get a set of rates that satisfies our rate constraint.

However, generally we do not have rate distortion functions available. In these cases we use whatever is available. For some cases we might have *operational* rate distortion curves available. By "operational" we mean performance curves for particular types of encoders operating on specific types of sources. For example, if we knew we were going to be using *pdf*-optimized nonuniform quantizers with entropy coding, we could estimate the distribution of the subband and use the performance curve for *pdf*-optimized nonuniform quantizers for

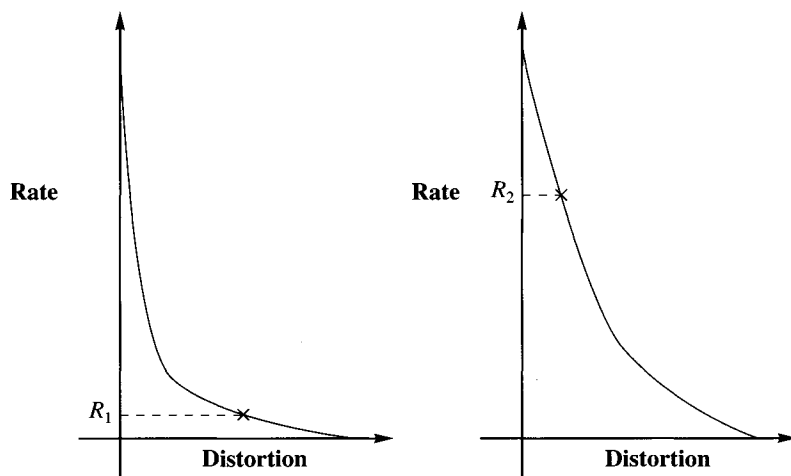


FIGURE 14. 28 Two rate distortion functions.

that distribution. We might only have the performance of the particular encoding scheme for a limited number of rates. In this case we need to have some way of obtaining the slope from a few points. We could estimate this numerically from these points. Or we could fit the points to a curve and estimate the slope from the curve. In these cases we might not be able to get exactly the average rate we wanted.

Finally, we have been talking about a situation where the number of samples in each subband is exactly the same, and therefore the total rate is simply the sum of the individual rates. If this is not true, we need to weight the rates of the individual subbands. The functional to be minimized becomes

$$J = \sum D_k + \lambda \sum \beta_k R_k \quad (14.116)$$

where β_k is the weight reflecting the relative length of the sequence generated by the k th filter. The distortion contribution from each subband might not be equally relevant, perhaps because of the filter construction or because of the perceptual weight attached to those frequencies [205]. In these cases we can modify our functional still further to include the unequal weighting of the distortion:

$$J = \sum w_k D_k + \lambda \sum \beta_k R_k. \quad (14.117)$$

14.10 Application to Speech Coding—G.722

The ITU-T recommendation G.722 provides a technique for wideband coding of speech signals that is based on subband coding. The basic objective of this recommendation is to provide high-quality speech at 64 kbits per second (kbps). The recommendation also contains two other modes that encode the input at 56 and 48 kbps. These two modes are used when an auxiliary channel is needed. The first mode provides for an auxiliary channel of 8 kbps; the second mode, for an auxiliary channel of 16 kbps.

The speech output or audio signal is filtered to 7 kHz to prevent aliasing, then sampled at 16,000 samples per second. Notice that the cutoff frequency for the anti-aliasing filter is 7 kHz, not 8 kHz, even though we are sampling at 16,000 samples per second. One reason for this is that the cutoff for the anti-aliasing filter is not going to be sharp like that of the ideal low-pass filter. Therefore, the highest frequency component in the filter output will be greater than 7 kHz. Each sample is encoded using a 14-bit uniform quantizer. This 14-bit input is passed through a bank of two 24-coefficient FIR filters. The coefficients of the low-pass QMF filter are shown in Table 14.7.

The coefficients for the high-pass QMF filter can be obtained by the relationship

$$h_{HP,n} = (-1)^n h_{LP,n}. \quad (14.118)$$

The low-pass filter passes all frequency components in the range of 0 to 4 kHz, while the high-pass filter passes all remaining frequencies. The output of the filters is downsampled by a factor of two. The downsampled sequences are encoded using adaptive differential PCM (ADPCM) systems.

The ADPCM system that encodes the downsampled output of the low-frequency filter uses 6 bits per sample, with the option of dropping 1 or 2 least significant bits in order to

TABLE 14.7 Transmit and receive QMF coefficient values.

h_0, h_{23}	3.66211×10^{-4}
h_1, h_{22}	-1.34277×10^{-3}
h_2, h_{21}	-1.34277×10^{-3}
h_3, h_{20}	6.46973×10^{-3}
h_4, h_{19}	1.46484×10^{-3}
h_5, h_{18}	-1.90430×10^{-2}
h_6, h_{17}	3.90625×10^{-3}
h_7, h_{16}	4.41895×10^{-2}
h_8, h_{15}	-2.56348×10^{-2}
h_9, h_{14}	-9.82666×10^{-2}
h_{10}, h_{13}	1.16089×10^{-1}
h_{11}, h_{12}	4.73145×10^{-1}

provide room for the auxiliary channel. The output of the high-pass filter is encoded using 2 bits per sample. Because the 2 least significant bits of the quantizer output of the low-pass ADPCM system could be dropped and then not available to the receiver, the adaptation and prediction at both the transmitter and receiver are performed using only the 4 most significant bits of the quantizer output.

If all 6 bits are used in the encoding of the low-frequency subband, we end up with a rate of 48 kbps for the low band. Since the high band is encoded at 2 bits per sample, the output rate for the high subband is 16 kbps. Therefore, the total output rate for the subband-ADPCM system is 64 kbps.

The quantizer is adapted using a variation of the Jayant algorithm [110]. Both ADPCM systems use the past two reconstructed values and the past six quantizer outputs to predict the next sample, in the same way as the predictor for recommendation G.726 described in Chapter 11. The predictor is adapted in the same manner as the predictor used in the G.726 algorithm.

At the receiver, after being decoded by the ADPCM decoder, each output signal is upsampled by the insertion of a zero after each sample. The upsampled signals are passed through the reconstruction filters. These filters are identical to the filters used for decomposing the signal. The low-pass reconstruction filter coefficients are given in Table 14.7, and the coefficients for the high-pass filter can be obtained using Equation (14.118).

14.11 Application to Audio Coding—MPEG Audio

The Moving Picture Experts Group (MPEG) has proposed an audio coding scheme that is based in part on subband coding. Actually, MPEG has proposed three coding schemes, called Layer I, Layer II, and Layer III coding. Each is more complex than the previous and provides higher compression. The coders are also “upward” compatible; a Layer N decoder is able to decode the bitstream generated by the Layer $N - 1$ encoder. In this section we will look primarily at the Layer 1 and Layer 2 coders.

The Layer 1 and Layer 2 coders both use a bank of 32 filters, splitting the input into 32 bands, each with a bandwidth of $f_s/64$, where f_s is the sampling frequency. Allowable

sampling frequencies are 32,000 samples per second, 44,100 samples per second, and 48,000 samples per second. Details of these coders are provided in Chapter 16.

14.12 Application to Image Compression

We have discussed how to separate a sequence into its components. However, all the examples we have used are one-dimensional sequences. What do we do when the sequences contain two-dimensional dependencies such as images? The obvious answer is that we need two-dimensional filters that separate the source output into components based on both the horizontal and vertical frequencies. Fortunately, in most cases, this two-dimensional filter can be implemented as two one-dimensional filters, which can be applied first in one dimension, then in the other. Filters that have this property are called *separable* filters. Two-dimensional non-separable filters do exist [206]; however, the gains are offset by the increase in complexity.

Generally, for subband coding of images we filter each row of the image separately using a high-pass and low-pass filter. The output of the filters is decimated by a factor of two. Assume that the images were of size $N \times N$. After this first stage, we will have two images of size $N \times \frac{N}{2}$. We then filter each column of the two subimages, decimating the outputs of the filters again by a factor of two. This results in four images of size $\frac{N}{2} \times \frac{N}{2}$. We can stop at this point or continue the decomposition process with one or more of the four subimages, resulting in 7, 10, 13, or 16 images. Generally, of the four original subimages, only one or two are further decomposed. The reason for not decomposing the other subimages is that many of the pixel values in the high-frequency subimages are close to zero. Thus, there is little reason to spend computational power to decompose these subimages.

Example 14.12.1:

Let's take the "image" in Table 14.8 and decompose it using the low-pass and high-pass filters of Example 14.2.1. After filtering each row with the low-pass filter, the output is decimated by a factor of two. Each output from the filter depends on the current input and the past input. For the very first input (that is, the pixels at the left edge of the image), we will assume that the past values of the input were zero. The decimated output of the low-pass and high-pass filters is shown in Table 14.9.

We take each of these subimages and filter them column by column using the low-pass and high-pass filters and decimate the outputs by two. In this case, the first input to the filters

TABLE 14.8 A sample "image."

10	14	10	12	14	8	14	12
10	12	8	12	10	6	10	12
12	10	8	6	8	10	12	14
8	6	4	6	4	6	8	10
14	12	10	8	6	4	6	8
12	8	12	10	6	6	6	6
12	10	6	6	6	6	6	6
6	6	6	6	6	6	6	6

TABLE 14.9 Filtered and decimated output.

Decimated Low-Pass Output				Decimated High-Pass Output			
5	12	13	11	5	-2	1	3
5	10	11	8	5	-2	-1	2
6	9	7	11	6	-1	1	1
4	5	5	7	4	-1	-1	1
7	11	7	5	7	-1	-1	1
6	10	8	6	6	2	-2	0
6	8	6	6	6	-2	0	0
3	6	6	6	3	0	0	0

TABLE 14.10 Four subimages.

Low-Low Image				Low-High Image			
2.5	6	6.5	5.5	2.5	6	6.5	5.5
5.5	9.5	9	9.5	0.5	-0.5	-2	1.5
5.5	8	6	6	1.5	3	1	-1
6	9	7	6	0	-1	-1	0
High-Low Image				High-High Image			
2.5	-1	0.5	1.5	2.5	-1	0.5	1.5
5.5	-1.5	0	1.5	0.5	0.5	1	-0.5
5.5	-1	-1	1	1.5	0	0	0
6	0	-1	0	0	-2	1	0

is the top element in each row. We assume that there is a zero row of pixels right above this row in order to provide the filter with “past” values. After filtering and decimation, we get four subimages (Table 14.10). The subimage obtained by low-pass filtering of the columns of the subimage (which was the output of the row low-pass filtering) is called the low-low (LL) image. Similarly, the other images are called the low-high (LH), high-low (HL), and high-high (HH) images. ♦

If we look closely at the final set of subimages in the previous example, we notice that there is a difference in the characteristics of the values in the left or top row and the interiors of some of the subimages. For example, in the high-low subimage, the values in the first column are significantly larger than the other values in the subimage. Similarly, in the low-high subimage, the values in the first row are generally very different than the other values in the subimage. The reason for this variance is our assumption that the “past” of the image above the first row and to the left of the column was zero. The difference between zero and the image values was much larger than the normal pixel-to-pixel differences. Therefore, we ended up adding some spurious structure to the image reflected in the subimages. Generally, this is undesirable because it is easier to select appropriate compression schemes when the characteristics of the subimages are as uniform as possible. For example, if we did not have

TABLE 14.11 **Alternate four subimages.**

Low-Low Image				Low-High Image			
10	12	13	11	0	0	-0.5	-0.5
11	9.5	9	9.5	1	-0.5	-2	1.5
11	8	6	6	3	3	1	-1
12	9	7	6	0	-1	-1	0
High-Low Image				High-High Image			
0	-2	1	3	0	0	0	0
0	-1.5	0	1.5	0	0.5	1	-0.5
0	-1	-1	1	0	0	0	0
0	0	-1	0	0	-2	1	0

the relatively large values in the first column of the high-low subimage, we could choose a quantizer with a smaller step size.

In this example, this effect was limited to a single row or column because the filters used a single past value. However, most filters use a substantially larger number of past values in the filtering operation, and a larger portion of the subimage is affected.

We can avoid this problem by assuming a different “past.” There are a number of ways this can be done. A simple method that works well is to reflect the values of the pixels at the boundary. For example, for the sequence 6 9 5 4 7 2 ..., which was to be filtered with a three-tap filter, we would assume the past as 9 6 6 9 5 4 7 2 If we use this approach for the image in Example 14.12.1, the four subimages would be as shown in Table 14.11.

Notice how much sparser each image is, except for the low-low image. Most of the energy in the original image has been compacted into the low-low image. Since the other subimages have very few values that need to be encoded, we can devote most of our resources to the low-low subimage.

14.12.1 Decomposing an Image

Earlier a set of filters was provided to be used in one-dimensional subband coding. We can use those same filters to decompose an image into its subbands.

Example 14.12.2:

Let’s use the eight-tap Johnston filter to decompose the Sinan image into four subbands. The results of the decomposition are shown in Figure 14.29. Notice that, as in the case of the image in Example 14.12.1, most of the signal energy is concentrated in the low-low subimage. However, there remains a substantial amount of energy in the higher bands. To see this more clearly, let’s look at the decomposition using the 16-tap Johnston filter. The results are shown in Figure 14.30. Notice how much less energy there is in the higher

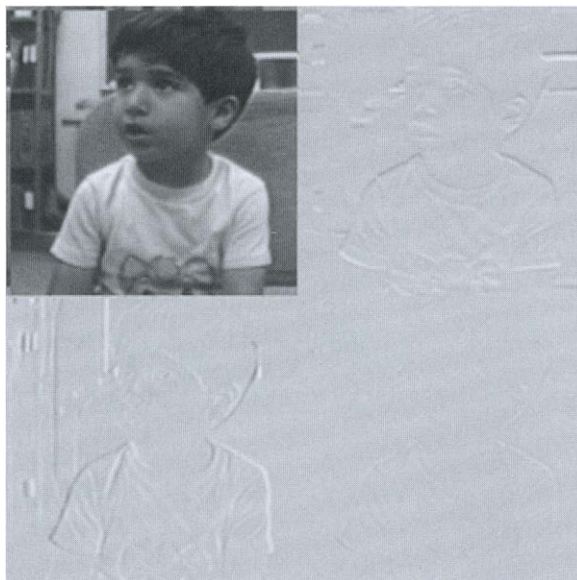


FIGURE 14. 29 **Decomposition of Sinan image using the eight-tap Johnston filter.**

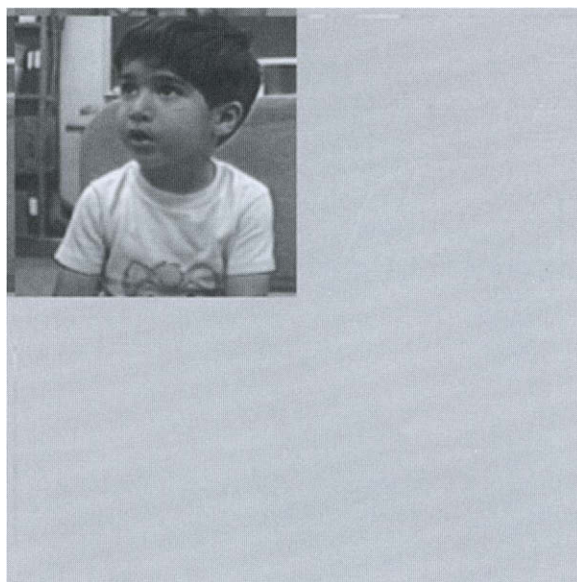


FIGURE 14. 30 **Decomposition of Sinan image using the 16-tap Johnston filter.**

subbands. In fact, the high-high subband seems completely empty. As we shall see later, this difference in *energy compaction* can have a drastic effect on the reconstruction.

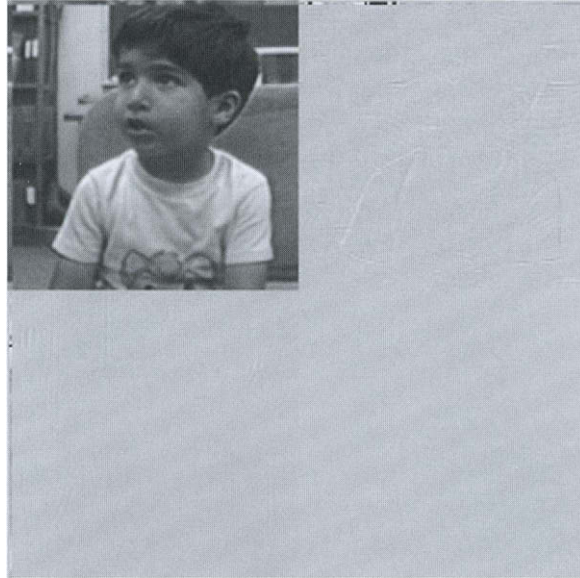


FIGURE 14. 31 **Decomposition of Sinan image using the the eight-tap Smith-Barnwell filter.**

Increasing the size of the filter is not necessarily the only way of improving the energy compaction. Figure 14.31 shows the decomposition obtained using the eight-tap Smith-Barnwell filter. The results are almost identical to the 16-tap Johnston filter. Therefore, rather than increase the computational load by going to a 16-tap filter, we can keep the same computational load and simply use a different filter. ♦

14.12.2 Coding the Subbands

Once we have decomposed an image into subbands, we need to find the best encoding scheme to use with each subband. The coding schemes we have studied to date are scalar quantization, vector quantization, and differential encoding. Let us encode some of the decomposed images from the previous section using two of the coding schemes we have studied earlier, scalar quantization and differential encoding.

Example 14.12.3:

In the previous example we noted the fact that the eight-tap Johnston filter did not compact the energy as well as the 16-tap Johnston filter or the eight-tap Smith-Barnwell filter. Let's see how this affects the encoding of the decomposed images.

When we encode these images at an average rate of 0.5 bits per pixel, there are $4 \times 0.5 = 2$ bits available to encode four values, one value from each of the four subbands. If we use the recursive bit allocation procedure on the eight-tap Johnston filter outputs, we end up allocating 1 bit to the low-low band and 1 bit to the high-low band. As the pixel-to-pixel difference in the low-low band is quite small, we use a DPCM encoder for the low-low band. The high-low band does not show this behavior, which means we can simply use scalar quantization for the high-low band. As there are no bits available to encode the other two bands, these bands can be discarded. This results in the image shown in Figure 14.32, which is far from pleasing. However, if we use the same compression approach with the image decomposed using the eight-tap Smith-Barnwell filter, the result is Figure 14.33, which is much more pleasing.



FIGURE 14. 32 Sinan image coded at 0.5 bits per pixel using the eight-tap Johnston filter.

To understand why we get such different results from using the two filters, we need to look at the way the bits were allocated to the different bands. In this implementation, we used the recursive bit allocation algorithm. In the image decomposed using the Johnston filter, there was significant energy in the high-low band. The algorithm allocated 1 bit to the low-low band and 1 bit to the high-low band. This resulted in poor encoding for both, and subsequently poor reconstruction. There was very little signal content in any of the bands other than the low-low band for the image decomposed using the Smith-Barnwell filter. Therefore, the bit allocation algorithm assigned both bits to the low-low band, which provided a reasonable reconstruction.



FIGURE 14. 33 Sinan image coded at 0.5 bits per pixel using the eight-tap Smith-Barnwell filter.

If the problem with the encoding of the image decomposed by the Johnston filter is an insufficient number of bits for encoding the low-low band, why not simply assign both bits to the low-low band? The problem is that the bit allocation scheme assigned a bit to the high-low band because there was a significant amount of information in that band. If both bits were assigned to the low-low band, we would have no bits left for use in encoding the high-low band, and we would end up throwing away information necessary for the reconstruction. ♦

The issue of energy compaction becomes a very important factor in reconstruction quality. Filters that allow for more energy compaction permit the allocation of bits to a smaller number of subbands. This in turn results in a better reconstruction.

The coding schemes used in this example were DPCM and scalar quantization, the techniques generally preferred in subband coding. The advantage provided by subband coding is readily apparent if we compare the result shown in Figure 14.33 to results in the previous chapters where we used either DPCM or scalar quantization without prior decomposition.

It would appear that the subband approach lends itself naturally to vector quantization. After decomposing an image into subbands, we could design separate codebooks for each subband to reflect the characteristics of that particular subband. The only problem with this idea is that the low-low subband generally requires a large number of bits per pixel. As we mentioned in Chapter 10, it is generally not feasible to operate the nonstructured vector quantizers at high rates. Therefore, when vector quantizers are used, they are generally

used only for encoding the higher frequency bands. This may change as vector quantization algorithms that operate at higher rates are developed.

14.13 Summary

In this chapter we introduced another approach to the decomposition of signals. In subband coding we decompose the source output into components. Each of these components can then be encoded using one of the techniques described in the previous chapters. The general subband encoding procedure can be summarized as follows:

- Select a set of filters for decomposing the source. We have provided a number of filters in this chapter. Many more filters can be obtained from the published literature (we give some references below).
- Using the filters, obtain the subband signals $\{y_{k,n}\}$:

$$y_{k,n} = \sum_{i=0}^{N-1} h_{k,i} x_{n-i} \quad (14.119)$$

where $\{h_{k,n}\}$ are the coefficients of the k th filter.

- Decimate the output of the filters.
- Encode the decimated output.

The decoding procedure is the inverse of the encoding procedure. When encoding images the filtering and decimation operations have to be performed twice, once along the rows and once along the columns. Care should be taken to avoid problems at edges, as described in Section 14.12.

Further Reading

1. *Handbook for Digital Signal Processing*, edited by S.K. Mitra and J.F. Kaiser [162], is an excellent source of information about digital filters.
2. *Multirate Systems and Filter Banks*, by P.P. Vaidyanathan [200], provides detailed information on QMF filters, as well as the relationship between wavelets and filter banks and much more.
3. The topic of subband coding is also covered in *Digital Coding of Waveforms*, by N.S. Jayant and P. Noll [123].
4. The MPEG-1 audio coding algorithm is described in “ISO-MPEG-1 Audio: A Generic Standard for Coding of High-Quality Digital Audio,” by K. Brandenburg and G. Stoll [28], in the October 1994 issue of the *Journal of the Audio Engineering Society*.
5. A review of the rate distortion method of bit allocation is provided in “Rate Distortion Methods for Image and Video Compression,” by A. Ortega and K. Ramachandran, in the November 1998 issue of *IEEE Signal Processing Magazine* [169].

14.14 Projects and Problems

1. A linear shift invariant system has the following properties:

- If for a given input sequence $\{x_n\}$ the output of the system is the sequence $\{y_n\}$, then if we delay the input sequence by k units to obtain the sequence $\{x_{n-k}\}$, the corresponding output will be the sequence $\{y_n\}$ delayed by k units.
- If the output corresponding to the sequence $\{x_n^{(1)}\}$ is $\{y_n^{(1)}\}$, and the output corresponding to the sequence $\{x_n^{(2)}\}$ is $\{y_n^{(2)}\}$, then the output corresponding to the sequence $\{\alpha x_n^{(1)} + \beta x_n^{(2)}\}$ is $\{\alpha y_n^{(1)} + \beta y_n^{(2)}\}$.

Use these two properties to show the convolution property given in Equation (14.18).

2. Let's design a set of simple four-tap filters that satisfies the perfect reconstruction condition.

- (a) We begin with the low-pass filter. Assume that the impulse response of the filter is given by $\{h_{1,k}\}_{k=0}^3$. Further assume that

$$|h_{1,k}| = |h_{1,j}| \quad \forall j, k.$$

Find a set of values for $\{h_{i,j}\}$ that satisfies Equation (14.91).

- (b) Plot the magnitude of the transfer function $H_1(z)$.
- (c) Using Equation (14.23), find the high-pass filter coefficients $\{h_{2,k}\}$.
- (d) Find the magnitude of the transfer function $H_2(z)$.

3. Given an input sequence

$$x_n = \begin{cases} (-1)^n & n = 0, 1, 2, \dots \\ 0 & \text{otherwise} \end{cases}$$

- (a) Find the output sequence y_n if the filter impulse response is

$$h_n = \begin{cases} \frac{1}{\sqrt{2}} & n = 0, 1 \\ 0 & \text{otherwise.} \end{cases}$$

- (b) Find the output sequence w_n if the impulse response of the filter is

$$h_n = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ -\frac{1}{\sqrt{2}} & n = 1 \\ 0 & \text{otherwise.} \end{cases}$$

- (c) Looking at the sequences y_n and w_n , what can you say about the sequence x_n ?

4. Given an input sequence

$$x_n = \begin{cases} 1 & n = 0, 1, 2, \dots \\ 0 & \text{otherwise} \end{cases}$$

- (a) Find the output sequence y_n if the filter impulse response is

$$h_n = \begin{cases} \frac{1}{\sqrt{2}} & n = 0, 1 \\ 0 & \text{otherwise.} \end{cases}$$

- (b) Find the output sequence w_n if the impulse response of the filter is

$$h_n = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ -\frac{1}{\sqrt{2}} & n = 1 \\ 0 & \text{otherwise.} \end{cases}$$

- (c) Looking at the sequences y_n and w_n , what can you say about the sequence x_n ?

5. Write a program to perform the analysis and downsampling operations and another to perform the upsampling and synthesis operations for an image compression application. The programs should read the filter parameters from a file. The synthesis program should read the output of the analysis program and write out the reconstructed images. The analysis program should also write out the subimages scaled so that they can be displayed. Test your program using the Johnston eight-tap filter and the Sena image.
6. In this problem we look at some of the many ways we can encode the subimages obtained after subsampling. Use the eight-tap Johnston filter to decompose the Sena image into four subimages.
 - (a) Encode the low-low band using an adaptive delta modulator (CFDM or CVSD). Encode all other bands using a 1-bit scalar quantizer.
 - (b) Encode the low-low band using a 2-bit adaptive DPCM system. Encode the low-high and high-low bands using a 1-bit scalar quantizer.
 - (c) Encode the low-low band using a 3-bit adaptive DPCM system. Encode the low-high and high-low band using a 0.5 bit/pixel vector quantizer.
 - (d) Compare the reconstructions obtained using the different schemes.