17

Analysis/Synthesis and Analysis by Synthesis Schemes

17.1 Overview

Analysis/synthesis schemes rely on the availability of a parametric model of the source output generation. When such a model exists, the transmitter analyzes the source output and extracts the model parameters, which are transmitted to the receiver. The receiver uses the model along with the transmitted parameters to synthesize an approximation to the source output. The difference between this approach and the techniques we have looked at in previous chapters is that what is transmitted is not a direct representation of the samples of the source output; instead, the transmitter informs the receiver how to go about regenerating those outputs. For this approach to work, a good model for the source has to be available. Since good models for speech production exist, this approach has been widely used for the low-rate coding of speech. We describe several different analysis/synthesis techniques for speech compression. In recent years the fractal approach to image compression has been gaining in popularity. Because this approach is also one in which the receiver regenerates the source output using "instructions" from the transmitter, we describe it in this chapter.

17.2 Introduction

In the previous chapters we have presented a number of lossy compression schemes that provide an estimate of each source output value to the receiver. Historically, an earlier approach towards lossy compression is to model the source output and send the model parameters to the source instead of the estimates of the source output. The receiver tries to synthesize the source output based on the received model parameters.

Consider an image transmission system that works as follows. At the transmitter, we have a person who examines the image to be transmitted and comes up with a description of the image. At the receiver, we have another person who then proceeds to create that image. For example, suppose the image we wish to transmit is a picture of a field of sunflowers. Instead of trying to send the picture, we simply send the words "field of sunflowers." The person at the receiver paints a picture of a field of sunflowers on a piece of paper and gives it to the user. Thus, an image of an object is transmitted from the transmitter to the receiver in a highly compressed form. This approach towards compression should be familiar to listeners of sports broadcasts on radio. It requires that both transmitter and receiver work with the same model. In terms of sports broadcasting, this means that the viewer has a mental picture of the sports arena, and both the broadcaster and listener attach the same meaning to the same terminology.

This approach works for sports broadcasting because the source being modeled functions under very restrictive rules. In a basketball game, when the referee calls a dribbling foul, listeners generally don't picture a drooling chicken. If the source violates the rules, the reconstruction would suffer. If the basketball players suddenly decided to put on a ballet performance, the transmitter (sportscaster) would be hard pressed to represent the scene accurately to the receiver. Therefore, it seems that this approach to compression can only be used for artificial activities that function according to man-made rules. Of the sources that we are interested in, only text fits this description, and the rules that govern the generation of text are complex and differ widely from language to language.

Fortunately, while natural sources may not follow man-made rules, they are subject to the laws of physics, which can prove to be quite restrictive. This is particularly true of speech. No matter what language is being spoken, the speech is generated using machinery that is not very different from person to person. Moreover, this machinery has to obey certain physical laws that substantially limit the behavior of outputs. Therefore, speech can be analyzed in terms of a model, and the model parameters can be extracted and transmitted to the receiver. At the receiver the speech can be synthesized using the model. This analysis/synthesis approach was first employed by Homer Dudley at Bell Laboratories, who developed what is known as the channel vocoder (described in the next section). Actually, the synthesis portion had been attempted even earlier by Kempelen Farkas Lovag (1734–1804). He developed a "speaking machine" in which the vocal tract was modeled by a flexible tube whose shape could be modified by an operator. Sound was produced by forcing air through this tube using bellows [219].

Unlike speech, images are generated in a variety of different ways; therefore, the analysis/synthesis approach does not seem very useful for image or video compression. However, if we restrict the class of images to "talking heads" of the type we would encounter in a videoconferencing situation, we might be able to satisfy the conditions required for this approach. When we talk, our facial gestures are restricted by the way our faces are constructed and by the physics of motion. This realization has led to the new field of model-based video coding (see Chapter 16).

A totally different approach to image compression based on the properties of self-similarity is the *fractal coding* approach. While this approach does not explicitly depend on some physical limitations, it fits in with the techniques described in this chapter; that is, what is stored or transmitted is not the samples of the source output, but a method for synthesizing the output. We will study this approach in Section 17.5.1.

17.3 Speech Compression

A very simplified model of speech synthesis is shown in Figure 17.1. As we described in Chapter 7, speech is produced by forcing air first through an elastic opening, the vocal cords, and then through the laryngeal, oral, nasal, and pharynx passages, and finally through the mouth and the nasal cavity. Everything past the vocal cords is generally referred to as the vocal tract. The first action generates the sound, which is then modulated into speech as it traverses through the vocal tract.

In Figure 17.1, the excitation source corresponds to the sound generation, and the vocal tract filter models the vocal tract. As we mentioned in Chapter 7, there are several different sound inputs that can be generated by different conformations of the vocal cords and the associated cartilages.

Therefore, in order to generate a specific fragment of speech, we have to generate a sequence of sound inputs or excitation signals and the corresponding sequence of appropriate vocal tract approximations.

At the transmitter, the speech is divided into segments. Each segment is analyzed to determine an excitation signal and the parameters of the vocal tract filter. In some of the schemes, a model for the excitation signal is transmitted to the receiver. The excitation signal is then synthesized at the receiver and used to drive the vocal tract filter. In other schemes, the excitation signal itself is obtained using an analysis-by-synthesis approach. This signal is then used by the vocal tract filter to generate the speech signal.

Over the years many different analysis/synthesis speech compression schemes have been developed, and substantial research into the development of new approaches and the improvement of existing schemes continues. Given the large amount of information, we can only sample some of the more popular approaches in this chapter. See [220, 221, 222] for more detailed coverage and pointers to the vast literature on the subject.

The approaches we will describe in this chapter include *channel vocoders*, which are of special historical interest; the *linear predictive coder*, which is the U.S. Government standard at the rate of 2.4 kbps; *code excited linear prediction* (CELP) based schemes; *sinusoidal coders*, which provide excellent performance at rates of 4.8 kbps and higher and are also a part of several national and international standards; and *mixed excitation linear prediction*, which is to be the new 2.4 kbps federal standard speech coder. In our description of these approaches, we will use the various national and international standards as examples.

17.3.1 The Channel Vocoder

In the channel vocoder [223], each segment of input speech is analyzed using a bank of band-pass filters called the *analysis filters*. The energy at the output of each filter is estimated at fixed intervals and transmitted to the receiver. In a digital implementation, the energy

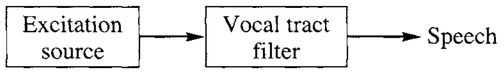

FIGURE 17.1 **A model for speech synthesis.**

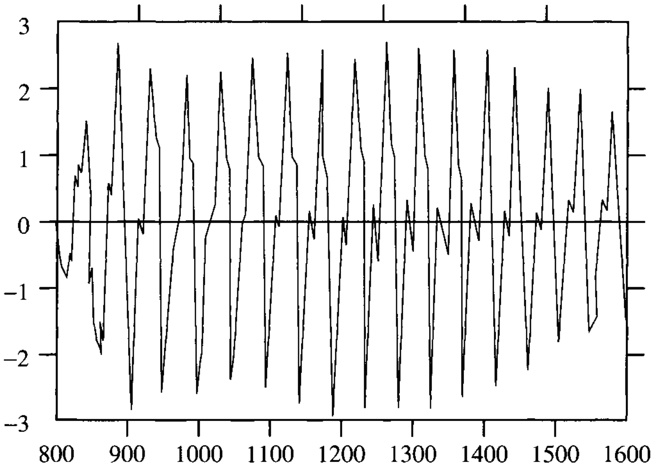

FIGURE 17.2 **The sound /e/ in *test*.**

estimate may be the average squared value of the filter output. In analog implementations, this is the sampled output of an envelope detector. Generally, an estimate is generated 50 times every second. Along with the estimate of the filter output, a decision is made as to whether the speech in that segment is voiced, as in the case of the sounds /a/ /e/ /o/, or unvoiced, as in the case for the sounds /s/ /f/. Voiced sounds tend to have a pseudoperiodic structure, as seen in Figure 17.2, which is a plot of the /e/ part of a male voice saying the word *test*. The period of the fundamental harmonic is called the *pitch* period. The transmitter also forms an estimate of the pitch period, which is transmitted to the receiver.

Unvoiced sounds tend to have a noiselike structure, as seen in Figure 17.3, which is the /s/ sound in the word *test*.

At the receiver, the vocal tract filter is implemented by a bank of band-pass filters. The bank of filters at the receiver, known as the *synthesis filters*, is identical to the bank of analysis filters. Based on whether the speech segment was deemed to be voiced or unvoiced, either a pseudonoise source or a periodic pulse generator is used as the input to the synthesis filter bank. The period of the pulse input is determined by the pitch estimate obtained for the segment being synthesized at the transmitter. The input is scaled by the energy estimate at the output of the analysis filters. A block diagram of the synthesis portion of the channel vocoder is shown in Figure 17.4.

Since the introduction of the channel vocoder, a number of variations have been developed. The channel vocoder matches the frequency profile of the input speech. There is no attempt to reproduce the speech samples per se. However, not all frequency components of speech are equally important. In fact, as the vocal tract is a tube of nonuniform cross section, it resonates at a number of different frequencies. These frequencies are known as *formants* [105]. The formant values change with different sounds; however, we can identify ranges in which they occur. For example, the first formant occurs in the range 200–800 Hz for a male speaker, and in the range 250–1000 Hz for a female speaker. The importance of these

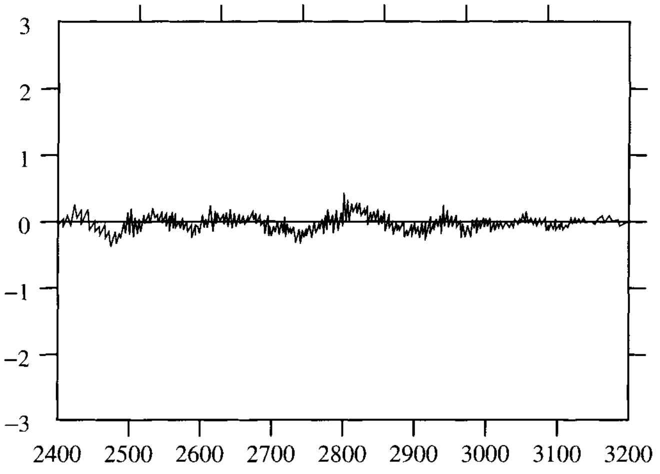

FIGURE 17. 3 **The sound /s/ in *test*.**

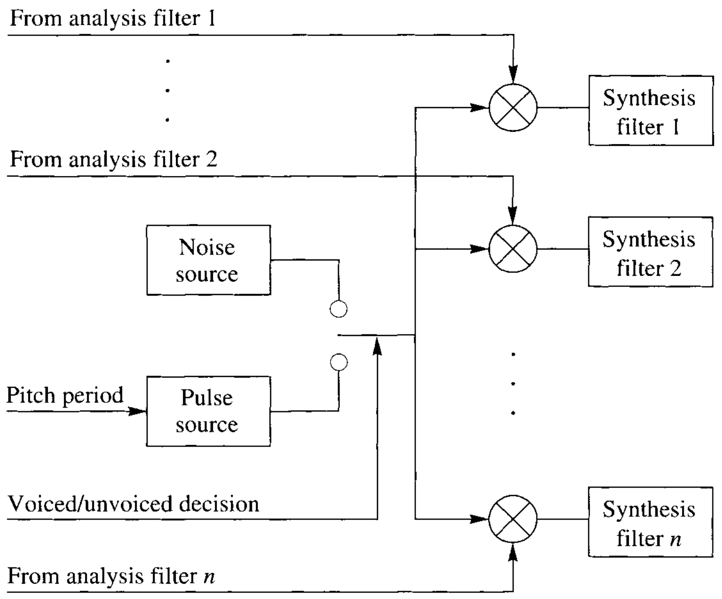

FIGURE 17. 4 **The channel vocoder receiver.**

formants has led to the development of *formant vocoders*, which transmit an estimate of the formant values (usually four formants are considered sufficient) and an estimate of the bandwidth of each formant. At the receiver the excitation signal is passed through tunable filters that are tuned to the formant frequency and bandwidth.

An important development in the history of the vocoders was an understanding of the importance of the excitation signal. Schemes that require the synthesis of the excitation signal at the receiver spend a considerable amount of computational resources to obtain accurate voicing information and accurate pitch periods. This expense can be avoided through the use of voice excitation. In the voice-excited channel vocoder, the voice is first filtered using a narrow-band low-pass filter. The output of the low-pass filter is sampled and transmitted to the receiver. At the receiver, this low-pass signal is passed through a nonlinearity to generate higher-order harmonics that, together with the low-pass signal, are used as the excitation signal. Voice excitation removes the problem of pitch extraction. It also removes the necessity for declaring every segment either voiced or unvoiced. As there are usually quite a few segments that are neither totally voiced or unvoiced, this can result in a substantial increase in quality. Unfortunately, the increase in quality is reflected in the high cost of transmitting the low-pass filtered speech signal.

The channel vocoder, although historically the first approach to analysis/synthesis—indeed the first approach to speech compression—is not as popular as some of the other schemes described here. However, all the different schemes can be viewed as descendants of the channel vocoder.

17.3.2 The Linear Predictive Coder (Government Standard LPC-10)

Of the many descendants of the channel vocoder, the most well-known is the linear predictive coder (LPC). Instead of the vocal tract being modeled by a bank of filters, in the linear predictive coder the vocal tract is modeled as a single linear filter whose output y_n is related to the input ϵ_n by

$$y_n = \sum_{i=1}^{M} b_i y_{n-i} + G\epsilon_n \tag{17.1}$$

where G is called the gain of the filter. As in the case of the channel vocoder, the input to the vocal tract filter is either the output of a random noise generator or a periodic pulse generator. A block diagram of the LPC receiver is shown in Figure 17.5.

At the transmitter, a segment of speech is analyzed. The parameters obtained include a decision as to whether the segment of speech is voiced or unvoiced, the pitch period if the segment is declared voiced, and the parameters of the vocal tract filter. In this section, we will take a somewhat detailed look at the various components that make up the linear predictive coder. As an example, we will use the specifications for the 2.4-kbit U.S. Government Standard LPC-10.

The input speech is generally sampled at 8000 samples per second. In the LPC-10 standard, the speech is broken into 180 sample segments, corresponding to 22.5 milliseconds of speech per segment.

The Voiced/Unvoiced Decision

If we compare Figures 17.2 and 17.3, we can see there are two major differences. Notice that the samples of the voiced speech have larger amplitude; that is, there is more energy in

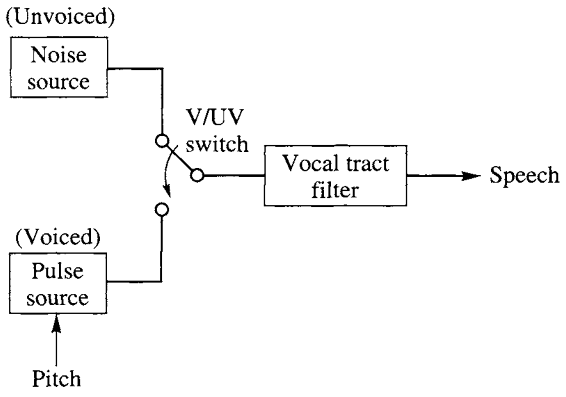

FIGURE 17.5 A model for speech synthesis.

the voiced speech. Also, the unvoiced speech contains higher frequencies. As both speech segments have average values close to zero, this means that the unvoiced speech waveform crosses the $x = 0$ line more often than the voiced speech sample. Therefore, we can get a fairly good idea about whether the speech is voiced or unvoiced based on the energy in the segment relative to background noise and the number of zero crossings within a specified window. In the LPC-10 algorithm, the speech segment is first low-pass filtered using a filter with a bandwidth of 1 kHz. The energy at the output relative to the background noise is used to obtain a tentative decision about whether the signal in the segment should be declared voiced or unvoiced. The estimate of the background noise is basically the energy in the unvoiced speech segments. This tentative decision is further refined by counting the number of zero crossings and checking the magnitude of the coefficients of the vocal tract filter. We will talk more about this latter point later in this section. Finally, it can be perceptually annoying to have a single voiced frame sandwiched between unvoiced frames. The voicing decision of the neighboring frames is considered in order to prevent this from happening.

Estimating the Pitch Period

Estimating the pitch period is one of the most computationally intensive steps of the analysis process. Over the years a number of different algorithms for pitch extraction have been developed. In Figure 17.2, it would appear that obtaining a good estimate of the pitch should be relatively easy. However, we should keep in mind that the segment shown in Figure 17.2 consists of 800 samples, which is considerably more than the samples available to the analysis algorithm. Furthermore, the segment shown here is noise-free and consists entirely of a voiced input. For a machine to extract the pitch from a short noisy segment, which may contain both voiced and unvoiced components, can be a difficult undertaking.

Several algorithms make use of the fact that the autocorrelation of a periodic function $R_{xx}(k)$ will have a maximum when k is equal to the pitch period. Coupled with the fact that the estimation of the autocorrelation function generally leads to a smoothing out of the noise, this makes the autocorrelation function a useful tool for obtaining the pitch period.

Unfortunately, there are also some problems with the use of the autocorrelation. Voiced speech is not exactly periodic, which makes the maximum lower than we would expect from a periodic signal. Generally, a maximum is detected by checking the autocorrelation value against a threshold; if the value is greater than the threshold, a maximum is declared to have occurred. When there is uncertainty about the magnitude of the maximum value, it is difficult to select a value for the threshold. Another problem occurs because of the interference due to other resonances in the vocal tract. There are a number of algorithms that resolve these problems in different ways. (see [105, 104] for details).

In this section, we will describe a closely related technique, employed in the LPC-10 algorithm, that uses the average magnitude difference function (AMDF). The AMDF is defined as

$$AMDF(P) = \frac{1}{N} \sum_{i=k_0+1}^{k_0+N} |y_i - y_{i-P}| \tag{17.2}$$

If a sequence $\{y_n\}$ is periodic with period P_0, samples that are P_0 apart in the $\{y_n\}$ sequence will have values close to each other, and therefore the AMDF will have a minimum at P_0. If we evaluate this function using the /e/ and /s/ sequences, we get the results shown in Figures 17.6 and 17.7. Notice that not only do we have a minimum when P equals the pitch period, but any spurious minimums we may obtain in the unvoiced segments are very shallow; that is, the difference between the minimum and average values is quite small. Therefore, the AMDF can serve a dual purpose: it can be used to identify the pitch period as well as the voicing condition.

The job of pitch extraction is simplified by the fact that the pitch period in humans tends to fall in a limited range. Thus, we do not have to evaluate the AMDF for all possible values of P. For example, the LPC-10 algorithm assumes that the pitch period is between 2.5 and

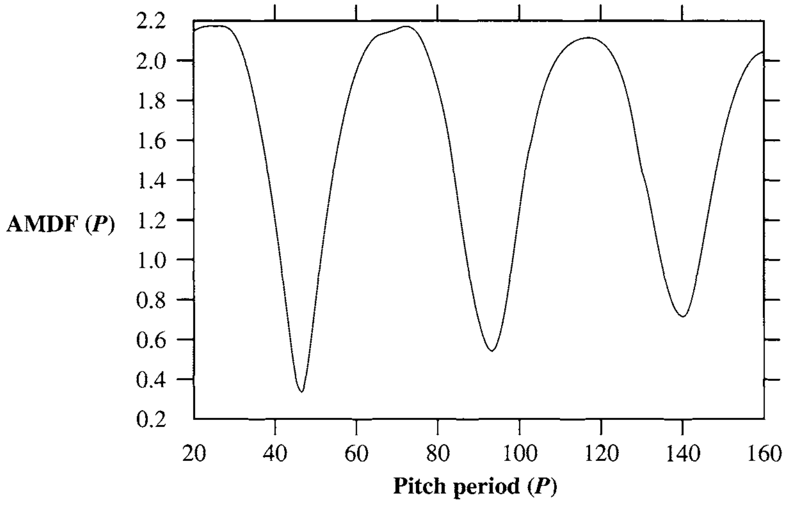

FIGURE 17.6 AMDF function for the sound /e/ in *test*.

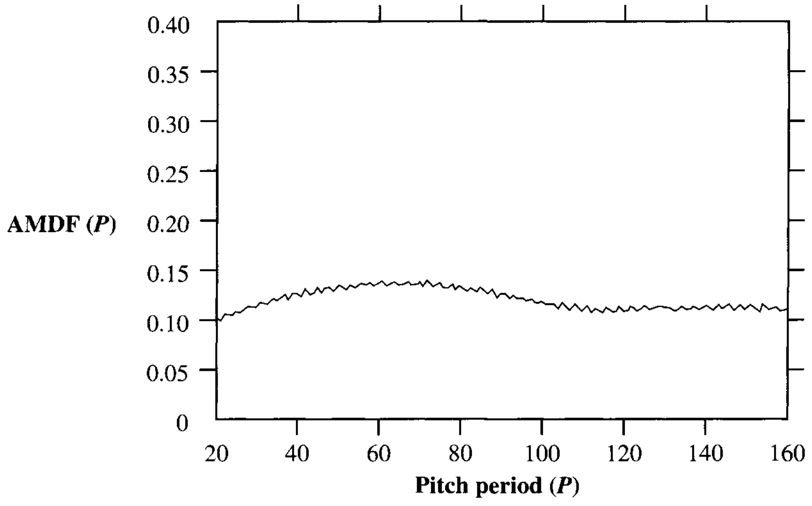

FIGURE 17.7 **AMDF function for the sound /s/ in *test*.**

19.5 milliseconds. Assuming a sampling rate of 8000 samples a second, this means that P is between 20 and 160.

Obtaining the Vocal Tract Filter

In linear predictive coding, the vocal tract is modeled by a linear filter with the input-output relationship shown in Equation (17.1). At the transmitter, during the analysis phase we obtain the filter coefficients that best match the segment being analyzed in a mean squared error sense. That is, if $\{y_n\}$ are the speech samples in that particular segment, then we want to choose $\{a_i\}$ to minimize the average value of e_n^2 where

$$e_n^2 = \left(y_n - \sum_{i=1}^{M} a_i y_{n-i} - G\epsilon_n\right)^2. \tag{17.3}$$

If we take the derivative of the expected value of e_n^2 with respect to the coefficients $\{a_j\}$, we get a set of M equations:

$$\frac{\delta}{\delta a_j} E\left[\left(y_n - \sum_{i=1}^{M} a_i y_{n-i} - G\epsilon_n\right)^2\right] = 0 \tag{17.4}$$

$$\Rightarrow -2E\left[\left(y_n - \sum_{i=1}^{M} a_i y_{n-i} - G\epsilon_n\right) y_{n-j}\right] = 0 \tag{17.5}$$

$$\Rightarrow \sum_{i=1}^{M} a_i E\left[y_{n-i} y_{n-j}\right] = E\left[y_n y_{n-j}\right] \tag{17.6}$$

where in the last step we have made use of the fact that $E[\epsilon_n y_{n-j}]$ is zero for $j \neq 0$. In order to solve (17.6) for the filter coefficients, we need to be able to estimate $E[y_{n-i}y_{n-j}]$. There are two different approaches for estimating these values, called the *autocorrelation* approach and the *autocovariance* approach, each leading to a different algorithm. In the autocorrelation approach, we assume that the $\{y_n\}$ sequence is stationary and therefore

$$E\left[y_{n-1}y_{n-j}\right] = R_{yy}(|i-j|) \tag{17.7}$$

Furthermore, we assume that the $\{y_n\}$ sequence is zero outside the segment for which we are calculating the filter parameters. Therefore, the autocorrelation function is estimated as

$$R_{yy}(k) = \sum_{n=n_0+1+k}^{n_0+N} y_n y_{n-k} \tag{17.8}$$

and the M equations of the form of (17.6) can be written in matrix form as

$$\mathbf{R}A = P \tag{17.9}$$

where

$$\mathbf{R} = \begin{bmatrix} R_y y(0) & R_{yy}(1) & R_{yy}(2) & \cdots & R_{yy}(M-1) \\ R_{yy}(1) & R_{yy}(0) & R_{yy}(1) & \cdots & R_{yy}(M-2) \\ R_{yy}(2) & R_{yy}(1) & R_{yy}(0) & \cdots & R_{yy}(M-3) \\ \vdots & \vdots & \vdots & & \vdots \\ R_{yy}(M-1) & R_{yy}(M-2) & R_{yy}(M-3) & \cdots & R_{yy}(0) \end{bmatrix} \tag{17.10}$$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_M \end{bmatrix} \tag{17.11}$$

and

$$P = \begin{bmatrix} R_{yy}(1) \\ R_{yy}(2) \\ R_{yy}(3) \\ \vdots \\ R_{yy}(M) \end{bmatrix} \tag{17.12}$$

This matrix equation can be solved directly to find the filter coefficients

$$A = \mathbf{R}^{-1}P. \tag{17.13}$$

However, the special form of the matrix $\mathbf{R}$ obviates the need for computing $\mathbf{R}^{-1}$. Note that not only is $\mathbf{R}$ symmetric, but also each diagonal of $\mathbf{R}$ consists of the same element. For example, the main diagonal contains only the element $R_{yy}(0)$, while the diagonals above and below the main diagonal contain only the element $R_{yy}(1)$. This special type of matrix is called a *Toeplitz matrix*, and there are a number of efficient algorithms available for the inversion of Toeplitz matrices [224]. Because $\mathbf{R}$ is Toeplitz, we can obtain a recursive solution to (17.9) that is computationally very efficient and that has an added attractive feature from the point of view of compression. This algorithm is known as the Levinson-Durbin algorithm [225, 226]. We describe the algorithm without derivation. For details of the derivation, see [227, 105].

In order to compute the filter coefficients of an Mth-order filter, the Levinson-Durbin algorithm requires the computation of all filters of order less than M. Furthermore, during the computation of the filter coefficients, the algorithm generates a set of constants k_i known as the *reflection* coefficients, or *partial correlation* (PARCOR) coefficients. In the algorithm description below, we denote the order of the filter using superscripts. Thus, the coefficients of the fifth-order filter would be denoted by $\{a_i^{(5)}\}$. The algorithm also requires the computation of the estimate of the average error $E[e_n^2]$. We will denote the average error using an mth-order filter by E_m. The algorithm proceeds as follows:

1. Set $E_0 = R_{yy}(0)$, $i = 0$.
2. Increment i by one.
3. Calculate $k_i = \left(\sum_{j=1}^{i-1} a_j^{(i-1)} R_{yy}(i-j+1) - R_{yy}(i)\right)/E_{i-1}$.
4. Set $a_i^{(i)} = k_i$.
5. Calculate $a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{i-1}$ for $j = 1, 2, \ldots, i-1$.
6. Calculate $E_i = \left(1 - k_i^2\right) E_{i-1}$.
7. If $i < M$, go to step 2.

In order to get an effective reconstruction of the voiced segment, the order of the vocal tract filter needs to be sufficiently high. Generally, the order of the filter is 10 or more. Because the filter is an IIR filter, error in the coefficients can lead to instability, especially for the high orders necessary in linear predictive coding. As the filter coefficients are to be transmitted to the receiver, they need to be quantized. This means that quantization error is introduced into the value of the coefficients, and that can lead to instability.

This problem can be avoided by noticing that if we know the PARCOR coefficients, we can obtain the filter coefficients from them. Furthermore, PARCOR coefficients have the property that as long as the magnitudes of the coefficients are less than one, the filter obtained from them is guaranteed to be stable. Therefore, instead of quantizing the coefficients $\{a_i\}$ and transmitting them, the transmitter quantizes and transmits the coefficients $\{k_i\}$. As long as we make sure that all the reconstruction values for the quantizer have magnitudes less than one, it is possible to use relatively high-order filters in the analysis/synthesis schemes.

The assumption of stationarity that was used to obtain (17.6) is not really valid for speech signals. If we discard this assumption, the equations to obtain the filter coefficients change. The term $E[y_{n-i}y_{n-j}]$ is now a function of both i and j. Defining

$$c_{ij} = E[y_{n-i}y_{n-j}] \tag{17.14}$$

we get the equation

$$\mathbf{C}A = S \tag{17.15}$$

where

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \cdots & c_{1M} \\ c_{21} & c_{22} & c_{23} & \cdots & c_{2M} \\ \vdots & \vdots & \vdots & & \vdots \\ c_{M1} & c_{M2} & c_{M3} & \cdots & c_{MM} \end{bmatrix} \tag{17.16}$$

and

$$S = \begin{bmatrix} c_{10} \\ c_{20} \\ c_{30} \\ \vdots \\ c_{M0} \end{bmatrix} \tag{17.17}$$

The elements c_{ij} are estimated as

$$c_{ij} = \sum_{n=n_0+1}^{n_0+N} y_{n-i}y_{n-j}. \tag{17.18}$$

Notice that we no longer assume that the values of y_n outside of the segment under consideration are zero. This means that in calculating the $\mathbf{C}$ matrix for a particular segment, we use samples from previous segments. This method of computing the filter coefficients is called the *covariance method.*

The $\mathbf{C}$ matrix is symmetric but no longer Toeplitz, so we can't use the Levinson-Durbin recursion to solve for the filter coefficients. The equations are generally solved using a technique called the *Cholesky decomposition.* We will not describe the solution technique here. (You can find it in most texts on numerical techniques; an especially good source is [178].) For an in-depth study of the relationship between the Cholesky decomposition and the reflection coefficients, see [228].

The LPC-10 algorithm uses the covariance method to obtain the reflection coefficients. It also uses the PARCOR coefficients to update the voicing decision. In general, for voiced signals the first two PARCOR coefficients have values close to one. Therefore, if both the first two PARCOR coefficients have very small values, the algorithm sets the voicing decision to unvoiced.

Transmitting the Parameters

Once the various parameters have been obtained, they need to be coded and transmitted to the receiver. There are a variety of ways this can be done. Let us look at how the LPC-10 algorithm handles this task.

The parameters that need to be transmitted include the voicing decision, the pitch period, and the vocal tract filter parameters. One bit suffices to transmit the voicing information. The pitch is quantized to 1 of 60 different values using a log-companded quantizer. The LPC-10 algorithm uses a 10th-order filter for voiced speech and a 4th-order filter for unvoiced speech. Thus, we have to send 11 values (10 reflection coefficients and the gain) for voiced speech and 5 for unvoiced speech.

The vocal tract filter is especially sensitive to errors in reflection coefficients that have magnitudes close to one. As the first few coefficients are most likely to have values close to one, the LPC-10 algorithm specifies the use of nonuniform quantization for k_1 and k_2. The nonuniform quantization is implemented by first generating the coefficients

$$g_i = \frac{1+k_i}{1-k_i} \tag{17.19}$$

which are then quantized using a 5-bit uniform quantizer. The coefficients k_3 and k_4 are both quantized using a 5-bit uniform quantizer. In the voiced segments, coefficients k_5 through k_8 are quantized using a 4-bit uniform quantizer, k_9 is quantized using a 3-bit uniform quantizer, and k_{10} is quantized using a 2-bit uniform quantizer. In the unvoiced segments, the 21 bits used to quantize k_5 through k_{10} in the voiced segments are used for error protection.

The gain G is obtained by finding the root mean squared (rms) value of the segment and quantized using 5-bit log-companded quantization. Including an additional bit for synchronization, we end up with a total of 54 bits per frame. Multiplying this by the total number of frames per second gives us the target rate of 2400 bits per second.

Synthesis

At the receiver, the voiced frames are generated by exciting the received vocal tract filter by a locally stored waveform. This waveform is 40 samples long. It is truncated or padded with zeros depending on the pitch period. If the frame is unvoiced, the vocal tract is excited by a pseudorandom number generator.

The LPC-10 coder provides intelligible reproduction at 2.4 kbits. The use of only two kinds of excitation signals gives an artificial quality to the voice. This approach also suffers when used in noisy environments. The encoder can be fooled into declaring segments of speech unvoiced because of background noise. When this happens, the speech information gets lost.

17.3.3 Code Excited Linear Predicton (CELP)

As we mentioned earlier, one of the most important factors in generating natural-sounding speech is the excitation signal. As the human ear is especially sensitive to pitch errors, a great deal of effort has been devoted to the development of accurate pitch detection algorithms.

However, no matter how accurate the pitch is in a system using the LPC vocal tract filter, the use of a periodic pulse excitation that consists of a single pulse per pitch period leads to a "buzzy twang" [229]. In 1982, Atal and Remde [230] introduced the idea of multipulse linear predictive coding (MP-LPC), in which several pulses were used during each segment. The spacing of these pulses is determined by evaluating a number of different patterns from a codebook of patterns.

A codebook of excitation patterns is constructed. Each entry in this codebook is an excitation sequence that consists of a few nonzero values separated by zeros. Given a segment from the speech sequence to be encoded, the encoder obtains the vocal tract filter using the LPC analysis described previously. The encoder then excites the vocal tract filter with the entries of the codebook. The difference between the original speech segment and the synthesized speech is fed to a perceptual weighting filter, which weights the error using a perceptual weighting criterion. The codebook entry that generates the minimum average weighted error is declared to be the best match. The index of the best-match entry is sent to the receiver along with the parameters for the vocal tract filter.

This approach was improved upon by Atal and Schroeder in 1984 with the introduction of the system that is commonly known as *code excited linear prediction* (CELP). In CELP, instead of having a codebook of pulse patterns, we allow a variety of excitation signals. For each segment the encoder finds the excitation vector that generates synthesized speech that best matches the speech segment being encoded. This approach is closer in a strict sense to a waveform coding technique such as DPCM than to the analysis/synthesis schemes. However, as the ideas behind CELP are similar to those behind LPC, we included CELP in this chapter. The main components of the CELP coder include the LPC analysis, the excitation codebook, and the perceptual weighting filter. Each component of the CELP coder has been investigated in great detail by a large number of researchers. For a survey of some of the results, see [220]. In the rest of the section, we give two examples of very different kinds of CELP coders. The first algorithm is the U.S. Government Standard 1016, a 4.8 kbps coder; the other is the CCITT (now ITU-T) G.728 standard, a low-delay 16 kbps coder.

Besides CELP, the MP-LPC algorithm had another descendant that has become a standard. In 1986, Kroon, Deprettere, and Sluyter [231] developed a modification of the MP-LPC algorithm. Instead of using excitation vectors in which the nonzero values are separated by an arbitrary number of zero values, they forced the nonzero values to occur at regularly spaced intervals. Furthermore, they allowed the nonzero values to take on a number of different values. They called this scheme *regular pulse excitation* (RPE) coding. A variation of RPE, called *regular pulse excitation with long-term prediction* (RPE-LTP) [232], was adopted as a standard for digital cellular telephony by the Group Speciale Mobile (GSM) subcommittee of the European Telecommunications Standards Institute at the rate of 13 kbps.

Federal Standard 1016

The vocal tract filter used by the CELP coder in FS 1016 is given by

$$y_n = \sum_{i=1}^{10} b_i y_{n-i} + \beta y_{n-P} + G\epsilon_n \tag{17.20}$$

where P is the pitch period and the term βy_{n-P} is the contribution due to the pitch periodicity. The input speech is sampled at 8000 samples per second and divided into 30-millisecond frames containing 240 samples. Each frame is divided into four subframes of length 7.5 milliseconds [233]. The coefficients $\{b_i\}$ for the 10th-order short-term filter are obtained using the autocorrelation method.

The pitch period P is calculated once every subframe. In order to reduce the computational load, the pitch value is assumed to lie between between 20 and 147 every odd subframe. In every even subframe, the pitch value is assumed to lie within 32 samples of the pitch value in the previous frame.

The FS 1016 algorithm uses two codebooks [234], a stochastic codebook and an adaptive codebook. An excitation sequence is generated for each subframe by adding one scaled element from the stochastic codebook and one scaled element from the adaptive codebook. The scale factors and indices are selected to minimize the perceptual error between the input and synthesized speech.

The stochastic codebook contains 512 entries. These entries are generated using a Gaussian random number generator, the output of which is quantized to -1, 0, or 1. If the input is less than -1.2, it is quantized to -1; if it is greater than 1.2, it is quantized to 1; and if it lies between -1.2 and 1.2, it is quantized to 0. The codebook entries are adjusted so that each entry differs from the preceding entry in only two places. This structure helps reduce the search complexity.

The adaptive codebook consists of the excitation vectors from the previous frame. Each time a new excitation vector is obtained, it is added to the codebook. In this manner, the codebook adapts to local statistics.

The FS 1016 coder has been shown to provide excellent reproductions in both quiet and noisy environments at rates of 4.8 kbps and above [234]. Because of the richness of the excitation signals, the reproduction does not suffer from the problem of sounding artificial. The lack of a voicing decision makes it more robust to background noise. The quality of the reproduction of this coder at 4.8 kbps has been shown to be equivalent to a delta modulator operating at 32 kbps [234]. The price for this quality is much higher complexity and a much longer coding delay. We will address this last point in the next section.

CCITT G.728 Speech Standard

By their nature, the schemes described in this chapter have some coding delay built into them. By "coding delay," we mean the time between when a speech sample is encoded to when it is decoded if the encoder and decoder were connected back-to-back (i.e., there were no transmission delays). In the schemes we have studied, a segment of speech is first stored in a buffer. We do not start extracting the various parameters until a complete segment of speech is available to us. Once the segment is completely available, it is processed. If the processing is real time, this means another segment's worth of delay. Finally, once the parameters have been obtained, coded, and transmitted, the receiver has to wait until at least a significant part of the information is available before it can start decoding the first sample. Therefore, if a segment contains 20 milliseconds' worth of data, the coding delay would be approximately somewhere between 40 to 60 milliseconds. This kind of delay may be acceptable for some applications; however, there are other applications where such long delays are not acceptable. For example, in some situations there are several intermediate tandem connections between the initial transmitter and the final receiver. In such situations, the total delay would be a multiple of the coding delay of a single connection. The size of the delay would depend on the number of tandem connections and could rapidly become quite large.

For such applications, CCITT approved recommendation G.728, a CELP coder with a coder delay of 2 milliseconds operating at 16 kbps. As the input speech is sampled at 8000 samples per second, this rate corresponds to an average rate of 2 bits per sample.

In order to lower the coding delay, the size of each segment has to be reduced significantly because the coding delay will be some multiple of the size of the segment. The G.728 recommendation uses a segment size of five samples. With five samples and a rate of 2 bits per sample, we only have 10 bits available to us. Using only 10 bits, it would be impossible to encode the parameters of the vocal tract filter as well as the excitation vector. Therefore, the algorithm obtains the vocal tract filter parameters in a backward adaptive manner; that is, the vocal tract filter coefficients to be used to synthesize the current segment are obtained by analyzing the previous decoded segments. The CCITT requirements for G.728 included the requirement that the algorithm operate under noisy channel conditions. It would be extremely difficult to extract the pitch period from speech corrupted by channel errors. Therefore, the G.728 algorithm does away with the pitch filter. Instead, the algorithm uses a 50th-order vocal tract filter. The order of the filter is large enough to model the pitch of most female speakers. Not being able to use pitch information for male speakers does not cause much degradation [235]. The vocal tract filter is updated every fourth frame, which is once every 20 samples or 2.5 milliseconds. The autocorrelation method is used to obtain the vocal tract parameters.

As the vocal tract filter is completely determined in a backward adaptive manner, we have all 10 bits available to encode the excitation sequence. Ten bits would be able to index 1024 excitation sequences. However, to examine 1024 excitation sequences every 0.625 milliseconds is a rather large computational load. In order to reduce this load, the G.728 algorithm uses a product codebook where each excitation sequence is represented by a normalized sequence and a gain term. The final excitation sequence is a product of the normalized excitation sequence and the gain. Of the 10 bits, 3 bits are used to encode the gain using a predictive encoding scheme, while the remaining 7 bits form the index to a codebook containing 127 sequences.

Block diagrams of the encoder and decoder for the CCITT G.728 coder are shown in Figure 17.8. The low-delay CCITT G.728 CELP coder operating at 16 kbps provides reconstructed speech quality superior to the 32 kbps CCITT G.726 ADPCM algorithm described in Chapter 10. Various efforts are under way to reduce the bit rate for this algorithm without compromising too much on quality and delay.

17.3.4 Sinusoidal Coders

A competing approach to CELP in the low-rate region is a relatively new form of coder called the sinusoidal coder [220]. Recall that the main problem with the LPC coder was the paucity of excitation signals. The CELP coder resolved this problem by using a codebook of excitation signals. The sinusoidal coders solve this problem by using an excitation signal

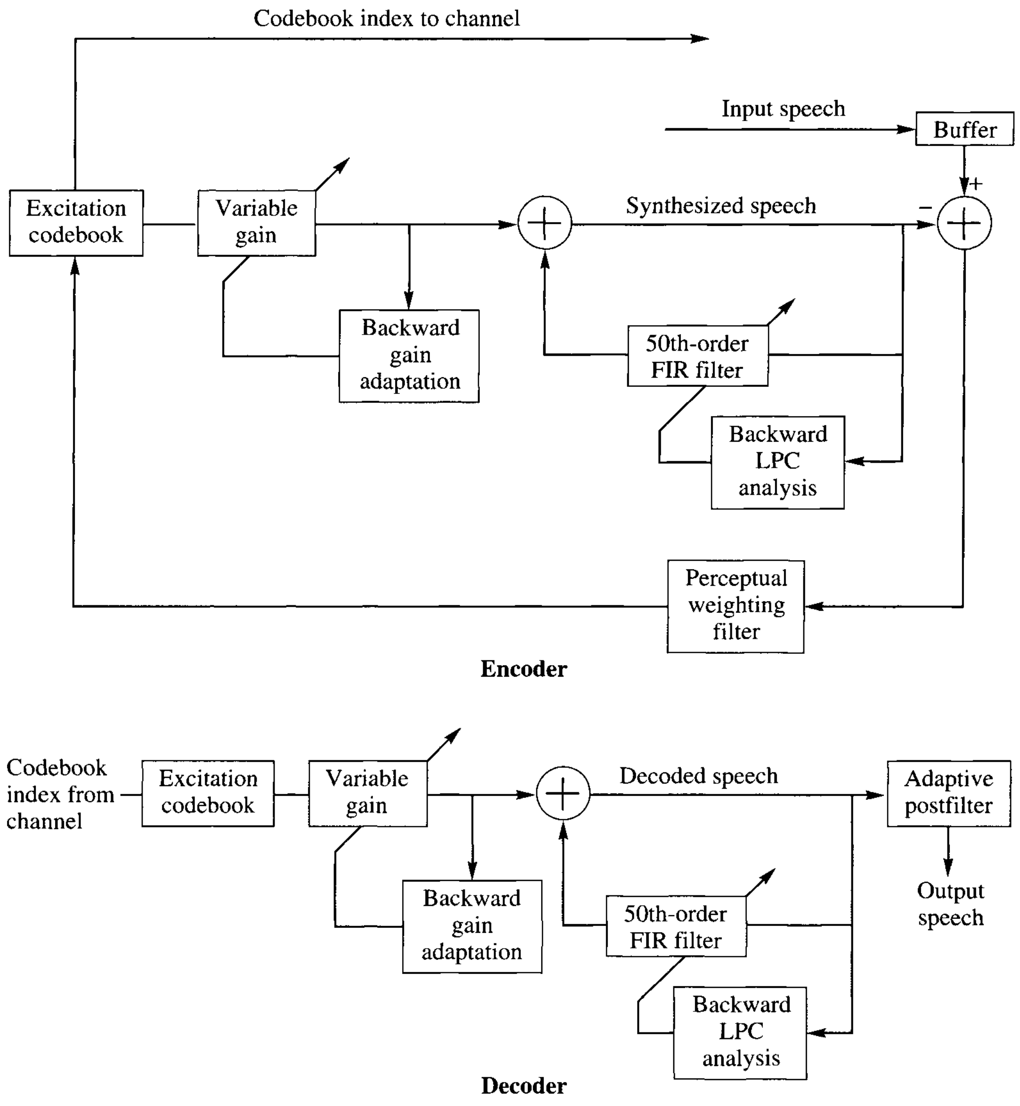

FIGURE 17. 8 Encoder and decoder for the CCITT G.728 16 kbps speech coder.

that is the sum of sine waves of arbitrary amplitudes, frequencies, and phases. Thus, the excitation signal is of the form

$$e_n = \sum_{l=1}^{L} a_l \cos(n\omega_l + \psi_l) \tag{17.21}$$

where the number of sinusoids L required for each frame depends on the contents of the frame. If the input to a linear system is a sinusoid with frequency ω_l, the output will also be a sinusoid with frequency ω_l, albeit with different amplitude and phase. The vocal tract filter is a linear system. Therefore, if the excitation signal is of the form of (17.21), the synthesized speech $\{s_n\}$ will be of the form

$$s_n = \sum_{i=1}^{L} A_l \cos(n\omega_l + \phi_l). \tag{17.22}$$

Thus, each frame is characterized by a set of spectral amplitudes A_l, frequencies ω_l, and phase terms ϕ_l. The number of parameters required to represent the excitation sequence is the same as the number of parameters required to represent the synthesized speech. Therefore, rather than estimate and transmit the parameters of both the excitation signal and vocal tract filter and then synthesize the speech at the receiver by passing the excitation signal through the vocal tract filter, the sinusoidal coders directly estimate the parameters required to synthesize the speech at the receiver.

Just like the coders discussed previously, the sinusoidal coders divide the input speech into frames and obtain the parameters of the speech separately for each frame. If we synthesized the speech segment in each frame independent of the other frames, we would get synthetic speech that is discontinuous at the frame boundaries. These discontinuities severely degrade the quality of the synthetic speech. Therefore, the sinusoidal coders use different interpolation algorithms to smooth the transition from one frame to another.

Transmitting all the separate frequencies ω_l would require significant transmission resources, so the sinusoidal coders obtain a fundamental frequency w_0 for which the approximation

$$\hat{y}_n = \sum_{k=1}^{K(\omega_0)} \hat{A}(k\omega_0)\cos(nk\omega_0 + \phi_k) \tag{17.23}$$

is close to the speech sequence y_n. Because this is a harmonic approximation, the approximate sequence $\{\hat{y}_n\}$ will be most different from the speech sequence $\{y_n\}$ when the segment of speech being encoded is unvoiced. Therefore, this difference can be used to decide whether the frame or some subset of it is unvoiced.

The two most popular sinusoidal coding techniques today are represented by the sinusoidal transform coder (STC) [236] and the multiband excitation coder (MBE) [237]. While the STC and MBE are similar in many respects, they differ in how they handle unvoiced speech. In the MBE coder, the frequency range is divided into bands, each consisting of several harmonics of the fundamental frequency ω_0. Each band is checked to see if it is unvoiced or voiced. The voiced bands are synthesized using a sum of sinusoids, while the unvoiced bands are obtained using a random number generator. The voiced and unvoiced bands are synthesized separately and then added together.

In the STC, the proportion of the frame that contains a voiced signal is measured using a "voicing probability" P_v. The voicing probability is a function of how well the harmonic model matches the speech segment. Where the harmonic model is close to the speech signal, the voicing probability is taken to be unity. The sine wave frequencies are then generated by

$$w_k = \begin{cases} kw_0 & \text{for } kw_0 \leq w_c P_v \\ k^* w_0 + (k - k^*) w_u & \text{for } kw_0 > w_c P_v \end{cases} \tag{17.24}$$

where w_c corresponds to the cutoff frequency (4 kHz), w_u is the unvoiced pitch corresponding to 100 Hz, and k^* is the largest value of k for which $k^* w_0 \leq w_c P_v$. The speech is then synthesized as

$$\hat{y}_n = \sum_{k=1}^{K} \hat{A}(w_k) \cos(nw_k + \phi_k). \tag{17.25}$$

Both the STC and the MBE coders have been shown to perform well at low rates. A version of the MBE coder known as the improved MBE (IMBE) coder has been approved by the Association of Police Communications Officers (APCO) as the standard for law enforcement.

17.3.5 Mixed Excitation Linear Prediction (MELP)

The mixed excitation linear prediction (MELP) coder was selected to be the new federal standard for speech coding at 2.4 kbps by the Defense Department Voice Processing Consortium (DDVPC). The MELP algorithm uses the same LPC filter to model the vocal tract. However, it uses a much more complex approach to the generation of the excitation signal.

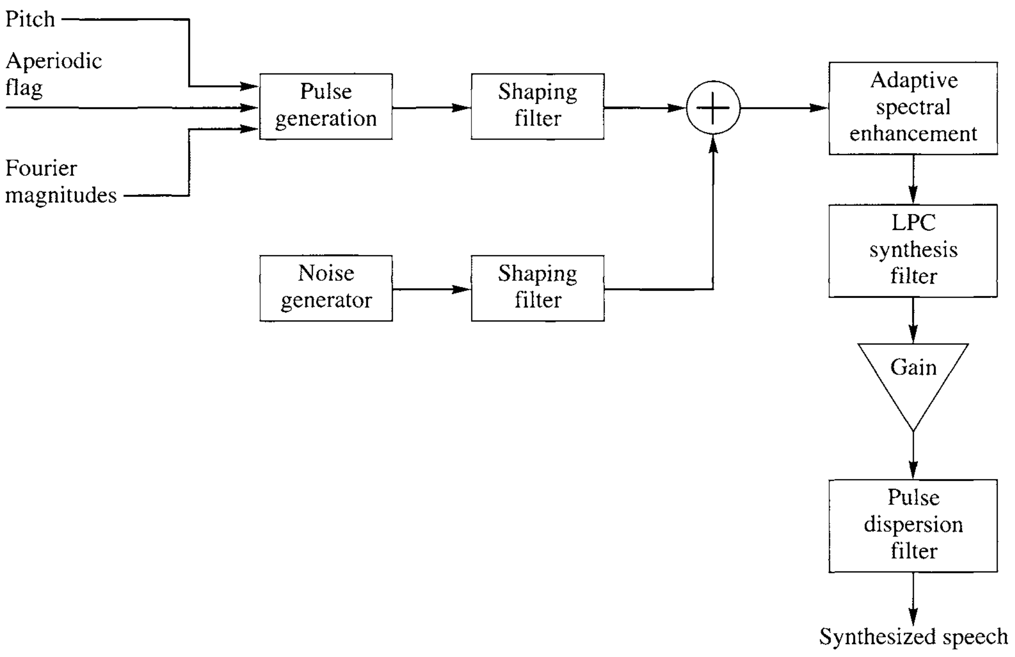

FIGURE 17.9 Block diagram of MELP decoder.

A block diagram of the decoder for the MELP system is shown in Figure 17.9. As evident from the figure, the excitation signal for the synthesis filter is no longer simply noise or a periodic pulse but a multiband mixed excitation. The mixed excitation contains both a filtered signal from a noise generator as well as a contribution that depends directly on the input signal.

The first step in constructing the excitation signal is pitch extraction. The MELP algorithm obtains the pitch period using a multistep approach. In the first step an integer pitch value P_1 is obtained by

1. first filtering the input using a low-pass filter with a cutoff of 1 kHz
2. computing the normalized autocorrelation for lags between 40 and 160

The normalized autocorrelation $r(\tau)$ is defined as

$$r(\tau) = \frac{c_\tau(0,\tau)}{\sqrt{c_\tau(0,0)c_\tau(\tau,\tau)}}$$

where

$$c_\tau(m,n) = \sum_{-\lfloor \tau/2 \rfloor - 80}^{-\lfloor \tau/2 \rfloor + 79} y_{k+m} y_{k+n}.$$

The first estimate of the pitch P_1 is obtained as the value of τ that maximizes the normalized autocorrelation function. This value is refined by looking at the signal filtered using a filter with passband in the 0–500 Hz range. This stage uses two values of P_1, one from the current frame and one from the previous frame, as candidates. The normalized autocorrelation values are obtained for lags from five samples less to five samples more than the candidate P_1 values. The lags that provide the maximum normalized autocorrelation value for each candidate are used for *fractional pitch refinement.* The idea behind fractional pitch refinement is that if the maximum value of $r(\tau)$ is found for some $\tau = T$, then the maximum could be in the interval $(T-1, T]$ or $[T, T+1)$. The fractional offset is computed using

$$\Delta = \frac{c_T(0,T+1)c_T(T,T) - c_T(0,T)c_T(T,T+1)}{c_T(0,T+1)[c_T(T,T) - c_T(T,T+1)] + c_T(0,T)[c_T(T+1,T+1) - c_T(T,T+1)]}. \tag{17.26}$$

The normalized autocorrelation at the fractional pitch values are given by

$$r(T+\Delta) = \frac{(1-\Delta)c_T(0,T) + \Delta c_T(0,T+1)}{\sqrt{c_T(0,0)[(1-\Delta)^2 c_T(T,T) + 2\Delta(1-\Delta)c_T(T,T+1) + \Delta^2 c_T(T+1,T+1)]}}. \tag{17.27}$$

The fractional estimate that gives the higher autocorrelation is selected as the refined pitch value P_2.

The final refinements of the pitch value are obtained using the linear prediction residuals. The residual sequence is generated by filtering the input speech signal with the filter obtained using the LPC analysis. For the purposes of pitch refinement the residual signal is filtered using a low-pass filter with a cutoff of 1 kHz. The normalized autocorrelation function is computed for this filtered residual signal for lags from five samples less to five samples more than the candidate P_2 value, and a candidate value of P_3 is obtained. If $r(P_3) \geq 0.6$, we check to make sure that P_3 is not a multiple of the actual pitch. If $r(P_3) < 0.6$, we do another fractional pitch refinement around P_2 using the input speech signal. If in the end $r(P_3) < 0.55$, we replace P_3 with a long-term average value of the pitch. The final pitch value is quantized on a logarithmic scale using a 99-level uniform quantizer.

The input is also subjected to a multiband voicing analysis using five filters with passbands 0–500, 500–1000, 1000–2000, 2000–3000, and 3000–4000 Hz. The goal of the analysis is to obtain the voicing strengths Vbp_i for each band used in the shaping filters. Noting that P_2 was obtained using the output of the lowest band filter, $r(P_2)$ is assigned as the lowest band voicing strength Vbp_1. For the other bands, Vbp_i is the larger of $r(P_2)$ for that band and the correlation of the envelope of the band-pass signal. If the value of Vbp_1 is small, this indicates a lack of low-frequency structure, which in turn indicates an unvoiced or transition input. Thus, if $Vbp_1 < 0.5$, the pulse component of the excitation signal is selected to be aperiodic, and this decision is communicated to the decoder by setting the aperiodic flag to 1. When $Vbp_1 > 0.6$, the values of the other voicing strengths are quantized to 1 if their value is greater than 0.6, and to 0 otherwise. In this way signal energy in the different bands is turned on or off depending on the voicing strength. There are several exceptions to this quantization rule. If Vbp_2, Vbp_3, and Vbp_4 all have magnitudes less than 0.6 and Vbp_5 has a value greater than 0.6, they are all (including Vbp_5) quantized to 0. Also, if the residual signal contains a few large values, indicating sudden transitions in the input signal, the voicing strengths are adjusted. In particular, the *peakiness* is defined as

$$\text{peakiness} = \frac{\sqrt{\frac{1}{160}\sum_{n=1}^{160} d_n^2}}{\frac{1}{160}\sum_{n=1}^{160} |d_n|}. \tag{17.28}$$

If this value exceeds 1.34, Vbp_1 is forced to 1. If the peakiness value exceeds 1.6, Vbp_1, Vbp_2, and Vbp_3 are all set to 1.

In order to generate the pulse input, the algorithm measures the magnitude of the discrete Fourier transform coefficients corresponding to the first 10 harmonics of the pitch. The prediction residual is generated using the quantized predictor coefficients. The algorithm searches in a window of width $\lfloor 512/\hat{P}_3 \rfloor$ samples around the initial estimates of the pitch harmonics for the actual harmonics where $\hat{P}_3$ is the quantized value of P_3. The magnitudes of the harmonics are quantized using a vector quantizer with a codebook size of 256. The codebook is searched using a weighted Euclidean distance that emphasizes lower frequencies over higher frequencies.

At the decoder, using the magnitudes of the harmonics and information about the periodicity of the pulse train, the algorithm generates one excitation signal. Another signal is generated using a random number generator. Both are shaped by the multiband shaping filter before being combined. This mixture signal is then processed through an *adaptive spectral enhancement filter*, which is based on the LPC coefficients, to form the final excitation signal. Note that in order to preserve continuity from frame to frame, the parameters used for generating the excitation signal are adjusted based on their corresponding values in neighboring frames.

17.4 Wideband Speech Compression—ITU-T G.722.2

One of the earliest forms of (remote) speech communication was over the telephone. This experience set the expectations for quality rather low. When technology advanced, people still did not demand higher quality in their voice communications. However, the multimedia revolution is changing that. With ever-increasing quality in video and audio there is an increasing demand for higher quality in speech communication. Telephone-quality speech is limited to the band between 200 Hz and 3400 Hz. This range of frequency contains enough information to make speech intelligible and provide some degree of speaker identification. To improve the quality of speech, it is necessary to increase the bandwidth of speech. Wideband speech is bandlimited to 50–7000 Hz. The higher frequencies give more clarity to the voice signal while the lower frequencies contribute timbre and naturalness. The ITU-T G.722.2 standard, approved in January of 2002, provides a multirate coder for wideband speech coding.

Wideband speech is sampled at 16,000 samples per second. The signal is split into two bands, a lower band from 50–6400 Hz and a narrow upper band from 6400–7000 Hz. The coding resources are devoted to the lower band. The upper band is reconstructed at the receiver based on information from the lower band and using random excitation. The lower band is downsampled to 12.8 kHz.

The coding method is a code-excited linear prediction method that uses an algebraic codebook as the fixed codebook. The adaptive codebook contains low-pass interpolated past excitation vectors. The basic idea is the same as in CELP. A synthesis filter is derived from the input speech. An excitation vector consisting of a weighted sum of the fixed and adaptive codebooks is used to excite the synthesis filter. The perceptual closeness of the output of the filter to the input speech is used to select the combination of excitation vectors. The selection, along with the parameters of the synthesis filter, is communicated to the receiver, which then synthesizes the speech. A voice activity detector is used to reduce the rate during silence intervals. Let us examine the various components in slightly more detail.

The speech is processed in 20-ms frames. Each frame is composed of four 5-ms subframes. The LP analysis is conducted once per frame using an overlapping 30-ms window. Autocorrelation values are obtained for the windowed speech and the Levinson-Durbin algorithm is used to obtain the LP coefficients. These coefficients are transformed to Immitance Spectral Pairs (ISP), which are quantized using a vector quantizer. The reason behind the transformation is that we will need to quantize whatever representation we have of the synthesis filters. The elements of the ISP representation are uncorrelated if the underlying process is stationary, which means that error in one coefficient will not cause the entire spectrum to get distorted.

Given a set of sixteen LP coefficients $\{a_i\}$, define two polynomials

$$f_1'(z) = A(z) + z^{-16}A(z^{-1}) \tag{17.29}$$

$$f_2'(z) = A(z) - z^{-16}A(z^{-1}) \tag{17.30}$$

Clearly, if we know the polynomials their sum will give us $A(z)$. Instead of sending the polynomials, we can send the roots of these polynomials. These roots are known to all lie on the unit circle, and the roots of the two polynomials alternate. The polynomial $f'_2(z)$ has two roots at $z = 1$ and $z = -1$. These are removed and we get the two polynomials

$$f_1(z) = f'_1(z) \tag{17.31}$$

$$f_2(z) = \frac{f'_2(z)}{1-z^{-2}} \tag{17.32}$$

These polynomials can now be factored as follows

$$f_1(z) = (1+a_{16}) \prod_{i=0,2,\ldots,14} \left(1-2q_i z^{-i}+z^{-2}\right) \tag{17.33}$$

$$f_2(z) = (1+a_{16}) \prod_{i=1,3,\ldots,13} \left(1-2q_i z^{-i}+z^{-2}\right) \tag{17.34}$$

where $q_i = \cos(\omega_i)$ and ω_i are the immitance spectral frequencies. The ISP coefficients are quantized using a combination of differential encoding and vector quantization. The vector of sixteen frequencies is split into subvectors and these vectors are quantized in two stages. The quantized ISPs are transformed to LP coefficients, which are then used in the fourth subframe for synthesis. The ISP coefficients used in the the other three subframes are obtained by interpolating the coefficients in the neighboring subframes.

For each 5-ms subframe we need to generate an excitation vector. As in CELP, the excitation is a sum of vectors from two codebooks, a fixed codebook and an adaptive codebook. One of the problems with vector codebooks has always been the storage requirements. The codebook should be large enough to provide for a rich set of excitations. However, with a dimension of 64 samples (for 5 ms), the number of possible combinations can get enormous. The G.722.2 algorithm solves this problem by imposing an algebraic structure on the fixed codebook. The 64 positions are divided into four tracks. The first track consists of positions $0, 4, 8, \ldots, 60$. The second track consists of the positions $1, 5, 9, \ldots, 61$. The third track consists of positions $2,\ 6,\ 10, \ldots, 62$ and the final track consists of the remaining positions. We can place a single signed pulse in each track by using 4 bits to denote the position and a fifth bit for the sign. This effectively gives us a 20-bit fixed codebook. This corresponds to a codebook size of 2^{20}. However, we do not need to store the codebook. By assigning more or fewer pulses per track we can dramatically change the "size" of the codebook and get different coding rates. The standard details a rapid search procedure to obtain the excitation vectors.

The voice activity detector allows the encoder to significantly reduce the rate during periods of speech pauses. During these periods the background noise is coded at a low rate by transmitting parameters describing the noise. This *comfort noise* is synthesized at the decoder.

17.5 Image Compression

Although there have been a number of attempts to mimic the linear predictive coding approach for image compression, they have not been overly successful. A major reason for this is that while speech can be modeled as the output of a linear filter, most images cannot.

However, a totally different analysis/synthesis approach, conceived in the mid-1980s, has found some degree of success—fractal compression.

17.5.1 Fractal Compression

There are several different ways to approach the topic of fractal compression. Our approach is to use the idea of fixed-point transformation. A function $f(\cdot)$ is said to have a fixed point x_0 if $f(x_0) = x_0$. Suppose we restrict the function $f(\cdot)$ to be of the form $ax+b$. Then, except for when $a = 1$, this equation always has a fixed point:

$$\begin{aligned} ax_0 + b &= x_o \\ \Rightarrow x_0 &= \frac{b}{1-a}. \end{aligned} \tag{17.35}$$

This means that if we wanted to transmit the value of x_0, we could instead transmit the values of a and b and obtain x_0 at the receiver using (17.35). We do not have to solve this equation to obtain x_0. Instead, we could take a guess at what x_0 should be and then refine the guess using the recursion

$$x_0^{(n+1)} = ax_0^{(n)} + b. \tag{17.36}$$

Example 17.5.1:

Suppose that instead of sending the value $x_0 = 2$, we sent the values of a and b as 0.5 and 1.0. The receiver starts out with a guess for x_0 as $x_0^{(0)} = 1$. Then

$$\begin{aligned} x_0^{(1)} &= ax_0^{(0)} + b = 1.5 \\ x_0^{(2)} &= ax_0^{(1)} + b = 1.75 \\ x_0^{(3)} &= ax_0^{(2)} + b = 1.875 \\ x_0^{(4)} &= ax_0^{(3)} + b = 1.9375 \\ x_0^{(5)} &= ax_0^{(4)} + b = 1.96875 \\ x_0^{(6)} &= ax_0^{(5)} + b = 1.984375 \end{aligned} \tag{17.37}$$

and so on. As we can see, with each iteration we come closer and closer to the actual x_0 value of 2. This would be true no matter what our initial guess was. ♦

Thus, the value of x_0 is accurately specified by specifying the fixed-point equation. The receiver can retrieve the value either by the solution of (17.35) or via the recursion (17.36).

Let us generalize this idea. Suppose that for a given image $\mathcal{I}$ (treated as an array of integers), there exists a function $f(\cdot)$ such that $f(\mathcal{I}) = \mathcal{I}$. If it was cheaper in terms of bits to represent $f(\cdot)$ than it was to represent $\mathcal{I}$, we could treat $f(\cdot)$ as the compressed representation of $\mathcal{I}$.

This idea was first proposed by Michael Barnsley and Alan Sloan [238] based on the idea of self-similarity. Barnsley and Sloan noted that certain natural-looking objects can be obtained as the fixed point of a certain type of function. If an image can be obtained as a fixed point of some function, can we then solve the *inverse* problem? That is, given an image, can we find the function for which the image is the fixed point? The first practical public answer to this came from Arnaud Jacquin in his Ph.D. dissertation [239] in 1989. The technique we describe in this section is from Jacquin's 1992 paper [240].

Instead of generating a single function directly for which the given image is a fixed point, we partition the image into blocks R_k, called *range* blocks, and obtain a transformation f_k for each block. The transformations f_k are not fixed-point transformations since they do not satisfy the equation

$$f_k(R_k) = R_k. \tag{17.38}$$

Instead, they are a mapping from a block of pixels D_k from some other part of the image. While each individual mapping f_k is not a fixed-point mapping, we will see later that we can combine all these mappings to generate a fixed-point mapping. The image blocks D_k are called *domain* blocks, and they are chosen to be larger than the range blocks. In [240], the domain blocks are obtained by sliding a $K \times K$ window over the image in steps of $K/2$ or $K/4$ pixels. As long as the window remains within the boundaries of the image, each $K \times K$ block thus encountered is entered into the domain pool. The set of all domain blocks does not have to partition the image. In Figure 17.10 we show the range blocks and two possible domain blocks.

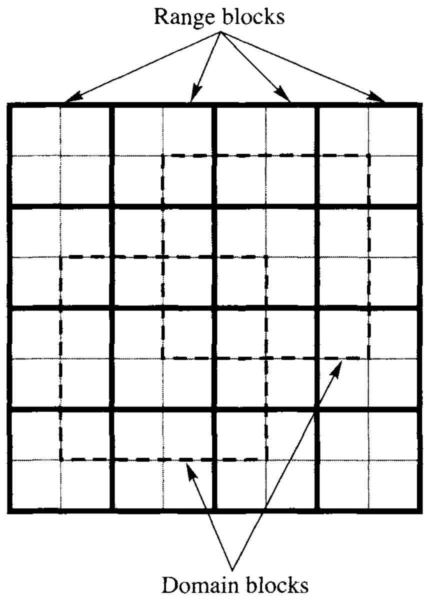

FIGURE 17.10 Range blocks and examples of domain blocks.

The transformations f_k are composed of a *geometric* transformation g_k and a *massic* transformation m_k. The geometric transformation consists of moving the domain block to the location of the range block and adjusting the size of the domain block to match the size of the range block. The massic transformation adjusts the intensity and orientation of the pixels in the domain block after it has been operated on by the geometric transform. Thus,

$$\hat{R}_k = f_k(D_k) = m_k(g_k(D_k)). \tag{17.39}$$

We have used $\hat{R}_k$ instead of R_k on the left-hand side of (17.39) because it is generally not possible to find an exact functional relationship between domain and range blocks. Therefore, we have to settle for some degree of loss of information. Generally, this loss is measured in terms of mean squared error.

The effect of all these functions together can be represented as the transformation $f(\cdot)$. Mathematically, this transformation can be viewed as a union of the transformations f_k:

$$f = \bigcup_k f_k. \tag{17.40}$$

Notice that while each transformation f_k maps a block of different size and location to the location of R_k, looking at it from the point of view of the entire image, it is a mapping from the image to the image. As the union of R_k is the image itself, we could represent all the transformations as

$$\hat{I} = f(\hat{I}) \tag{17.41}$$

where we have used $\hat{I}$ instead of I to account for the fact that the reconstructed image is an approximation to the original.

We can now pose the encoding problem as that of obtaining D_k, g_k, and m_k such that the difference $d(R_k, \hat{R}_k)$ is minimized, where $d(R_k, \hat{R}_k)$ can be the mean squared error between the blocks R_k and $\hat{R}_k$.

Let us first look at how we would obtain g_k and m_k assuming that we already know which domain block D_k we are going to use. We will then return to the question of selecting D_k.

Knowing which domain block we are using for a given range block automatically specifies the amount of displacement required. If the range blocks R_k are of size $M \times M$, then the domain blocks are usually taken to be of size $2M \times 2M$. In order to adjust the size of D_k to be the same as that of R_k, we generally replace each 2×2 block of pixels with their average value. Once the range block has been selected, the geometric transformation is easily obtained.

Let's define $T_k = g_k(D_k)$, and t_{ij} as the ijth pixel in T_k $i, j = 0, 1, \ldots, M-1$. The massic transformation m_k is then given by

$$m_k(t_{ij}) = i(\alpha_k t_{ij} + \Delta_k) \tag{17.42}$$

where $i(\cdot)$ denotes a shuffling or rearrangement of the pixels with the block. Possible rearrangements (or *isometries*) include the following:

1. Rotation by 90 degrees, $i(t_{ij}) = t_{j(M-1-i)}$
2. Rotation by 180 degrees, $i(t_{ij}) = t_{(M-1-i)(M-1-j)}$

3. Rotation by -90 degrees, $i(t_{ij}) = t_{(M-1-i)j}$
4. Reflection about midvertical axis, $i(t_{ij}) = t_{i(M-1-j)}$
5. Reflection about midhorizontal axis, $i(t_{ij}) = t_{(M-1-i)j}$
6. Reflection about diagonal, $i(t_{ij}) = t_{ji}$
7. Reflection about cross diagonal, $i(t_{ij}) = t_{(M-1-j)(M-1-i)}$
8. Identity mapping, $i(t_{ij}) = t_{ij}$

Therefore, for each massic transformation m_k, we need to find values of α_k, Δ_k, and an isometry. For a given range block R_k, in order to find the mapping that gives us the closest approximation $\hat{R}_k$, we can try all possible combinations of transformations and domain blocks—a massive computation. In order to reduce the computations, we can restrict the number of domain blocks to search. However, in order to get the best possible approximation, we would like the pool of domain blocks to be as large as possible. Jacquin [240] resolves this situation in the following manner. First, he generates a relatively large pool of domain blocks by the method described earlier. The elements of the domain pool are then divided into *shade blocks, edge blocks,* and *midrange blocks*. The shade blocks are those in which the variance of pixel values within the block is small. The edge block, as the name implies, contains those blocks that have a sharp change of intensity values. The midrange blocks are those that fit into neither category—not too smooth but with no well-defined edges. The shade blocks are then removed from the domain pool. The reason is that, given the transformations we have described, a shade domain block can only generate a shade range block. If the range block is a shade block, it is much more cost effective simply to send the average value of the block rather than attempt any more complicated transformations.

The encoding procedure proceeds as follows. A range block is first classified into one of the three categories described above. If it is a shade block, we simply send the average value of the block. If it is a midrange block, the massic transformation is of the form $\alpha_k t_{ij} + \Delta_k$. The isometry is assumed to be the identity isometry. First α_k is selected from a small set of values—Jacquin [240] uses the values (0.7, 0.8, 0.9, 1.0)—such that $d(R_k, \alpha_k T_k)$ is minimized. Thus, we have to search over the possible values of α and the midrange domain blocks in the domain pool in order to find the (α_k, D_k) pair that will minimize $d(R_k, \alpha_k T_k)$. The value of Δ_k is then selected as the difference of the average values of R_k and $\alpha_k T_k$.

If the range block R_k is classified as an edge block, selection of the massic transformation is a somewhat more complicated process. The block is first divided into a bright and a dark region. The dynamic range of the block $r_d(R_k)$ is then computed as the difference of the average values of the light and dark regions. For a given domain block, this is then used to compute the value of α_k by

$$\alpha_k = \min\left\{\frac{r_d(R_k)}{r_d(T_j)}, \alpha_{\max}\right\} \tag{17.43}$$

where $\alpha_{\max}$ is an upper bound on the scaling factor. The value of α_k obtained in this manner is then quantized to one of a small set of values. Once the value of α_k has been obtained, Δ_k is obtained as the difference of either the average values of the bright regions or the average values of the dark regions, depending on whether we have more pixels in the dark regions or the light regions. Finally, each of the isometries is tried out to find the one that gives the closest match between the transformed domain block and the range block.

Once the transformations have been obtained, they are communicated to the receiver in terms of the following parameters: the location of the selected domain block and a single bit denoting whether the block is a shade block or not. If it is a shade block, the average intensity value is transmitted; if it is not, the quantized scale factor and offset are transmitted along with the label of the isometry used.

The receiver starts out with some arbitrary initial image I_0. The transformations are then applied for each of the range blocks to obtain the first approximation. Then the transformations are applied to the first approximation to get the second approximation, and so on. Let us see an example of the decoding process.

Example 17.5.2:

The image Elif, shown in Figure 17.11, was encoded using the fractal approach. The original image was of size 256×256, and each pixel was coded using 8 bits. Therefore, the storage space required was 65,536 bytes. The compressed image consisted of the transformations described above. The transformations required a total of 4580 bytes, which translates to an average rate of 0.56 bits per pixel. The decoding process started with the transformations being applied to an all-zero image. The first six iterations of the decoding process are shown in Figure 17.12. The process converged in nine iterations. The final image is shown in Figure 17.13. Notice the difference in this reconstructed image and the low-rate reconstructed image obtained using the DCT. The blocking artifacts are for the most part gone. However,

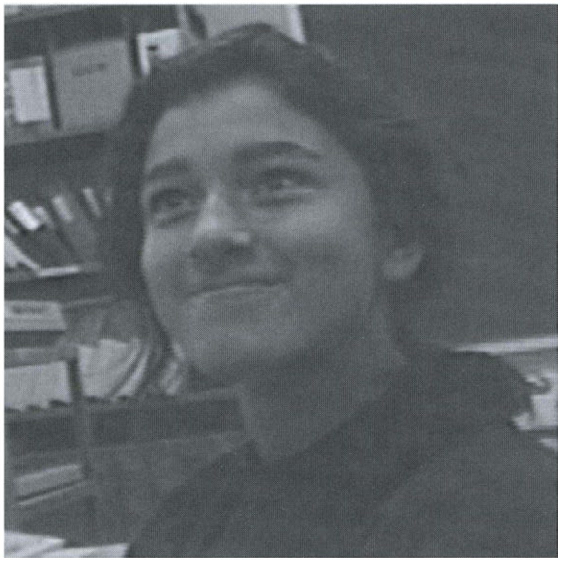

FIGURE 17.11 **Original Elif image.**

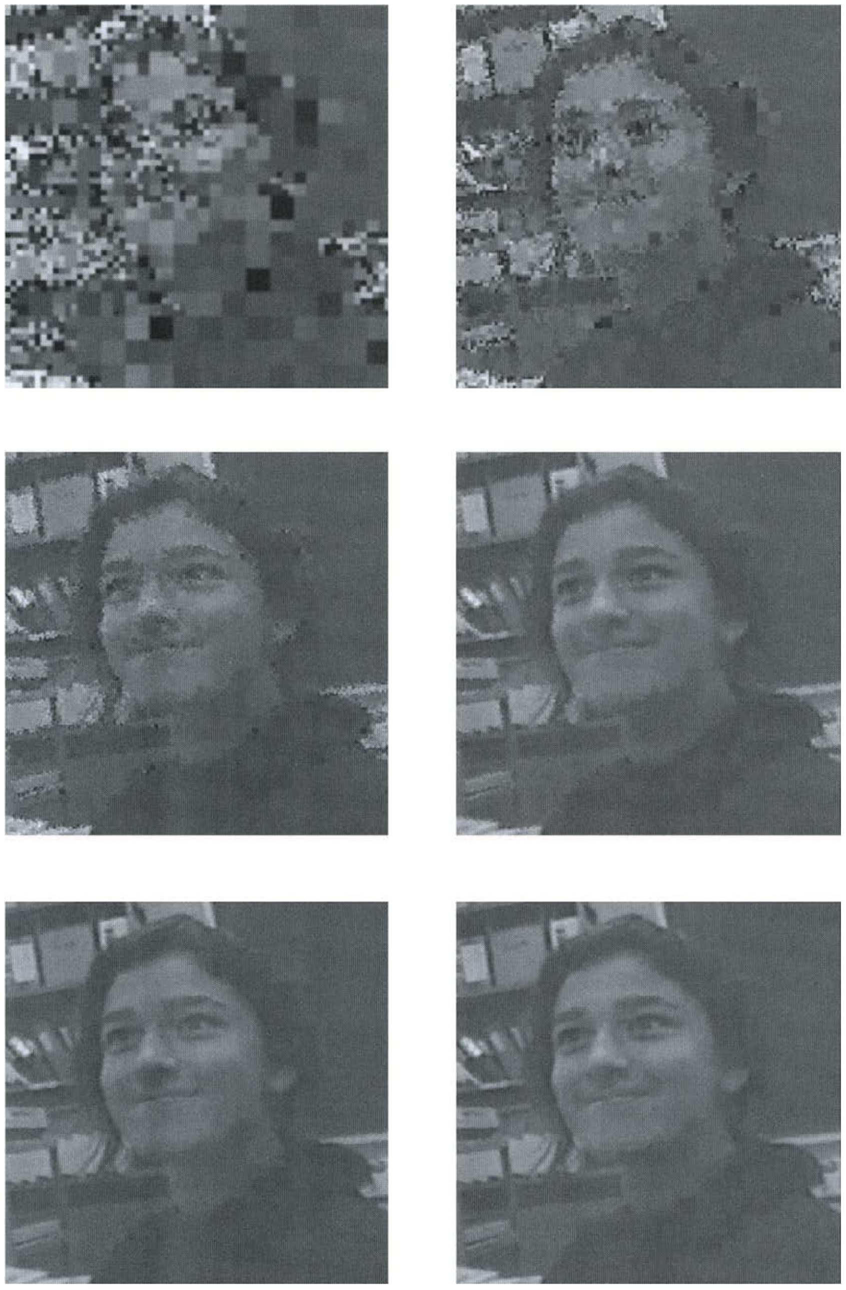

FIGURE 17. 12 The first six iterations of the fractal decoding process.

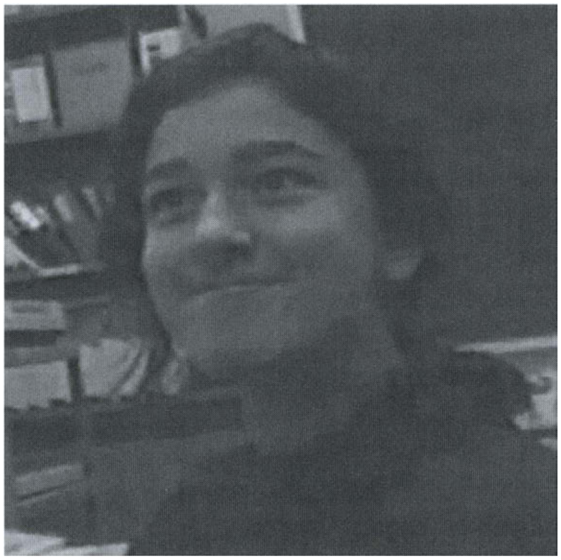

FIGURE 17.13 Final reconstructed Elif image.

this does not mean that the reconstruction is free of distortions and artifacts. They are especially visible in the chin and neck region. ♦

In our discussion (and illustration) we have assumed that the size of the range blocks is constant. If so, how large should we pick the range block? If we pick the size of the range block to be large, we will have to send fewer transformations, thus improving the compression. However, if the size of the range block is large, it becomes more difficult to find a domain block that, after appropriate transformation, will be close to the range block, which in turn will increase the distortion in the reconstructed image. One compromise between picking a large or small value for the size of the range block is to start out with a large size and, if a good enough match is not found, to progressively reduce the size of the range block until we have either found a good match or reached a minimum size. We could also compute a weighted sum of the rate and distortion

$$J = D + \beta R$$

where D is a measure of the distortion, and R represents the number of bits required to represent the block. We could then either subdivide or not depending on the value of J.

We can also start out with range blocks that have the minimum size (also called the *atomic blocks*) and obtain larger blocks via merging smaller blocks.

There are a number of ways in which we can perform the subdivision. The most commonly known approach is *quadtree partitioning*, initially introduced by Samet [241]. In quadtree partitioning we start by dividing up the image into the maximum-size range

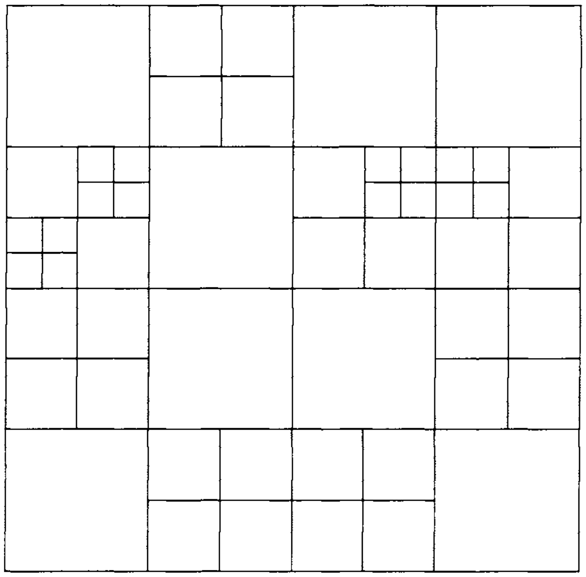

FIGURE 17. 14 **An example of quadtree partitioning.**

blocks. If a particular block does not have a satisfactory reconstruction, we can divide it up into four blocks. These blocks in turn can also, if needed, be divided into four blocks. An example of quadtree partitioning can be seen in Figure 17.14. In this particular case there are three possible sizes for the range blocks. Generally, we would like to keep the minimum size of the range block small if fine detail in the image is of greater importance [242]. Since we have multiple sizes for the range blocks, we also need multiple sizes for the domain blocks.

Quadtree partitioning is not the only method of partitioning available. Another popular method of partitioning is the HV method. In this method we allow the use of rectangular regions. Instead of dividing a square region into four more square regions, rectangular regions are divided either vertically or horizontally in order to generate more homogeneous regions. In particular, if there are vertical or horizontal edges in a block, it is partitioned along these edges. One way to obtain the locations of partitions for a given $M \times N$ range block is to calculate the biased vertical and horizontal differences:

$$v_i = \frac{\min(i, N-i-1)}{N-1}\left(\sum_j \mathcal{I}_{i,j} - \sum_j \mathcal{I}_{i+1,j}\right)$$

$$h_j = \frac{\min(j, M-j-1)}{M-1}\left(\sum_j \mathcal{I}_{i,j} - \sum_j \mathcal{I}_{i,j+1}\right).$$

The values of i and j for which $|v_i|$ and $|h_j|$ are the largest indicate the row and column for which there is maximum difference between two halves of the block. Depending on whether $|v_i|$ or $|h_j|$ is larger, we can divide the rectangle either vertically or horizontally.

Finally, partitioning does not have to be rectangular, or even regular. People have experimented with triangle partitions as well as irregular-shaped partitions [243].

The fractal approach is a novel way of looking at image compression. At present the quality of the reconstructions using the fractal approach is about the same as the quality of the reconstruction using the DCT approach employed in JPEG. However, the fractal technique is relatively new, and further research may bring significant improvements. The fractal approach has one significant advantage: decoding is simple and fast. This makes it especially useful in applications where compression is performed once and decompression is performed many times.

17.6 Summary

We have looked at two very different ways of using the analysis/synthesis approach. In speech coding the approach works because of the availability of a mathematical model for the speech generation process. We have seen how this model can be used in a number of different ways, depending on the constraints of the problem. Where the primary objective is to achieve intelligible communication at the lowest rate possible, the LPC algorithm provides a very nice solution. If we also want the quality of the speech to be high, CELP and the different sinusoidal techniques provide higher quality at the cost of more complexity and processing delay. If delay also needs to be kept below a threshold, one particular solution is the low-delay CELP algorithm in the G.728 recommendation. For images, fractal coding provides a very different way to look at the problem. Instead of using the physical structure of the system to generate the source output, it uses a more abstract view to obtain an analysis/synthesis technique.

Further Reading

1. For information about various aspects of speech processing, *Voice and Speech Processing,* by T. Parsons [105], is a \ery readable source.
2. The classic tutorial on linear prediction is "Linear Prediction," by J. Makhoul [244], which appeared in the April 1975 issue of the *Proceedings of the IEEE.*
3. For a thorough review of recent activity in speech compression, see "Advances in Speech and Audio Compression," by A. Gersho [220], which appeared in the June 1994 issue of the *Proceedings of the IEEE.*
4. An excellent source for information about speech coders is *Digital Speech: Coding for Low Bit Rate Communication Systems*, by A. Kondoz [127].
5. An excellent description of the G.728 algorithm can be found in "A Low Delay CELP Coder for the CCITT 16 kb/s Speech Coding Standard," by J.-H. Chen, R.V. Cox,

Y.-C. Lin, N. Jayant, and M.J. Melchner [235], in the June 1992 issue of the *IEEE Journal on Selected Areas in Communications*.

6. A good introduction to fractal image compression is *Fractal Image Compression: Theory and Application*, Y. Fisher (ed.) [242], New York: Springer-Verlag, 1995.

7. The October 1993 issue of the *Proceedings of the IEEE* contains a special section on fractals with a tutorial on fractal image compression by A. Jacquin.

17.7 Projects and Problems

1. Write a program for the detection of voiced and unvoiced segments using the AMDF function. Test your algorithm on the `test.snd` sound file.

2. The `testf.raw` file is a female voice saying the word *test*. Isolate 10 voiced and unvoiced segments from the `testm.raw` file and the `testf.snd` file. (Try to pick the same segments in the two files.) Compute the number of zero crossings in each segment and compare your results for the two files.

3. **(a)** Select a voiced segment from the `testf.raw` file. Find the fourth-, sixth-, and tenth-order LPC filters for this segment using the Levinson-Durbin algorithm.

   (b) Pick the corresponding segment from the `testf.snd` file. Find the fourth-, sixth-, and tenth-order LPC filters for this segment using the Levinson-Durbin algorithm.

   (c) Compare the results of (a) and (b).

4. Select a voiced segment from the `test.raw` file. Find the fourth-, sixth-, and tenth-order LPC filters for this segment using the Levinson-Durbin algorithm. For each of the filters, find the multipulse sequence that results in the closest approximation to the voiced signal.