11

11.1 Overview

Sources such as speech and images have a great deal of correlation from sample to sample. We can use this fact to predict each sample based on its past and only encode and transmit the differences between the prediction and the sample value. Differential encoding schemes are built around this premise. Because the prediction techniques are rather simple, these schemes are much easier to implement than other compression schemes. In this chapter, we will look at various components of differential encoding schemes and study how they are used to encode sources—in particular, speech. We will also look at a widely used international differential encoding standard for speech encoding.

11.2 Introduction

In the last chapter we looked at vector quantization—a rather complex scheme requiring a significant amount of computational resources—as one way of taking advantage of the structure in the data to perform lossy compression. In this chapter, we look at a different approach that uses the structure in the source output in a slightly different manner, resulting in a significantly less complex system.

When we design a quantizer for a given source, the size of the quantization interval depends on the variance of the input. If we assume the input is uniformly distributed, the variance depends on the dynamic range of the input. In turn, the size of the quantization interval determines the amount of quantization noise incurred during the quantization process.

In many sources of interest, the sampled source output $\{x_n\}$ does not change a great deal from one sample to the next. This means that both the dynamic range and the variance of the sequence of differences $\{d_n = x_n - x_{n-1}\}$ are significantly smaller than that of the source output sequence. Furthermore, for correlated sources the distribution of d_n is highly peaked

at zero. We made use of this skew, and resulting loss in entropy, for the lossless compression of images in Chapter 7. Given the relationship between the variance of the quantizer input and the incurred quantization error, it is also useful, in terms of lossy compression, to look at ways to encode the difference from one sample to the next rather than encoding the actual sample value. Techniques that transmit information by encoding differences are called *differential encoding techniques.*

Example 11.2.1:

Consider the half cycle of a sinusoid shown in Figure 11.1 that has been sampled at the rate of 30 samples per cycle. The value of the sinusoid ranges between 1 and -1. If we wanted to quantize the sinusoid using a uniform four-level quantizer, we would use a step size of 0.5, which would result in quantization errors in the range $[-0.25, 0.25]$. If we take the sample-to-sample differences (excluding the first sample), the differences lie in the range $[-0.2, 0.2]$. To quantize this range of values with a four-level quantizer requires a step size of 0.1, which results in quantization noise in the range $[-0.05, 0.05]$.

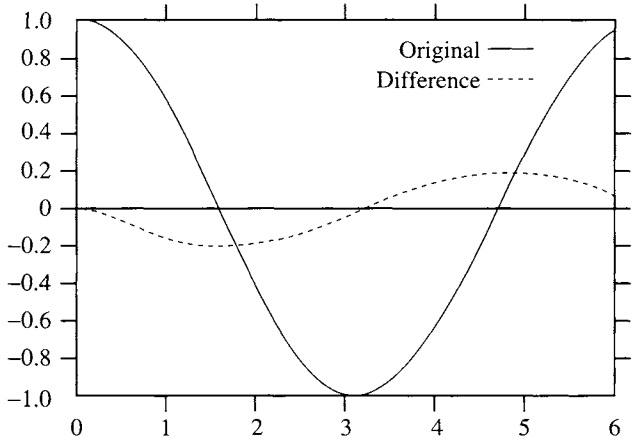

FIGURE 11.1 Sinusoid and sample-to-sample differences. ♦

The sinusoidal signal in the previous example is somewhat contrived. However, if we look at some of the real-world sources that we want to encode, we see that the dynamic range that contains most of the differences is significantly smaller than the dynamic range of the source output.

Example 11.2.2:

Figure 11.2 is the histogram of the Sinan image. Notice that the pixel values vary over almost the entire range of 0 to 255. To represent these values exactly, we need 8 bits per

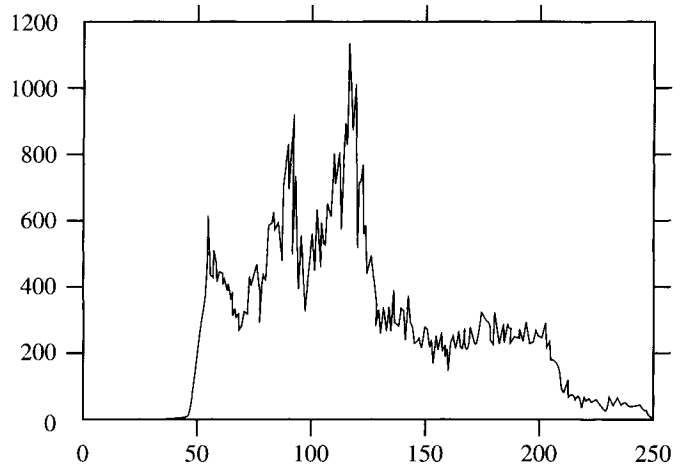

FIGURE 11.2 Histogram of the Sinan image.

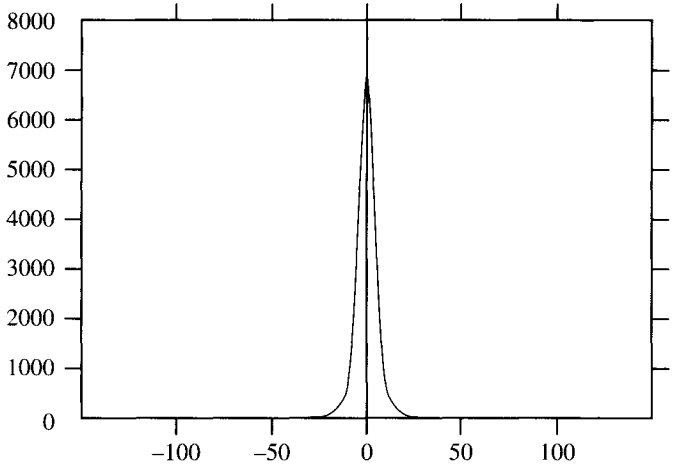

FIGURE 11.3 Histogram of pixel-to-pixel differences of the Sinan image.

pixel. To represent these values in a lossy manner to within an error in the least significant bit, we need 7 bits per pixel. Figure 11.3 is the histogram of the differences.

More than 99% of the pixel values lie in the range −31 to 31. Therefore, if we were willing to accept distortion in the least significant bit, for more than 99% of the difference values we need 5 bits per pixel rather than 7. In fact, if we were willing to have a small percentage of the differences with a larger error, we could get by with 4 bits for each difference value. ♦

In both examples, we have shown that the dynamic range of the differences between samples is substantially less than the dynamic range of the source output. In the following sections we describe encoding schemes that take advantage of this fact to provide improved compression performance.

11.3 The Basic Algorithm

Although it takes fewer bits to encode differences than it takes to encode the original pixel, we have not said whether it is possible to recover an acceptable reproduction of the original sequence from the quantized difference value. When we were looking at lossless compression schemes, we found that if we encoded and transmitted the first value of a sequence, followed by the encoding of the differences between samples, we could losslessly recover the original sequence. Unfortunately, a strictly analogous situation does not exist for lossy compression.

Example 11.3.1:

Suppose a source puts out the sequence

$$6.2\ 9.7\ 13.2\ 5.9\ 8\ 7.4\ 4.2\ 1.8$$

We could generate the following sequence by taking the difference between samples (assume that the first sample value is zero):

$$6.2\ 3.5\ 3.5\ -7.3\ 2.1\ -0.6\ -3.2\ -2.4$$

If we losslessly encoded these values, we could recover the original sequence at the receiver by adding back the difference values. For example, to obtain the second reconstructed value, we add the difference 3.5 to the first received value 6.2 to obtain a value of 9.7. The third reconstructed value can be obtained by adding the received difference value of 3.5 to the second reconstructed value of 9.7, resulting in a value of 13.2, which is the same as the third value in the original sequence. Thus, by adding the nth received difference value to the $(n-1)$th reconstruction value, we can recover the original sequence exactly.

Now let us look at what happens if these difference values are encoded using a lossy scheme. Suppose we had a seven-level quantizer with output values $-6, -4, -2, 0, 2, 4, 6$. The quantized sequence would be

$$6\ 4\ 4\ -6\ 2\ 0\ -4\ -2$$

If we follow the same procedure for reconstruction as we did for the lossless compression scheme, we get the sequence

$$6\ 10\ 14\ 8\ 10\ 10\ 6\ 4$$

The difference or error between the original sequence and the reconstructed sequence is

$$0.2\ -0.3\ -0.8\ -2.1\ -2\ -2.6\ -1.8\ -2.2$$

Notice that initially the magnitudes of the error are quite small (0.2, 0.3). As the reconstruction progresses, the magnitudes of the error become significantly larger (2.6, 1.8, 2.2). ♦

To see what is happening, consider a sequence $\{x_n\}$. A difference sequence $\{d_n\}$ is generated by taking the differences $x_n - x_{n-1}$. This difference sequence is quantized to obtain the sequence $\{\hat{d}_n\}$:

$$\hat{d}_n = Q[d_n] = d_n + q_n$$

where q_n is the quantization error. At the receiver, the reconstructed sequence $\{\hat{x}_n\}$ is obtained by adding $\hat{d}_n$ to the previous reconstructed value $\hat{x}_{n-1}$:

$$\hat{x}_n = \hat{x}_{n-1} + \hat{d}_n.$$

Let us assume that both transmitter and receiver start with the same value x_0, that is, $\hat{x}_0 = x_0$. Follow the quantization and reconstruction process for the first few samples:

$$d_1 = x_1 - x_0 \tag{11.1}$$

$$\hat{d}_1 = Q[d_1] = d_1 + q_1 \tag{11.2}$$

$$\hat{x}_1 = x_0 + \hat{d}_1 = x_0 + d_1 + q_1 = x_1 + q_1 \tag{11.3}$$

$$d_2 = x_2 - x_1 \tag{11.4}$$

$$\hat{d}_2 = Q[d_2] = d_2 + q_2 \tag{11.5}$$

$$\hat{x}_2 = \hat{x}_1 + \hat{d}_2 = x_1 + q_1 + d_2 + q_2 \tag{11.6}$$

$$= x_2 + q_1 + q_2. \tag{11.7}$$

Continuing this process, at the nth iteration we get

$$\hat{x}_n = x_n + \sum_{k=1}^{n} q_k. \tag{11.8}$$

We can see that the quantization error accumulates as the process continues. Theoretically, if the quantization error process is zero mean, the errors will cancel each other out in the long run. In practice, often long before that can happen, the finite precision of the machines causes the reconstructed value to overflow.

Notice that the encoder and decoder are operating with different pieces of information. The encoder generates the difference sequence based on the original sample values, while the decoder adds back the quantized difference onto a distorted version of the original signal. We can solve this problem by forcing both encoder and decoder to use the same information during the differencing and reconstruction operations. The only information available to the receiver about the sequence $\{x_n\}$ is the reconstructed sequence $\{\hat{x}_n\}$. As this information is also available to the transmitter, we can modify the differencing operation to use the reconstructed value of the previous sample, instead of the previous sample itself, that is,

$$d_n = x_n - \hat{x}_{n-1}. \tag{11.9}$$

Using this new differencing operation, let's repeat our examination of the quantization and reconstruction process. We again assume that $\hat{x}_0 = x_0$.

$$d_1 = x_1 - x_0 \tag{11.10}$$

$$\hat{d}_1 = Q[d_1] = d_1 + q_1 \tag{11.11}$$

$$\hat{x}_1 = x_0 + \hat{d}_1 = x_0 + d_1 + q_1 = x_1 + q_1 \tag{11.12}$$

$$d_2 = x_2 - \hat{x}_1 \tag{11.13}$$

$$\hat{d}_2 = Q[d_2] = d_2 + q_2 \tag{11.14}$$

$$\hat{x}_2 = \hat{x}_1 + \hat{d}_2 = \hat{x}_1 + d_2 + q_2 \tag{11.15}$$

$$= x_2 + q_2 \tag{11.16}$$

At the nth iteration we have

$$\hat{x}_n = x_n + q_n, \tag{11.17}$$

and there is no accumulation of the quantization noise. In fact, the quantization noise in the nth reconstructed sequence is the quantization noise incurred by the quantization of the nth difference. The quantization error for the difference sequence is substantially less than the quantization error for the original sequence. Therefore, this procedure leads to an overall reduction of the quantization error. If we are satisfied with the quantization error for a given number of bits per sample, then we can use fewer bits with a differential encoding procedure to attain the same distortion.

Example 11.3.2:

Let us try to quantize and then reconstruct the sinusoid of Example 11.2.1 using the two different differencing approaches. Using the first approach, we get a dynamic range of

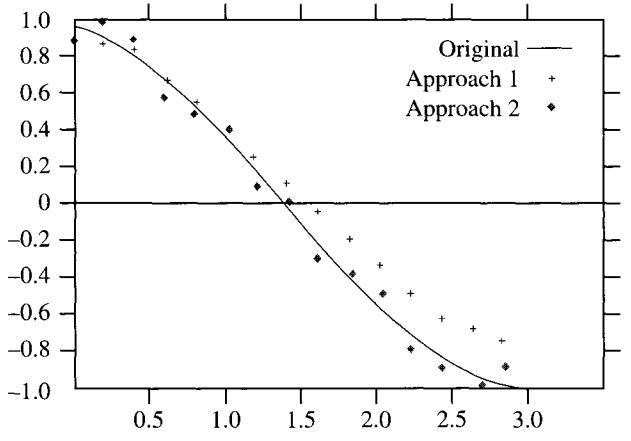

FIGURE 11.4 Sinusoid and reconstructions.

differences from −0.2 to 0.2. Therefore, we use a quantizer step size of 0.1. In the second approach, the differences lie in the range [−0.4, 0.4]. In order to cover this range, we use a step size in the quantizer of 0.2. The reconstructed signals are shown in Figure 11.4.

Notice in the first case that the reconstruction diverges from the signal as we process more and more of the signal. Although the second differencing approach uses a larger step size, this approach provides a more accurate representation of the input. ♦

A block diagram of the differential encoding system as we have described it to this point is shown in Figure 11.5. We have drawn a dotted box around the portion of the encoder that mimics the decoder. The encoder must mimic the decoder in order to obtain a copy of the reconstructed sample used to generate the next difference.

We would like our difference value to be as small as possible. For this to happen, given the system we have described to this point, $\hat{x}_{n-1}$ should be as close to x_n as possible. However, $\hat{x}_{n-1}$ is the reconstructed value of x_{n-1}; therefore, we would like $\hat{x}_{n-1}$ to be close to x_{n-1}. Unless x_{n-1} is always very close to x_n, some function of past values of the reconstructed sequence can often provide a better prediction of x_n. We will look at some of these *predictor* functions later in this chapter. For now, let's modify Figure 11.5 and replace the delay block with a predictor block to obtain our basic differential encoding system as shown in Figure 11.6. The output of the predictor is the prediction sequence $\{p_n\}$ given by

$$p_n = f(\hat{x}_{n-1}, \hat{x}_{n-2}, \ldots, \hat{x}_0). \tag{11.18}$$

This basic differential encoding system is known as the differential pulse code modulation (DPCM) system. The DPCM system was developed at Bell Laboratories a few years after World War II [164]. It is most popular as a speech-encoding system and is widely used in telephone communications.

As we can see from Figure 11.6, the DPCM system consists of two major components, the predictor and the quantizer. The study of DPCM is basically the study of these two components. In the following sections, we will look at various predictor and quantizer designs and see how they function together in a differential encoding system.

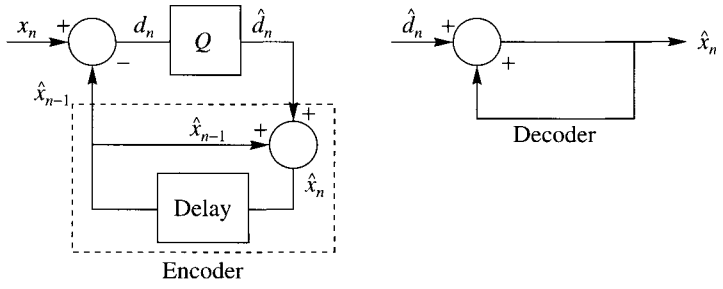

FIGURE 11.5 A simple differential encoding system.

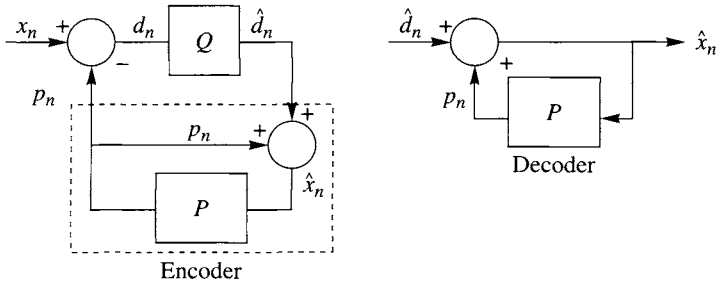

FIGURE 11.6 The basic algorithm.

11.4 Prediction in DPCM

Differential encoding systems like DPCM gain their advantage by the reduction in the variance and dynamic range of the difference sequence. How much the variance is reduced depends on how well the predictor can predict the next symbol based on the past reconstructed symbols. In this section we will mathematically formulate the prediction problem. The analytical solution to this problem will give us one of the more widely used approaches to the design of the predictor. In order to follow this development, some familiarity with the mathematical concepts of expectation and correlation is needed. These concepts are described in Appendix A.

Define σ_d^2, the variance of the difference sequence, as

$$\sigma_d^2 = E[(x_n - p_n)^2] \tag{11.19}$$

where $E[]$ is the expectation operator. As the predictor outputs p_n are given by (11.18), the design of a good predictor is essentially the selection of the function $f(\cdot)$ that minimizes σ_d^2. One problem with this formulation is that $\hat{x}_n$ is given by

$$\hat{x}_n = x_n + q_n$$

and q_n depends on the variance of d_n. Thus, by picking $f(\cdot)$, we affect σ_d^2, which in turn affects the reconstruction $\hat{x}_n$, which then affects the selection of $f(\cdot)$. This coupling makes an explicit solution extremely difficult for even the most well-behaved source [165]. As most real sources are far from well behaved, the problem becomes computationally intractable in most applications.

We can avoid this problem by making an assumption known as the *fine quantization assumption*. We assume that quantizer step sizes are so small that we can replace $\hat{x}_n$ by x_n, and therefore

$$p_n = f(x_{n-1}, x_{n-2}, \ldots, x_0). \tag{11.20}$$

Once the function $f(\cdot)$ has been found, we can use it with the reconstructed values $\hat{x}_n$ to obtain p_n. If we now assume that the output of the source is a stationary process, from the study of random processes [166], we know that the function that minimizes σ_d^2

is the conditional expectation $E[x_n \mid x_{n-1}, x_{n-2}, \ldots, x_0]$. Unfortunately, the assumption of stationarity is generally not true, and even if it were, finding this conditional expectation requires the knowledge of nth-order conditional probabilities, which would generally not be available.

Given the difficulty of finding the best solution, in many applications we simplify the problem by restricting the predictor function to be linear. That is, the prediction p_n is given by

$$p_n = \sum_{i=1}^{N} a_i \hat{x}_{n-i}. \tag{11.21}$$

The value of N specifies the order of the predictor. Using the fine quantization assumption, we can now write the predictor design problem as follows: Find the $\{a_i\}$ so as to minimize σ_d^2.

$$\sigma_d^2 = E\left[\left(x_n - \sum_{i=1}^{N} a_i x_{n-i}\right)^2\right] \tag{11.22}$$

where we assume that the source sequence is a realization of a real valued wide sense stationary process. Take the derivative of σ_d^2 with respect to each of the a_i and set this equal to zero. We get N equations and N unknowns:

$$\frac{\delta\sigma_d^2}{\delta a_1} = -2E\left[\left(x_n - \sum_{i=1}^{N} a_i x_{n-i}\right) x_{n-1}\right] = 0 \tag{11.23}$$

$$\frac{\delta\sigma_d^2}{\delta a_2} = -2E\left[\left(x_n - \sum_{i=1}^{N} a_i x_{n-i}\right) x_{n-2}\right] = 0 \tag{11.24}$$

$$\vdots \quad \vdots$$

$$\frac{\delta\sigma_d^2}{\delta a_N} = -2E\left[\left(x_n - \sum_{i=1}^{N} a_i x_{n-i}\right) x_{n-N}\right] = 0. \tag{11.25}$$

Taking the expectations, we can rewrite these equations as

$$\sum_{i=1}^{N} a_i R_{xx}(i-1) = R_{xx}(1) \tag{11.26}$$

$$\sum_{i=1}^{N} a_i R_{xx}(i-2) = R_{xx}(2) \tag{11.27}$$

$$\vdots \quad \vdots$$

$$\sum_{i=1}^{N} a_i R_{xx}(i-N) = R_{xx}(N) \tag{11.28}$$

where $R_{xx}(k)$ is the autocorrelation function of x_n:

$$R_{xx}(k) = E[x_n x_{n+k}]. \tag{11.29}$$

We can write these equations in matrix form as

$$\mathbf{RA} = \mathbf{P} \tag{11.30}$$

where

$$\mathbf{R} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) & R_{xx}(2) & \cdots & R_{xx}(N-1) \\ R_{xx}(1) & R_{xx}(0) & R_{xx}(1) & \cdots & R_{xx}(N-2) \\ R_{xx}(2) & R_{xx}(1) & R_{xx}(0) & \cdots & R_{xx}(N-3) \\ \vdots & \vdots & & & \vdots \\ R_{xx}(N-1) & R_{xx}(N-2) & R_{xx}(N-3) & \cdots & R_{xx}(0) \end{bmatrix} \tag{11.31}$$

$$\mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \end{bmatrix} \tag{11.32}$$

$$\mathbf{P} = \begin{bmatrix} R_{xx}(1) \\ R_{xx}(2) \\ R_{xx}(3) \\ \vdots \\ R_{xx}(N) \end{bmatrix} \tag{11.33}$$

where we have used the fact that $R_{xx}(-k) = R_{xx}(k)$ for real valued wide sense stationary processes. These equations are referred to as the discrete form of the Wiener-Hopf equations. If we know the autocorrelation values $\{R_{xx}(k)\}$ for $k = 0, 1, \ldots, N$, then we can find the predictor coefficients as

$$\mathbf{A} = \mathbf{R}^{-1}\mathbf{P}. \tag{11.34}$$

Example 11.4.1:

For the speech sequence shown in Figure 11.7, let us find predictors of orders one, two, and three and examine their performance. We begin by estimating the autocorrelation values from the data. Given M data points, we use the following average to find the value for $R_{xx}(k)$:

$$R_{xx}(k) = \frac{1}{M-k} \sum_{i=1}^{M-k} x_i x_{i+k}. \tag{11.35}$$

Using these autocorrelation values, we obtain the following coefficients for the three different predictors. For $N = 1$, the predictor coefficient is $a_1 = 0.66$; for $N = 2$, the coefficients are $a_1 = 0.596$, $a_2 = 0.096$; and for $N = 3$, the coefficients are $a_1 = 0.577$, $a_2 = -0.025$, and $a_3 = 0.204$. We used these coefficients to generate the residual sequence. In order to see the reduction in variance, we computed the ratio of the source output variance to the variance of

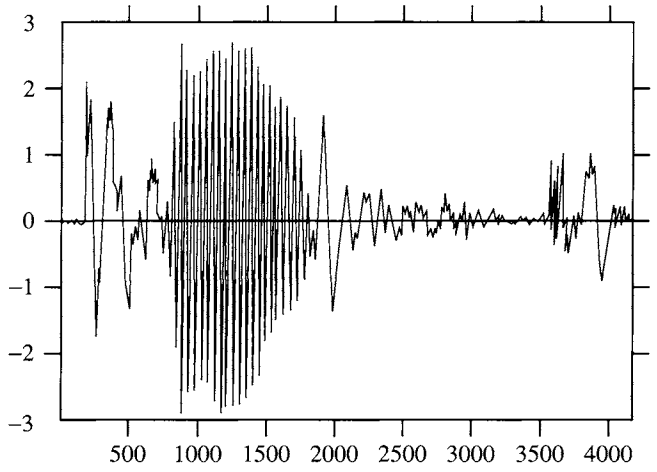

FIGURE 11.7 A segment of speech: a male speaker saying the word "test."

the residual sequence. For comparison, we also computed this ratio for the case where the residual sequence is obtained by taking the difference of neighboring samples. The sample-to-sample differences resulted in a ratio of 1.63. Compared to this, the ratio of the input variance to the variance of the residuals from the first-order predictor was 2.04. With a second-order predictor, this ratio rose to 3.37, and with a third-order predictor, the ratio was 6.28.

The residual sequence for the third-order predictor is shown in Figure 11.8. Notice that although there has been a reduction in the dynamic range, there is still substantial structure

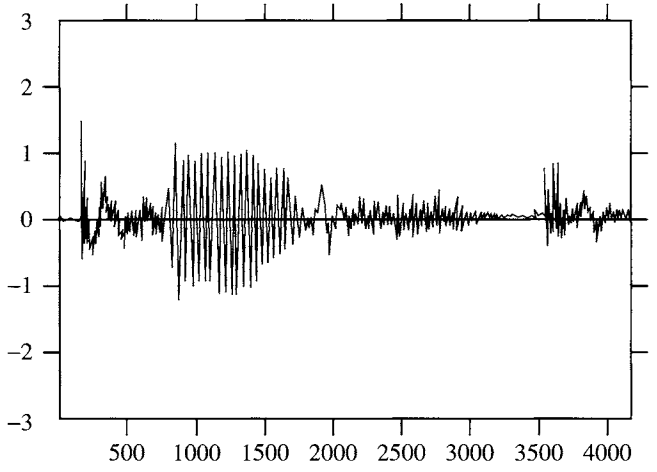

FIGURE 11.8 The residual sequence using a third-order predictor.

in the residual sequence, especially in the range of samples from about the 700th sample to the 2000th sample. We will look at ways of removing this structure when we discuss speech coding.

Let us now introduce a quantizer into the loop and look at the performance of the DPCM system. For simplicity, we will use a uniform quantizer. If we look at the histogram of the residual sequence, we find that it is highly peaked. Therefore, we will assume that the input to the quantizer will be Laplacian. We will also adjust the step size of the quantizer based on the variance of the residual. The step sizes provided in Chapter 9 are based on the assumption that the quantizer input has a unit variance. It is easy to show that when the variance differs from unity, the optimal step size can be obtained by multiplying the step size for a variance of one with the standard deviation of the input. Using this approach for a four-level Laplacian quantizer, we obtain step sizes of 0.75, 0.59, and 0.43 for the first-, second-, and third-order predictors, and step sizes of 0.3, 0.4, and 0.5 for an eight-level Laplacian quantizer. We measure the performance using two different measures, the signal-to-noise ratio (SNR) and the signal-to-prediction-error ratio. These are defined as follows:

$$\text{SNR(dB)} = \frac{\sum_{i=1}^{M} x_i^2}{\sum_{i=1}^{M} (x_i - \hat{x}_i)^2} \tag{11.36}$$

$$\text{SPER(dB)} = \frac{\sum_{i=1}^{M} x_i^2}{\sum_{i=1}^{M} (x_i - p_i)^2}. \tag{11.37}$$

The results are tabulated in Table 11.1. For comparison we have also included the results when no prediction is used; that is, we directly quantize the input. Notice the large difference between using a first-order predictor and a second-order predictor, and then the relatively minor increase when going from a second-order predictor to a third-order predictor. This is fairly typical when using a fixed quantizer.

Finally, let's take a look at the reconstructed speech signal. The speech coded using a third-order predictor and an eight-level quantizer is shown in Figure 11.9. Although the reconstructed sequence looks like the original, notice that there is significant distortion in areas where the source output values are small. This is because in these regions the input to the quantizer is close to zero. Because the quantizer does not have a zero output level,

TABLE 11.1 Performance of DPCM system with different predictors and quantizers.

Quantizer	Predictor Order	SNR (dB)	SPER (dB)
Four-level	None	2.43	0
	1	3.37	2.65
	2	8.35	5.9
	3	8.74	6.1
Eight-level	None	3.65	0
	1	3.87	2.74
	2	9.81	6.37
	3	10.16	6.71

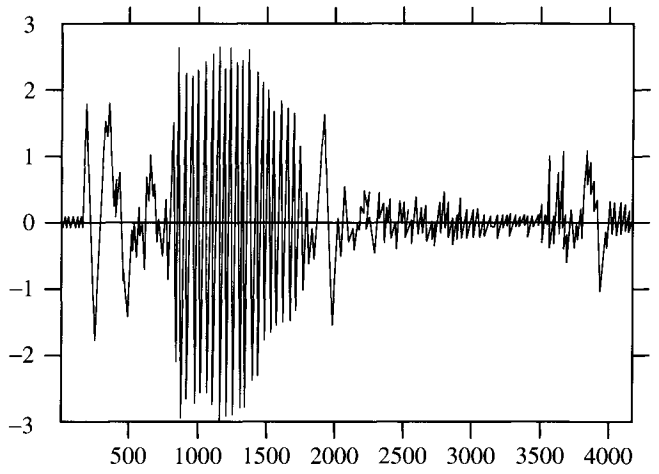

FIGURE 11.9 **The reconstructed sequence using a third-order predictor and an eight-level uniform quantizer.**

the output of the quantizer flips between the two inner levels. If we listened to this signal, we would hear a hissing sound in the reconstructed signal.

The speech signal used to generate this example is contained among the data sets accompanying this book in the file `testm.raw`. The function `readau.c` can be used to read the file. You are encouraged to reproduce the results in this example and listen to the resulting reconstructions. ♦

If we look at the speech sequence in Figure 11.7, we can see that there are several distinct segments of speech. Between sample number 700 and sample number 2000, the speech looks periodic. Between sample number 2200 and sample number 3500, the speech is low amplitude and noiselike. Given the distinctly different characteristics in these two regions, it would make sense to use different approaches to encode these segments. Some approaches to dealing with these issues are specific to speech coding, and we will discuss these approaches when we specifically discuss encoding speech using DPCM. However, the problem is also much more widespread than when encoding speech. A general response to the nonstationarity of the input is the use of adaptation in prediction. We will look at some of these approaches in the next section.

11.5 Adaptive DPCM

As DPCM consists of two main components, the quantizer and the predictor, making DPCM adaptive means making the quantizer and the predictor adaptive. Recall that we can adapt a system based on its input or output. The former approach is called forward adaptation; the latter, backward adaptation. In the case of forward adaptation, the parameters of the

system are updated based on the input to the encoder, which is not available to the decoder. Therefore, the updated parameters have to be sent to the decoder as side information. In the case of backward adaptation, the adaptation is based on the output of the encoder. As this output is also available to the decoder, there is no need for transmission of side information.

In cases where the predictor is adaptive, especially when it is backward adaptive, we generally use adaptive quantizers (forward or backward). The reason for this is that the backward adaptive predictor is adapted based on the quantized outputs. If for some reason the predictor does not adapt properly at some point, this results in predictions that are far from the input, and the residuals will be large. In a fixed quantizer, these large residuals will tend to fall in the overload regions with consequently unbounded quantization errors. The reconstructed values with these large errors will then be used to adapt the predictor, which will result in the predictor moving further and further from the input.

The same constraint is not present for quantization, and we can have adaptive quantization with fixed predictors.

11.5.1 Adaptive Quantization in DPCM

In forward adaptive quantization, the input is divided into blocks. The quantizer parameters are estimated for each block. These parameters are transmitted to the receiver as side information. In DPCM, the quantizer is in a feedback loop, which means that the input to the quantizer is not conveniently available in a form that can be used for forward adaptive quantization. Therefore, most DPCM systems use backward adaptive quantization.

The backward adaptive quantization used in DPCM systems is basically a variation of the backward adaptive Jayant quantizer described in Chapter 9. In Chapter 9, the Jayant algorithm was used to adapt the quantizer to a stationary input. In DPCM, the algorithm is used to adapt the quantizer to the local behavior of nonstationary inputs. Consider the speech segment shown in Figure 11.7 and the residual sequence shown in Figure 11.8. Obviously, the quantizer used around the 3000th sample should not be the same quantizer that was used around the 1000th sample. The Jayant algorithm provides an effective approach to adapting the quantizer to the variations in the input characteristics.

Example 11.5.1:

Let's encode the speech sample shown in Figure 11.7 using a DPCM system with a backward adaptive quantizer. We will use a third-order predictor and an eight-level quantizer. We will also use the following multipliers [110]:

$$M_0 = 0.90, \quad M_1 = 0.90, \quad M_2 = 1.25, \quad M_3 = 1.75.$$

The results are shown in Figure 11.10. Notice the region at the beginning of the speech sample and between the 3000th and 3500th sample, where the DPCM system with the fixed quantizer had problems. Because the step size of the adaptive quantizer can become quite small, these regions have been nicely reproduced. However, right after this region, the speech output has a larger spike than the reconstructed waveform. This is an indication that the quantizer is not expanding rapidly enough. This can be remedied by increasing the

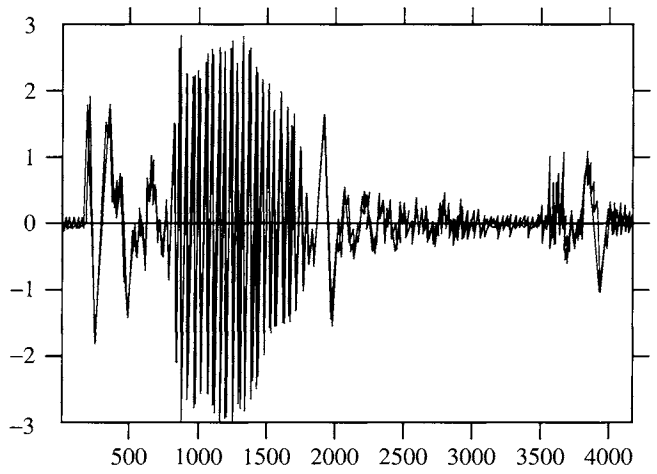

FIGURE 11.10 **The reconstructed sequence using a third-order predictor and an eight-level Jayant quantizer.**

value of M_3. The program used to generate this example is `dpcm_aqb`. You can use this program to study the behavior of the system for different configurations. ♦

11.5.2 Adaptive Prediction in DPCM

The equations used to obtain the predictor coefficients were derived based on the assumption of stationarity. However, we see from Figure 11.7 that this assumption is not true. In the speech segment shown in Figure 11.7, different segments have different characteristics. This is true for most sources we deal with; while the source output may be locally stationary over any significant length of the output, the statistics may vary considerably. In this situation, it is better to adapt the predictor to match the local statistics. This adaptation can be forward adaptive or backward adaptive.

DPCM with Forward Adaptive Prediction (DPCM-APF)

In forward adaptive prediction, the input is divided into segments or blocks. In speech coding this block consists of about 16 ms of speech. At a sampling rate of 8000 samples per second, this corresponds to 128 samples per block [123, 167]. In image coding, we use an 8×8 block [168].

The autocorrelation coefficients are computed for each block. The predictor coefficients are obtained from the autocorrelation coefficients and quantized using a relatively high-rate quantizer. If the coefficient values are to be quantized directly, we need to use at least 12 bits per coefficient [123]. This number can be reduced considerably if we represent the predictor coefficients in terms of *parcor coefficients*; we will describe how to obtain

the parcor coefficients in Chapter 17. For now, let's assume that the coefficients can be transmitted with an expenditure of about 6 bits per coefficient.

In order to estimate the autocorrelation for each block, we generally assume that the sample values outside each block are zero. Therefore, for a block length of M, the autocorrelation function for the lth block would be estimated by

$$R_{xx}^{(l)}(k) = \frac{1}{M-k} \sum_{i=(l-1)M+1}^{lM-k} x_i x_{i+k} \tag{11.38}$$

for k positive, or

$$R_{xx}^{(l)}(k) = \frac{1}{M+k} \sum_{i=(l-1)M+1-k}^{lM} x_i x_{i+k} \tag{11.39}$$

for k negative. Notice that $R_{xx}^{(l)}(k) = R_{xx}^{(l)}(-k)$, which agrees with our initial assumption.

DPCM with Backward Adaptive Prediction (DPCM-APB)

Forward adaptive prediction requires that we buffer the input. This introduces delay in the transmission of the speech. As the amount of buffering is small, the use of forward adaptive prediction when there is only one encoder and decoder is not a big problem. However, in the case of speech, the connection between two parties may be several links, each of which may consist of a DPCM encoder and decoder. In such tandem links, the amount of delay can become large enough to be a nuisance. Furthermore, the need to transmit side information makes the system more complex. In order to avoid these problems, we can adapt the predictor based on the output of the encoder, which is also available to the decoder. The adaptation is done in a sequential manner [169, 167].

In our derivation of the optimum predictor coefficients, we took the derivative of the statistical average of the squared prediction error or residual sequence. In order to do this, we had to assume that the input process was stationary. Let us now remove that assumption and try to figure out how to adapt the predictor to the input algebraically. To keep matters simple, we will start with a first-order predictor and then generalize the result to higher orders.

For a first-order predictor, the value of the residual squared at time n would be given by

$$d_n^2 = (x_n - a_1 \hat{x}_{n-1})^2. \tag{11.40}$$

If we could plot the value of d_n^2 against a_1, we would get a graph similar to the one shown in Figure 11.11. Let's take a look at the derivative of d_n^2 as a function of whether the current value of a_1 is to the left or right of the optimal value of a_1—that is, the value of a_1 for which d_n^2 is minimum. When a_1 is to the left of the optimal value, the derivative is negative. Furthermore, the derivative will have a larger magnitude when a_1 is further away from the optimal value. If we were asked to adapt a_1, we would add to the current value of a_1. The amount to add would be large if a_1 was far from the optimal value, and small if a_1 was close to the optimal value. If the current value was to the right of the optimal value, the derivative would be positive, and we would subtract some amount from a_1 to adapt it. The

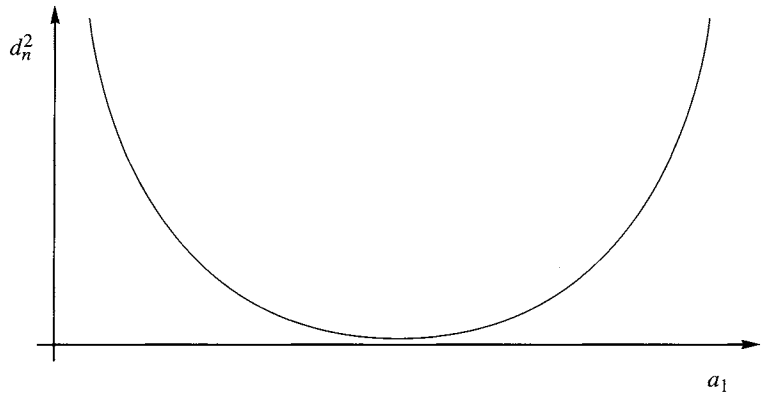

FIGURE 11.11 A plot of the residual squared versus the predictor coefficient.

amount to subtract would be larger if we were further from the optimal, and as before, the derivative would have a larger magnitude if a_1 were further from the optimal value.

At any given time, in order to adapt the coefficient at time $n+1$, we add an amount proportional to the magnitude of the derivative with a sign that is opposite to that of the derivative of d_n^2 at time n:

$$a_1^{(n+1)} = a_1^{(n)} - \alpha \frac{\delta d_n^2}{\delta a_1} \tag{11.41}$$

where α is some proportionality constant.

$$\frac{\delta d_n^2}{\delta a_1} = -2(x_n - a_1 \hat{x}_{n-1})\hat{x}_{n-1} \tag{11.42}$$

$$= -2d_n \hat{x}_{n-1}. \tag{11.43}$$

Substituting this into (11.41), we get

$$a_1^{(n+1)} = a_1^{(n)} + \alpha d_n \hat{x}_{n-1} \tag{11.44}$$

where we have absorbed the 2 into α. The residual value d_n is available only to the encoder. Therefore, in order for both the encoder and decoder to use the same algorithm, we replace d_n by $\hat{d}_n$ in (11.44) to obtain

$$a_1^{(n+1)} = a_1^{(n)} + \alpha \hat{d}_n \hat{x}_{n-1}. \tag{11.45}$$

Extending this adaptation equation for a first-order predictor to an Nth-order predictor is relatively easy. The equation for the squared prediction error is given by

$$d_n^2 = \left(x_n - \sum_{i=1}^{N} a_i \hat{x}_{n-i}\right)^2. \tag{11.46}$$

Taking the derivative with respect to a_j will give us the adaptation equation for the jth predictor coefficient:

$$a_j^{(n+1)} = a_j^{(n)} + \alpha \hat{d}_n \hat{x}_{n-j}. \tag{11.47}$$

We can combine all N equations in vector form to get

$$\mathbf{A}^{(n+1)} = \mathbf{A}^{(n)} + \alpha \hat{d}_n \hat{X}_{n-1} \tag{11.48}$$

where

$$\hat{X}_n = \begin{bmatrix} \hat{x}_n \\ \hat{x}_{n-1} \\ \vdots \\ \hat{x}_{n-N+1} \end{bmatrix}. \tag{11.49}$$

This particular adaptation algorithm is called the least mean squared (LMS) algorithm [170].

11.6 Delta Modulation

A very simple form of DPCM that has been widely used in a number of speech-coding applications is the delta modulator (DM). The DM can be viewed as a DPCM system with a 1-bit (two-level) quantizer. With a two-level quantizer with output values $\pm\Delta$, we can only represent a sample-to-sample difference of Δ. If, for a given source sequence, the sample-to-sample difference is often very different from Δ, then we may incur substantial distortion. One way to limit the difference is to sample more often. In Figure 11.12 we see a signal that has been sampled at two different rates. The lower-rate samples are shown by open circles, while the higher-rate samples are represented by $+$. It is apparent that the lower-rate samples are further apart in value.

The rate at which a signal is sampled is governed by the highest frequency component of a signal. If the highest frequency component in a signal is W, then in order to obtain an exact reconstruction of the signal, we need to sample it at least at twice the highest frequency, or $2W$. In systems that use delta modulation, we usually sample the signal at much more than twice the highest frequency. If F_s is the sampling frequency, then the ratio of F_s to $2W$ can range from almost 1 to almost 100 [123]. The higher sampling rates are used for high-quality A/D converters, while the lower rates are more common for low-rate speech coders.

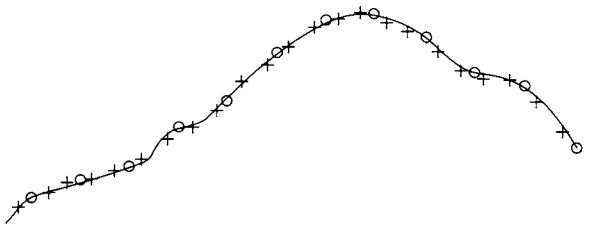

FIGURE 11.12 **A signal sampled at two different rates.**

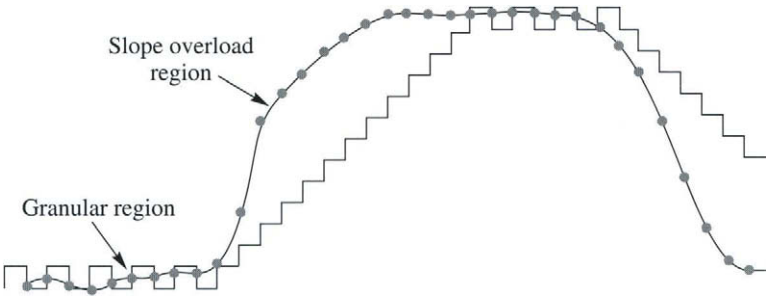

FIGURE 11.13 A source output sampled and coded using delta modulation.

If we look at a block diagram of a delta modulation system, we see that, while the block diagram of the encoder is identical to that of the DPCM system, the standard DPCM decoder is followed by a filter. The reason for the existence of the filter is evident from Figure 11.13, where we show a source output and the unfiltered reconstruction. The samples of the source output are represented by the filled circles. As the source is sampled at several times the highest frequency, the staircase shape of the reconstructed signal results in distortion in frequency bands outside the band of frequencies occupied by the signal. The filter can be used to remove these spurious frequencies.

The reconstruction shown in Figure 11.13 was obtained with a delta modulator using a fixed quantizer. Delta modulation systems that use a fixed step size are often referred to as linear delta modulators. Notice that the reconstructed signal shows one of two behaviors. In regions where the source output is relatively constant, the output alternates up or down by Δ; these regions are called the *granular regions*. In the regions where the source output rises or falls fast, the reconstructed output cannot keep up; these regions are called the *slope overload regions*. If we want to reduce the granular error, we need to make the step size Δ small. However, this will make it more difficult for the reconstruction to follow rapid changes in the input. In other words, it will result in an increase in the overload error. To avoid the overload condition, we need to make the step size large so that the reconstruction can quickly catch up with rapid changes in the input. However, this will increase the granular error.

One way to avoid this impasse is to adapt the step size to the characteristics of the input, as shown in Figure 11.14. In quasi-constant regions, make the step size small in order to reduce the granular error. In regions of rapid change, increase the step size in order to reduce overload error. There are various ways of adapting the delta modulator to the local characteristics of the source output. We describe two of the more popular ways here.

11.6.1 Constant Factor Adaptive Delta Modulation (CFDM)

The objective of adaptive delta modulation is clear: increase the step size in overload regions and decrease it in granular regions. The problem lies in knowing when the system is in each of these regions. Looking at Figure 11.13, we see that in the granular region the output of

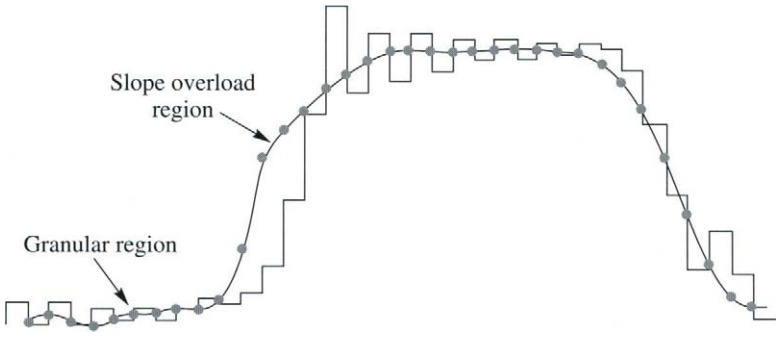

FIGURE 11.14 **A source output sampled and coded using adaptive delta modulation.**

the quantizer changes sign with almost every input sample; in the overload region, the sign of the quantizer output is the same for a string of input samples. Therefore, we can define an overload or granular condition based on whether the output of the quantizer has been changing signs. A very simple system [171] uses a history of one sample to decide whether the system is in overload or granular condition and whether to expand or contract the step size. If s_n denotes the sign of the quantizer output $\hat{d}_n$,

$$s_n = \begin{cases} 1 & \text{if } \hat{d}_n > 0 \\ -1 & \text{if } \hat{d}_n < 0 \end{cases} \tag{11.50}$$

the adaptation logic is given by

$$\Delta_n = \begin{cases} M_1 \Delta_{n-1} & s_n = s_{n-1} \\ M_2 \Delta_{n-1} & s_n \neq s_{n-1} \end{cases} \tag{11.51}$$

where $M_1 = \frac{1}{M_2} = M > 1$. In general, $M < 2$.

By increasing the memory, we can improve the response of the CFDM system. For example, if we looked at two past samples, we could decide that the system was moving from overload to granular condition if the sign had been the same for the past two samples and then changed with the current sample:

$$s_n \neq s_{n-1} = s_{n-2}. \tag{11.52}$$

In this case it would be reasonable to assume that the step size had been expanding previously and, therefore, needed a sharp contraction. If

$$s_n = s_{n-1} \neq s_{n-2}, \tag{11.53}$$

then it would mean that the system was probably entering the overload region, while

$$s_n = s_{n-1} = s_{n-2} \tag{11.54}$$

would mean the system was in overload and the step size should be expanded rapidly.

For the encoding of speech, the following multipliers M_i are recommended by [172] for a CFDM system with two-sample memory:

$$s_n \neq s_{n-1} = s_{n-2} \qquad M_1 = 0.4 \tag{11.55}$$

$$s_n \neq s_{n-1} \neq s_{n-2} \qquad M_2 = 0.9 \tag{11.56}$$

$$s_n = s_{n-1} \neq s_{n-2} \qquad M_3 = 1.5 \tag{11.57}$$

$$s_n = s_{n-1} = s_{n-2} \qquad M_4 = 2.0. \tag{11.58}$$

The amount of memory can be increased further with a concurrent increase in complexity. The space shuttle used a delta modulator with a memory of seven [173].

11.6.2 Continuously Variable Slope Delta Modulation

The CFDM systems described use a rapid adaptation scheme. For low-rate speech coding, it is more pleasing if the adaptation is over a longer period of time. This slower adaptation results in a decrease in the granular error and generally an increase in overload error. Delta modulation systems that adapt over longer periods of time are referred to as *syllabically* companded. A popular class of syllabically companded delta modulation systems is the continuously variable slope delta modulation systems.

The adaptation logic used in CVSD systems is as follows [123]:

$$\Delta_n = \beta\Delta_{n-1} + \alpha_n\Delta_0 \tag{11.59}$$

where β is a number less than but close to one, and α_n is equal to one if J of the last K quantizer outputs were of the same sign. That is, we look in a window of length K to obtain the behavior of the source output. If this condition is not satisfied, then α_n is equal to zero. Standard values for J and K are $J = 3$ and $K = 3$.

11.7 Speech Coding

Differential encoding schemes are immensely popular for speech encoding. They are used in the telephone system, voice messaging, and multimedia applications, among others. Adaptive DPCM is a part of several international standards (ITU-T G.721, ITU G.723, ITU G.726, ITU-T G.722), which we will look at here and in later chapters.

Before we do that, let's take a look at one issue specific to speech coding. In Figure 11.7, we see that there is a segment of speech that looks highly periodic. We can see this periodicity if we plot the autocorrelation function of the speech segment (Figure 11.15).

The autocorrelation peaks at a lag value of 47 and multiples of 47. This indicates a periodicity of 47 samples. This period is called the *pitch period.* The predictor we originally designed did not take advantage of this periodicity, as the largest predictor was a third-order predictor, and this periodic structure takes 47 samples to show up. We can take advantage of this periodicity by constructing an outer prediction loop around the basic DPCM structure

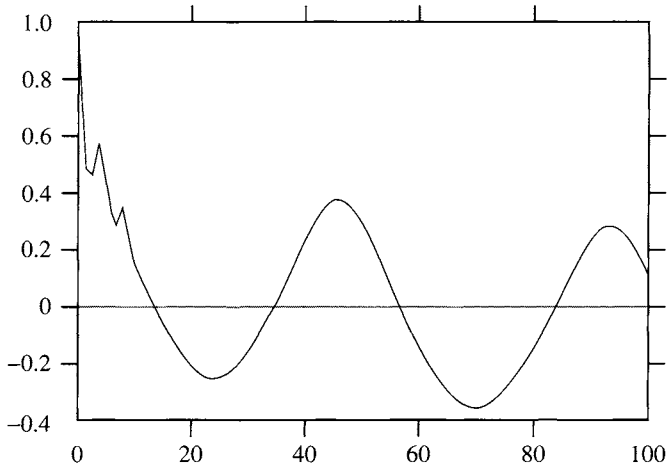

FIGURE 11.15 **Autocorrelation function for `test.snd`.**

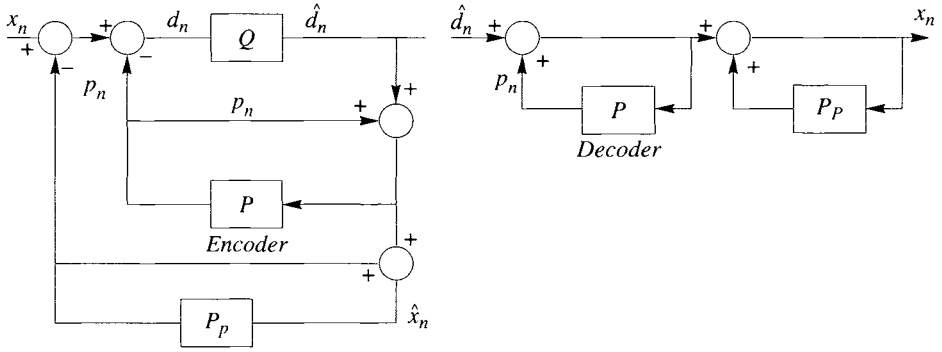

FIGURE 11.16 **The DPCM structure with a pitch predictor.**

as shown in Figure 11.16. This can be a simple single coefficient predictor of the form $b\hat{x}_{n-\tau}$, where τ is the pitch period. Using this system on `testm.raw`, we get the residual sequence shown in Figure 11.17. Notice the decrease in amplitude in the periodic portion of the speech.

Finally, remember that we have been using mean squared error as the distortion measure in all of our discussions. However, perceptual tests do not always correlate with the mean squared error. The level of distortion we perceive is often related to the level of the speech signal. In regions where the speech signal is of higher amplitude, we have a harder time perceiving the distortion, but the same amount of distortion in a different frequency band might be very perceptible. We can take advantage of this by shaping the quantization error so that most of the error lies in the region where the signal has a higher amplitude. This variation of DPCM is called *noise feedback coding* (NFC) (see [123] for details).

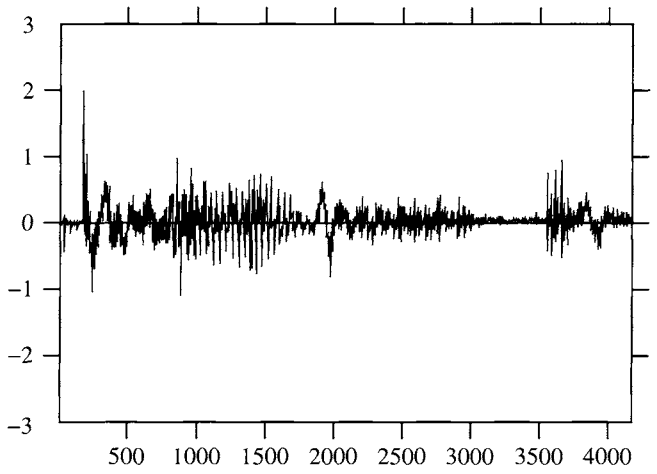

FIGURE 11.17 **The residual sequence using the DPCM system with a pitch predictor.**

11.7.1 G.726

The International Telecommunications Union has published recommendations for a standard ADPCM system, including recommendations G.721, G.723, and G.726. G.726 supersedes G.721 and G.723. In this section we will describe the G.726 recommendation for ADPCM systems at rates of 40, 32, 24, and 16 kbits.

The Quantizer

The recommendation assumes that the speech output is sampled at the rate of 8000 samples per second, so the rates of 40, 32, 24, and 16 kbits per second translate 5 bits per sample, 4 bits per sample, 3 bits per sample, and 2 bits per sample. Comparing this to the PCM rate of 8 bits per sample, this would mean compression ratios of 1.6:1, 2:1, 2.67:1, and 4:1. Except for the 16 kbits per second system, the number of levels in the quantizer are $2^{nb} - 1$, where nb is the number of bits per sample. Thus, the number of levels in the quantizer is odd, which means that for the higher rates we use a midtread quantizer.

The quantizer is a backward adaptive quantizer with an adaptation algorithm that is similar to the Jayant quantizer. The recommendation describes the adaptation of the quantization interval in terms of the adaptation of a scale factor. The input d_k is normalized by a scale factor α_k. This normalized value is quantized, and the normalization removed by multiplying with α_k. In this way the quantizer is kept fixed and α_k is adapted to the input. Therefore, for example, instead of expanding the step size, we would increase the value of α_k.

The fixed quantizer is a nonuniform midtread quantizer. The recommendation describes the quantization boundaries and reconstruction values in terms of the log of the scaled input. The input-output characteristics for the 24 kbit system are shown in Table 11.2. An output value of $-\infty$ in the table corresponds to a reconstruction value of 0.

TABLE 11.2 Recommended input-output characteristics of the quantizer for 24-kbits-per-second operation.

Input Range	Label	Output
$\log_2 \frac{d_k}{\alpha_k}$	$\lvert I_k \rvert$	$\log_2 \frac{d_k}{\alpha_k}$
$[2.58, \infty)$	3	2.91
$[1.70, 2.58)$	2	2.13
$[0.06, 1.70)$	1	1.05
$(-\infty, -0.06)$	0	$-\infty$

The adaptation algorithm is described in terms of the logarithm of the scale factor

$$y(k) = \log_2 \alpha_k. \tag{11.60}$$

The adaptation of the scale factor α or its log $y(k)$ depends on whether the input is speech or speechlike, where the sample-to-sample difference can fluctuate considerably, or whether the input is voice-band data, which might be generated by a modem, where the sample-to-sample fluctuation is quite small. In order to handle both these situations, the scale factor is composed of two values, a *locked* slow scale factor for when the sample-to-sample differences are quite small, and an *unlocked* value for when the input is more dynamic:

$$y(k) = a_l(k)y_u(k-1) + (1 - a_l(k))y_l(k-1). \tag{11.61}$$

The value of $a_l(k)$ depends on the variance of the input. It will be close to one for speech inputs and close to zero for tones and voice band data.

The unlocked scale factor is adapted using the Jayant algorithm with one slight modification. If we were to use the Jayant algorithm, the unlocked scale factor could be adapted as

$$\alpha_u(k) = \alpha_{k-1} M[I_{k-1}] \tag{11.62}$$

where $M[\cdot]$ is the multiplier. In terms of logarithms, this becomes

$$y_u(k) = y(k-1) + \log M[I_{k-1}]. \tag{11.63}$$

The modification consists of introducing some memory into the adaptive process so that the encoder and decoder converge following transmission errors:

$$y_u(k) = (1-\epsilon)y(k-1) + \epsilon W[I_{k-1}] \tag{11.64}$$

where $W[\cdot] = \log M[\cdot]$, and $\epsilon = 2^{-5}$.

The locked scale factor is obtained from the unlocked scale factor through

$$y_l(k) = (1-\gamma)y_l(k-1) + \gamma y_u(k), \qquad \gamma = 2^{-6}. \tag{11.65}$$

The Predictor

The recommended predictor is a backward adaptive predictor that uses a linear combination of the past two reconstructed values as well as the six past quantized differences to generate the prediction

$$p_k = \sum_{i=1}^{2} a_i^{(k-1)} \hat{x}_{k-i} + \sum_{i=1}^{6} b_i^{(k-1)} \hat{d}_{k-i}. \tag{11.66}$$

The set of predictor coefficients is updated using a simplified form of the LMS algorithm.

$$a_1^{(k)} = (1-2^{-8})a_1^{(k-1)} + 3 \times 2^{-8}\text{sgn}[z(k)]\text{sgn}[z(k-1)] \tag{11.67}$$

$$a_2^{(k)} = (1-2^{-7})a_2^{(k-1)} + 2^{-7}(\text{sgn}[z(k)]\text{sgn}[z(k-2)] - f\left(a_1^{(k-1)}\text{sgn}[z(k)]\text{sgn}[z(k-1)]\right)\Big) \tag{11.68}$$

where

$$z(k) = \hat{d}_k + \sum_{i=1}^{6} b_i^{(k-1)} \hat{d}_{k-i} \tag{11.69}$$

$$f(\beta) = \begin{cases} 4\beta & |\beta| \le \frac{1}{2} \\ 2\text{sgn}(\beta) & |\beta| > \frac{1}{2}. \end{cases} \tag{11.70}$$

The coefficients $\{b_i\}$ are updated using the following equation:

$$b_i^{(k)} = (1-2^{-8})b_i^{(k-1)} + 2^{-7}\text{sgn}[\hat{d}_k]\text{sgn}[\hat{d}_{k-i}]. \tag{11.71}$$

Notice that in the adaptive algorithms we have replaced products of reconstructed values and products of quantizer outputs with products of their signs. This is computationally much simpler and does not lead to any significant degradation of the adaptation process. Furthermore, the values of the coefficients are selected such that multiplication with these coefficients can be accomplished using shifts and adds. The predictor coefficients are all set to zero when the input moves from tones to speech.

11.8 Image Coding

We saw in Chapter 7 that differential encoding provided an efficient approach to the lossless compression of images. The case for using differential encoding in the lossy compression of images has not been made as clearly. In the early days of image compression, both differential encoding and transform coding were popular forms of lossy image compression. At the current time differential encoding has a much more restricted role as part of other compression strategies. Several currently popular approaches to image compression decompose the image into lower and higher frequency components. As low-frequency signals have high sample-to-sample correlation, several schemes use differential encoding to compress the low-frequency components. We will see this use of differential encoding when we look at subband- and wavelet-based compression schemes and, to a lesser extent, when we study transform coding.

For now let us look at the performance of a couple of stand-alone differential image compression schemes. We will compare the performance of these schemes with the performance of the JPEG compression standard.

Consider a simple differential encoding scheme in which the predictor $p[j,k]$ for the pixel in the jth row and the kth column is given by

$$p[j,k] = \begin{cases} \hat{x}[j,k-1] & \text{for } k > 0 \\ \hat{x}[j-1,k] & \text{for } k = 0 \text{ and } j > 0 \\ 128 & \text{for } j = 0 \text{ and } k = 0 \end{cases}$$

where $\hat{x}[j,k]$ is the reconstructed pixel in the jth row and kth column. We use this predictor in conjunction with a fixed four-level uniform quantizer and code the quantizer output using an arithmetic coder. The coding rate for the compressed image is approximately 1 bit per pixel. We compare this reconstructed image with a JPEG-coded image at the same rate in Figure 11.18. The signal-to-noise ratio for the differentially encoded image is 22.33 dB (PSNR 31.42 dB) and for the JPEG-encoded image is 32.52 dB (PSNR 41.60 dB), a difference of more than 10 dB!

However, this is an extremely simple system compared to the JPEG standard, which has been fine-tuned for encoding images. Let's make our differential encoding system slightly more complicated by replacing the uniform quantizer with a recursively indexed quantizer and the predictor by a somewhat more complicated predictor. For each pixel (except for the boundary pixels) we compute the following three values:

$$\begin{aligned} p_1 &= 0.5 \times \hat{x}[j-1,k] + 0.5 \times \hat{x}[j,k-1] \\ p_2 &= 0.5 \times \hat{x}[j-1,k-1] + 0.5 \times \hat{x}[j,k-1] \\ p_3 &= 0.5 \times \hat{x}[j-1,k-1] + 0.5 \times \hat{x}[j-1,k] \end{aligned} \tag{11.72}$$

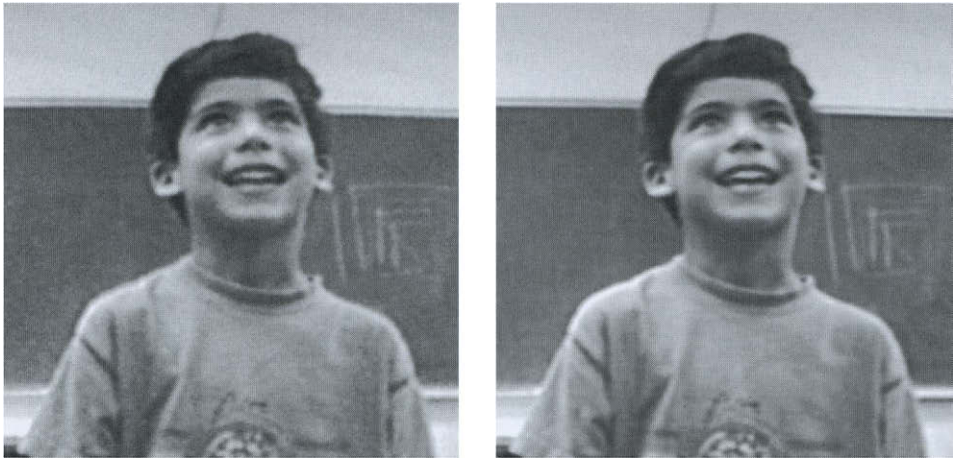

FIGURE 11.18 **Left: Reconstructed image using differential encoding at 1 bit per pixel. Right: Reconstructed image using JPEG at 1 bit per pixel.**

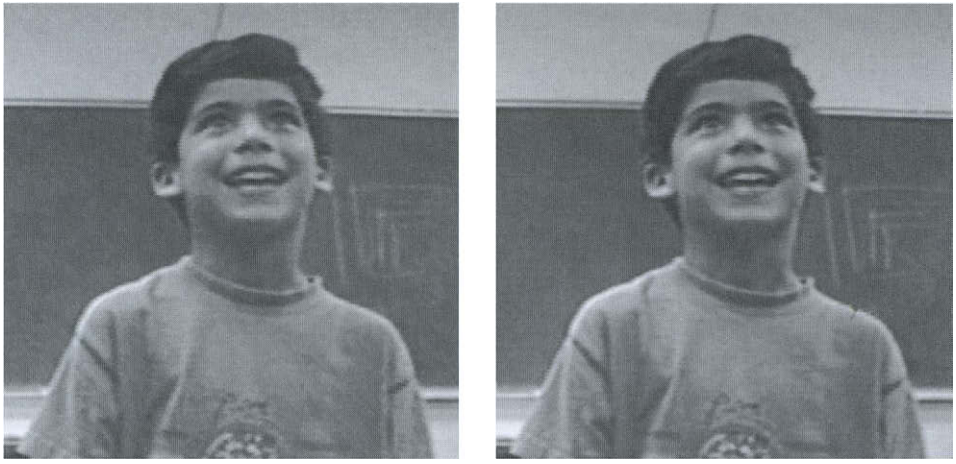

FIGURE 11.19 **Left: Reconstructed image using differential encoding at 1 bit per pixel using median predictor and recursively indexed quantizer. Right: Reconstructed image using JPEG at 1 bit per pixel.**

then obtain the predicted value as

$$p[j,k] = \text{median}\{p_1, p_2, p_3\}.$$

For the boundary pixels we use the simple prediction scheme. At a coding rate of 1 bit per pixel, we obtain the image shown in Figure 11.19. For reference we show it next to the JPEG-coded image at the same rate. The signal-to-noise ratio for this reconstruction is 29.20 dB (PSNR 38.28 dB). We have made up two-thirds of the difference using some relatively minor modifications. We can see that it might be feasible to develop differential encoding schemes that are competitive with other image compression techniques. Therefore, it makes sense not to dismiss differential encoding out of hand when we need to develop image compression systems.

11.9 Summary

In this chapter we described some of the more well-known differential encoding techniques. Although differential encoding does not provide compression as high as vector quantization, it is very simple to implement. This approach is especially suited to the encoding of speech, where it has found broad application. The DPCM system consists of two main components, the quantizer and the predictor. We spent a considerable amount of time discussing the quantizer in Chapter 9, so most of the discussion in this chapter focused on the predictor. We have seen different ways of making the predictor adaptive, and looked at some of the improvements to be obtained from source-specific modifications to the predictor design.

Further Reading

1. *Digital Coding of Waveforms*, by N.S. Jayant and P. Noll [123], contains some very detailed and highly informative chapters on differential encoding.
2. "Adaptive Prediction in Speech Differential Encoding Systems," by J.D. Gibson [167], is a comprehensive treatment of the subject of adaptive prediction.
3. A real-time video coding system based on DPCM has been developed by NASA. Details can be found in [174].

11.10 Projects and Problems

1. Generate an AR(1) process using the relationship

$$x_n = 0.9 \times x_{n-1} + \epsilon_n$$

where ϵ_n is the output of a Gaussian random number generator (this is option 2 in `rangen`).

   (a) Encode this sequence using a DPCM system with a one-tap predictor with predictor coefficient 0.9 and a three-level Gaussian quantizer. Compute the variance of the prediction error. How does this compare with the variance of the input? How does the variance of the prediction error compare with the variance of the $\{\epsilon_n\}$ sequence?

   (b) Repeat using predictor coefficient values of 0.5, 0.6, 0.7, 0.8, and 1.0. Comment on the results.

2. Generate an AR(5) process using the following coefficients: 1.381, 0.6, 0.367, −0.7, 0.359.

   (a) Encode this with a DPCM system with a 3-bit Gaussian nonuniform quantizer and a first-, second-, third-, fourth-, and fifth-order predictor. Obtain these predictors by solving (11.30). For each case compute the variance of the prediction error and the SNR in dB. Comment on your results.

   (b) Repeat using a 3-bit Jayant quantizer.

3. DPCM can also be used for encoding images. Encode the Sinan image using a one-tap predictor of the form

$$\hat{x}_{i,j} = a \times x_{i,j-1}$$

and a 2-bit quantizer. Experiment with quantizers designed for different distributions. Comment on your results.

4. Repeat the image-coding experiment of the previous problem using a Jayant quantizer.
5. DPCM-encode the Sinan, Elif, and bookshelf1 images using a one-tap predictor and a four-level quantizer followed by a Huffman coder. Repeat using a five-level quantizer. Compute the SNR for each case, and compare the rate distortion performances.

6. We want to DPCM-encode images using a two-tap predictor of the form

$$\hat{x}_{i,j} = a \times x_{i,j-1} + b \times x_{i-1,j}$$

and a four-level quantizer followed by a Huffman coder. Find the equations we need to solve to obtain coefficients a and b that minimize the mean squared error.

7. **(a)** DPCM-encode the Sinan, Elif, and bookshelf1 images using a two-tap predictor and a four-level quantizer followed by a Huffman coder.

(b) Repeat using a five-level quantizer. Compute the SNR and rate (in bits per pixel) for each case.

(c) Compare the rate distortion performances with the one-tap case.

(d) Repeat using a five-level quantizer. Compute the SNR for each case, and compare the rate distortion performances using a one-tap and two-tap predictor.