CHAPTER 10

INFORMATION THEORY AND CODING

Information theory provides a different perspective for evaluating the performance of communication systems, and significant insight into the performance characteristics of communication systems can often be gained through the study of information theory. More explicitly, information theory provides a quantitative measure of the information contained in message signals and allows us to determine the capability of a system to transfer this information from source to destination. Coding, the major application area of information theory, will be briefly presented in this chapter. Through the use of source coding, unsystematic redundancy can be removed from message signals so that channels can be used with maximum efficiency. In addition, through the use of channel, or error-correcting, coding, systematic redundancy can be induced into the transmitted signal so that errors caused by nonperfect practical channels can be corrected.

Information theory also provides us with the performance characteristics of an *ideal*, or optimum, communication system. The performance of an ideal system provides a meaningful basis against which to compare the performance of the realizable systems studied in previous chapters. Performance characteristics of ideal systems illustrate the gain in performance that can be obtained by implementing more complicated transmission and detection schemes.

Motivation for the study of information theory is provided by *Shannon's coding theorem*, sometimes referred to as *Shannon's second theorem*, which can be stated as follows: "If a source has an information rate less than the channel capacity, there exists a coding procedure such that the source output can be transmitted over the channel with an arbitrarily small probability of error." This is a powerful result. Shannon tells us that transmission and reception can be accomplished with *negligible* error, even in the presence of noise. An understanding of this process called *coding*, and an understanding of its impact on the design and performance of communication systems, require an understanding of several basic concepts of information theory.

10.1 BASIC CONCEPTS

Consider a hypothetical classroom situation occurring early in a course at the end of a class period. The professor makes one of the following statements to the class:

A. I will see you next period.

B. My colleague will lecture next period.

C. Everyone gets an A in the course, and there will be no more class meetings.

What is the relative information conveyed to the students by each of these statements, assuming that there had been no previous discussion on the subject? Obviously, there is little information conveyed by statement (A), since the class would normally assume that their regular professor would lecture; that is, the probability, $P(A)$, of the regular professor lecturing is nearly unity. Intuitively, we know that statement (B) contains more information, and the probability of a colleague lecturing $P(B)$ is relatively low. Statement (C) contains a vast amount of information for the entire class, and most would agree that such a statement has a very low probability of occurrence in a typical classroom situation. It appears that the lower the probability of a statement, the greater is the information conveyed by that statement. Stated another way, the students' surprise on hearing a statement seems to be a good measure of the information contained in that statement. Information is defined consistent with this intuitive example.

10.1.1 Information

Let x_j be an event that occurs with probability $p(x_j)$. If we are told that event x_j has occurred, we say that we have received

$$I(x_j) = \log_a \frac{1}{p(x_j)} = -\log_a p(x_j) \tag{10.1}$$

units of information. This definition is consistent with the previous example since $I(x_j)$ increases as $p(x_j)$ decreases. The base of the logarithm in (10.1) is arbitrary and determines the units by which information is measured. R. V. Hartley,[1] who first suggested the logarithmic measure of information in 1928, used logarithms to the base 10, and the measure of information was the *hartley*. Today it is standard to use logarithms to the base 2, and the unit of information is the binary unit, or *bit*. Logarithms to the base e are sometimes utilized, and the corresponding unit is the *nat*, or natural unit.

There are several reasons for us to be consistent in using the base 2 logarithm to measure information. The simplest random experiment that one can imagine is an experiment with two equally likely outcomes, such as the flipping of an unbiased coin. Knowledge of each outcome has associated with it one bit of information. Also, since the digital computer is a binary machine, each logical 0 and each logical 1 has associated with it one bit of information, assuming that each of these logical states is equally likely.

EXAMPLE 10.1

Consider a random experiment with 16 equally likely outcomes. The information associated with each outcome is

$$I(x_j) = -\log_2 \frac{1}{16} = \log_2 16 = 4 \text{ bits} \tag{10.2}$$

where j ranges from 1 to 16. The information is greater than one bit, since the probability of each outcome is less than one-half. ■

10.1.2 Entropy

In general, the average information associated with the outcome of an experiment is of interest rather than the information associated with each particular event. The average information associated with a discrete random variable X is defined as the entropy $H(X)$. Thus

$$H(X) = E[I(x_j)] = -\sum_{j=1}^{n} p(x_j) \log_2 p(x_j) \tag{10.3}$$

where n is the total number of possible outcomes. Entropy can be regarded as average uncertainty and therefore should be maximum when each outcome is equally likely.

EXAMPLE 10.2

For a binary source, $p(1) = \alpha$ and $p(0) = 1 - \alpha = \beta$. Derive the entropy of the source as a function of α, and sketch $H(\alpha)$ as α varies from zero to 1.

[1] Hartley (1928).

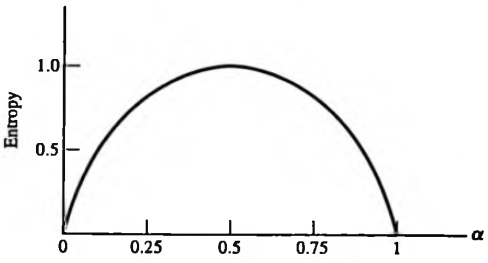


FIGURE 10.1 Entropy of a binary source.

Solution: From (10.3),

$$H(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha) \tag{10.4}$$

This is sketched in Figure 10.1 We should note the maximum. If $\alpha = \frac{1}{2}$, each symbol is equally likely, and our uncertainty is a maximum. If $\alpha \neq \frac{1}{2}$, one symbol is more likely to occur than the other, and we are less uncertain as to which symbol appears on the source output. If α or β is equal to zero, our uncertainty is zero, since we know exactly which symbol will occur. ■

From Example 10.2 we conclude, at least for the special case illustrated in Figure 10.1, that the entropy function has a maximum, and the maximum occurs when all probabilities are equal. This fact is of sufficient importance to warrant a more complete derivation. Assume that a chance experiment has n possible outcomes and that p_n is a dependent variable depending on the other probabilities. Thus

$$p_n = 1 - (p_1 + p_2 + \cdots + p_k + \cdots + p_{n-1}) \tag{10.5}$$

where p_j is concise notation for $p(x_j)$. The entropy associated with the chance experiment is

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{10.6}$$

In order to find the maximum value of entropy, the entropy is differentiated with respect to p_k, holding all probabilities constant except p_k and p_n. This gives a relationship between p_k and p_n that yields the maximum value of H. Since all derivatives are zero except the ones involving p_k and p_n,

$$\frac{dH}{dp_k} = \frac{d}{dp_k}(-p_k \log_2 p_k - p_n \log_2 p_n) \tag{10.7}$$

Using (10.7) and

$$\frac{d}{dx}\log_a u = \frac{1}{u}\log_a e \frac{du}{dx} \tag{10.8}$$

gives

$$\frac{dH}{dp_k} = -p_k \frac{1}{p_k}\log_2 e - \log_2 p_k + p_n \frac{1}{p_n}\log_2 e + \log_2 p_n \tag{10.9}$$

or

$$\frac{dH}{dp_k} = \log_2 \frac{p_n}{p_k} \tag{10.10}$$

which is zero if $p_k = p_n$. Since p_k is arbitrary,

$$p_1 = p_2 = \cdots = p_n = \frac{1}{n} \tag{10.11}$$

From (10.6), the case where all probabilities are equal yields $H = \log_2 n$.

10.1.3 Discrete Channel Models

Throughout most of this chapter we will assume the communications channel to be *memoryless*. For such channels, the channel output at a given time is a function of the channel input *at that time* and is not a function of previous channel inputs. Discrete memoryless channels are completely specified by the set of conditional probabilities that relate the probability of each output state to the input probabilities. An example illustrates the technique. A diagram of a channel with two inputs and three outputs is illustrated in Figure 10.2. Each possible input-to-output path is indicated along with a conditional probability p_{ij}, which is concise notation for $p(y_j|x_i)$. Thus p_{ij} is the conditional probability of obtaining output y_j given that the input is x_i and is called a *channel transition probability*.

We can see from Figure 10.2 that the channel is completely specified by the set of transition probabilities. Accordingly, a discrete channel is specified by the matrix of transition probabilities $[P(Y|X)]$, where, for the channel of Figure 10.2,

$$[P(Y|X)] = \begin{bmatrix} p(y_1|x_1) & p(y_2|x_1) & p(y_3|x_1) \\ p(y_1|x_2) & p(y_2|x_2) & p(y_3|x_2) \end{bmatrix} \tag{10.12}$$

Since each input to the channel results in some output, each new row of the channel matrix must sum to unity.

The channel matrix is useful in deriving the output probabilities given the input probabilities. For example, if the input probabilities $P(X)$ are represented by the row matrix

$$[P(X)] = [p(x_1)\ \ p(x_2)] \tag{10.13}$$

then

$$[P(Y)] = [p(y_1)\ \ p(y_2)\ \ p(y_3)] \tag{10.14}$$

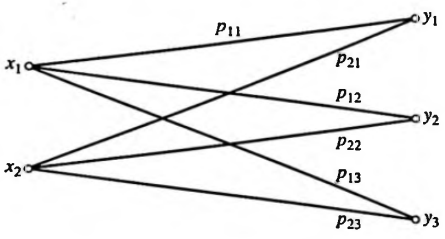


FIGURE 10.2 Channel diagram.

which is computed by

$$[P(Y)] = [P(X)][P(Y|X)] \tag{10.15}$$

If $[P(X)]$ is written as a diagonal matrix, (10.15) yields a matrix $[P(X, Y)]$. Each element in the matrix has the form $p(x_i)p(y_j|x_i)$ or $p(x_i, y_j)$. This matrix is known as the *joint probability matrix*, and the term $p(x_i, y_j)$ is the joint probability of transmitting x_i and receiving y_j.

EXAMPLE 10.3

Consider the binary input-output channel shown in Figure 10.3. The matrix of transition probabilities is

$$[P(Y|X)] = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \tag{10.16}$$

If the input probabilities are $P(x_1) = 0.5$ and $P(x_2) = 0.5$, the output probabilities are

$$[P(Y)] = [0.5 \quad 0.5] \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} = [0.55 \quad 0.45] \tag{10.17}$$

and the joint probability matrix is

$$[P(X, Y)] = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} = \begin{bmatrix} 0.35 & 0.15 \\ 0.2 & 0.3 \end{bmatrix} \tag{10.18}$$

■

As we first observed in Chapter 8, a binary satellite communication system can often be represented by the cascade combination of two binary channels. This is illustrated in Figure 10.4(a), in which the first binary channel represents the uplink and the second binary channel represents the downlink. These channels can be combined as shown in Figure 10.4(b).

By determining all possible paths from x_i to z_j, it is clear that the following probabilities define the overall channel illustrated in Figure 10.4(b):

$$p_{11} = \alpha_1\beta_1 + \alpha_2\beta_3 \tag{10.19}$$

$$p_{12} = \alpha_1\beta_2 + \alpha_2\beta_4 \tag{10.20}$$

$$p_{21} = \alpha_3\beta_1 + \alpha_4\beta_3 \tag{10.21}$$

$$p_{22} = \alpha_3\beta_2 + \alpha_4\beta_4 \tag{10.22}$$

Thus the overall channel matrix

$$[P(Z|X)] = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \tag{10.23}$$

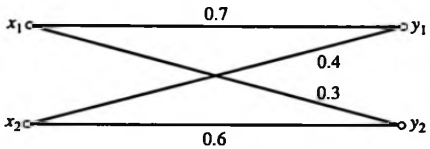


FIGURE 10.3 Binary channel.

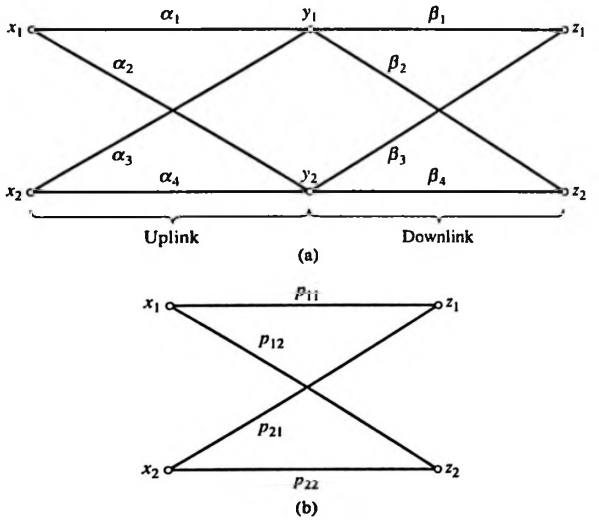


FIGURE 10.4 Two-hop satellite system. (a) Binary satellite channel. (b) Composite satellite channel.

can be represented by the matrix multiplication

$$[P(Z|X)] = \begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{bmatrix} \begin{bmatrix} \beta_1 & \beta_2 \\ \beta_3 & \beta_4 \end{bmatrix} \tag{10.24}$$

The right-hand side of the preceding expression is simply the uplink channel matrix multiplied by the downlink channel matrix.

10.1.4 Joint and Conditional Entropy

If we use the input probabilities $p(x_i)$, the output probabilities $p(y_j)$, the transition probabilities $p(y_j|x_i)$, and the joint probabilities $p(x_i, y_j)$, we can define several different entropy functions for a channel with n inputs and m outputs. These are

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{10.25}$$

$$H(Y) = -\sum_{j=1}^{m} p(y_j) \log_2 p(y_j) \tag{10.26}$$

$$H(Y|X) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log_2 p(y_j|x_i) \tag{10.27}$$

and

$$H(X, Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log_2 p(x_i, y_j) \tag{10.28}$$

An important and useful entropy, $H(X|Y)$, which is sometimes called *equivocation*, is defined as

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2 p(x_i|y_j) \tag{10.29}$$

These entropies are easily interpreted. $H(X)$ is the average uncertainty of the source, whereas $H(Y)$ is the average uncertainty of the received symbol. Similarly, $H(X|Y)$ is a measure of our average uncertainty of the transmitted symbol after we have received a symbol. The function $H(Y|X)$ is the average uncertainty of the received symbol given that X was transmitted. The joint entropy $H(X, Y)$ is the average uncertainty of the communication system as a whole.

Two important and useful relationships, which can be obtained directly from the definitions of the various entropies, are

$$H(X, Y) = H(X|Y) + H(Y) \tag{10.30}$$

and

$$H(X, Y) = H(Y|X) + H(X) \tag{10.31}$$

These are developed in Problem 10.13.

10.1.5 Channel Capacity

Consider for a moment an observer at the channel output. The observer's average uncertainty concerning the channel input will have some value before the reception of an output, and this average uncertainty of the input will usually decrease when the output is received. In other words, $H(X|Y) \leq H(X)$. The decrease in the observer's average uncertainty of the transmitted signal when the output is received is a measure of the average transmitted information. This is defined as *mutual information* $I(X; Y)$. Thus

$$I(X; Y) = H(X) - H(X|Y) \tag{10.32}$$

It follows from (10.30) and (10.31) that we can also write (10.32) as

$$I(X; Y) = H(Y) - H(Y|X) \tag{10.33}$$

It should be observed that mutual information is a function of the source probabilities as well as of the channel transition probabilities. It is easy to show mathematically that

$$H(X) \geq H(X|Y) \tag{10.34}$$

by showing that

$$H(X|Y) - H(X) = -I(X; Y) \leq 0 \tag{10.35}$$

Substitution of (10.29) for $H(X|Y)$ and (10.25) for $H(X)$ allows us to write $-I(X; Y)$ as

$$-I(X; Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2 \frac{p(x_i)}{p(x_i|y_j)} \tag{10.36}$$

Since

$$\log_2 x = \frac{\ln x}{\ln 2} \tag{10.37}$$

and

$$\frac{p(x_i)}{p(x_i|y_j)} = \frac{p(x_i)p(y_j)}{p(x_i, y_j)} \tag{10.38}$$

we can write $-I(X;Y)$ as

$$-I(X;Y) = \frac{1}{\ln 2}\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\ln\frac{p(x_i)p(y_j)}{p(x_i, y_j)} \tag{10.39}$$

In order to carry the derivation further, we need the often-used inequality

$$\ln(x) \leq x - 1 \tag{10.40}$$

which we can easily prove by considering the function

$$f(x) = \ln(x) - (x - 1) \tag{10.41}$$

The derivative of $f(x)$

$$\frac{df}{dx} = \frac{1}{x} - 1 \tag{10.42}$$

is equal to zero at $x = 1$. It follows that $f(1) = 0$ is the maximum value of $f(x)$, since we can make $f(x)$ as small as we wish by choosing x sufficiently large. Using the inequality (10.42) in (10.39) results in

$$-I(X;Y) \leq \frac{1}{\ln 2}\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\left[\frac{p(x_i)p(y_j)}{p(x_i, y_j)} - 1\right] \tag{10.43}$$

which yields

$$-I(X;Y) \leq \frac{1}{\ln 2}\left[\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i)p(y_j) - \sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\right] \tag{10.44}$$

Since both the double sums equal 1, we have the desired result

$$-I(X;Y) \leq 0 \tag{10.45}$$

Thus we have shown that mutual information is always nonnegative and, consequently, that $H(X) \geq H(X|Y)$.

The *channel capacity* C is defined as the maximum value of mutual information, which is the maximum average information per symbol that can be transmitted through the channel. This is

$$C = \max\,[I(X;Y)] \tag{10.46}$$

The maximization is with respect to the source probabilities, since the transition probabilities are fixed by the channel. However, the channel capacity is a function of only the channel transition probabilities, since the maximization process eliminates the dependence on the source probabilities. The following examples illustrate the method.

EXAMPLE 10.4

Find the channel capacity of the noiseless discrete channel illustrated in Figure 10.5.

Solution: We start with

$$I(X;Y) = H(X) - H(X|Y)$$

and write

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2 p(x_i|y_j) \tag{10.47}$$

For the noiseless channel, all $p(x_i, y_j)$ and $p(x_i|y_j)$ are zero unless $i = j$. For $i = j$, $p(x_i|y_j) = 1$. Thus $H(X|Y)$ is zero for the noiseless channel, and

$$I(X;Y) = H(X) \tag{10.48}$$

We have seen that the entropy of a source is maximum if all source symbols are equally likely. Thus

$$C = \sum_{i=1}^{n}\frac{1}{n}\log_2 n = \log_2 n \tag{10.49}$$

■

EXAMPLE 10.5

Find the channel capacity of the *binary symmetric channel* illustrated in Figure 10.6.

Solution: This problem has considerable practical importance in the area of binary digital communications. We will determine the capacity by maximizing

$$I(X;Y) = H(Y) - H(Y|X)$$

where

$$H(Y|X) = -\sum_{i=1}^{2}\sum_{j=1}^{2} p(x_i, y_j)\log_2 p(y_j|x_i) \tag{10.50}$$

Using the source and transition probabilities defined in Figure 10.6, we obtain

$$\begin{aligned} H(Y|X) = &-\alpha p\log_2 p - (1-\alpha)p\log_2 p \\ &-\alpha q\log_2 q - (1-\alpha)q\log_2 q \end{aligned} \tag{10.51}$$

or

$$H(Y|X) = -p\log_2 p - q\log_2 q \tag{10.52}$$

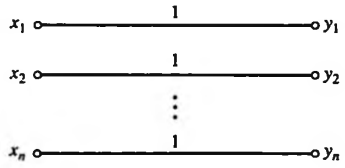


FIGURE 10.5 Noiseless channel.

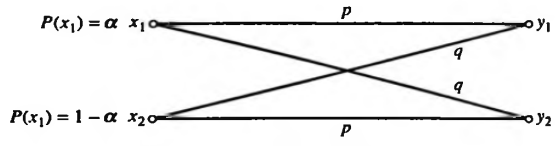


FIGURE 10.6 Binary symmetric channel.

Thus

$$I(X;Y) = H(Y) + p\log_2 p + q\log_2 q \tag{10.53}$$

which is maximum when $H(Y)$ is maximum. Since the system output is binary, $H(Y)$ is maximum when each output has a probability of $\frac{1}{2}$ and, for a symmetrical channel, is achieved for *equally likely inputs*. For this case, $H(Y)$ is unity, and the channel capacity is

$$C = 1 + p\log_2 p + q\log_2 q = 1 - H(p) \tag{10.54}$$

where $H(p)$ is defined in (10.4).

The capacity of a binary symmetric channel is sketched in Figure 10.7. As expected, if $p = 0$ or 1, the channel output is completely determined by the channel input, and the capacity is 1 bit per symbol. If p is equal to 0.5, an input symbol yields either output with equal probability, and the capacity is zero. ■

It is worth noting that the capacity of the channel illustrated in Figure 10.5 is most easily found by starting with (10.32), while the capacity of the channel illustrated in Figure 10.6 is most easily found starting with (10.33). Choosing the appropriate expression for $I(X;Y)$ can often save considerable effort. It sometimes takes insight and careful study of a problem to choose the expression for $I(X;Y)$ that yields the capacity with minimum computational effort.

The error probability P_E of a binary symmetric channel is easily computed. From

$$P_E = \sum_{i=1}^{2} p(e|x_i)p(x_i) \tag{10.55}$$

where $p(e|x_i)$ is the error probability given input x_i, we have

$$P_E = qp(x_1) + qp(x_2) \tag{10.56}$$

Thus

$$P_E = q$$

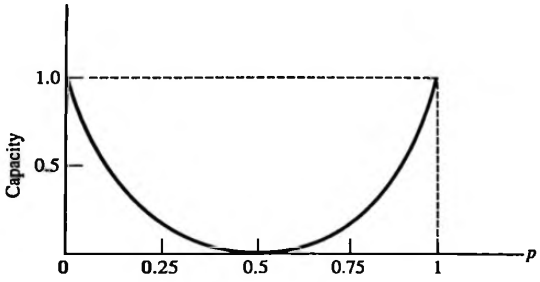


FIGURE 10.7 Capacity of a binary symmetric channel.

which states that the *unconditional error probability* P_E is equal to the conditional error probability $p(y_j|x_i)$, $i \neq j$.

In Chapter 7 we showed that P_E is a decreasing function of the energy of the received symbols. Since the symbol energy is the received power multiplied by the symbol period, it follows that *if the transmitter power is fixed, the error probability can be reduced by decreasing the source rate*. This can be accomplished by removing the redundancy at the source through a process called *source encoding*.

EXAMPLE 10.6

In Chapter 7 we showed that for binary coherent FSK systems, the probability of symbol error is the same for each transmitted symbol. Thus a binary symmetrical channel model is a suitable model for FSK transmission. Assume that the transmitter power is 1000 W and that the attenuation in the channel from transmitter to detector input is 30 dB. Also assume that the source rate r is 10,000 symbols per second and that the noise power spectral density N_0 is 2×10^{-5} W/Hz. Determine the channel matrix.

Solution: Since the attenuation is 30 dB, the signal power P_R at the input to the detector is

$$P_R = (1000)(10^{-3}) = 1 \text{ W} \tag{10.57}$$

This corresponds to a received energy per symbol of

$$E_s = P_R T = \frac{1}{10{,}000} = 10^{-4} \text{ J} \tag{10.58}$$

In Chapter 7 we saw that the error probability for an FSK receiver is

$$P_E = Q\left[\sqrt{\frac{E_s}{N_0}}\right] \tag{10.59}$$

which, with the given values, is $P_E = 0.0127$. Thus the channel matrix is

$$[P(Y|X)] = \begin{bmatrix} 0.9873 & 0.0127 \\ 0.0127 & 0.9873 \end{bmatrix} \tag{10.60}$$

It is interesting to compute the change in the channel matrix resulting from a moderate reduction in source symbol rate with all other parameters held constant. If the source symbol rate is reduced 25 percent to 7500 symbols per second, the received energy per symbol is

$$E_s = \frac{1}{7500} = 1.333 \times 10^{-4} \text{ J} \tag{10.61}$$

With the other given parameters, the symbol error probability becomes $P_E = 0.0049$, which yields the channel matrix

$$[P(Y|X)] = \begin{bmatrix} 0.9951 & 0.0049 \\ 0.0049 & 0.9951 \end{bmatrix} \tag{10.62}$$

Thus the 25 percent reduction in source symbol rate results in an improvement of the system symbol error probability by a factor of almost 3. In Section 10.2 we will investigate a technique that sometimes allows the source symbol rate to be reduced without reducing the source information rate. ■

10.2 SOURCE CODING

We determined in the preceding section that the information from a source that produced different symbols according to some probability scheme could be described by the entropy $H(X)$. Since entropy has units of bits per symbol, we also must know the symbol

rate in order to specify the source information rate in bits per second. In other words, the source information rate R_s is given by

$$R_s = rH(X) \text{ bits/s} \tag{10.63}$$

where $H(X)$ is the source entropy in bits per symbol and r is the symbol rate in symbols per second.

Let us assume that this source is the input to a channel with capacity C bits per symbol or SC bits per second, where S is the available symbol rate for the channel. An important theorem of information theory, the *noiseless coding theorem*, sometimes referred to as *Shannon's first theorem*, is stated: *Given a channel and a source that generates information at a rate less than the channel capacity, it is possible to encode the source output in such a manner that it can be transmitted through the channel.* A proof of this theorem is beyond the scope of this introductory treatment of information theory and can be found in any of the standard information theory textbooks.[2] However, we demonstrate the theorem by a simple example.

10.2.1 An Example of Source Encoding

Let us consider a discrete binary source that has two possible outputs A and B that have probabilities 0.9 and 0.1, respectively. Assume also that the source rate r is 3.5 symbols per second. The source output is connected to a binary channel that can transmit a binary 0 or 1 at a rate of 2 symbols per second with negligible error, as shown in Figure 10.8. Thus, from Example 10.5 with $p = 1$, the channel capacity is 1 bit per symbol which, in this case, is an information rate of 2 bits per second.

It is clear that the source *symbol* rate is greater than the channel capacity, so the source symbols cannot be placed directly into the channel. However, the source entropy is

$$H(X) = -0.1 \log_2 0.1 - 0.9 \log_2 0.9 = 0.469 \text{ bits/symbol} \tag{10.64}$$

which corresponds to a source *information* rate of

$$rH(X) = 3.5(0.469) = 1.642 \text{ bits/s} \tag{10.65}$$

Thus, the *information* rate is less than the channel capacity, so transmission is possible.

Transmission is accomplished by the process called *source encoding*, whereby codewords are assigned to n-symbol groups of source symbols. The shortest codeword is assigned to the most probable group of source symbols, and the longest codeword is assigned to the least probable group of source symbols. Thus source encoding decreases the average symbol rate, which allows the source to be matched to the channel. The n-symbol groups of source symbols are known as the nth-order extension of the original source.

Table 10.1 illustrates the first-order extension of the original source. Clearly, the symbol rate at the encoder output is equal to the symbol rate of the source. Thus the symbol rate at the channel input is still larger than the channel can accommodate.

The second-order extension of the original source which is formed by taking the source symbols two at a time, as illustrated in Table 10.2. The average word length $\bar{L}$ for

[2]See for example, Gallagher (1968).

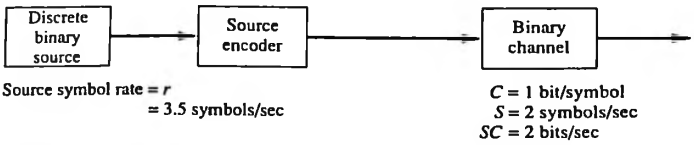


FIGURE 10.8 Transmission scheme.

this case is

$$\overline{L} = \sum_{i=1}^{2^n} p(x_i) l_i = 1.29 \tag{10.66}$$

where $p(x_i)$ is the probability of the ith symbol of the extended source and l_i is the length of the codeword corresponding to the ith symbol. Since the source is binary, there are 2^n symbols in the extended source output, each of length n. Thus, for the second-order extension,

$$\frac{\overline{L}}{n} = \frac{1}{n} \sum P(x_i) l_i = \frac{1.29}{2} = 0.645 \text{ code symbols/source symbol} \tag{10.67}$$

and the symbol rate at the encoder output is

$$r \frac{\overline{L}}{n} = 3.5(0.645) = 2.258 \text{ code symbols/second} \tag{10.68}$$

which is still greater than the 2 symbols per second that the channel can accept. It is clear that the symbol rate has been reduced, and this provides motivation to try again.

Table 10.3 shows the third-order extension. For this case, the source symbols are grouped three at a time. The average word length $\overline{L}$ is 1.598, and

$$\frac{\overline{L}}{n} = \frac{1}{n} \sum P(x_i) l_i = \frac{1.598}{3} = 0.533 \text{ code symbols/source symbol} \tag{10.69}$$

The symbol rate at the encoder output is

$$r \frac{\overline{L}}{n} = 3.5(0.533) = 1.864 \text{ code symbols/second} \tag{10.70}$$

This rate can be accepted by the channel and therefore transmission is possible using the third-order source extension.

It is worth noting in passing that if the source symbols appear at a constant rate, the code symbols at the encoder output do not appear at a constant rate. As is apparent

TABLE 10.1 First-order Extension

Source Symbol	Symbol Probability $P(x_i)$	Codeword	l_i	$P(x_i)l_i$
A	0.9	0	1	0.9
B	0.1	1	1	0.1
				$\overline{L} = 1.0$

TABLE 10.2 Second-order Source Extension

Source Symbol	Symbol Probability $P(x_i)$	Codeword	l_i	$P(x_i)l_i$
AA	0.81	0	1	0.81
AB	0.09	10	2	0.18
BA	0.09	110	3	0.27
BB	0.01	111	3	0.03

in Table 10.3, the source output *AAA* results in a single symbol at the encoder output, whereas the source output *BBB* results in five symbols at the encoder output. Thus symbol buffering must be provided at the encoder output if the symbol rate into the channel is to be constant.

Figure 10.9 shows the behavior of $\overline{L}/n$ as a function of n. One can see that $\overline{L}/n$ always exceeds the source entropy and converges to the source entropy for large n. This is a fundamental result of information theory.

To illustrate the method used to select the codewords in this example, let us look at the general problem of source encoding.

10.2.2 Several Definitions

Before we discuss in detail the method of deriving codewords, we pause to make a few definitions that will clarify our work.

Each codeword is constructed from an *alphabet* that is a collection of symbols used for communication through a channel. For example, a binary codeword is constructed from a two-symbol alphabet, wherein the two symbols are usually taken as 0 and 1. The *word length* of a codeword is the number of symbols in the codeword.

There are several major subdivisions of codes. For example, a code can be either *block* or *nonblock*. A block code is one in which each block of source symbols is encoded into a fixed-length sequence of code symbols. A *uniquely decipherable* code is a block code in which the codewords may be deciphered without using spaces. These codes can be further classified as *instantaneous* or *noninstantaneous*, according to whether or not it is

TABLE 10.3 Third-order Source Extension

Source Symbol	Symbol Probability $P(x_i)$	Codeword	l_i	$P(x_i)l_i$
AAA	0.729	0	1	0.729
AAB	0.081	100	3	0.243
ABA	0.081	101	3	0.243
BAA	0.081	110	3	0.243
ABB	0.009	11100	5	0.045
BAB	0.009	11101	5	0.045
BBA	0.009	11110	5	0.045
BBB	0.001	11111	5	0.005

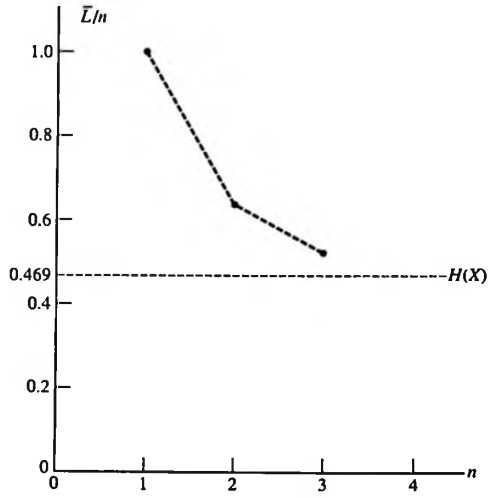


FIGURE 10.9 Behavior of $\bar{L}/n$.

possible to decode each word in sequence without reference to succeeding code symbols. Alternatively, noninstantaneous codes require reference to succeeding code symbols, as illustrated in Table 10.4. It should always be remembered that a noninstantaneous code can be uniquely decipherable.

A useful measure of goodness of a source code is the *efficiency*, which is defined as the ratio of the minimum average word length of the codewords $\bar{L}_{\min}$ to the average word length of the codeword $\bar{L}$. Thus

$$\text{Efficiency} = \frac{\bar{L}_{\min}}{\bar{L}} = \frac{\bar{L}_{\min}}{\sum_{i=1}^{n} p(x_i) l_i} \tag{10.71}$$

where $p(x_i)$ is the probability of the ith source symbol and l_i is the length of the codeword corresponding to the ith source symbol. It can be shown that the minimum average word length is given by

$$\bar{L}_{\min} = \frac{H(X)}{\log_2 D} \tag{10.72}$$

where $H(X)$ is the entropy of the message ensemble being encoded and D is the number

TABLE 10.4 Instantaneous and Noninstantaneous Codes

Source Symbol	Code 1 Noninstantaneous	Code 2 Instantaneous
x_1	0	0
x_2	01	10
x_3	011	110
x_4	0111	1110

of symbols in the encoding alphabet. This yields

$$\text{Efficiency} = \frac{H(X)}{\overline{L}\log_2 D} \tag{10.73}$$

or

$$\text{Efficiency} = \frac{H(X)}{\overline{L}} \tag{10.74}$$

for a *binary* alphabet. Note that if the efficiency of a code is 100 percent, the average word length $\overline{L}$ is equal to the entropy, $H(X)$.

10.2.3 Entropy of an Extended Binary Source

In many problems of practical interest, the efficiency is improved by coding the nth-order extension of a source. This is exactly the scheme used in the example of source encoding in a previous subsection. Computation of the efficiency of each of the three schemes used involves calculating the efficiency of the extended source. The efficiency can, of course, be calculated directly, using the symbol probabilities of the extended source, but there is an easier method.

The entropy of the nth-order extension of a discrete memoryless source $H(X^n)$ is given by

$$H(X^n) = nH(X) \tag{10.75}$$

This is easily shown by representing a message sequence from the output of the nth source extension as $(i_1, i_2, \ldots, i_n)$ where i_k can take on one of two states with probability p_{i_k}. The entropy of the nth-order extension of a binary source is

$$H(X^n) = -\sum_{i_1=1}^{2}\sum_{i_2=1}^{2}\cdots\sum_{i_n=1}^{2}(p_{i_1}p_{i_2}\cdots p_{i_n})\log_2(p_{i_1}p_{i_2}\cdots p_{i_n}) \tag{10.76}$$

or

$$H(X^n) = -\sum_{i_1=1}^{2}\sum_{i_2=1}^{2}\cdots\sum_{i_n=1}^{2}(p_{i_1}p_{i_2}\cdots p_{i_n})(\log_2 p_{i_1} + \log_2 p_{i_2} + \cdots + \log_2 p_{i_n}) \tag{10.77}$$

We can write the preceding expression as

$$\begin{aligned} H(X^n) = &-\sum_{i_1=1}^{2} p_{i_1}\log_2 p_{i_1}\left[\sum_{i_2=1}^{2} p_{i_2}\sum_{i_3=1}^{2} p_{i_3}\cdots\sum_{i_n=1}^{2} p_{i_n}\right] \\ &-\left[\sum_{i_1=1}^{2} p_{i_1}\right]\sum_{i_2=1}^{2} p_{i_2}\log_2 p_{i_2}\left[\sum_{i_3=1}^{2} p_{i_3}\cdots\sum_{i_n=1}^{2} p_{i_n}\right]\cdots \\ &-\left[\sum_{i_1=1}^{2} p_{i_1}\sum_{i_2=1}^{2} p_{i_2}\cdots\sum_{i_{n-1}=1}^{2} p_{i_{n-1}}\right]\sum_{i_n=1}^{2} p_{i_n}\log_2 p_{i_n} \end{aligned} \tag{10.78}$$

Each term in brackets is equal to 1. Thus

$$H(X^n) = -\sum_{k=1}^{n}\sum_{i_k=1}^{2} p_{i_k}\log_2 p_{i_k} = \sum_{k=1}^{n} H(X) \tag{10.79}$$

which yields

$$H(X^n) = nH(X)$$

This is a general result for all D. The efficiency of the extended source is

$$\text{Efficiency} = \frac{nH(X)}{\overline{L}} \tag{10.80}$$

If efficiency tends to 100 percent as n approaches infinity, it follows that $\overline{L}/n$ tends to the entropy of the extended source. This is exactly the observation made from Figure 10.9.

10.2.4 Shannon-Fano Encoding

There are several methods of encoding a source output so that an instantaneous code results. We consider two such methods here. First, we consider the Shannon-Fano method, which is very easy to apply and usually yields source codes having reasonably high efficiency. We then consider the Huffman encoding technique, which yields the source code having the shortest average word length for a given source entropy.

Assume that we are given a set of source outputs that are to be encoded. These source outputs are first ranked in order of nonincreasing probability of occurrence, as illustrated in Figure 10.10. The set is then partitioned into two sets (indicated by line A-A') that are as close to equiprobable as possible, and 0's are assigned to the upper set and 1's to the lower set, as shown in the first column of the codewords. This process is continued, each time partitioning the sets with as nearly equal probabilities as possible, until further partitioning is not possible. This scheme will give a 100 percent efficient code if the partitioning always results in equiprobable sets; otherwise, the code will have an efficiency of less than 100 percent. For this particular example,

$$\text{Efficiency} = \frac{H(X)}{\overline{L}} = \frac{2.75}{2.75} = 1 = 100 \text{ percent} \tag{10.81}$$

since equiprobable partitioning is always possible.

Source words	Probability	Code word	(Length)·(Probability)	
X_1	0.2500	00	2 (0.25)	= 0.50
X_2	0.2500	01	2 (0.25)	= 0.50
		A -------- A'		
X_3	0.1250	100	3 (0.125)	= 0.375
X_4	0.1250	101	3 (0.125)	= 0.375
X_5	0.0625	1100	4 (0.0625)	= 0.25
X_6	0.0625	1101	4 (0.0625)	= 0.25
X_7	0.0625	1110	4 (0.0625)	= 0.25
X_8	0.0625	1111	4 (0.0625)	= 0.25
			Average word length	= 2.75

FIGURE 10.10 Shannon-Fano encoding.

10.2.5 Huffman Encoding

Huffman encoding results in an optimum code in the sense that the Huffman code has the minimum average word length for a source of given entropy. Thus it is the code that has the highest efficiency. We will illustrate the Huffman coding procedure using the same source output of eight messages used to illustrate the Shannon-Fano encoding procedure.

Figure 10.11 illustrates the Huffman encoding procedure. The source output consists of messages X_1, X_2, X_3, X_4, X_5, X_6, X_7, and X_8. They are listed in order of nonincreasing probability, as was done for Shannon-Fano encoding. The first step of the Huffman procedure is to combine the two source messages having the lowest probability, X_7 and X_8.

The upper message, X_7, is assigned a binary 0 as the last symbol in the codeword, and the lower message, X_8, is assigned a binary 1 as the last symbol in the codeword. The combination of X_7 and X_8 can be viewed as a composite message having a probability equal to the sum of the probabilities of X_7 and X_8, which in this case is 0.1250, as shown. This composite message is denoted X_4'. After this initial step, the new set of messages, denoted X_1, X_2, X_3, X_4, X_5, X_6, and X_4' are arranged in order of nonincreasing probability. Note that X_4' could be placed at any point between X_2 and X_5, although it was given the name X_4' because it was placed after X_4. The same procedure is then applied once again. The messages X_5 and X_6 are combined. The resulting composite message is combined with X_4'. This procedure is continued as far as possible. The resulting tree structure is then traced in reverse to determine the codewords. The resulting codewords are shown in Figure 10.11.

The codewords resulting from the Huffman procedure are different from the codewords resulting from the Shannon-Fano procedure because at several points the placement of composite messages resulting from previous combinations was arbitrary. The assignment of binary 0's or binary 1's to the upper or lower messages was arbitrary. Note, however, that the average word length is the same for both procedures. This must be the case for the example chosen because the Shannon-Fano procedure yielded 100 percent efficiency and the Huffman procedure can be no worse. There are cases in which the two procedures do not result in equal average word lengths.

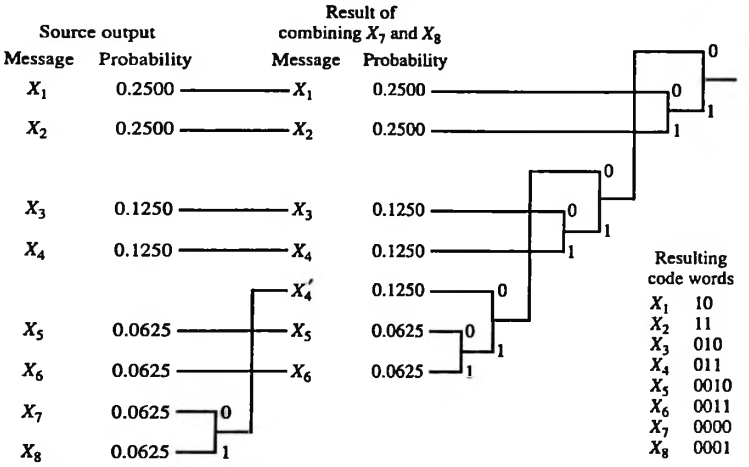


FIGURE 10.11 Example of Huffman encoding.

10.3 RELIABLE COMMUNICATION IN THE PRESENCE OF NOISE

We now turn our attention to methods for achieving reliable communication in the presence of noise by combating the effects of that noise. We undertake our study with a promise from Claude Shannon of considerable success. *Shannon's theorem*, sometimes referred to as the *fundamental theorem of information theory*, is stated: *Given a discrete memoryless channel (each symbol is perturbed by noise independently of all other symbols) with capacity C and a source with positive rate R, where $R < C$, there exists a code such that the output of the source can be transmitted over the channel with an arbitrarily small probability of error.*

Thus Shannon's theorem predicts essentially error-free transmission in the presence of noise. Unfortunately, the theorem tells us only of the existence of codes and tells nothing of how to construct these codes.

Before we start our study of constructing codes for noisy channels, we will take a minute to discuss the continuous channel. This detour will yield considerable insight that will prove useful.

10.3.1 The Capacity of a White Additive Gaussian Noise Channel

The capacity, in bits per second, of a continuous channel with additive white Gaussian noise is given by

$$C_c = B \log_2 \left(1 + \frac{S}{N}\right) \tag{10.82}$$

where B is the channel bandwidth in hertz and S/N is the signal-to-noise power ratio. This particular formulation is known as the *Shannon-Hartley law*. The subscript is used to distinguish (10.82) from (10.46). Capacity, as expressed by (10.46), has units of bits per symbol, while (10.82) has units of bits per second.

The trade-off between bandwidth and signal-to-noise ratio can be seen from the *Shannon-Hartley law*. For infinite signal-to-noise ratio, which is the noiseless case, the capacity is infinite for any nonzero bandwidth. We will show, however, that the capacity cannot be made arbitrarily large by increasing bandwidth if noise is present.

In order to understand the behavior of the Shannon-Hartley law for the large-bandwidth case, it is desirable to place (10.82) in a slightly different form. The energy per bit E_b is equal to the bit time T_b multiplied by the signal power S. At capacity, the bit rate R_b is equal to the capacity. Thus $T_b = 1/C_c$ seconds per bit. This yields, at capacity,

$$E_b = ST_b = \frac{S}{C_c} \tag{10.83}$$

The total noise power in bandwidth B is given by

$$N = N_0 B \tag{10.84}$$

where N_0 is the single-sided noise power spectral density in watts per hertz. The signal-to-noise ratio can therefore be expressed as

$$\frac{S}{N} = \frac{E_b}{N_0}\frac{C_c}{B} \tag{10.85}$$

This allows the Shannon-Hartley law to be written in the equivalent form

$$\frac{C_c}{B} = \log_2\left(1 + \frac{E_b}{N_0}\frac{C_c}{B}\right) \tag{10.86}$$

Solving for E_b/N_0 yields

$$\frac{E_b}{N_0} = \frac{B}{C_c}(2^{C_c/B} - 1) \tag{10.87}$$

This expression establishes performance of the ideal system. For the case in which $B >> C_c$

$$2^{C_c/B} = e^{(C_c/B)\ln 2} \cong 1 + \frac{C_c}{B}\ln 2 \tag{10.88}$$

where the approximation $e^x \cong 1 + x,\ x \ll 1$, has been used. Substitution of (10.88) into (10.87) gives

$$\frac{E_b}{N_0} \cong \ln 2 = -1.6 \text{ dB},\ B \gg C_c \tag{10.89}$$

Thus, for the ideal system, in which $R_b = C_c$, E_b/N_0 approaches the limiting value of -1.6 dB as the bandwidth grows without bound.

A plot of E_b/N_0, expressed in decibels, as a function of R_b/B is illustrated in Figure 10.12. The ideal system is defined by $R_b = C_c$ and corresponds to (10.87). There are two regions of interest. The first region, for which $R_b < C_c$, is the region in which arbitrarily small error probabilities can be obtained. Clearly this is the region in which we wish to operate. The other region, for which $R_b > C_c$, does not allow the error probability to be made arbitrarily small.

An important trade-off can be deduced from Figure 10.12. If the bandwidth factor R_b/B is large so that the bit rate is much greater than the bandwidth, then a significantly larger value of E_b/N_0 is necessary to ensure operation in the $R_b < C_c$ region than is the case if R_b/B is small. Stated another way, assume that the source bit rate is fixed at R_b bits per second and the available bandwidth is large so that $B \gg R_b$. For this case, operation in

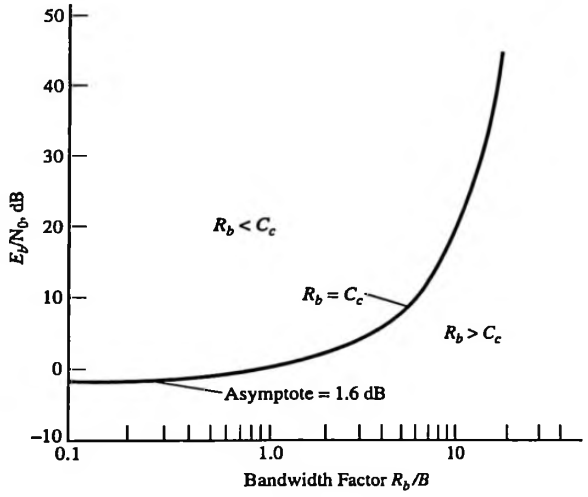


FIGURE 10.12 $R_b = C_c$ relationship for additive white Gaussian noise channel.

the $R_b < C_c$ region requires only that E_b/N_0 is slightly greater than -1.6 dB. The required signal power is

$$S \cong R_b N_0 W (\ln 2) \tag{10.90}$$

This is the minimum signal power for operation in the $R_b < C_c$ region. Therefore, operation in this region is desired for *power-limited operation*.

Now assume that bandwidth is limited so that $R_b \gg B$. Figure 10.12 shows that a much larger value of E_b/N_0 is necessary for operation in the $R_b < C_c$ region. Thus the required signal power is much greater than that given by (10.90). This is referred to as *bandwidth-limited operation*.

The preceding paragraphs illustrate that, at least in the additive white Gaussian noise channel where the Shannon-Hartley law applies, a trade-off exists between power and bandwidth. This trade-off is of fundamental importance in the design of communication systems.

10.3.2 Reliable Communication Using Orthogonal Signals

Realizing that we can theoretically achieve perfect system performance, even in the presence of noise, we start our search for system configurations that yield the performance promised by Shannon's theorem. Actually one such system was analyzed in Chapter 9. Orthogonal signals were chosen for transmission through the channel, and a correlation receiver structure was chosen for demodulation. The system performance is illustrated in Figure 9.7. Shannon's bound is clearly illustrated.

10.3.3 Block Coding for Binary Systems

An alternative technique for approaching Shannon's limit is to use coding schemes that allow for detection and correction of errors. One class of codes for this purpose is known as *block codes*. In order to understand the meaning of a block code, let us consider a source that produces a serial stream of binary symbols at a rate of R symbols per second. Assume that these symbols are grouped into blocks T seconds long, so that each block contains $RT = k$ source or information symbols. To each of these k-symbol blocks is added a group of redundant check symbols to produce a codeword n symbols long. The $n - k$ check symbols should supply sufficient information to the decoder to allow for the correction of most errors that might occur in the channel. An encoder that operates in this manner is said to produce an (n, k) block code.

Codes can either correct or merely detect errors, depending on the amount of redundancy contained in the check symbols. Codes that can correct errors are known as *error-correcting codes*. Codes that can only detect errors are also useful. A feedback channel can be used to request a retransmission of the codeword found to be in error, and often the error can be corrected. We will discuss error-detection feedback later. Often the codewords with detected errors are simply discarded by the decoder. This is practical when errors are more serious than a lost codeword.

An understanding of how codes can detect and correct errors can be gained from a geometric point of view. A binary codeword is a sequence of 1's and 0's that is n symbols in length. The *Hamming weight* $w(s_j)$ of codeword s_j is defined as the number if 1's in that codeword. The *Hamming distance* $d(s_i, s_j)$ or d_{ij} between codewords s_i and s_j is defined as the number of positions in which s_i and s_j differ. It follows that Hamming distance can

be written in terms of Hamming weight as

$$d_{ij} = w(s_i \oplus s_j) \tag{10.91}$$

where the symbol $\oplus$ denotes modulo-2 addition, which is binary addition without a carry.

EXAMPLE 10.7

Compute the Hamming distance between s_1 and 101101 and $s_2 = 001100$.

Solution: Since

$$101101 \oplus 001100 = 100001$$

we have

$$d_{12} = w(100001) = 2$$

which simply means that s_1 and s_2 differ in 2 positions. ■

A geometric representation of two codewords is shown in Figure 10.13. The C's represent two codewords that are distance 5 apart. The codeword on the left is the reference codeword. The first "×" to the right of the reference represents a binary sequence distance 1 from the reference codeword, where distance is understood to denote the Hamming distance. The second "×" to the right of the reference codeword is distance 2 from the reference, and so on. Assuming that the two codewords shown are the closest in Hamming distance of all the codewords for a given code, the code is then a distance 5 code. Figure 10.13 illustrates the concept of a minimum-distance decoding, in which a given received sequence is assigned to the codeword closest, in Hamming distance, to the received sequence. A minimum distance decoder will therefore assign the ×'s to the left of the vertical line to the codeword on the left and the ×'s to the right of the vertical line to the codeword on the right, as shown.

We deduce that a minimum-distance decoder can always correct as many as e errors, where e is the largest integer not to exceed

$$\frac{1}{2}(d_m - 1)$$

where d_m is the minimum distance between codewords. It follows that if d_m is odd, all received words can be assigned to a codeword. However, if d_m is even, a received word can lie halfway between two codewords. For this case, errors are detected that cannot be corrected.

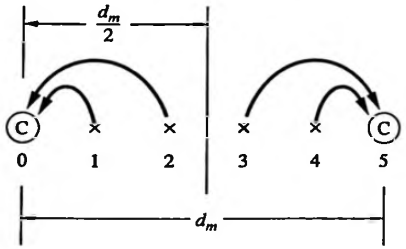


FIGURE 10.13 Geometric representation of two codewords.

EXAMPLE 10.8

A code consists of eight codewords [0001011, 1110000, 1000110, 1111011, 0110110, 1001101, 0111101, 0000000]. If 1101011 is received, what is the decoded codeword?

Solution: The decoded codeword is the codeword closest in Hamming distance to 1101011. The calculations are

$$w(0001011 \oplus 1101011) = 2 \quad w(0110110 \oplus 1101011) = 5$$

$$w(1110000 \oplus 1101011) = 4 \quad w(1001101 \oplus 1101011) = 3$$

$$w(1000110 \oplus 1101011) = 4 \quad w(0111101 \oplus 1101011) = 4$$

$$w(1111011 \oplus 1101011) = 1 \quad w(0000000 \oplus 1101011) = 5$$

The decoded codeword is therefore 1111011. ■

10.3.4 Single-Parity-Check Codes

A simple code capable of detecting, but not capable of correcting, single errors is formed by adding one check symbol to each block of k information symbols. This yields a $(k+1, k)$ code. Thus the rate is $k/(k+1)$. The added symbol is called a *parity-check symbol*, and it is added so that the Hamming weight of all codewords is odd or even. If the received word contains an *even* number of errors, the decoder will not detect the errors. If the number of errors is *odd*, the decoder will detect that an odd number of errors, most likely one, has been made.

10.3.5 Repetition Codes

The simplest code that allows for correction of errors consists of transmitting each symbol n times, which results in $n-1$ check symbols. This technique produces an $(n, 1)$ code having two codewords—one of all 0's and one of all 1's. A received word is decoded as a 0 if the majority of the received symbols are 0's and as a 1 if the majority are 1's. This is equivalent to minimum-distance decoding, wherein $\frac{1}{2}(n-1)$ errors can be corrected. Repetition codes have great error-correcting capability if the symbol error probability is low but have the disadvantage of transmitting many redundant symbols. For example, if the information rate of the source is R bits per symbol, the rate R_c out of the encoder is

$$R_c = \left(\frac{k}{n}\right) R = \frac{1}{n} R \text{ bits/symbol} \tag{10.92}$$

The factor k/n is called the *code rate*.

The process of repetition coding for a rate $\frac{1}{3}$ repetition code is illustrated in detail in Figure 10.14. The encoder maps the data symbols 0 and 1 into the corresponding codewords 000 and 111. There are eight possible received sequences, as shown. The mapping from the transmitted sequence to the received sequence is random, and the statistics of the mapping are determined by the channel characteristics derived in Chapters 7 and 8. The decoder maps the received sequence into one of the two codewords by a minimum Hamming distance decoding rule. Each decoded codeword corresponds to a data symbol, as shown.

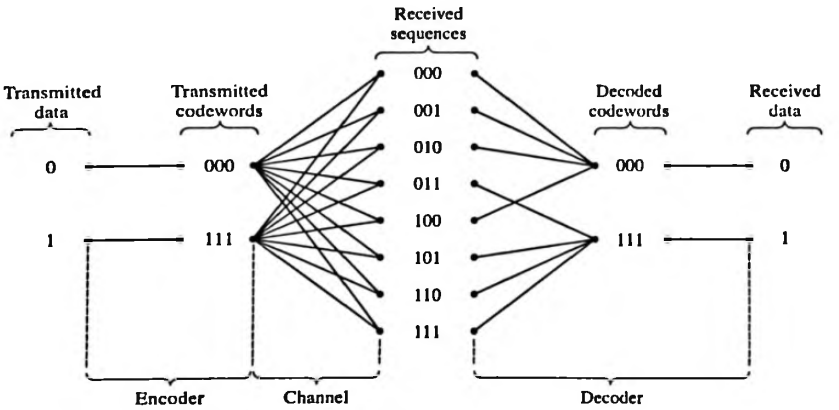


FIGURE 10.14 Example of rate $\frac{1}{3}$ repetition code.

EXAMPLE 10.9

Investigate the error-correcting capability of a repetition code having a code rate of $\frac{1}{3}$.

Solution: Assume that the code is used with a binary symmetric channel with a conditional error probability equal to $(1 - p)$, that is,

$$P(y_j|x_i) = 1 - p, \quad i \neq j \tag{10.93}$$

Each source 0 is encoded as 000, and each source 1 is encoded as 111. An error is made if two or three symbols undergo a change in passing through the channel. Assuming that the source outputs are equally likely, the error probability P_e becomes

$$P_e = 3(1 - p)^2 p + (1 - p)^3 \tag{10.94}$$

For $(1 - p) = 0.1$, $P_e = 0.028$, implying an improvement factor of slightly less than 4. For $(1 - p) = 0.01$, the improvement factor is approximately 33. Thus the code performs best when $(1 - p)$ is small.

We will see later that this simple example can be misleading since the error probability, $1 - p$, with coding is not equal to the error probability, $1 - p$, without coding. The example implies that performance increases as n, the Hamming distance between the codewords, becomes larger. However, as n increases, the code rate decreases. In most cases of practical interest, the *information rate* must be maintained constant which, for this example, requires that three code symbols be transmitted for each bit of information. An increase in redundancy results in an increase in symbol rate. Thus, coded symbols are transmitted with less energy than uncoded symbols. This changes the channel matrix so that $1 - p$ with coding is greater than $1 - p$ without coding. We will consider this effect in some detail in Example 10.13. ■

10.3.6 Parity-Check Codes for Correction of Single Errors

Repetition codes and single-parity-check codes are examples of codes that have either high error-correction capability or high information rate, but not both. Only codes that have a reasonable combination of these characteristics are practical for use in digital communication systems. We now examine a class of parity-check codes that satisfies these requirements.

A general codeword consisting of r parity-check symbols can be written in the form

$$a_1\ a_2 \cdots a_k\ c_1\ c_2 \cdots c_r$$

where a_i is the ith information symbol and c_j is the jth check symbol. The word length $n = k + r$. The problem is selecting the r parity-check symbols so that good performance is obtained. Desirable codes are those providing good error-correcting properties at high code rates.

There is another desirable property of good codes. That is, decoders must be easily implemented. This, in turn, requires that the code has a simple structure. Keep in mind that 2^k different codewords can be constructed from information sequences of length k. Since the codewords are of length n there are 2^n possible received sequences. Of these 2^n possible received sequences, 2^k represent valid codewords and the remaining $2^n - 2^k$ represent received sequences containing errors resulting from noise or other channel impairments. Shannon showed that for $n \gg k$, one can simply randomly assign one of the 2^n sequences of length n to each of the 2^k information sequences and, most of the time, a "good" code will result. The encoder then consists of a table with these assignments. The difficulty with this strategy is that the codewords lack structure and therefore table lookup is required for decoding. Table lookup is not desirable for most applications since it is slow and usually requires excessive memory. Thus, structure is required. We now examine a structured technique for assigning information sequences to n-symbol codewords.

Codes for which the first k symbols of the codeword are the information symbols are called systematic codes. The r parity check symbols are chosen to satisfy the r linear equations

$$\begin{aligned} 0 &= h_{11}a_1 \oplus h_{12}a_2 \oplus \cdots \oplus h_{1k}a_k \oplus c_1 \\ 0 &= h_{21}a_1 \oplus h_{22}a_2 \oplus \cdots \oplus h_{2k}a_k \oplus c_2 \\ &\vdots \qquad\qquad\qquad \vdots \\ 0 &= h_{r1}a_1 \oplus h_{r2}a_2 \oplus \cdots \oplus h_{rk}a_k \oplus c_r \end{aligned} \tag{10.95}$$

Equation (10.95) can be written as

$$[H][T] = [0] \tag{10.96}$$

where $[H]$ is the parity-check matrix

$$[H] = \begin{bmatrix} h_{11} & h_{12} \cdots h_{1k} & 1\ 0 \cdots 0 \\ h_{21} & h_{22} \cdots h_{2k} & 0\ 1 \cdots 0 \\ \vdots & \vdots \ddots \vdots & \vdots\ \vdots \ddots \vdots \\ h_{r1} & h_{r2} \cdots h_{rk} & 0\ 0 \cdots 1 \end{bmatrix} \tag{10.97}$$

and $[T]$ is the codeword vector

$$[T] = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \\ c_1 \\ \vdots \\ c_r \end{bmatrix} \tag{10.98}$$

Now let the received word be $[R]$. If

$$[H][R] \neq [0] \tag{10.99}$$

we know that $[R]$ is not a codeword, that is, $[R] \neq [T]$, and at least one error has been made. If

$$[H][R] = [0] \tag{10.100}$$

we know that $[R]$ is a valid codeword and, since the probability of symbol error on the channel is assumed small, it is *most likely* the transmitted codeword.

Since $[R]$ is the received codeword, it can be written

$$[R] = [T] \oplus [E] \tag{10.101}$$

where $[E]$ represents the error pattern of length n induced by the channel. The decoding problem essentially reduces to determining $[E]$, since the codeword can be reconstructed from $[R]$ and $[E]$.

As the first step in computing $[E]$, we multiply the received word $[R]$ by the parity-check matrix $[H]$. This yields

$$[S] = [H][R] = [H][T] \oplus [H][E] \tag{10.102}$$

or

$$[S] = [H][E] \tag{10.103}$$

The matrix $[S]$ is known as the *syndrome*. Note that we cannot solve (10.103) directly since $[H]$ is not a square matrix and, therefore, the inverse of $[H]$ does not exist.

Assuming that a single error has taken place, the error vector will be of the form

$$[E] = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Multiplying $[E]$ by $[H]$ on the left-hand side shows that the syndrome is the ith column of the matrix $[H]$, where the error is in the ith position. The following example illustrates this method. Note that since the probability of symbol error on the channel is assumed small, the error vector having the smallest Hamming weight is the most likely error vector. Error patterns containing single errors are therefore the most likely.

EXAMPLE 10.10

A code has the parity-check matrix

$$[H] = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{10.104}$$

Assuming that 111011 is received, determine if an error has been made, and if so, determine the decoded codeword.

Solution: First, we compute the syndrome, remembering that all operations are modulo-2. This gives

$$[S] = [H][R] = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{10.105}$$

Since the syndrome is the third column of the parity-check matrix, the third symbol of the received word is assumed to be in error. Thus the decoded codeword is 110011. This can be proved by showing that 110011 has a zero syndrome. ■

We now pause to examine the parity-check code in more detail. It follows from (10.95) and (10.97) that the parity checks can be written as

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_r \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ h_{r1} & h_{r2} & \cdots & h_{rk} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} \tag{10.106}$$

Thus the codeword vector $[T]$ can be written

$$[T] = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \\ c_1 \\ \vdots \\ c_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ h_{11} & h_{12} & \cdots & h_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ h_{r1} & h_{r2} & \cdots & h_{rk} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} \tag{10.107}$$

or

$$[T] = [G][A] \tag{10.108}$$

where $[A]$ is the vector of k information symbols,

$$[A] = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} \tag{10.109}$$

and $[G]$, which is called the *generator matrix*, is

$$[G] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ h_{11} & h_{12} & \cdots & h_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ h_{r1} & h_{r2} & \cdots & h_{rk} \end{bmatrix} \tag{10.110}$$

The relationship between the generator matrix $[G]$ and the parity-check matrix $[H]$ is apparent if we compare (10.97) and (10.110). If the m by m identity matrix is identified by $[I_m]$ and the matrix $[H_p]$ is defined by

$$[H_p] = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ h_{r1} & h_{r2} & \cdots & h_{rk} \end{bmatrix} \tag{10.111}$$

it follows that the generator matrix is given by

$$[G] = \begin{bmatrix} I_k \\ \cdots \\ H_p \end{bmatrix} \tag{10.112}$$

and that the parity-check matrix is given by

$$[H] = [H_p \vdots I_r] \tag{10.113}$$

which establishes the relationship between the generator and parity-check matrices for systematic codes.

Codes defined by (10.110) are referred to as *linear codes*, since the $k + r$ codeword symbols are formed as a linear combination of the k information symbols. It is also worthwhile to note that if two different information sequences are summed to give a third sequence, then the codeword for the third sequence is the sum of the two codewords corresponding to the original two information sequences. This is easily shown. If two information sequences are summed, the resulting vector of information symbols is

$$[A_3] = [A_1] \oplus [A_2] \tag{10.114}$$

The codeword corresponding to $[A_3]$ is

$$[T_3] = [G][A_3] = [G]\{[A_1] \oplus [A_2]\} = [G][A_1] \oplus [G][A_2] \tag{10.115}$$

Since

$$[T_1] = [G][A_1] \tag{10.116}$$

and

$$[T_2] = [G][A_2] \tag{10.117}$$

it follows that

$$[T_3] = [T_1] \oplus [T_2] \tag{10.118}$$

Codes that satisfy this property are known as *group codes*.

10.3.7 Hamming Codes

A Hamming code is a particular parity-check code having distance 3. Therefore, all single errors can be corrected. The parity-check matrix for the code has dimensions $2^{n-k} - 1$ by $n - k$ and is very easy to construct. The ith column of the matrix $[H]$ is the binary representation of the number i. This code has the interesting property in that, for a single error, the syndrome is the binary representation of the position in error.

EXAMPLE 10.11

Determine the parity-check matrix for a (7, 4) code and the decoded codeword if the received word is 1110001.

Solution: Since the ith column of the matrix $[H]$ is the binary representation of i, we have

$$[H] = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{10.119}$$

For the received word 1110001, the syndrome is

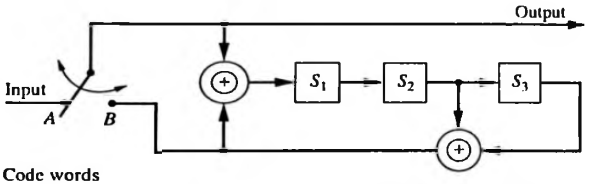
$$[S] = [H][R] = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{10.120}$$

Thus the error is in the seventh position, and the decoded codeword is 1110000.

We must note in passing that for the (7, 4) Hamming code, the parity checks are in the first, second, and fourth positions in the codewords, since these are the only columns of the parity-check matrix containing only one nonzero element.Thus the code defined by (10.119) is not systematic. ■

10.3.8 Cyclic Codes

The preceding subsections dealt primarily with the mathematical properties of parity-check codes. We have avoided any discussion of the implementation of parity-check encoders or decoders. Indeed, if we were to examine the implementation of these devices, we would find that fairly complex hardware configurations are required. However, there is a class of parity-check codes, known as *cyclic codes*, that are easily implemented using feedback shift registers. A cyclic code derives its name from the fact that a cyclic permutation of any codeword produces another codeword. For example, if $x_1x_2 \cdots x_{n-1}x_n$ is a codeword, so is $x_nx_1x_2 \cdots x_{n-1}$. In this section we examine not the underlying theory of cyclic codes but the implementation of encoders and decoders. We will accomplish this by means of an example.



Code words

0000000
1000101
1100010
1110100
1111111
0001011
0011101
0111010
1011000
1101001
1001110
0110001
0100111
0010110
0101100
1010011

Register contents for input word 1101

Shift	Register content	Output
1	100	1
2	110	1
3	111	0
4	111	1
Switch set to position B		
5	011	0
6	001	0
7	000	1

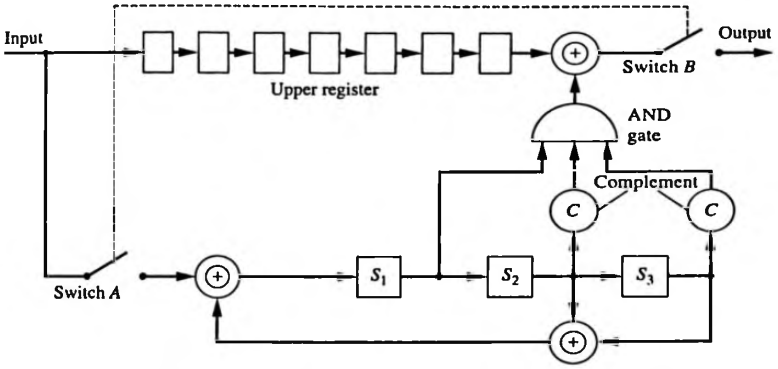
FIGURE 10.15 Coder for (7, 4) cyclic code.

An (n, k) cyclic code can easily be generated with an $n - k$ stage shift register with appropriate feedback. The register illustrated in Figure 10.15 produces a (7, 4) cyclic code. The switch is initially in position A, and the shift register stages initially contain all zeros. The $k = 4$ information symbols are then shifted into the encoder. As each information symbol arrives, it is routed to the output and added to the value of $S_2 \oplus S_3$. The resulting sum is then placed into the first stage of the shift register. Simultaneously, the contents of S_1 and S_2 are shifted to S_2 and S_3, respectively. After all information symbols have arrived, the switch is moved to position B, and the shift register is shifted $n - k = 3$ times to clear it. On each shift, the sum of S_2 and S_3 appears at the output. This sum added to itself produces a 0 which is fed into S_1. After $n - k$ shifts, a codeword has been generated that contains $k = 4$ information symbols and $n - k = 3$ parity-check symbols. It also should be noted that at this time the register contains all 0's so that the encoder is ready to receive the next $k = 4$ information symbols.

All $2^k = 16$ codewords that can be generated with the example encoder are also illustrated in Figure 10.16. The $k = 4$ information symbols, which are the first four symbols of each codeword, were shifted into the encoder beginning with the left-hand symbol. Also shown in Figure 10.16 are the contents of the register and the output symbol after each shift for the codeword 1101.

The decoder for the (7, 4) cyclic code is illustrated in Figure 10.16. The upper register is used for storage, and the lower register and feedback arrangement are identical to the feedback shift register used in the encoder. Initially, switch A is closed and switch B is open. The n received symbols are shifted into the two registers. If there are no errors, the lower register will contain all 0's when the upper register is full. The switch positions are then reversed, and the codeword that is stored in the upper register is shifted out. This operation is illustrated in Figure 10.16 for the received word 1101001.

If, after the received word is shifted into the decoder, the lower register does not contain all 0's, an error has been made. The error is corrected automatically by the decoder, since, when the incorrect symbol appears at the output of the shift register, a 1 appears at



Received word 1101001 (no errors)

	Shift	Input	Lower register content	Output
Switch *A* closed Switch *B* open	1	1	100	
	2	1	110	
	3	0	111	
	4	1	111	
	5	0	011	
	6	0	001	
	7	1	000	
Switch *A* open Switch *B* closed	8		000	1
	9		000	1
	10		000	0
	11		000	1
	12		000	0
	13		000	0
	14		000	1

Received word 1101011 (one errors)

Shift	Input	Lower register content	Output
1	1	100	
2	1	110	
3	0	111	
4	1	111	
5	0	011	
6	1	101	
7	1	010	
8		101	1
9		110	1
10		111	0
11		011	1
12		001	0
13		100	0
14		010	1

AND gate output inverts upper register output

FIGURE 10.16 Decoder for a (7, 4) cyclic code.

the output of the AND gate. This 1 inverts the upper register output and is produced by the sequence 100 in the lower register. The operation is illustrated in Figure 10.16.

10.3.9 Comparison of Errors in Block-Coded and Uncoded Systems

We will now compare the relative performance of coded and uncoded systems for block codes. The basic assumption will be that the *information rate* is the same for both systems. Since the coded and uncoded words contain the same information, the word duration T_w will be the same under the equal-information-rate assumption. Since the codeword contains more symbols than the uncoded word, due to the addition of parity symbols, the symbol rate will be higher for the coded system than for the uncoded system. If constant transmitter power is assumed, it follows that the energy per transmitted symbol is decreased by the use

of coding, resulting in a higher probability of symbol error. We must determine whether or not coding can overcome this increase in symbol error probability to the extent that a significant decrease in word error probability can be obtained.

Assume that q_u and q_c represent the probability of *symbol* error for the uncoded and coded systems, respectively. Also assume that P_{eu} and P_{ec} are the *word* error probabilities for the uncoded and coded systems. The word error probability for the uncoded system is relatively easy to compute. An uncoded word is in error if any of the symbols in that word are in error. The probability that a symbol will be received correctly is $(1 - q_u)$, and since all symbol errors are assumed independent, the probability that all k symbols in a word are received correctly is $(1 - q_u)^k$. Thus the uncoded word error probability is

$$P_{eu} = 1 - (1 - q_u)^k \tag{10.121}$$

The probability of word error for the coded system is more difficult to compute because symbol errors can possibly be corrected by the decoder, depending on the code used. If a code is capable of correcting up to e errors, the probability of word error P_{ec} is equal to the probability that more than e errors are present in the received codeword. Thus

$$P_{ec} = \sum_{i=e+1}^{n} \binom{n}{i} (1 - q_c)^{n-i} q_c^i \tag{10.122}$$

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{10.123}$$

is the number of combinations of n symbols taken i at a time.

For single-error-correcting codes $e = 1$, by definition. Thus P_{ec} becomes

$$P_{ec} = \sum_{i=2}^{n} \binom{n}{i} (1 - q_c)^{n-i} q_c^i \tag{10.124}$$

By comparing (10.121) and (10.124), we determine the improvement in word error probability gained through the use of coding. Comparing different codes requires that we evaluate the decoded bit error probabilities. This is a more difficult problem. (See Computer Exercise 10.3 at the end of this chapter.)

EXAMPLE 10.12

We now investigate the effectiveness of a (7, 4) single error-correcting code by comparing the word error probabilities for the coded and uncoded systems. Assume that the code is used with a BPSK transmission system. As shown in Chapter 7 (with $m = 0$), the symbol error probability is

$$q = Q[\sqrt{2z}] \tag{10.125}$$

The symbol energy E_s is the transmitter power S times the word time T_w divided by k, since the total energy in each word is divided by k. Thus, the symbol error probability without coding is

$$q_u = Q\left[\sqrt{\frac{2ST_w}{kN_0}}\right] \tag{10.126}$$

Assuming equal *word rates* for both the coded and uncoded system gives

$$q_c = Q\left[\sqrt{\frac{2ST_w}{nN_0}}\right] \tag{10.127}$$

for the coded symbol error probability, since the energy available for k information symbols must be

spread over $n > k$ symbols when coding is used. Thus it follows that the *symbol* error probability is increased by the use of coding. However, we will show that the error-correcting capability of the code can overcome the increased symbol error probability and indeed yield a net gain in word error probability for certain ranges of the signal-to-noise ratio. The uncoded word error probability is given by (10.121) with and $k = 4$. Thus

$$P_{eu} = 1 - (1 - q_u)^4 \tag{10.128}$$

Also, from (10.124), the word error probability for the coded case is

$$P_{ec} = \sum_{i=2}^{7} \binom{7}{i} (1 - q_c)^{7-i} q_c^i \tag{10.129}$$

Given the complexity of the preceding equations, and given that we wish to calculate q_u, q_c, P_{eu}, and P_{ec} for a variety of values of ST_w/N_0, it is convenient to develop a computer program for performing the calculations and for plotting the results. ■

COMPUTER EXAMPLE 10.1

The MATLAB program for performing the calculations outlined in the preceding example follows.

```
n = 7; k = 4; t = 1;              % code parameters
zdB = 0:0.1:14;                   % set STw/No in dB
z = 10.^(zdB/10);                 % STw/No
lenz = length(z);                 % length of vector
qc = qfn(sqrt(2*z/n));            % coded symbol error prob.
qu = qfn(sqrt(2*z/k));            % uncoded symbol error prob.
peu = 1-((1-qu).^k);              % uncoded word error prob.
pec = zeros(1,lenz);              % initialize
for j=1:lenz
%
% The following code determines pec
%
pc = qc(j);                         % jth symbol error prob.
s = 0;                              % initialize
for i=(t+1):n
 a = sum(log(1:n));                 % ln of n!
 b = sum(log(1:i));                 % ln of k!
 c = sum(log(1:(n-i)));             % ln of (n-k)!
 nchoosek = round(exp(a-b-c));      % n!/k!/(n-k)!
 termi = (pc^i)*((1-pc)^(n-i));
 s = s+nchoosek*termi;
pec(1,j) = s;                       % coded word error
                                      probability
end
%
% End of pec calculation.
%
end
qq = [qc',qu',peu',pec'];
semilogy(zdB',qq)
xlabel('STw/No in dB')              % label x axis
ylabel('Probability')               % label y
% This function computes the Gaussian Q-function
% function Q=qfn(x)
Q = 0.5*erfc(x/sqrt(2));
```

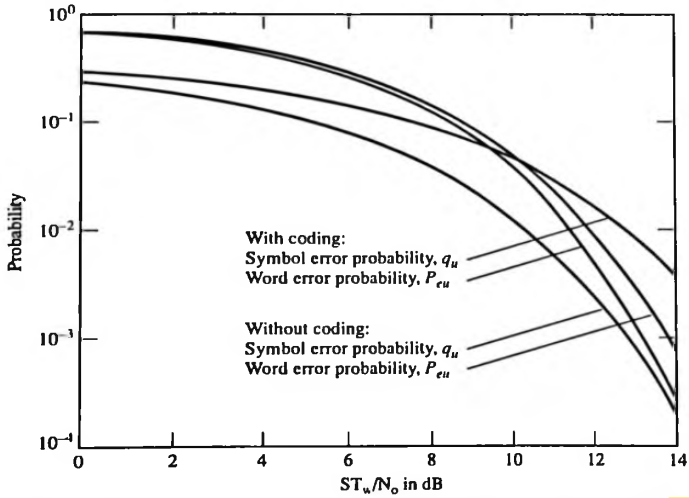


FIGURE 10.17 Comparison of uncoded and coded systems assuming a (7, 4) code.

Note the manner in which logarithms are used in the calculation of the binomial coefficient. While not necessary for most values of n and k, very large values of n and k often lead to overflow problems when the binomial coefficient is calculated. The technique used here is suitable in such situations. The "round" command is used simply to remove any round-off errors.[3]

The word error probabilities for the coded and uncoded systems are illustrated in Figure 10.17. The curves are plotted as a function of ST_W/N_0, which is word energy divided by the noise power spectral density.

You should note that coding has little effect on system performance unless the value of ST_W/N_0 is in the neighborhood of 11 dB or above. Also, the improvement afforded by a (7, 4) code is quite modest unless ST_W/N_0 is large, in which case system performance may be satisfactory without coding. In many systems, however, even small improvements are very important. Also illustrated in Figure 10.17 are the uncoded and coded symbol error probabilities q_u and q_c, respectively. The effect of spreading the available energy per word over a larger number of symbols is evident. ■

EXAMPLE 10.13

In this example we examine the performance of repetition codes in two different channels. Both cases utilize FSK modulation and a noncoherent receiver structure. In the first case, an *additive white Gaussian noise* (AWGN) channel is assumed. The second case assumes a Rayleigh fading channel. Distinctly different results will be obtained.

[3]The function `nchoosek` is a standard MATLAB m-file and could have been used in this program for calculating the binomial coefficient rather than using the logarithm technique illustrated in the program. The logarithm technique is given, however, since it is useful in cases where N is so large that computing N ! directly leads to overflows.

Case 1

As was shown in Chapter 7, the error probability for a noncoherent FSK system in an additive white Gaussian noise (AWGN) channel is given by

$$q_u = \frac{1}{2}e^{-0.5z} \tag{10.130}$$

where z is the ratio of signal power to noise power at the output of the receiver bandpass filter. Equivalently, if the receiver filter is matched to the received signal envelope, z is the ratio of the energy per bit to the noise power spectral density at the receiver input. The SNR is defined as

$$z = \frac{A^2}{2N_0B_T} \tag{10.131}$$

where N_0B_T is the noise power in the signal bandwidth B_T. The performance of the system is illustrated by the $n = 1$ curve in Figure 10.18. When an n-symbol repetition code is used with this system, the symbol error probability is given by

$$q_c = \frac{1}{2}e^{-z/2n} \tag{10.132}$$

This result occurs since encoding a single information symbol (bit) as n code symbols requires that the symbol duration with coding be $1/n$ times the information symbol duration without coding or, equivalently, that the signal bandwidth be increased by a factor of n with coding. Thus, with coding

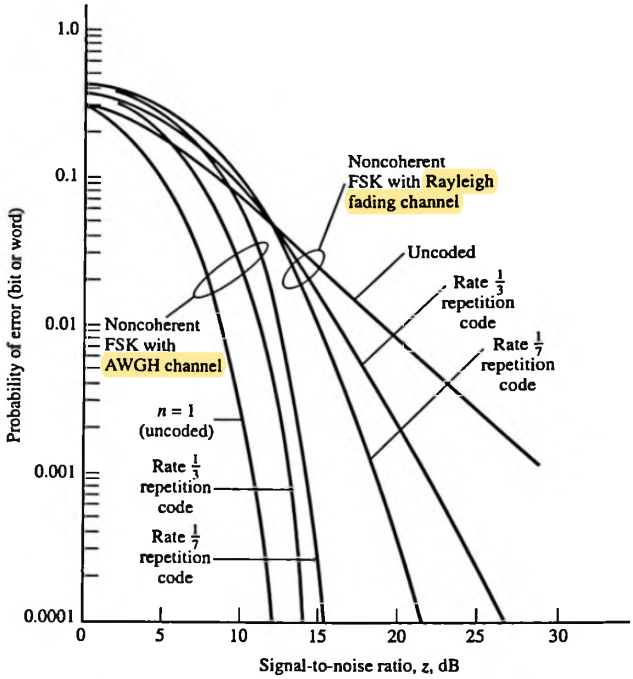


FIGURE 10.18 Performance of repetition codes on AWGN and Rayleigh fading channels.

B_T in q_u is replaced by nB_T to give q_c. The word error probability is given by (10.122) with

$$e = \frac{1}{2}(n-1) \tag{10.133}$$

Since each codeword carries one bit of information, the word error probability is equal to the bit error probability for the repetition code.

The performance of a noncoherent FSK system with an AWGN channel with rate $\frac{1}{3}$ and $\frac{1}{7}$ repetition codes is illustrated in Figure 10.18. It should be noted that system performance is degraded through the use of repetition coding. This result occurs because the increase in symbol error probability with coding is greater than can be overcome by the error-correction capability of the code. This same result occurs with coherent FSK and BPSK as well as with ASK, illustrating that the low rate of the repetition code prohibits its effective use in systems in which the dependence on symbol error probability and signal-to-noise ratio is essentially exponential.

Case 2

An example of a system in which repetition coding can be used effectively is an FSK system operating in a Rayleigh fading environment. Such a system was analyzed in Chapter 9. We showed that the symbol error probability can be written as

$$q_u = \frac{1}{2}\frac{1}{1+\frac{E_a}{2N_0}} \tag{10.134}$$

in which E_a is the average received energy per symbol (or bit). The use of a repetition code spreads the energy E_a over the n symbols in a codeword. Thus, with coding,

$$q_c = \frac{1}{2}\frac{1}{1+\frac{E_a}{2nN_0}} \tag{10.135}$$

As in Case 1, the decoded bit error probability is given by (10.122) with e given by (10.133). The Rayleigh fading results are also shown in Figure 10.18 for rate 1, $\frac{1}{3}$, and $\frac{1}{7}$ repetition codes, where, for this case, the signal-to-noise ratio z is E_a/N_0. We see that the repetition code can be used to advantage in a Rayleigh fading environment if E_a/N_0 is sufficiently large.

Repetition coding, which is one form of diversity transmission, in which the available signal energy is divided equally among n subpaths. In Problem 9.17, it was shown that the optimal combining of the receiver outputs prior to making a decision on the transmitted information bit is as shown in Figure 10.19(a). The receiver model for the repetition code considered in this example is shown in Figure 10.19(b). The essential difference is that a "hard decision" on each symbol of the n-symbol codeword is made at the output of the n receivers. The decoded information bit is then in favor of the majority of the n symbols of the codeword.

When a hard decision is made at the receiver output, information is clearly lost, and the result is a degradation of performance. This can be seen in Figure 10.20, which illustrates the performance of the $n = 7$ optimal system of Figure 10.19(a) and that of the rate $\frac{1}{7}$ repetition code of Figure 10.19(b). Also shown for reference is the performance of the uncoded system. ■

10.3.10 Convolutional Codes

The convolutional code is an example of a nonblock code. Rather than the parity-check symbols being calculated for a block of code symbols, the parity checks are calculated over a span of information symbols. This span, which is referred to as the *constraint span*, is shifted one information symbol each time an information symbol is input to the encoder.

A general convolutional encoder is illustrated in Figure 10.21. The encoder is rather simple and consists of three component parts. The heart of the encoder is a shift register that holds k information symbols, where k is the constraint span of the code. The shift register

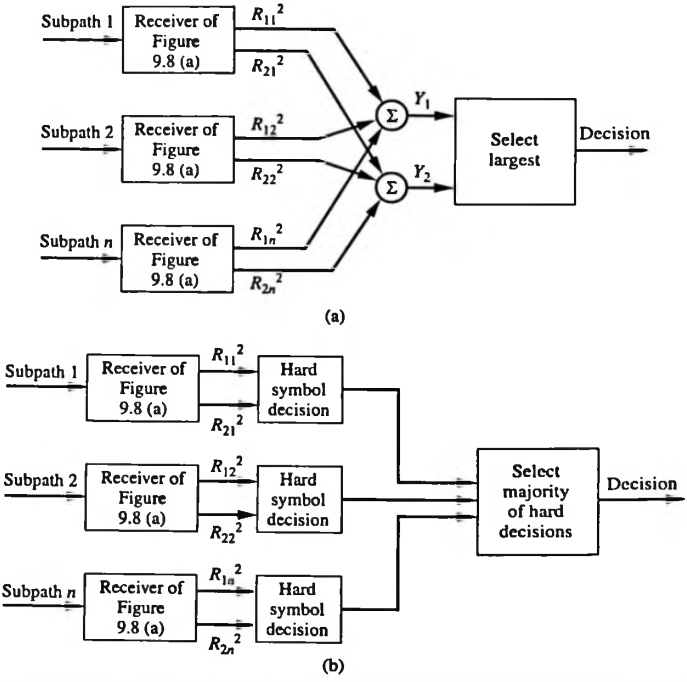


FIGURE 10.19 Comparison of optimum and suboptimum systems. (a) Optimum system. (b) Repetition code model.

stages are connected to v modulo-2 adders as indicated. Not all stages are connected to all adders. In fact, the connections are "somewhat random" and can have considerable impact on the performance of the code generated. Each time a new information symbol is shifted into the encoder, the adder outputs are sampled by the commutator. Thus v output symbols are generated for each input symbol yielding a code of rate $1/v$.

A rate $\frac{1}{3}$ convolutional encoder is illustrated in Figure 10.22. For each input, the output of the encoder is the sequence $v_1 v_2 v_3$. For the encoder of Figure 10.22,

$$v_1 = S_1 \oplus S_2 \oplus S_3 \tag{10.136}$$

$$v_2 = S_1 \tag{10.137}$$

$$v_3 = S_1 \oplus S_2 \tag{10.138}$$

Thus we see that the input sequence

$$101001\cdots$$

results in the output sequence

$$111101011101100111\cdots$$

At some point the sequence is terminated in a way that allows for unique decoding. This will be illustrated when we consider the Viterbi algorithm.

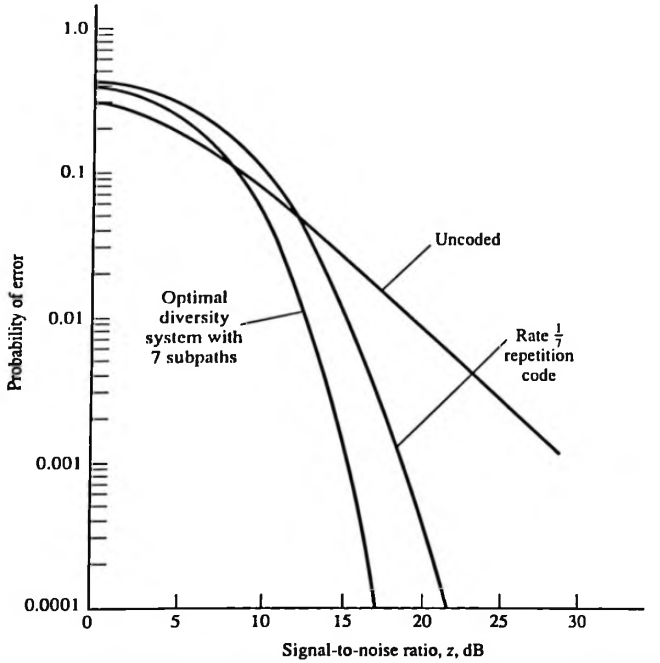


FIGURE 10.20 Performance of noncoherent FSK in a Rayleigh fading channel.

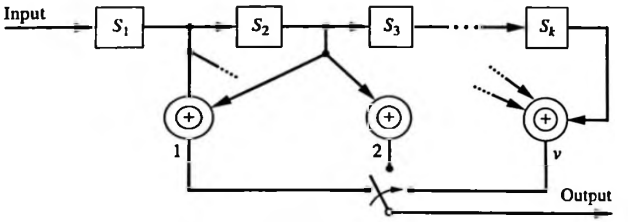


FIGURE 10.21 General convolutional encoder.

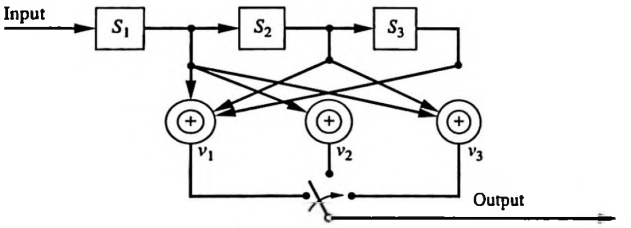


FIGURE 10.22 A rate $\frac{1}{3}$ convolutional encoder.

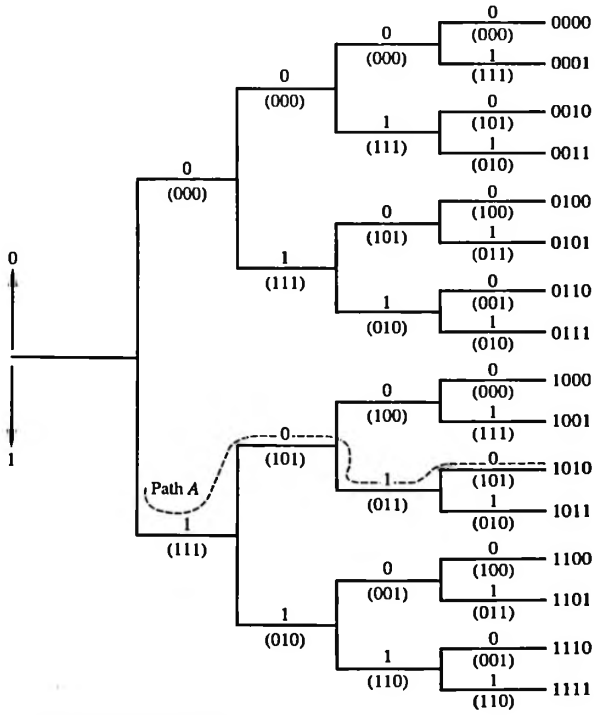


FIGURE 10.23 Code tree.

A number of techniques have been developed for decoding convolutional codes. We discuss two techniques here: the tree-searching technique, because of its fundamental nature, and the Viterbi algorithm, because of its widespread use. The tree search is considered first. A portion of the code tree for the encoder of Figure 10.22 is illustrated in Figure 10.23. In the latter figure, the single binary symbols are inputs to the encoder, and the three binary symbols in parentheses are the output symbols corresponding to each input symbol. For example, if 1010 is fed into the encoder, the output is 111101011101 or path A.

The decoding procedure also follows from Figure 10.23. To decode a received sequence, we search the code tree for the path closest in Hamming distance to the input sequence. For example, the input sequence 110101011111 is decoded as 1010, indicating an error in the third and eleventh positions of the input sequence.

The exact implementation of tree-searching techniques is not practical for many applications since the code tree grows exponentially with the number of information symbols. For example, N information symbols generate 2^N branches of the code tree and storage of the complete tree is impractical for large N. Several algorithms have been developed that yield excellent performance with reasonable hardware requirements. Prior to taking a brief look at the most popular of these techniques, the Viterbi algorithm, we look at the trellis diagram, which is essentially a code tree in compact form.

The key to construction of the trellis diagram is recognition that the code tree is repetitive after k branches, where k is the constraint span of the encoder. This is easily recognized from the code tree shown in Figure 10.23. After the fourth input of an information

symbol, 16 branches have been generated in the code tree. The encoder outputs for the first 8 branches match exactly the encoder outputs for the second 8 branches, except for the first symbol. After a little thought, you should see that this is obvious. The encoder output depends only on the latest k inputs. In this case, the constraint span, k, is 3. Thus the output corresponding to the fourth information symbol depends only on the second, third, and fourth encoder inputs. It makes no difference whether the first information symbol was a binary 0 or a binary 1. (This should clarify the meaning of a constraint span.)

The analysis is simplified by defining the state of the encoder as $S_1 S_2$, or the contents of the input register prior to inputting the current information symbol. When the current information symbol is input, S_1 is shifted to S_2 and S_2 is shifted to S_3. The state and current input then determine S_1, S_2, and S_3, which in turn determine the output. This information is summarized in Table 10.5. The outputs corresponding to given state transitions are shown in parentheses, consistent with Figure 10.23.

TABLE 10.5 States, Transitions, and Outputs for the Convolutional Encoder Shown in Figure 10.23

(a) Definition of States

State	S_1	S_2
A	0	0
B	0	1
C	1	0
D	1	1

(b) State Transitions

Previous				Current				
State	S_1	S_2	Input	S_1	S_2	S_3	State	Output
A	0	0	0	0	0	0	*A*	(000)
			1	1	0	0	*C*	(111)
B	0	1	0	0	0	1	*A*	(100)
			1	1	0	1	*C*	(011)
C	1	0	0	0	1	0	*B*	(101)
			1	1	1	0	*D*	(010)
D	1	1	0	0	1	1	*B*	(001)
			1	1	1	1	*D*	(110)

(c) Encoder Output for State Transition $x \rightarrow y$

Transition	Output
$A \rightarrow A$	(000)
$A \rightarrow C$	(111)
$B \rightarrow A$	(100)
$B \rightarrow C$	(011)
$C \rightarrow B$	(101)
$C \rightarrow D$	(010)
$D \rightarrow B$	(001)
$D \rightarrow D$	(110)

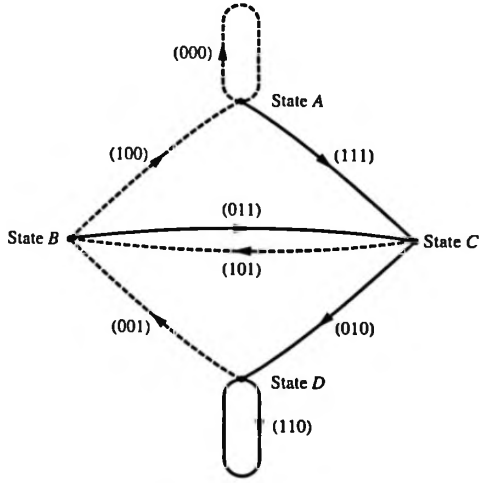


FIGURE 10.24 State diagram for the example convolutional encoder.

It should be noted that states A and C can only be reached from states A and B. Also, states B and D can only be reached from states C and D. The information in Table 10.5 is often shown in a state diagram, such as in Figure 10.24. In the state diagram, an input of binary 0 results in the transition denoted by a dashed line, and an input of binary 1 results in the transition designated by a solid line. The resulting encoder output is denoted by the three symbols in parentheses. For any given sequence of inputs, the resulting state transitions and encoder outputs can be traced on the state diagram. This is a very convenient method for determining the outputs resulting from a given sequence of inputs.

The trellis diagram (Figure 10.25) results directly from the state diagram. Initially, the encoder is assumed to be in state A (all contents are 0's). An input of binary 0 results in the encoder remaining in state A, as indicated by the dashed line, and an input of binary 1 results in a transition to state C, as indicated by the solid line. Any of the four states can be reached by a sequence of two inputs. The third input results in the possible transitions shown. The fourth input results in exactly the same set of possible transitions. Therefore, after the second input, the trellis becomes completely repetitive, and the possible transitions are those labeled steady-state transitions. The encoder can always be returned to state A by inputting two binary 0's as shown in Figure 10.25. As before, the output sequence resulting from any transition is shown by the sequence in parentheses.

In order to illustrate the Viterbi algorithm, we consider the received sequence that we previously considered to illustrate decoding using a code tree—namely, the sequence 110101011111. The first step is to compute the Hamming distances between the initial node (state A) and each of the four states three levels deep into the trellis. We must look three levels deep into the trellis because the constraint span of the example encoder is 3. Since each of the four nodes can be reached from only two preceding nodes, eight paths must be identified, and the Hamming distance must be computed for each path. We therefore initially look three levels deep into the trellis, and since the example encoder has rate $\frac{1}{3}$, the first nine received symbols are initially considered. Thus the Hamming distances between the input sequence 110101011 and the eight paths terminating three levels deep into the trellis are computed. These calculations are summarized in Table 10.6. After the eight Hamming distances are computed, the path having the minimum Hamming distance

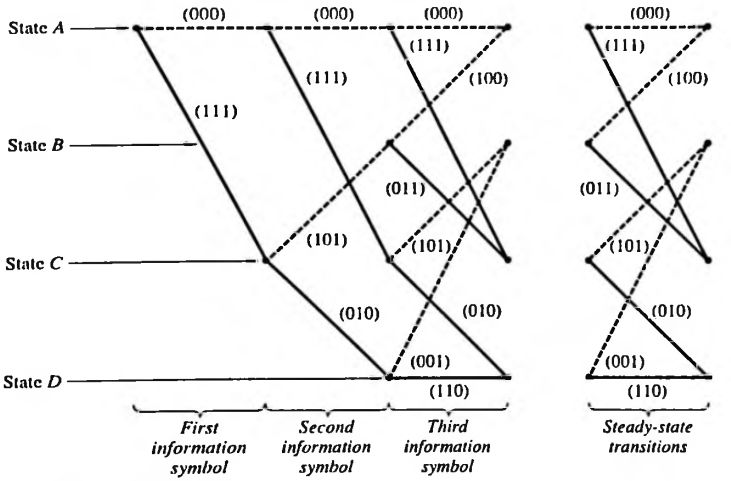


FIGURE 10.25 Trellis diagram.

to *each* of the four nodes is retained. These four retained paths are known as *survivors*. The other four paths are discarded from further consideration. The four survivors are identified in Table 10.6.

The next step in the application of the Viterbi algorithm is to consider the next three received symbols, which are 111 in the example being considered. The scheme is to compute once again the Hamming distance to the four states, this time four levels deep in the trellis. As before, each of the four states can be reached from only two previous states. Thus, once again, eight Hamming distances must be computed. Each of the four previous survivors, along with their respective Hamming distances, is extended to the two states reached by each surviving path. The Hamming distance of each new segment is computed by comparing the

TABLE 10.6 Calculations for Viterbi Algorithm: Step One (Received Sequence = 110101011)

Path[1]	Corresponding Symbols	Hamming Distance	Survivor?
AAAA	000000000	6	No
ACBA	111101100	4	Yes
ACDB	111010001	5	Yes[2]
AACB	000111101	5	No[2]
AAAC	000000111	5	No
ACBC	111101011	1	Yes
ACDD	111010110	6	No
AACD	000111010	4	Yes

[1]The initial and terminal states are identified by the first and fourth letters, respectively. The second and third letters correspond to intermediate states.

[2]If two or more paths have the same Hamming distance, it makes no difference which is retained as a survivor.

TABLE 10.7 Calculations for Viterbi Algorithm: Step Two (Received Sequence = 110101011111)

Path[1]	Previous Survivor's Distance	New Segiment	Added Distance	New Distance	Survivor?
ACBAA	4	*AA*	3	7	Yes
ACDBA	5	*BA*	2	7	No
ACBCB	1	*CB*	1	2	Yes
AACDB	4	*DB*	2	6	No
ACBAC	4	*AC*	0	4	Yes
ACDBC	5	*BC*	1	6	No
ACBCD	1	*CD*	2	3	Yes
AACDD	4	*DD*	1	5	No

[1] An underscore indicates the previous survivor.

encoder output, corresponding to each of the new segments, with 111. The calculations are summarized in Table 10.7. The path having the smallest new distance is path *ACBCB*. This corresponds to information sequence 1010 and is in agreement with the previous tree search.

For a general received sequence, the process identified in Table 10.7 is continued. After each new set of calculations, involving the next three received symbols, only the four surviving paths and the accumulated Hamming distances need be retained. At the end of the process, it is necessary to reduce the number of surviving paths from four to one. This is accomplished by inserting two dummy 0's at the end of the information sequence, corresponding to the transmission of six code symbols. As shown in Figure 10.26, this forces the trellis to terminate at state *A*.

The Viterbi algorithm has found widespread application in practice, especially in satellite and cellular communications systems. It can be shown that the Viterbi algorithm is

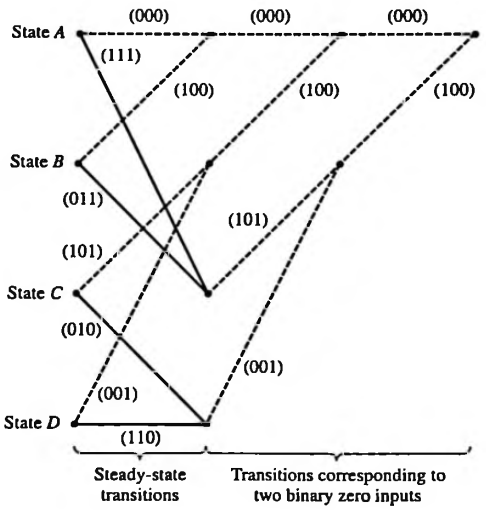


FIGURE 10.26 Termination of the trellis diagram.

a maximum-likelihood decoder, and in that sense, it is optimal. Viterbi and Omura (1979) give an excellent analysis of the Viterbi algorithm. A paper by Heller and Jacobs (1971) summarizes a number of performance characteristics of the Viterbi algorithm.

10.3.11 Burst-Error-Correcting Codes

Many practical communication channels, such as those encountered in mobile communication systems, exhibit fading in which errors tend to group together in bursts. Thus, errors are no longer independent. Much attention has been devoted to code development for improving the performance of systems exhibiting burst-error characteristics. Most of these codes tend to be more complex than the simple codes considered here. A code for correction of a single burst, however, is rather simple to understand and leads to a technique known as interleaving, which is useful in a number of situations.

As an example, assume that the output of a source is encoded using an (n, k) block code. The ith codeword will be of the form

$$\lambda_{i1}\ \lambda_{i2}\ \lambda_{i3} \cdots \lambda_{in}$$

Assume that m of these codewords are read into a table by rows so that the ith row represents the ith codeword. This yields the n by m array

$$\begin{array}{cccc} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1n} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2n} \\ \lambda_{31} & \lambda_{32} & \cdots & \lambda_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m1} & \lambda_{m2} & \cdots & \lambda_{mn} \end{array}$$

If transmission is accomplished by reading out of this table by columns, the transmitted stream of symbols will be

$$\lambda_{11}\ \lambda_{21} \cdots \lambda_{m1}\ \lambda_{12}\ \lambda_{22} \cdots \lambda_{m2} \cdots \lambda_{1n}\ \lambda_{2n} \cdots \lambda_{mn}$$

The received symbols must be deinterleaved prior to decoding as illustrated in Figure 10.27. The deinterleaver performs the inverse operation to the interleaver and reorders the received symbols into blocks of n symbols per block. Each block corresponds to a codeword, which may exhibit errors due to channel effects. Specifically, if a burst of errors affects m consecutive symbols, then each codeword (length n) will have exactly one error.

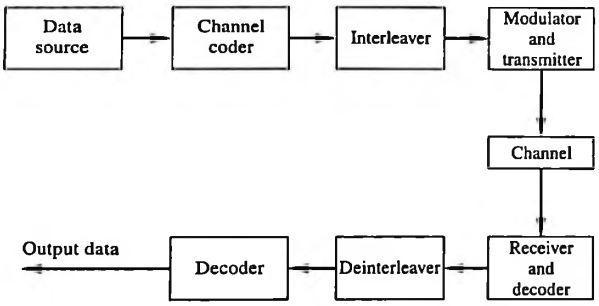


FIGURE 10.27 Communication system with interleaving.

An error-correcting code capable of correcting single errors, such as a Hamming code, will correct a burst of m channel errors induced by the channel *if* there are no other errors in the transmitted stream of mn symbols. Similarly, a double-error-correcting code can be used to correct a single burst spanning $2m$ *symbols*. These codes are known as *interleaved codes* since m codewords, each of length n, are interleaved to form the sequence of length mn. Interleaving plays a key role in the implementation and the performance of turbo codes, which are briefly discussed in the following section.

The net effect of the interleaver is to randomize the errors so that the correlation of error events are reduced. The interleaver illustrated here is called a block interleaver. There are many other types of interleavers possible but their consideration is beyond the scope of this simple introduction. We will see in a following section that the interleaving process plays a critical role in turbo coding.

10.3.12 Turbo Coding

The study of coding theory has been a search for the coding scheme that yields a communications system having the performance closely approaching the Shannon bound. For the most part progress has been incremental. A large step in this quest for nearly ideal performance was revealed in 1993 with the publication of a paper by Berrou, Glavieux, and Thitimajshima. It is remarkable that this paper was not the result of a search for a more powerful coding scheme, but was a result of a study of efficient clocking techniques for concatenated circuits. Their discovery, however, has revolutionized coding theory.

The turbo coding, and especially decoding, are complex tasks and a study of even simple implementations is well beyond the scope of this text. We will present, however, a few important concepts as motivation for further study.

The basic architecture of a turbo coder is illustrated in Figure 10.28. Note that the turbo coder consists of an interleaver, much as we studied previously, and a pair of recursive systematic convolutional encoders (RSCCs). An RSCC is shown in Figure 10.29. Note that the RSCC is much like the convolutional coders previously studied with one important difference. That difference lies in the feedback path from the delay elements back to the input. The conventional convolutional coder does not have this feedback path and therefore it behaves as a FIR digital filter. With the feedback path the filter becomes an IIR, or recursive, filter and here lies one of the attributes of the turbo code. The RSCC shown in Figure 10.29 is a rate 1/2 convolutional coder for which the input x_i generates an output sequence $x_i p_i$. Since the first symbol in the output sequence is the information symbol, the coder is systematic.

The two RSCCs shown in Figure 10.29 are usually identical and, with the parallel architecture shown, generates a rate 1/3 code. The input symbol x_i produces the output

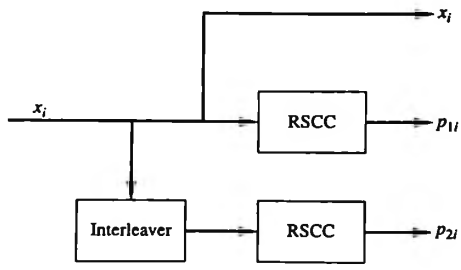


FIGURE 10.28 Turbo coder.

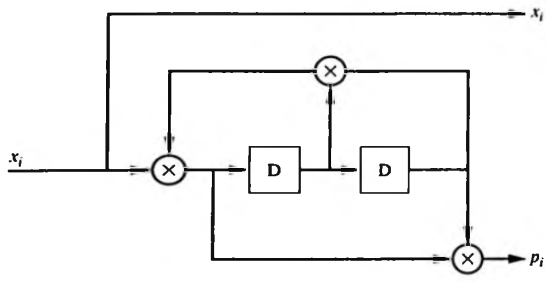


FIGURE 10.29 Recursive, systematic, convolutional coder.

sequence $x_i\, p_{1i}\, p_{2i}$. As we know, good code performance (low error probability) results if the Hamming distance between codewords is large. Because of the recursive nature of the coder, a single binary 1 in the input sequence will produce a periodic parity sequence p_1, with period T_p. Strictly speaking, a sequence of unity weight on the input will produce a sequence of infinite weight for p_1. However, if the input sequence consists of a pair of binary ones separated by T_p, the parity sequence will be the sum of two periodic sequences with period T_p. Since binary arithmetic has an addition table that results in a zero when two identical binary numbers are added, the sum of the two sequences is zero except for the first period of the offset. This, of course will reduce the Hamming weight of the first parity sequence, which is an undesirable effect.

This is where the interleaver comes into play. The interleaver will change the separation between the two binary ones and therefore cancellation will, with high probability, not occur. It therefore follows that if one of the parity sequences has large Hamming weight, the other one will not.

Figure 10.30 illustrates the performance of a turbo code for two different interleaver sizes. The larger interleaver produces better performance results because larger interleaves can do a better job of "randomizing" the interleaver input.

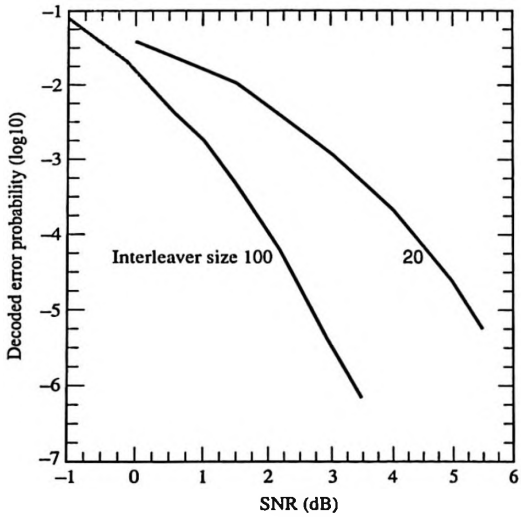


FIGURE 10.30 Performance curves for turbo code.

Decoders for turbo codes are well beyond the scope of this brief introduction. However, it is important to point out that most turbo decoding algorithms are based on the MAP estimation principle studied in the previous chapter. Of perhaps more importance is the fact that turbo decoding algorithms, unlike other decoding tools, are iterative in nature so that a given sequence passes through the decoder a number of times with the error probability decreasing with each pass. As a result a trade-off exists between performance and decoding time. This attribute allows one the freedom to develop application-specific decoding algorithms. This freedom is not available in other techniques. For example one can target various decoders for a given quality of service (QoS) by adjusting the number of iterations used in the decoding process. Decoders can also be customized to take advantage of latency/performance trade-offs. As an example, data communications requires low bit error probabilities but latency is not often a problem. Voice communications, however, requires low latency but higher error probabilities can be tolerated.

10.3.13 Feedback Channels

In many practical systems, a feedback channel is available from receiver to transmitter. When available, this channel can be utilized to achieve a specified performance with decreased complexity of the encoding scheme. Many such schemes are possible: decision feedback, error-detection feedback, and information feedback. In a decision-feedback scheme, a null-zone receiver is used, and the feedback channel is utilized to inform the transmitter either that no decision was possible on the previous symbol and to retransmit or that a decision was made and to transmit the next symbol. The null-zone receiver is often modeled as a binary-erasure channel.

Error-detection feedback involves the combination of coding and a feedback channel. With this scheme, retransmission of codewords is requested when errors are detected.

In general, feedback schemes tend to be rather difficult to analyze. Thus only the simplest scheme, the decision-feedback channel with perfect feedback assumed, will be treated here. Assume a binary transmission scheme with matched-filter detection. The signaling waveforms are $s_1(t)$ and $s_2(t)$. The conditional probability density functions of the matched filter output at time T, conditioned on $s_1(t)$ and $s_2(t)$, were derived in Chapter 7 and are illustrated in Figure 7.7, which is redrawn for our application in Figure 10.31. We will assume that both $s_1(t)$ and $s_2(t)$ have equal *a priori* probabilities. For the null-zone receiver, two thresholds, a_1 and a_2, are established. If the sampled matched-filter output, denoted V, lies between a_1 and a_2, no decision is made, and the feedback channel is used to request a retransmission. This event is denoted an erasure and occurs with probability P_2. Assuming $s_1(t)$ transmitted, an error is made if $V > a_2$. The probability of this event is denoted P_1. By symmetry, these probabilities are the same for $s_2(t)$ transmitted.

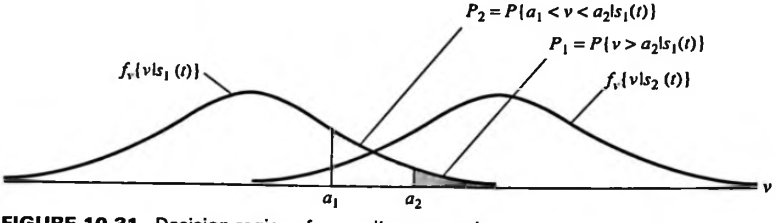


FIGURE 10.31 Decision regions for a null-zone receiver.

Assuming independence, the probability of j erasures followed by an error is

$$P(j \text{ transmissions, error}) = P_2^{j-1} P_1 \tag{10.139}$$

The overall probability of error is the summation of this probability over all j. This is[4]

$$P_E = \sum_{j=1}^{\infty} P_2^{j-1} P_1 \tag{10.140}$$

which is

$$P_E = \frac{P_1}{1 - P_2} \tag{10.141}$$

The expected number of transmissions N is also easily derived. The result is

$$N = \frac{1}{1 - P_2} \tag{10.142}$$

which is typically only slightly greater than one.

It follows from these results that the error probability can be reduced considerably without significantly increasing N. Thus performance is improved without a great sacrifice in information rate.

COMPUTER EXAMPLE 10.2

We now consider a baseband communications system with an integrate-and-dump detector. The output of the integrate-and-dump detector is given by

$$V = \begin{cases} +AT + N, & \text{if } +A \text{ is sent} \\ -AT + N, & \text{if } -A \text{ is sent} \end{cases}$$

where N is a random variable representing the noise at the detector output at the sampling instant. The detector uses two thresholds, a_1 and a_2, where $a_1 = -\gamma AT$ and $a_2 = \gamma AT$. A retransmission occurs if $a_1 < V < a_2$. The goal of this exercise is to compute and plot both the probability of error and the expected number of transmissions as a function of $z = A^2T/N_0$.

The probability density function of the sampled matched filter output, conditioned on the transmission of $-A$, is

$$f_V(v| -A) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(v + AT)^2}{2\sigma_n^2}\right) \tag{10.143}$$

The probability of erasure is

$$P(\text{Erasure}| -A) = \frac{1}{\sqrt{2\pi}\sigma_n} \int_{a_1}^{a_2} \exp\left(-\frac{(v + AT)^2}{2\sigma_n^2}\right) dv \tag{10.144}$$

With

$$y = \frac{v + AT}{\sigma_n} \tag{10.145}$$

(10.144) becomes

$$P(\text{Erasure}| -A) = \frac{1}{\sqrt{2\pi}} \int_{(1-\gamma)AT/\sigma_n}^{(1+\gamma)AT/\sigma_n} \exp\left(-\frac{y^2}{2}\right) dy \tag{10.146}$$

[4]Note that $j = 0$ is not included since $j = 0$ corresponds to a correct transmission resulting from a single transmission.

This may be expressed in terms of the Gaussian-Q function. The result is

$$P(\text{Erasure}|-A) = Q\left(\frac{(1-\gamma)AT}{\sigma_n}\right) - Q\left(\frac{(1+\gamma)AT}{\sigma_n}\right) \tag{10.147}$$

By symmetry

$$P(\text{Erasure}|-A) = P(\text{Erasure}|+A) \tag{10.148}$$

In addition, $+A$ and $-A$ are transmitted with equal probability. Thus

$$P(\text{Erasure}) = P_2 = Q\left(\frac{(1-\gamma)AT}{\sigma_n}\right) - Q\left(\frac{(1+\gamma)AT}{\sigma_n}\right) \tag{10.149}$$

It was shown in Chapter 7 that the variance of N for an integrate-and-dump detector is

$$\sigma_n^2 = \frac{1}{2}N_0T \tag{10.150}$$

Thus

$$\frac{AT}{\sigma_n} = AT\sqrt{\frac{2}{N_0T}} = \sqrt{\frac{2A^2T}{N_0}} \tag{10.151}$$

which is

$$\frac{AT}{\sigma_n} = \sqrt{2z} \tag{10.152}$$

With this substitution the probability of an erasure becomes

$$P_2 = Q[(1-\gamma)\sqrt{2z}] - Q[(1+\gamma)\sqrt{2z}] \tag{10.153}$$

The probability of error, conditioned on the transmission of $-A$, is

$$P(\text{Error}|-A) = \frac{1}{\sqrt{2\pi}\sigma_n}\int_{a_2}^{\infty}\exp\left(-\frac{(v+AT)^2}{2\sigma_n^2}\right)dv \tag{10.154}$$

Using the same steps as used to determine the probability of erasure gives

$$P(\text{Error}) = P_1 = Q[(1+\gamma)\sqrt{2z}]$$

The MATLAB code for calculating and plotting the error probability and the expected number of transmissions is shown below. For comparison purposes, the probability of error for a single-threshold integrate-and-dump detector is also determined [simply let $\gamma = 0$ in (10.154)] for comparison purposes. Results are shown in the following two illustrations. The probability of error is shown in Figure 10.32, and Figure 10.33 illustrates the average number of transmissions.

```
g = 0.2;                          % gamma
zdB = 0:0.1:10;                   % z in dB
z = 10.^(zdB/10);                 % vector of z values
q1 = qfn((1-g)*sqrt(2*z));
q2 = qfn((1+g)*sqrt(2*z));
qt = Q(sqrt(2*z));                % gamma=0 case
p2 = q1-q2;                       % P2
p1 = q2;                          % P1
pe = p1./(1-p2);                  % probability of error
semilogy(zdB,pe,zdB,qt)
xlabel('z - dB')
ylabel('Probability of Error')
pause
```

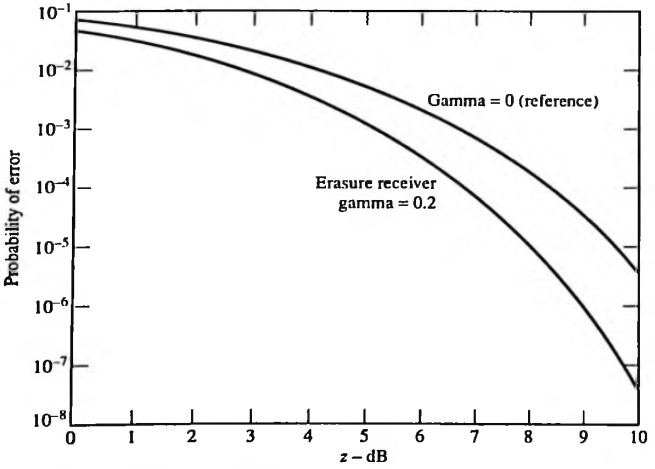


FIGURE 10.32 Probability of error.

```
N = 1./(1-p2);
plot(zdB,N)
xlabel('Z - dB')
ylabel('Expected Number of Transmissions')
```

In the preceding program the Gaussian Q-function is calculated using the MATLAB routine

```
function out=qfn(x)
% Gaussian Q Function
out=0.5*erfc(x/sqrt(2));
```

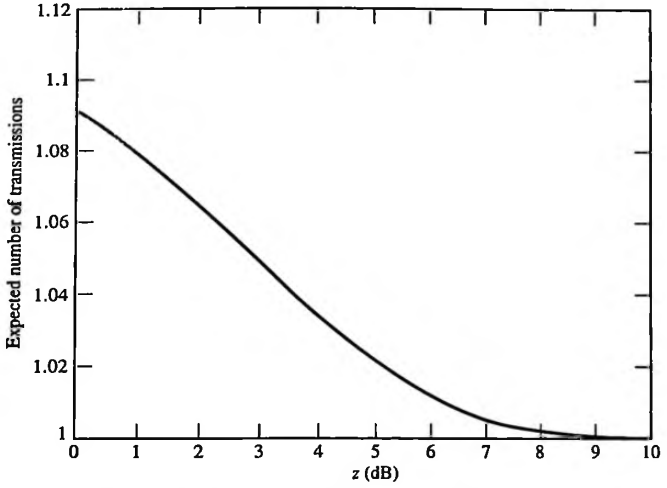


FIGURE 10.33 Expected number of transmissions. ■

10.4 MODULATION AND BANDWIDTH EFFICIENCY

In Chapter 6, signal-to-noise ratios were computed at various points in a communication system. Of particular interest were the signal-to-noise ratio at the input to the demodulator and the signal-to-noise ratio of the demodulated output. These were referred to as the *predetection SNR*, $(\text{SNR})_T$, and the *postdetection SNR*, $(\text{SNR})_D$, respectively. The ratio of these parameters, the detection gain, has been widely used as a figure of merit for the system. In this section we will compare the behavior of $(\text{SNR})_D$ as a function of $(\text{SNR})_T$ for several systems. First, however, we investigate the behavior of an *optimum*, but *unrealizable* system. This study will provide a firm basis for comparison and also provide additional insight into the concept of the trade-off of bandwidth for signal-to-noise ratio.

10.4.1 Bandwidth and SNR

The block diagram of a communication system is illustrated in Figure 10.34. We will focus on the receiver portion of the system. The SNR at the output of the predetection filter, $(\text{SNR})_T$, yields the maximum rate at which information may arrive at the receiver. From the Shannon-Hartley law, this rate, C_T is

$$C_T = B_T \log [1 + (\text{SNR})_T] \tag{10.155}$$

where B_T, the predetection bandwidth, is typically the bandwidth of the modulated signal. Since (10.155) is based on the Shannon-Hartley law, it is valid only for additive *white* Gaussian noise cases. The SNR of the demodulated output, $(\text{SNR})_D$, yields the maximum rate at which information may leave the receiver. This rate, denoted C_D is given by

$$C_D = W \log[1 + (\text{SNR})_D] \tag{10.156}$$

where W is the bandwidth of the message signal.

An optimal modulation is defined as one for which $C_D = C_T$. For this system, demodulation is accomplished, in the presence of noise, without loss of information. Equating C_D to C_T yields

$$(\text{SNR})_D = [1 + (SNR)_T]^{B_T/W} - 1 \tag{10.157}$$

which shows that the optimum exchange of bandwidth for SNR is *exponential*. Recall that we first encountered the trade-off between bandwidth and system performance, in terms of the SNR at the output of the demodulator in Chapter 6 when the performance of angle modulation in the presence of noise was studied.

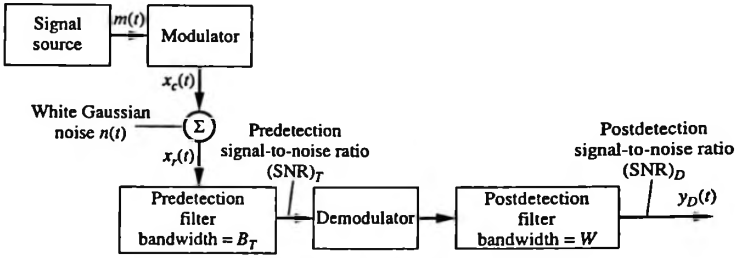


FIGURE 10.34 Block diagram of a communication system.

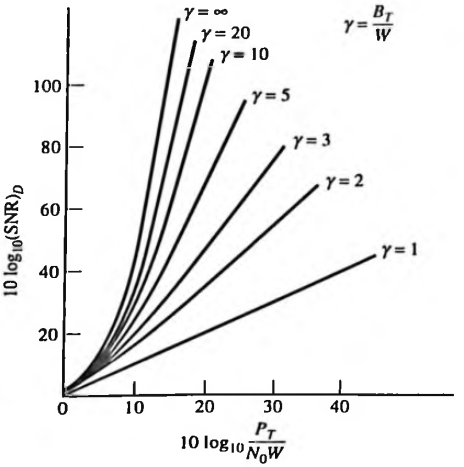


FIGURE 10.35 Performance of an optimum modulation system.

The ratio of transmission bandwidth B_T to the message bandwidth W is referred to as the *bandwidth expansion factor* γ. To fully understand the role of this parameter, we write the predetection SNR as

$$(\text{SNR})_T = \frac{P_T}{N_0 B_T} = \frac{W}{B_T} \frac{P_T}{N_0 W} = \frac{1}{\gamma} \frac{P_T}{N_0 W} \tag{10.158}$$

Thus (10.157) can be expressed as

$$(\text{SNR})_D = \left[1 + \frac{1}{\gamma} \left(\frac{P_T}{N_0 W}\right)\right]^{\gamma} - 1 \tag{10.159}$$

The relationship between $(\text{SNR})_D$ and $\frac{P_T}{N_0 W}$ is illustrated in Figure 10.35.

10.4.2 Comparison of Modulation Systems

The concept of an optimal modulation system provides a basis for comparing system performance. For example, an ideal single-sideband system has a bandwidth expansion factor of unity, since the transmission bandwidth is ideally equal to the message bandwidth. Thus the postdetection SNR of the optimal modulation system is, from (10.159) with γ equal to 1,

$$(\text{SNR})_D = \frac{P_T}{N_0 W} \tag{10.160}$$

This is exactly the same result as that obtained in Chapter 6 for an SSB system using coherent demodulation with a perfect phase reference. Therefore, if the transmission bandwidth B_T of an SSB system is *exactly* equal to the message bandwidth W, SSB is optimal, assuming that there are no other error sources. Of course, this can never be achieved in practice, since *ideal* filters would be required in addition to *perfect* phase coherence of the demodulation carrier.

The story is quite different with DSB, AM, and QDSB. For these systems, $\gamma = 2$. In Chapter 6 we saw that the postdetection SNR for DSB and QDSB, assuming perfect

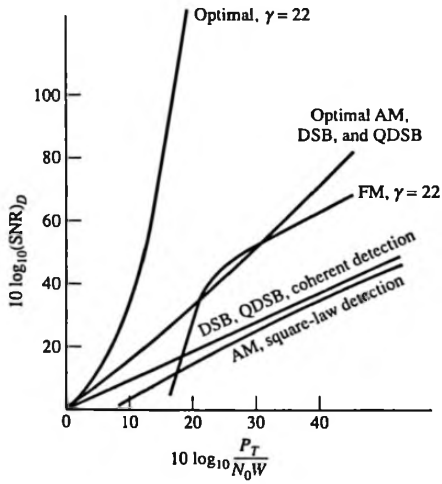


FIGURE 10.36 Performance comparison of analog systems.

coherent demodulation, is

$$(\text{SNR})_D = \frac{P_T}{N_0 W} \tag{10.161}$$

whereas for the optimal system it is given by (10.159) with $\gamma = 2$.

These results are shown in Figure 10.36, along with the result for AM with square-law demodulation. It can be seen that these systems are far from optimal, especially for large values of $\frac{P_T}{N_0 W}$.

Also shown in Figure 10.35 is the result for FM without preemphasis, with sinusoidal modulation, assuming a modulation index of 10. With this modulation index, the bandwidth expansion factor is

$$\gamma = \frac{2(\beta + 1) W}{W} = 22 \tag{10.162}$$

The realizable performance of the FM system is taken from Figure 6.18. It can be seen that realizable systems fall far short of optimal if γ and $\frac{P_T}{N_0 W}$ are large.

10.5 BANDWIDTH AND POWER-EFFICIENT MODULATION

A desirable characteristic of any modulation scheme is the simultaneous conservation of bandwidth and power. Since the late 1970s, the approach to this challenge has been to combine coding and modulation. There have been two approaches: (1) continuous phase modulation (CPM)[5] with memory extended over several modulation symbols by cyclical use

[5]CPM has been explored by many investigators. For introductory treatments see C.-E. Sundberg, "Continuous Phase Modulation," *IEEE Communications Magazine*, Vol. 24, April 1986, pp. 25–38, and J. B. Anderson and C.-E. Sundberg, "Advances in Constant Envelope Coded Modulation," *IEEE Communications Magazine*, Vol. 29, Dec. 1991, pp. 36–45.

of a set of modulation indices; and (2) combining coding with an M-ary modulation scheme, referred to as *trellis-coded modulation* (TCM).[6] We briefly explore the latter approach in this section. For an introductory discussion of the former approach, see Ziemer and Peterson (2001), Chapter 4. Sklar (2001) is a well-written reference with more examples on TCM than given in this short section.

In Chapter 8 it was illustrated through the use of signal-space diagrams that the most probable errors in an M-ary modulation scheme result from mistaking a signal point closest in Euclidian distance to the transmitted signal point as corresponding to the actual transmitted signal. Ungerboeck's solution to this problem was to use coding in conjunction with M-ary modulation to increase the minimum Euclidian distance between those signal points most likely to be confused without increasing the average power or bandwidth over an uncoded scheme transmitting the same number of bits per second. We illustrate the procedure with a specific example.

We wish to compare a TCM system and a QPSK system operating at the same data rates. Since the QPSK system transmits 2 bits per signal phase (signal-space point), we can keep that same data rate with the TCM system by employing an 8-PSK modulator, which carries 3 bits per signal phase, in conjunction with a convolutional encoder that produces three encoded symbols for every two input data bits, that is, a rate $\frac{2}{3}$ encoder. Figure 10.37(a) shows an encoder for accomplishing this and Figure 10.37(b) shows the corresponding trellis diagram. The encoder operates by taking the first data bit as the input to a rate $\frac{1}{2}$ convolutional encoder that produces the first and second encoded symbols, and the second data bit directly as the third encoded symbol. These are then used to select the particular signal phase to be transmitted according to the following rules:

1. All parallel transitions in the trellis are assigned the maximum possible Euclidian distance. Since these transitions differ by one code symbol (the one corresponding to the uncoded bit in this example), an error in decoding these transitions amounts to a single bit error, which is minimized by this procedure.
2. All transitions emanating or converging into a trellis state are assigned the next to largest possible Euclidian distance separation.

The application of these rules to assigning the encoded symbols to a signal phase in an 8-PSK system can be done with a technique known as set partitioning, which is illustrated in Figure 10.38. If the encoded symbol c_1 is a 0, the left branch is chosen in the first tier of the tree, whereas if c_1 is a 1, the right branch is chosen. A similar procedure is followed for tiers 2 and 3 of the tree, with the result being that a unique signal phase is chosen for each possible encoded output.

To decode the TCM signal, the received signal plus noise in each signaling interval is correlated with each possible transition in the trellis and a search is made through the trellis by means of a Viterbi algorithm using the sum of these cross-correlations as metrics rather than Hamming distance as discussed in conjunction with Figure 10.25 (this is called the use of a soft decision metric). Also note that the decoding procedure is twice as complicated since two branches correspond to a path from one trellis state to the next due to the uncoded bit becoming the third symbol in the code. In choosing the two decoded bits for a surviving branch, the first decoded bit of the pair corresponds to the input bit b_1 that

[6]Three introductory treatments of TCM can be found in G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Transactions on Information Theory*, Vol. IT-28, Jan. 1982, pp. 55–66, G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part I: Introduction," *IEEE Communications Magazine*, Vol. 25, Feb. 1987, pp. 5–11, and G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part II: State of the Art," *IEEE Communications Magazine*, Vol. 25, Feb. 1987, pp. 12–21.

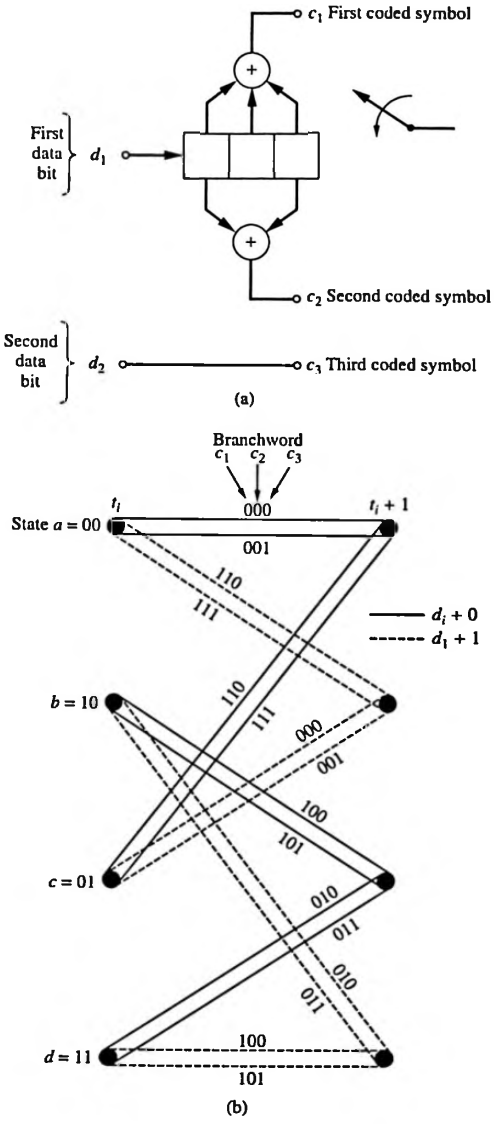


FIGURE 10.37 (a) Convolutional encoder and (b) trellis diagram corresponding to 4-state, 8-PSK TCM.

produced the state transition of the branch being decoded. The second decoded bit of the pair is the same as the third symbol c_3 of that branch word, since c_3 is the same as the uncoded bit b_2.

Ungerboeck has characterized the event error probability performance of a signaling method in terms of the free distance of the signal set. For high signal-to-noise ratios, the probability of an error event (i.e., the probability that at any given time the VA makes a

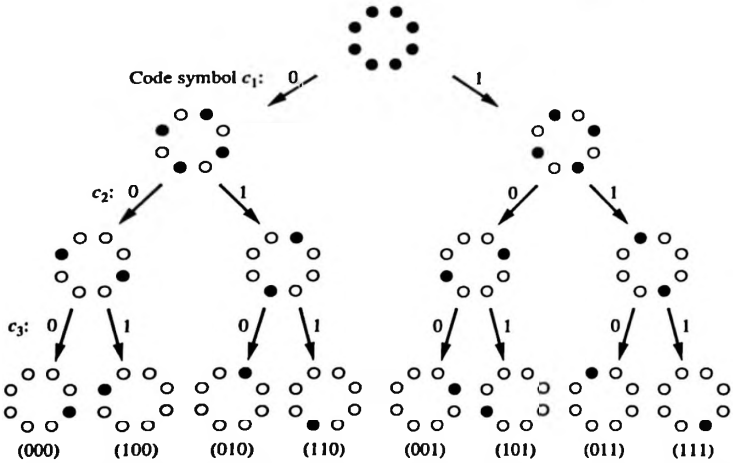


FIGURE 10.38 Set partitioning for assigning a rate 2/3 encoder output to 8-PSK signal points while obeying the rules for maximizing free distance. (From G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Transactions on Information Theory*, Vol. IT-28, Jan. 1982, pp. 55–66.)

wrong decision among the signals associated with parallel transitions, or starts to make a sequence of wrong decisions along some path diverging from more than one transition from the correct path) is well approximated by

$$P(\text{error event}) = N_{\text{free}} Q\left(\frac{d_{\text{free}}}{2\sigma}\right) \tag{10.163}$$

where N_{free} denotes the number of nearest-neighbor signal sequences with distance d_{free} that diverge at any state from a transmitted signal sequence, and reemerge with it after one or more transitions. (The free distance is often calculated by assuming the signal energy has been normalized to unity and that the noise standard deviation σ accounts for this normalization.)

For uncoded QPSK, we have $d_{\text{free}} = 2^{1/2}$ and $N_{\text{free}} = 2$ (there are two adjacent signal points at distance $d_{\text{free}} = 2^{1/2}$), whereas for 4-state-coded 8-PSK we have $d_{\text{free}} = 2$ and $N_{\text{free}} = 1$. Ignoring the factor N_{free}, we have an asymptotic gain due to TCM over uncoded QPSK of $2^2/(2^{1/2})^2 = 2 = 3$ dB. Figure 10.39, also from Ungerboeck, compares the asymptotic lower bound for the error event probability with simulation results.

It should be clear that the TCM encoding/modulation procedure can be generalized to higher-level M-ary schemes. Ungerboeck shows that this observation can be generalized as follows: (1) Of the m bits to be transmitted per encoder/modulator operation, $k \leq m$ bits are expanded to $k + 1$ coded symbols by a binary rate $k/(k + 1)$ convolutional encoder; (2) the $k + 1$ coded symbols select one of 2^{k+1} subsets of a redundant 2^{m+1}-ary signal set; (3) the remaining $m - k$ symbols determine one of 2^{m-k} signals within the selected subset. It should also be stated that one may use block codes or other modulation schemes, such as M-ary ASK or QASK, to implement a TCM system.

Another parameter that influences the performance of a TCM system is the constraint length of the code, ν, which is equivalent to saying that the encoder has 2^{ν} states. Ungerboeck

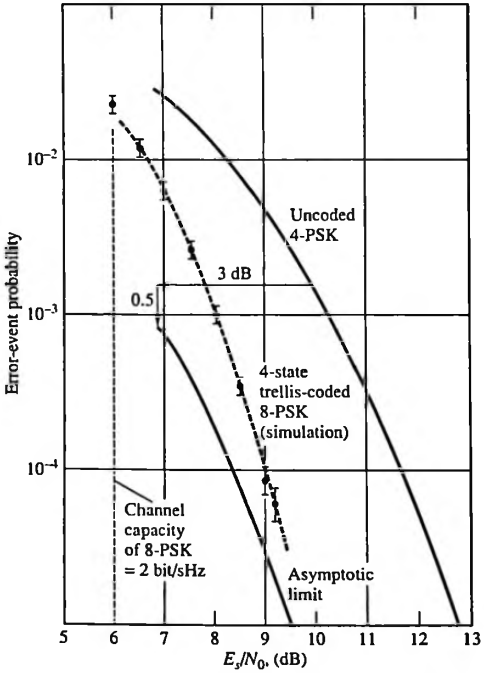


FIGURE 10.39 Performance for a 4-state, 8-PSK TCM signaling scheme. (From G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Set, Part I: Introduction," *IEEE Communications Magazine*, Feb. 1987, Vol. 25, pp. 5–11.)

has published asymptotic gains for TCM systems with various constraint lengths. These are given in Table 10.8.

Finally, the paper by Viterbi et al. (1989) gives a simplified scheme for M-ary PSK that uses a single rate $\frac{1}{2}$, 64-state binary convolutional code for which VLSI circuit implementations are plentiful. A technique known as puncturing converts it to r0te $(n-1)/n$.

10.6 SUMMARY

1. The information associated with the occurrence of an event is defined as the logarithm of the probability of the event. If a base 2 logarithm is used, the measure of information is the bit.
2. The average information associated with a set of source outputs is known as the entropy of the source. The entropy function has a maximum, and the maximum occurs when all source states are equally likely. Entropy is average uncertainty.
3. A channel with n inputs and m outputs is represented by the nm transition probabilities of the form $P(y_j|x_i)$. The channel model can be a diagram showing the transition probabilities or a matrix of the transition probabilities.

TABLE 10.8 Asymptotic Coding Gains for TCM Systems

No. of States, 2^r	k	Asymtotic Coding Gain (dB) $G_{8PSK/QPSK}$ $m = 2$	Asymtotic Coding Gain (dB) $G_{16PSK/8PSK}$ $m = 3$
4	1	3.01	—
8	2	3.60	—
16	2	4.13	—
32	2	4.59	—
64	2	5.01	—
128	2	5.17	—
256	2	5.75	—
4	1	—	3.54
8	1	—	4.01
16	1	—	4.44
32	1	—	5.13
64	1	—	5.33
128	1	—	5.33
256	2	—	5.51

SOURCE : Adapted from G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Part II: States of the Art," *IEEE Communications Magazine*, Vol. 25. Feb. 1987, pp. 12–21.

4. A number of entropies can be defined for a system. The entropies $H(X)$ and $H(Y)$ denote the average uncertainty of the channel input and output, respectively. The quantity $H(X|Y)$ is the average uncertainty of the channel input given the output, and $H(Y|X)$ is the average uncertainty of the channel output given the input. The quantity $H(X, Y)$ is the average uncertainty of the communication system as a whole.
5. The mutual information between the input and output of a channel is given by

$$I(X;Y) = H(X) - H(X|Y)$$

or

$$I(X;Y) = H(Y) - H(Y|X)$$

The maximum value of mutual information, where the maximization is with respect to the source probabilities, is known as the channel capacity.

6. Source encoding is used to remove redundancy from a source output so that the information per transmitted symbol can be maximized. If the source rate is less than the channel capacity, it is possible to encode the source output so that the information can be transmitted through the channel. This is accomplished by forming source extensions and encoding the symbols of the extended source into codewords having minimum average word length. The minimum average word length $\overline{L}$ approaches $H(X^n) = nH(X)$, where $H(X^n)$ is the entropy of the nth order extension of a source having entropy $H(X)$, as n increases.
7. Two source encoding schemes are the Shannon-Fano technique and the Huffman technique. The Huffman technique yields an optimum source code, which is the source code having minimum average word length.
8. Error-free transmission on a noisy channel can be accomplished if the source rate is less than the channel capacity. This is accomplished using channel codes.

9. The capacity of an additive white Gaussian noise (AWGN) channel is

$$C_c = B \log_2 \left(1 + \frac{S}{N}\right)$$

where B is the channel bandwidth and S/N is the signal-to-noise ratio. This is known as the Shannon-Hartley law.

10. An (n, k) block code is generated by appending $r = n - k$ parity symbols to a k-symbol source sequence. This yields an n-symbol codeword.

11. Decoding is typically accomplished by computing the Hamming distance from the received n-symbol sequence to each of the possible transmitted codewords. The codeword closest in Hamming distance to the received sequence is the most likely transmitted codeword. The two codewords closest in Hamming distance determine the minimum distance of the code d_m. The code can correct $\frac{1}{2}(d_m - 1)$ errors.

12. A single-parity-check code is formed by adding a single-parity symbol to the information sequence. This code can detect single errors but provides no error-correcting capability.

13. The rate of a block code is k/n. The best codes provide powerful error correction capabilities in combination with high rate.

14. Repetition codes are formed by transmitting each source symbol an odd number of times. Repetition codes provide powerful error-correction capabilities but have very low rate.

15. The parity-check matrix $[H]$ is defined such that $[H][T] = [0]$, where $[T]$ is the transmitted codeword written as a column vector. If the received sequence is denoted by the column vector $[R]$, the syndrome $[S]$ is determined from $[S] = [H][R]$. This can be shown to be equivalent to $[S] = [H][E]$, where $[E]$ is the error sequence. If a single error occurs in the transmission of a codeword, the syndrome is the column of $[H]$ corresponding to the error position.

16. The generator matrix $[G]$ of a parity-check code is determined such that $[T] = [G][A]$, where $[T]$ is the n-symbol transmitted sequence and $[A]$ is the k-symbol information sequence. Both $[T]$ and $[A]$ are written as column vectors.

17. For a group code, the modulo-2 sum of any two codewords is another codeword.

18. A Hamming code is a single-error-correcting code such that the columns of the parity-check matrix correspond to the binary representation of the column index.

19. Cyclic codes are a class of block codes; such a cyclic shift of the codeword symbols always yields another codeword. These codes are very useful because implementation is simple. Implementation of both the encoder and decoder is accomplished using shift registers and basic logic components.

20. The channel symbol error probability of a coded system is greater than the symbol error probability of an uncoded system because the available energy for transmission of k information symbols must be spread over the n-symbol codeword rather than just the k-information symbols. The error-correcting capability of the code often allows a net performance gain to be realized. The performance gain depends on the choice of code and the channel characteristics.

21. Convolutional codes are easily generated using simple shift registers and modulo-2 adders. Decoding is accomplished using a tree-search technique, which is often implemented using the Viterbi algorithm. The constraint span is the code parameter having the most significant impact on performance.

22. Interleaved codes are useful for burst noise environments.
23. The feedback channel system makes use of a null-zone receiver, and a retransmission is requested if the receiver decision falls within the null zone. If a feedback channel is available, the error probability can be significantly reduced with only a slight increase in the required number of transmissions.
24. Use of the Shannon-Hartley law yields the concept of optimum modulation for a system operating in an AWGN environment. The result is the performance of an optimum system in terms of predetection and postdetection bandwidth. The trade-off between bandwidth and signal-to-noise ratio is easily seen.
25. Trellis-coded modulation is a scheme for combining M-ary modulation with coding in a way that increases the Euclidian distance between those signal points for which errors are most likely without increasing the average power or bandwidth over an uncoded scheme having the same bit rate. Decoding is accomplished using a Viterbi decoder that accumulates decision metrics (soft decisions) rather than Hamming distances (hard decisions).

10.7 FURTHER READING

An exposition of information theory and coding that was anywhere near complete would, of necessity, be presented at a level beyond that intended for this text. The purpose in the present chapter is to present some of the basic ideas of information theory at a level consistent with the rest of this book. You should be motivated by this to further study.

The original paper by Shannon (1948) is stimulating reading at about the same level as this chapter. This paper is available as a paperback with an interesting postscript by W. Weaver (Shannon and Weaver, 1963).

Several standard texts on information theory are available. Those by Blahut (1987) and Mansuripur (1987) are written at a slightly higher level than this text. The textbook by Gallager (1968) is a very complete exposition of the subject and is written at the graduate level. Another standard text, which is very complete, is the one by Blahut (1984). This volume is also written at the beginning graduate level. The text by Lin and Costello (1982) is less rigorous than the other two and contains much information on the implementation of decoders. An interesting perspective on coding theory is provided by Arazi (1988). Many other well-written books and review articles are available.

Several other books deserve special mention. The book by Clark and Cain (1981) contains a wealth of practical information concerning coder and decoder design, in addition to the usual theoretical background material. The last chapter, entitled "System Applications," contains a large number of performance curves for different channel models. The book by Viterbi and Omura (1979) is an excellent basic treatment of information theory and coding, and it contains an excellent analysis of the Viterbi algorithm.

As mentioned in the last section of this chapter, the subject of bandwidth and power-efficient communications is very important to the implementation of modern systems. Continuous phase modulation is treated in the text by Ziemer and Peterson (2001). An introductory treatment of trellis-coded modulation, (TCM), including a discussion of coding gain, is contained in the book by Sklar (2001). The book by Biglieri, Divsalar, McLane, and Simon (1991) is a complete treatment of TCM theory, performance, and implementation. A recent book on turbo codes is Heegard (1999).

10.8 PROBLEMS

SECTION 10.1

10.1. A message occurs with a probability of 0.8. Determine the information associated with the message in bits, nats, and hartleys.

10.2. Assume that you have a standard deck of 52 cards (jokers have been removed).

(a) What is the information associated with the drawing of a single card from the deck?

(b) What is the information associated with the drawing of a pair of cards, assuming that the first card drawn is replaced in the deck prior to drawing the second card?

(c) What is the information associated with the drawing of a pair of cards assuming that the first card drawn is not replaced in the deck prior to drawing the second card?

10.3. A source has five outputs denoted $[m_1, m_2, m_3, m_4, m_5]$ with respective probabilities [0.3, 0.25, 0.25, 0.15, 0.05]. Determine the entropy of the source. What is the maximum entropy of a source with five outputs?

10.4. A source consists of six outputs denoted $[A, B, C, D, E, F]$ with respective probabilities [0.3, 0.2, 0.15, 0.15, 0.1, 0.1]. Determine the entropy of the source.

10.5. A channel has the following transition matrix:

$$\begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.25 & 0.65 \end{bmatrix}$$

(a) Sketch the channel diagram showing all transition probabilities.

(b) Determine the channel output probabilities assuming that the input probabilities are equal.

(c) Determine the channel input probabilities that result in equally likely channel outputs.

(d) Determine the joint probability matrix using (c) above.

10.6. Describe the channel transition probability matrix and joint probability matrix for a noiseless channel.

10.7. A binary symmetric channel has an error probability of 0.001. How many of these channels can be cascaded before the overall error probability exceeds 0.08?

10.8. Show that $H(Y) \geq H(Y|X)$.

10.9. A channel is described by the transition probability matrix

$$[P(Y \mid X)] = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Determine the channel capacity and the source probabilities that yield capacity.

10.10. A channel is described by the matrix

$$[P(Y \mid X)] = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Determine the channel capacity and the source probabilities that yield capacity.

10.11. A channel has two inputs, (0, 1), and three outputs, $(0, e, 1)$, where e indicates an erasure; that is, there is no output for the corresponding input. The channel matrix is

$$\begin{bmatrix} p & 1-p & 0 \\ 0 & 1-p & p \end{bmatrix}$$

Compute the channel capacity.

10.12. Determine the capacity of the channel described by the channel matrix shown below. Sketch your result as a function of p and give an intuitive argument that supports your sketch. (*Note*: $q = 1 - p$.)

$$\begin{bmatrix} p & q & 0 & 0 \\ q & p & 0 & 0 \\ 0 & 0 & p & q \\ 0 & 0 & q & p \end{bmatrix}$$

10.13. From the entropy definition given in Equations (10.25) through (10.29), derive Equations (10.30) and (10.31).

10.14. The input to a quantizer is a random signal having an amplitude probability density function

$$f_X(x) = \begin{cases} ae^{-ax}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The signal is to be quantized using four quantizing levels x_i as shown in Figure 10.40. Determine the values x_i, $i = 1, 2, 3$, in terms of a so that the entropy at the quantizer output is maximized.

10.15. Repeat the preceding problem assuming that the input to the quantizer has the Rayleigh probability

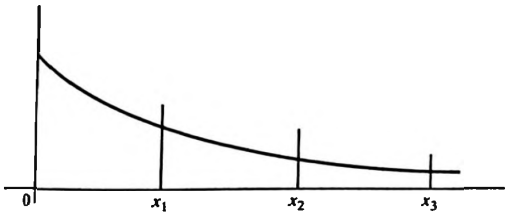


FIGURE 10.40

density function

$$f_X(x) = \begin{cases} \frac{x}{a^2} e^{-x^2/2a^2}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

10.16. A signal has a Gaussian amplitude-density function with zero mean and variance σ^2. The signal is sampled at 500 samples per second. The samples are quantized according to the following table. Determine the entropy at the quantizer output and the information rate in samples per second.

Quantizer Input	Quantizer Output
$-\infty < x_i < -2\sigma$	m_0
$-2\sigma < x_i < -\sigma$	m_1
$-\sigma < x_i < 0$	m_2
$0 < x_i < \sigma$	m_3
$\sigma < x_i < 2\sigma$	m_4
$2\sigma < x_i < \infty$	m_5

10.17. Determine the quantizing levels, in terms of σ, so that the entropy at the output of a quantizer is maximized. Assume that there are six quantizing levels and that the quantizer is a zero-mean Gaussian process as in the previous problem.

10.18. Two binary symmetrical channels are in cascade, as shown in Figure 10.41. Determine the capacity of each channel. The overall system with inputs x_1 and x_2 and outputs z_1 and z_2 can be represented as shown with p_{11}, p_{12}, p_{21}, and p_{22} properly chosen. Determine these four probabilities and the capacity of the overall system. Comment on the results.

10.19. A two-hop satellite communications channel uses BPSK signaling. The uplink signal-to-noise ratio is 8 dB and the downlink signal-to-noise ratio is 5 dB, where the signal-to-noise ratio is the signal power divided by the noise power in the bit-rate bandwidth. Determine the overall error probability.

SECTION 10.2

10.20. A source has two outputs $[A, B]$ with respective probabilities $[3/4, 1/4]$. Determine the entropy of the fourth-order extension of this source using two different methods.

10.21. Calculate the entropy of the fourth-order extension of the source defined in Table 10.1. Determine $\bar{L}/n$ for $n = 4$ and add this result to those shown in Figure 10.9. Determine the efficiency of the resulting codes for $n = 1, 2, 3$, and 4.

10.22. A source has five equally likely output messages. Determine a Shannon-Fano code for the source and determine the efficiency of the resulting code. Repeat for the Huffman code and compare the results.

10.23. A source has five outputs denoted $[m_1, m_2, m_3, m_4, m_5]$ with respective probabilities $[0.41, 0.19, 0.16, 0.15, 0.9]$. Determine the codewords to represent the source outputs using both the Shannon-Fano and the Huffman techniques.

10.24. A binary source has output probabilities $[0.85, 0.15]$. The channel can transmit 350 binary symbols per second at the capacity of 1 bit symbol. Determine the maximum source symbol rate if transmission is to be accomplished.

10.25. A source output consists of nine equally likely messages. Encode the source output using both binary Shannon-Fano and Huffman codes. Compute the efficiency of both of the resulting codes and compare the results.

10.26. Repeat the preceding problem assuming that the source has 12 equally likely outputs.

10.27. An analog source has an output described by the probability density function

$$f_X(x) = \begin{cases} 2x, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

The output of the source is quantized into 10 messages using the 9 quantizing levels

$$x_i = 0.1k, \quad k = 0, 1, \ldots, 10$$

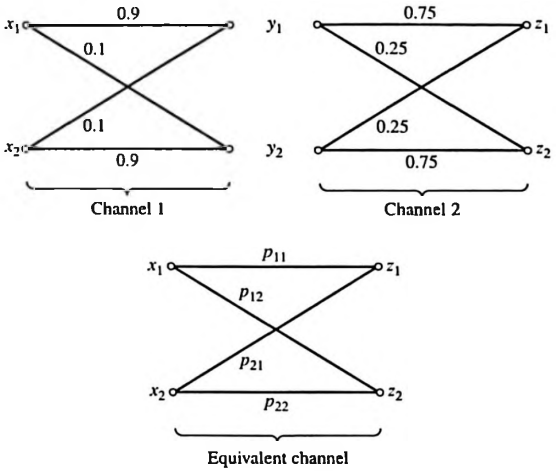


FIGURE 10.41

The resulting messages are encoded using a binary Huffman code. Assuming that 250 samples of the source are transmitted each second, determine the resulting binary symbol rate in symbols per second. Also determine the information rate in bits per second.

10.28. A source output consists of four messages $[m_1, m_2, m_3, m_4]$ with respective probabilities [0.4, 0.3, 0.2, 0.1]. The second-order extension of the source is to be encoded using a code with a three-symbol alphabet. Determine the codewords using the Shannon-Fano technique and determine the efficiency of the resulting code.

10.29. It can be shown that a necessary and sufficient condition for the existence of an instantaneous binary code with word lengths l_i, $1 \leq i \leq N$, is that

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1$$

This is known as the Kraft inequality. Show that the Kraft inequality is satisfied by the codewords given in Table 10.3. (*Note*: The inequality given above must also be satisfied for uniquely decipherable codes.)

SECTION 10.3

10.30. A continuous bandpass channel can be modeled as illustrated in Figure 10.42. Assuming a signal power of 40 W and a noise power spectral density of 10^{-4} W/Hz, plot the capacity of the channel as a function of the channel bandwidth.

10.31. For the preceding problem determine the capacity in the limit as $B \to \infty$.

10.32. Derive an equation, similar to (10.94), that gives the error probability for a rate $\frac{1}{5}$ repetition code. Plot together, on the same set of axes, the error probability of both a rate of $\frac{1}{3}$ and rate $\frac{1}{5}$ repetition code as a function of $q = 1 - p$.

10.33. Show that a (15, 11) Hamming code is a distance 3 code. (*Hint*: It is not necessary to find all codewords.)

10.34. Write the parity-check matrix and the generator matrix for a (15, 11) single error-correcting code. Assume that the code is systematic. Calculate the codeword corresponding to the all one's information sequence. Calculate the syndrome corresponding to an error in the third position assuming the codeword corresponding to the all one's input sequence.

10.35. A parity-check code has the parity-check matrix

$$[H] = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

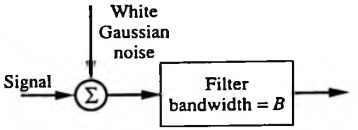


FIGURE 10.42

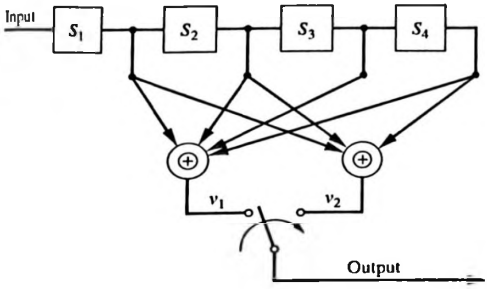


FIGURE 10.43

Determine the generator matrix and find all possible codewords.

10.36. For the code described in the preceding problem, find the codewords $[T_1]$ and $[T_2]$ corresponding to the information sequences

$$[A_1] = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad [A_2] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Using these two codewords, illustrate the group property.

10.37. Determine the generator matrix for a rate $\frac{1}{3}$ and a rate $\frac{1}{5}$ repetition code. Describe the generator matrix for a rate $1/n$ repetition code.

10.38. Determine the relationship between n and k for a Hamming code. Use this result to show that the code rate approaches 1 for large n.

10.39. Determine the generator matrix for the encoder illustrated in Figure 10.16. Use the generator matrix to generate the complete set of codewords and use the results to check the codewords shown in Figure 10.16. Show that these codewords constitute a cyclic code.

10.40. Use the result of the preceding problem to determine the parity-check matrix for the encoder shown in Figure 10.16. Use the parity-check matrix to decode the received sequences 1101001 and 1101011. Compare your result with that shown in Figure 10.17.

10.41. Modify the encoder illustrated in Figure 10.16 by using feedback taps from stages S_1 and S_3 rather than from stages S_2 and S_3. Determine the resulting generator matrix and the complete set of 16 codewords. Is the resulting code a cyclic code?

10.42. Construct a set of performance curves similar to those of Figure 10.18 for a (15, 11) Hamming code.

10.43. Consider the coded system examined in Example 10.12. Show that the probability of three symbol errors in a codeword is negligible compared to the probability of two symbol errors in a codeword for signal-to-noise ratios above a certain level.

10.44. The Hamming code was defined as a code for which the ith column of the parity-check

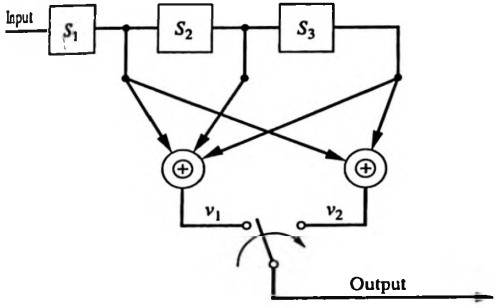


FIGURE 10.44

matrix is the binary representation of the number i. With a little thought it becomes clear that the columns of the parity-check matrix can be permuted without changing the distance properties, and therefore the error-correcting capabilities, of the code. Using this fact, determine a generator matrix and the corresponding parity-check matrix of the systematic code equivalent to a (7, 4) Hamming code. How many different generator matrices can be found?

10.45. What is the constraint span of the encoder shown in Figure 10.43? How many states are required to define the state diagram and the trellis diagram? Draw the state diagram, giving the output for each state transition.

10.46. Determine the state diagram for the convolutional encoder shown in Figure 10.44. Draw the trellis diagram through the first set of steady-state transitions. On a second trellis diagram, show the termination of the trellis to the all-zero state.

10.47. A source produces binary symbols at a rate of 2000 symbols per second. The channel is subjected to error bursts lasting 0.15 sec. Devise an encoding scheme using an interleaved (n, k) Hamming code which allows full correction of the error burst. Assume that the information rate out of the encoder is equal to the information rate into the encoder. What is the minimum time between bursts if your system is to operate properly?

SECTION 10.4

10.48. Compare FM with preemphasis to an optimal modulation system for $\beta = 1, 5$, and 10. Consider only operation above threshold, and assume 20 dB as the value of P_T/N_0W at threshold.

10.9 COMPUTER EXERCISES

10.1. Develop a computer program that allows you to plot the entropy of a source with variable output probabilities. We wish to observe that the maximum source entropy does indeed occur when the source outputs are equally likely. Start with a simple two-output source $[m_1, m_2]$ with respective probabilities $[a, 1-a]$ and plot the entropy as a function of the parameter a. Then consider more complex cases such as a three output source $[m_1, m_2, m_3]$ with respective probabilities $[a, b, 1-a-b]$. Be creative with the manner in which the results are displayed.

10.2. Generate a set of performance curves similar to those illustrated in Figure 10.17 for a single error correcting (15, 11) Hamming code. Repeat for a (31, 26) Hamming code.

10.3. In Exercise 10.12 we compared a coded and an uncoded system based on word error probability. Usually we have more interest in the bit error probability. It is difficult to calculate the bit error probability for a coded system. However, for block codes a very good approximation has been developed, which is

$$P_b = \frac{d}{n}\sum_{i=e+1}^{d}\binom{n}{i}P_{sc}^{i}(1-P_{sc})^{n-i} + \sum_{i=d+1}^{n}\binom{n-1}{i-1}P_{sc}^{i}(1-P_{sc})^{n-i}$$

where d is the minimum distance of the code, e is the number of correctable errors per code word, P_{sc} is the symbol error probability for the coded system, and P_b is the decoded bit-error probability.[7] Develop a computer program for evaluating this expression. Use the resulting progran for calculating and plotting the decoded bit error probability for a (7, 4) Hamming code as a function of E_b/N_0. Compare this result with the performance of an uncoded system.

10.4. Using the computer program developed in the previous computer exercise, compare the performance of a (7, 4) Hamming code and a (23, 12) triple-error-correcting Golay code.

10.5. Develop a computer program for producing the performance curves shown in Figure 10.18.

10.6. Develop a computer program for producing the performance curves shown in Figure 10.20.

[7]D. J. Torrieri, *Principles of Secure Communication Systems*, 2d ed. Norwood, MA: Artech House, 1992, p. 47.