8

BLOCK AND CONVOLUTIONAL CHANNEL CODES

In Chapter 7, we treated channel coding and decoding from a general viewpoint, and showed that even randomly selected codes on the average yield performances close to the capacity of a channel. In the case of orthogonal signals, we demonstrated that the channel capcity limit can be achieved as the number of signals approaches infinity.

In this chapter, we describe specific codes and evaluate their performance for the additive white gaussian noise channel. In particular, we treat two classes of codes, namely, linear block codes and convolutional codes. The code performance is evaluated for both hard-decision decoding and soft-decision decoding.

8-1 LINEAR BLOCK CODES

A block code consists of a set of fixed-length vectors called *code words*. The length of a code word is the number of elements in the vector and is denoted by n. The elements of a code word are selected from an alphabet of q elements. When the alphabet consists of two elements, 0 and 1, the code is a binary code and the elements of any code word are called bits. When the elements of a code word are selected from an alphabet having q elements ($q > 2$), the code is nonbinary. It is interesting to note that when q is a power of 2, i.e., $q = 2^b$ where b is a positive integer, each q-ary element has an equivalent binary representation consisting of b bits, and, thus, a nonbinary code of block length N can be mapped into a binary code of block length $n = bN$.

There are 2^n possible code words in a binary block code of length n. From these 2^n code words, we may select $M = 2^k$ code words ($k < n$) to form a code. Thus, a block of k information bits is mapped into a code word of length n selected from the set of $M = 2^k$ code words. We refer to the resulting block code as an (n, k) code, and the ratio $k/n \equiv R_c$ is defined to be the *rate* of the code. More generally, in a code having q elements, there are q^n possible code words. A subset of $M = 2^k$ code words may be selected to transmit k-bit blocks of information.

Besides the code rate parameter R_c, an important parameter of a code word is its *weight*, which is simply the number of nonzero elements that it contains. In general, each code word has its own weight. The set of all weights in a code constitutes the *weight distribution* of the code. When all the M code words have equal weight, the code is called a *fixed-weight code* or a *constant-weight code*.

The encoding and decoding functions involve the arithmetic operations of addition and multiplication performed on code words. These arithmetic operations are performed according to the conventions of the algebraic field that has as its elements the symbols contained in the alphabet. For example, the symbols in a binary alphabet are 0 and 1; hence, the field has two elements. In general, a field F consists of a set of elements that has two arithmetic operations defined on its elements, namely, addition and multiplication, that satisfy the following properties (axioms).

Addition

1 The set F is closed under addition, i.e., if $a, b \in F$ than $a + b \in F$.

2 Addition is associative, i.e., if a, b, and c are elements of F then $a + (b + c) = (a + b) + c$.

3 Addition is commutative, i.e., $a + b = b + a$.

4 The set contains an element called *zero* that satisfies the condition $a + 0 = a$.

5 Every element in the set has its own negative element. Hence, if b is an element, its negative is denoted by $-b$. The subtraction of two elements, such as $a - b$, is defined as $a + (-b)$.

Multiplication

1 The set F is closed under multiplication, i.e., if $a, b \in F$ then $ab \in F$.

2 Multiplication is associative, i.e., $a(bc) = (ab)c$.

3 Multiplication is commutative, i.e., $ab = ba$.

4 Multiplication is distributive over addition, i.e., $(a + b)c = ac + bc$.

5 The set F contains an element, called the *identity*, that satisfies the condition $a(1) = a$, for any element $a \in F$.

6 Every element of F, except zero, has an inverse. Hence, if $b \in F$ ($b \neq 0$)

then its inverse is defined as b^{-1}, and $bb^{-1} = 1$. The division of two elements, such as $a \div b$, is defined as ab^{-1}

We are very familiar with the field of real numbers and the field of complex numbers. These fields have an infinite number of elements. However, as indicated above, codes are constructed from fields with a finite number of elements. A finite field with q elements is generally called a *Galois field* and denoted by GF(q).

Every field must have a *zero* element and a *one* element. Hence, the simplest field is GF(2). In general, when q is a prime, we can construct the finite field GF(q) consisting of the elements $\{0, 1, \dots, q-1\}$. The addition and multiplication operations on the elements of GF(q) are defined modulo q and denoted as (mod q). For example, the addition and multiplication tables for GF(2) are

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

which are operations (mod 2). Similarly, the field GF(5) is a set consisting of the elements $\{0, 1, 2, 3, 4\}$. The addition and multiplication tables for GF(5) are

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

In general, the finite field GF(q) can be constructed only if q is a prime or a power of a prime. When q is a prime, multiplication and addition are based on modulo-q arithmetic as illustrated above. If $q = p^m$ where p is a prime and m is any positive integer, it is possible to extend the field GF(p) to the field GF(p^m). This is called the *extension field* of GF(p). Multiplication and addition of the elements in the extension field are based on modulo-p arithmetic.

With this brief introduction to the arithmetic operations that may be performed on the elements of code words, let us now consider some basic characteristics of block codes.

Suppose $\mathbf{C}_i$ and $\mathbf{C}_j$ are any two code words in an (n, k) block code. A measure of the difference between the code words is the number of corresponding elements or positions in which they differ. This measure is called the *Hamming distance* between the two code words and is denoted as d_{ij}.

Clearly, d_{ij} for $i \neq j$ satisfies the condition $0 < d_{ij} \leq n$. The smallest value of the set $\{d_{ij}\}$ for the M code words is called the *minimum distance* of the code and is denoted as $d_{\min}$. Since the Hamming distance is a measure of the separation between pairs of code words, it is intimately related to the cross-correlation coefficient between corresponding pairs of waveforms generated from the code words. The relationship is discussed in Section 8-1-4.

Besides characterizing a code as being binary or nonbinary, one can also describe it as either linear or nonlinear. Suppose $\mathbf{C}_i$ and $\mathbf{C}_j$ are two code words in an (n, k) block code and let α_1 and α_2 be any two elements selected from the alphabet. Then the code is said to be linear if and only if $\alpha_1\mathbf{C}_i + \alpha_2\mathbf{C}_j$ is also a code word. This definition implies that a linear code must contain the all-zero code word. Consequently a constant-weight code is nonlinear.

Suppose we have a binary linear block code, and let $\mathbf{C}_i$, $i = 1, 2, \ldots, M$, denote the M code words. For convenience, let $\mathbf{C}_1$ denote the all-zero code word, i.e., $\mathbf{C}_1 = [0\,0\ldots0]$, and let w_r denote the weight of the rth code word. It follows that w_r is the Hamming distance between the code words $\mathbf{C}_r$ and $\mathbf{C}_1$. Thus, the distance $d_{1r} = w_r$. In general, the distance d_{ij} between any pair of code words $\mathbf{C}_i$ and $\mathbf{C}_j$ is simply equal to the weight of the code word formed by taking the difference between $\mathbf{C}_i$ and $\mathbf{C}_j$. Since the code is linear, the difference (equivalent to taking the modulo-2 sum for a binary code) between $\mathbf{C}_i$ and $\mathbf{C}_j$ is also a code word having a weight included in the set $\{w_r\}$. Hence, the weight distribution of a linear code completely characterizes the distance properties of the code. The minimum distance of the code is, therefore,

$$d_{\min} = \min_{r, r \neq 1} \{w_r\} \tag{8-1-1}$$

A number of elementary concepts from linear algebra are particularly useful in dealing with linear block codes. Specifically, the set of all n-tuples (vectors with n elements) form a vector space S. If we select a set of $k < n$ linearly independent vectors from S and from these construct the set of all linear combinations of these vectors, the resulting set forms a subspace of S, say S_c, of dimension k. Any set of k linearly independent vectors in the subspace S_c constitutes a basis. Now consider the set of vectors in S that are orthogonal to every vector in a basis for S_c (and, hence, orthogonal to all vectors in S_c). This set of vectors is also a subspace of S and is called the *null space* of S_c. If the dimension of S_c is k, the dimension of the null space is $n - k$.

Expressed in terms appropriate for binary block codes, the vector space S consists of the 2^n binary valued n-tuples. The linear (n, k) code is a set of 2^k n-tuples called *code words*, which forms a subspace S_c over the field of two elements. Since there are 2^k code words in S_c, a basis for S_c has k code words. That is, k linearly independent code words are required to construct 2^k linear combinations, thus generating the entire code. The null space of S_c is another linear code, which consists of 2^{n-k} code words of block length n and $n - k$ information bits. Its dimension is $n - k$. In Section 8-1-1, we consider these relationships in greater detail.

8-1-1 The Generator Matrix and the Parity Check Matrix

Let $x_{m1}, x_{m2}, \ldots, x_{mk}$ denote the k information bits encoded into the code word $\mathbf{C}_m$. Throughout this chapter, we follow the established convention in coding of representing code words as row vectors. Thus, the vector of k information bits into the encoder is denoted by

$$\mathbf{X}_m = [x_{m1} \quad x_{m2} \quad \ldots \quad x_{mk}]$$

and the output of the encoder is the vector

$$\mathbf{C}_m = [c_{m1} \quad c_{m2} \quad \ldots \quad c_{mn}]$$

The encoding operation performed in a linear binary block encoder can be represented by a set of n equations of the form

$$c_{mj} = x_{m1}g_{1j} + x_{m2}g_{2j} + \ldots + x_{mk}g_{kj}, \qquad j = 1, 2, \ldots, n \qquad (8\text{-}1\text{-}2)$$

where $g_{ij} = 0$ or 1 and $x_{mi}g_{ij}$ represents the product of x_{mi} and g_{ij}. The linear equations (8-1-2) may also be represented in a matrix form as

$$\mathbf{C}_m = \mathbf{X}_m\mathbf{G} \qquad (8\text{-}1\text{-}3)$$

where $\mathbf{G}$, called the *generator matrix* of the code, is

$$\mathbf{G} = \begin{bmatrix} \leftarrow \mathbf{g}_1 \rightarrow \\ \leftarrow \mathbf{g}_2 \rightarrow \\ \vdots \\ \leftarrow \mathbf{g}_k \rightarrow \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix} \qquad (8\text{-}1\text{-}4)$$

Note that any code word is simply a linear combination of the vectors $\{\mathbf{g}_i\}$ of $\mathbf{G}$, i.e.,

$$\mathbf{C}_m = x_{m1}\mathbf{g}_1 + x_{m2}\mathbf{g}_2 + \ldots + x_{mk}\mathbf{g}_k \qquad (8\text{-}1\text{-}5)$$

Since the linear (n, k) code with 2^k code words is a subspace of dimension k, the row vectors $\{\mathbf{g}_i\}$ of the generator matrix $\mathbf{G}$ must be linearly independent, i.e., they must span a subspace of k dimensions. In other words, the $\{\mathbf{g}_i\}$ must be a basis for the (n, k) code. We note that the set of basis vectors is not unique, and, hence, $\mathbf{G}$ is not unique. We also note that, since the subspace has dimension k, the rank of $\mathbf{G}$ is k.

Any generator matrix of an (n, k) code can be reduced by row operations (and column permutations) to the "systematic form."

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}] = \left[\begin{array}{ccccc|cccc} 1 & 0 & 0 & \ldots & 0 & p_{11} & p_{12} & \cdots & p_{1n-k} \\ 0 & 1 & 0 & \ldots & 0 & p_{21} & p_{22} & \cdots & p_{2n-k} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \ldots & 1 & p_{k1} & p_{k2} & \cdots & p_{kn-k} \end{array}\right] \qquad (8\text{-}1\text{-}6)$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n-k)$ matrix that determines the $n-k$ redundant bits or parity check bits. Note that a generator matrix of the systematic form generates a linear block code in which the first k bits of each code word are identical to the information bits to be transmitted, and the remaining $n-k$ bits of each code word are linear combinations of the k information bits. These $n-k$ redundant bits are called *parity check bits*. The resulting (n, k) code is called a *systematic code*.

An (n, k) code generated by a generator matrix that is not in the systematic form (8-1-6) is called *nonsystematic*. However, such a generator matrix is equivalent to a generator matrix of the systematic form in the sense that one can be obtained from the other by elementary row operations and column permutations. The two (n, k) linear codes generated by the two equivalent generator matrices are said to be *equivalent*, and one can be obtained from the other by a permutation of the places of every element. Thus, every linear (n, k) code is equivalent to a linear systematic (n, k) code.

Example 8-1-1

Consider a (7, 4) code with generator matrix

$$\mathbf{G} = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] = [\mathbf{I}_4 \,\vdots\, \mathbf{P}] \tag{8-1-7}$$

A typical code word may be expressed as

$$\mathbf{C}_m = [x_{m1} \quad x_{m2} \quad x_{m3} \quad x_{m4} \quad c_{m5} \quad c_{m6} \quad c_{m7}]$$

where the $\{x_{mj}\}$ represents the four information bits and the $\{c_{mj}\}$ represent the three parity check bits given by

$$\begin{aligned} c_{m5} &= x_{m1} + x_{m2} + x_{m3} \\ c_{m6} &= x_{m2} + x_{m3} + x_{m4} \\ c_{m7} &= x_{m1} + x_{m2} + x_{m4} \end{aligned} \tag{8-1-8}$$

A linear systematic (n, k) binary block encoder may be implemented by using a k-bit shift register and $n-k$ modulo-2 adders tied to the appropriate stages of the shift register. The $n-k$ adders generate the parity check bits, which are subsequently stored temporarily in a second shift register of length $n-k$. The k-bit block of information bits shifted into the k-bit shift register and the $n-k$ parity check bits are computed. Then the k information bits followed by the $n-k$ parity check bits are shifted out of the two shift registers

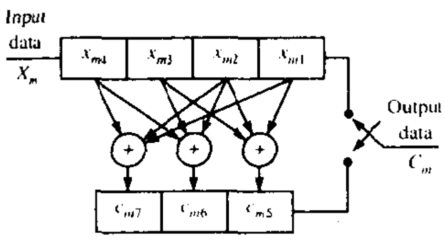


FIGURE 8-1-1 A linear shift register for generating a (7, 4) binary code.

and fed to the modulator. This encoding is illustrated in Fig. 8-1-1 for the (7, 4) code of Example 8-1-1.

Associated with any linear (n, k) code is the dual code of dimension $n - k$. The dual code is a linear $(n, n - k)$ code with 2^{n-k} code vectors, which is the null space of the (n, k) code. The generator matrix for the dual code, denoted by $\mathbf{H}$, consists of $n - k$ linearly independent code vectors selected from the null space. Any code word $\mathbf{C}_m$ of the (n, k) code is orthogonal to any code word in the dual code. Hence, any code word of the (n, k) code is orthogonal to every row of the matrix $\mathbf{H}$, i.e.,

$$\mathbf{C}_m\mathbf{H}' = \mathbf{0} \tag{8-1-9}$$

where $\mathbf{0}$ denotes an all-zero row vector with $n - k$ elements, and $\mathbf{C}_m$ is a code word of the (n, k) code. Since (8-1-9) holds for every code word of the (n, k) code, it follows that

$$\mathbf{G}\mathbf{H}' = \mathbf{0} \tag{8-1-10}$$

where $\mathbf{0}$ is now a $k \times (n - k)$ matrix with all-zero elements.

Now suppose that the linear (n, k) code is systematic and its generator matrix $\mathbf{G}$ is given by the systematic form (8-1-6). Then, since $\mathbf{GH}' = \mathbf{0}$, it follows that

$$\mathbf{H} = [-\mathbf{P}' \mid \mathbf{I}_{n-k}] \tag{8-1-11}$$

The negative sign in (8-1-11) may be dropped when dealing with binary codes, since modulo-2 subtraction is identical to modulo-2 addition.

Example 8-1-2

For the systematic (7, 4) code generated by matrix $\mathbf{G}$ given by (8-1-7), we have, according to (8-1-11), the matrix $\mathbf{H}$ in the form

$$\mathbf{H} = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \tag{8-1-12}$$

Now, the product $\mathbf{C}_m\mathbf{H}'$ yields the three equations

$$\begin{aligned} x_{m1} + x_{m2} + x_{m3} + c_{m5} &= 0 \\ x_{m2} + x_{m3} + x_{m4} + c_{m6} &= 0 \\ x_{m1} + x_{m2} + x_{m4} + c_{m7} &= 0 \end{aligned} \tag{8-1-13}$$

Thus, we observe that the product $\mathbf{C}_m\mathbf{H}'$ is equivalent to adding the parity check bits to the corresponding linear combinations of the information bits used to compute c_{mj}, $j = 5, 6, 7$. That is, (8-1-13) are equivalent to (8-1-8). The matrix $\mathbf{H}$ may be used by the decoder to check that a received code word $\mathbf{Y}$ satisfies the condition (8-1-13), i.e., $\mathbf{YH}' = \mathbf{0}$. In so doing, the decoder checks the received parity check bits with the corresponding linear combination of the bits y_1, y_2, y_3, and y_4 that formed the parity check bits at the transmitter. It is, therefore, appropriate to call $\mathbf{H}$ the *parity check matrix* associated with the (n, k) code.

We make the following observation regarding the relation of the minimum distance of a code to its parity check matrix $\mathbf{H}$. The product $\mathbf{C}_m\mathbf{H}'$ with $\mathbf{C}_m \neq \mathbf{0}$ represents a linear combination of the n columns of $\mathbf{H}'$. Since $\mathbf{C}_m\mathbf{H}' = \mathbf{0}$, the column vectors of $\mathbf{H}$ are linearly dependent. Suppose $\mathbf{C}_j$ denotes the minimum weight code word of a linear (n, k) code. It must satisfy the condition $\mathbf{C}_j\mathbf{H}' = \mathbf{0}$. Since the minimum weight is equal to the minimum distance, it follows that $d_{\min}$ of the columns of $\mathbf{H}$ are linearly dependent. Alternatively, we may say that no more than $d_{\min} - 1$ columns of $\mathbf{H}$ are linearly independent. Since the rank of $\mathbf{H}$ is at most $n - k$, we have $n - k \geq d_{\min} - 1$. Therefore, $d_{\min}$ is upper-bounded as

$$d_{\min} \leq n - k + \tag{8-1-14}$$

Given a linear binary (n, k) code with minimum distance $d_{\min}$, we can construct a linear binary $(n + 1, k)$ code by appending one additional parity check bit to each code word. The check bit is usually selected to be a check bit on all the bits in the code word. Thus the added check bit is a 0 if the original code word has an even number of 1s and it is a 1 if the code word has an odd number of 1s. Consequently, if the minimum weight and, hence, the minimum distance of the code is odd, the added parity check bit increases the minimum distance by 1. We call the $(n + 1, k)$ code an *extended code*. Its parity check matrix is

$$\boldsymbol{H}_e = \left[\begin{array}{ccccc:c} & & & & & 0 \\ & & & & & 0 \\ & & \mathbf{H} & & & \vdots \\ & & & & & 0 \\ \hdashline 1 & 1 & 1 & \ldots & 1 & 1 \end{array}\right] \tag{8-1-15}$$

where $\mathbf{H}$ is the parity check matrix of the original code.

A systematic (n, k) code can also be shortened by setting a number of the information bits to zero. That is, a linear (n, k) code consisting of k information bits and $n-k$ check bits can be shortened into a $(n-l, k-l)$ linear code by setting the first l bits to zero. These l bits are not transmitted. The $n-k$ check bits are computed in the usual manner, as in the original code. Since

$$\mathbf{C}_m = \mathbf{X}_m \mathbf{G}$$

the effect of setting the first l bits of $\mathbf{X}_m$ to 0 is equivalent to reducing the number of rows of $\mathbf{G}$ by removing the first l rows. Equivalently, since

$$\mathbf{C}_m \mathbf{H}' = \mathbf{0}$$

we may remove the first l columns of $\mathbf{H}$. The shortened $(n-l, k-l)$ code consists of 2^{k-l} code words. The minimum distance of these 2^{k-l} code words is at least as large as the minimum distance of the original (n, k) code.

8-1-2 Some Specific Linear Block Codes

In this subsection, we shall briefly describe three types of linear block codes that are frequently encountered in practice and list their important parameters.

Hamming Codes There are both binary and nonbinary Hamming codes. We limit our discussion to the properties of binary Hamming codes. These comprise a class of codes with the property that

$$(n, k) = (2^m - 1, 2^m - 1 - m) \tag{8-1-16}$$

where m is any positive integer. For example, if $m = 3$, we have a (7, 4) code.

The parity check matrix $\mathbf{H}$ of a Hamming code has a special property that allows us to describe the code rather easily. Recall that the parity check matrix of an (n, k) code has $n-k$ rows and n columns. For the binary (n, k) Hamming code, the $n = 2^m - 1$ columns consist of all possible binary vectors with $n - k = m$ elements, except the all-zero vector. For example, the (7, 4) code considered in Examples 8-1-1 and 8-1-2 is a Hamming code. Its parity check matrix consists of the seven column vectors (001), (010), (011), (100), (101), (110), (111).

If we desire to generate a systematic Hamming code, the parity check matrix $\mathbf{H}$ can be easily arranged in the systematic form (8-1-11). Then the corresponding generator matrix $\mathbf{G}$ can be obtained from (8-1-11).

We make the observation that no two columns of $\mathbf{H}$ are linearly dependent, for otherwise the two columns would be identical. However, for $m > 1$, it is possible to find three columns of $\mathbf{H}$ that add to zero. Consequently, $d_{\min} = 3$ for an (n, k) Hamming code.

By adding an overall parity bit, a Hamming (n, k) code can be modified to yield an $(n+1, k)$ code with $d_{\min} = 4$. On the other hand, an (n, k) Hamming code may be shortened to $(n-l, k-l)$ by removing l rows of its generator matrix $\mathbf{G}$ or, equivalently, by removing l columns of its parity check matrix $\mathbf{H}$.

The weight distribution for the class of Hamming (n, k) codes is known and is expressed in compact form by the weight enumerating polynomial

$$\begin{aligned} A(z) &= \sum_{i=0}^{n} A_i z^i \\ &= \frac{1}{n+1}\left[(1+z)^n + n(1+z)^{(n-1)/2}(1-z)^{(n+1)/2}\right] \end{aligned} \tag{8-1-17}$$

where A_i is the number of code words of weight i.

Hadamard Codes A Hadamard code is obtained by selecting as code words the rows of a Hadamard matrix. A Hadamard matrix $\mathbf{M}_n$ is an $n \times n$ matrix (n an even integer) of 1s and 0s with the property that any row differs from any other row in exactly $\frac{1}{2}n$ positions.† One row of the matrix contains all zeros. The other rows contain $\frac{1}{2}n$ zeros and $\frac{1}{2}n$ ones.

For $n = 2$, the Hadamard matrix is

$$\mathbf{M}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{8-1-18}$$

Furthermore, from $\mathbf{M}_n$, we can generate the Hadamard matrix $\mathbf{M}_{2n}$ according to the relation

$$M_{2n} = \begin{bmatrix} \mathbf{M}_n & \mathbf{M}_n \\ \mathbf{M}_n & \bar{\mathbf{M}}_n \end{bmatrix} \tag{8-1-19}$$

where $\bar{\mathbf{M}}_n$ denotes the complement (0s replaced by 1s and vice versa) of $\mathbf{M}_n$. Thus, by substituting (8-1-18) into (8-1-19), we obtain

$$\mathbf{M}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \tag{8-1-20}$$

The complement of $\mathbf{M}_4$ is

$$\bar{\mathbf{M}}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \tag{8-1-21}$$

Now the rows of $\mathbf{M}_4$ and $\bar{\mathbf{M}}_4$ form a linear binary code of block length $n = 4$ having $2n = 8$ code words. The minimum distance of the code is $d_{\min} = \frac{1}{2}n = 2$.

By repeated application of (8-1-19), we can generate Hadamard codes with block length $n = 2^m$, $k = \log_2 2n = \log_2 2^{m+1} = m + 1$, and $d_{\min} = \frac{1}{2}n = 2^{m-1}$, where m is a positive integer. In addition to the important special case where $n = 2^m$, Hadamard codes of other block lengths are possible, but the codes are not linear.

† Sometimes the elements of the Hadamard matrix are denoted by $+1$ and -1. Then the rows of the Hadamard matrix are mutually orthogonal. We also note that the $M = 2^k$ signal waveforms, constructed from Hadamard code words by mapping each bit in a code word into a binary PSK signal, are orthogonal.

TABLE 8-1-1 WEIGHT DISTRIBUTION OF GOLAY (23, 12) AND EXTENDED GOLAY (24, 12) CODES

	Number of code words	
Weight	**(23, 12) code**	**(24, 12) code**
0	1	1
7	253	0
8	506	759
11	1288	0
12	1288	2576
15	506	0
16	253	759
23	1	0
24	0	1

Source: Peterson and Weldon (1972).

Golay Code The Golay code is a binary linear (23, 12) code with $d_{\min} = 7$. The extended Golay code obtained by adding an overall parity to the (23, 12) is a binary linear (24, 12) code with $d_{\min} = 8$. Table 8-1-1 lists the weight distribution of the code words in the Golay (23, 12) and the extended Golay (24, 12) codes. We discuss the generation of the Golay code in Section 8-1-3.

8-1-3 Cyclic Codes

Cyclic codes are a subset of the class of linear codes that satisfy the following cyclic shift property: if $\mathbf{C} = [c_{n-1}c_{n-2} \ldots c_1c_0]$ is a code word of a cyclic code then $[c_{n-2}c_{n-3} \ldots c_0c_{n-1}]$, obtained by a cyclic shift of the elements of $\mathbf{C}$, is also a code word. That is, all cyclic shifts of $\mathbf{C}$ are code words. As a consequence of the cyclic property, the codes possess a considerable amount of structure which can be exploited in the encoding and decoding operations. A number of efficient encoding and hard-decision decoding algorithms have been devised for cyclic codes that make it possible to implement long block codes with a large number of code words in practical communications systems. A description of specific algorithms is beyond the scope of this book. Our primary objective is to briefly describe a number of characteristics of cyclic codes.

In dealing with cyclic codes, it is convenient to associate with a code word $\mathbf{C} = [c_{n-1}c_{n-2} \ldots c_1c_0]$ a polynomial $C(p)$ of degree $\leq n-1$, defined as

$$C(p) = c_{n-1}p^{n-1} + c_{n-2}p^{n-2} + \ldots + c_1p + c_0 \qquad (8\text{-}1\text{-}22)$$

For a binary code, each of the coefficients of the polynomial is either zero or one.

Now suppose we form the polynomial

$$pC(p) = c_{n-1}p^n + c_{n-2}p^{n-1} + \ldots + c_1p^2 + c_0p$$

This polynomial cannot represent a code word, since its degree may be equal to n (when $c_{n-1}=1$). However, if we divide $pC(p)$ by p^n+1, we obtain

$$\frac{pC(p)}{p^n+1}=c_{n-1}+\frac{C_1(p)}{p^n+1} \tag{8-1-23}$$

where

$$C_1(p)=c_{n-2}p^{n-1}+c_{n-3}p^{n-2}+\ldots+c_0p+c_{n-1}$$

Note that the polynomial $C_1(p)$ represents the code word $\mathbf{C}_1=[c_{n-2}\cdots c_0c_{n-1}]$, which is just the code word $\mathbf{C}$ shifted cyclicly by one position. Since $C_1(p)$ is the remainder obtained by dividing $pC(p)$ by p^n+1, we say that

$$C_1(p)=pC(p) \quad \mathrm{mod}\,(p^n+1) \tag{8-1-24}$$

In a similar manner, if $C(p)$ represents a code word in a cyclic code then $p^iC(p)$ mod (p^n+1) is also a code word of the cyclic code. Thus we may write

$$p^iC(p)=Q(p)(p^n+1)+C_i(p) \tag{8-1-25}$$

where the remainder polynomial $C_i(p)$ represents a code word of the cyclic code and $Q(p)$ is the quotient.

We can generate a cyclic code by using a *generator polynomial* $g(p)$ of degree $n-k$. The generator polynomial of an (n,k) cyclic code is a factor of p^n+1 and has the general form

$$g(p)=p^{n-k}+g_{n-k-1}p^{n-k-1}+\ldots+g_1p+1 \tag{8-1-26}$$

We also define a *message polynomial* $X(p)$ as

$$X(p)=x_{k-1}p^{k-1}+x_{k-2}p^{k-2}+\ldots+x_1p+x_0 \tag{8-1-27}$$

where $[x_{k-1}x_{k-2}\cdots x_1x_0]$ represent the k information bits. Clearly, the product $X(p)g(p)$ is a polynomial of degree less than or equal to $n-1$, which may represent a code word. We note that there are 2^k polynomials $\{X_i(p)\}$, and, hence, there are 2^k possible code words that can be formed from a given $g(p)$.

Suppose we denote these code words as

$$C_m(p)=X_m(p)g(p), \qquad m=1,2,\ldots,2^k \tag{8-1-28}$$

To show that the code words in (8-1-28) satisfy the cyclic property, consider any code word $C(p)$ in (8-1-28). A cyclic shift of $C(p)$ produces

$$C_1(p)=pC(p)+c_{n-1}(p^n+1) \tag{8-1-29}$$

and, since $g(p)$ divides both p^n+1 and $C(p)$, it also divides $C_1(p)$, i.e., $C_1(p)$ can be represented as

$$C_1(p)=X_1(p)g(p)$$

Therefore, a cyclic shift of any code word $C(p)$ generated by (8-1-28) yields another code word.

From the above, we see that code words possessing the cyclic property can be generated by multiplying the 2^k message polynomials with a unique polynomial $g(p)$, called the generator polynomial of the (n, k) cyclic code, which divides $p^n + 1$ and has degree $n - k$. The cyclic code generated in this manner is a subspace S_c of the vector space S. The dimension of S_c is k.

Example 8-1-3

Consider a code with block length $n = 7$. The polynomial $p^7 + 1$ has the following factors:

$$p^7 + 1 = (p + 1)(p^3 + p^2 + 1)(p^3 + p + 1) \tag{8-1-30}$$

To generate a (7, 4) cyclic code, we may take as a generator polynomial one of the following two polynomials:

$$\begin{aligned} g_1(p) &= p^3 + p^2 + 1 \\ g_2(p) &= p^3 + p + 1 \end{aligned} \tag{8-1-31}$$

The codes generated by $g_1(p)$ and $g_2(p)$ are equivalent. The code words in the (7, 4) code generated by $g_1(p) = p^3 + p^2 + 1$ are given in Table 8-1-2.

In general, the polynomial $p^n + 1$ may be factored as

$$p^n + 1 = g(p)h(p)$$

where $g(p)$ denotes the generator polynomial for the (n, k) cyclic code and

TABLE 8-1-2 (7, 4) CYCLIC CODE
Generator Polynomial: $g_1(p) = p^3 + p^2 + 1$

Information bits				Code words						
p^3	p^2	p^1	p^0	p^6	p^5	p^4	p^3	p^2	p^1	p^0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1	0	1
0	0	1	0	0	0	1	1	0	1	0
0	0	1	1	0	0	1	0	1	1	1
0	1	0	0	0	1	1	0	1	0	0
0	1	0	1	0	1	1	1	0	0	1
0	1	1	0	0	1	0	1	1	1	0
0	1	1	1	0	1	0	0	0	1	1
1	0	0	0	1	1	0	1	0	0	0
1	0	0	1	1	1	0	0	1	0	1
1	0	1	0	1	1	1	0	0	1	0
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	1	0	1	0	0	0	1
1	1	1	0	1	0	0	0	1	1	0
1	1	1	1	1	0	0	1	0	1	1

$h(p)$ denotes the *parity polynomial* that has degree k. The latter may be used to generate the dual code.

For this purpose, we define the *reciprocal polynomial* of $h(p)$ as

$$\begin{aligned} p^k h(p^{-1}) &= p^k(p^{-k} + h_{k-1}p^{-k+1} + h_{k-2}p^{-k+2} + \dots + h_1 p^{-1} + 1) \\ &= 1 + h_{k-1}p + h_{k-2}p^2 + \dots + h_1 p^{k-1} + p^k \qquad (8\text{-}1\text{-}32) \end{aligned}$$

Clearly, the reciprocal polynomial is also a factor of $p^n + 1$. Hence, $p^k h(p^{-1})$ is the generator polynomial of an $(n, n-k)$ cyclic code. This cyclic code is the dual code to the (n, k) code generated from $g(p)$. Thus, the $(n, n-k)$ dual code constitutes the null space of the (n, k) cyclic code.

Example 8-1-4

Let us consider the dual code to the $(7, 4)$ cyclic code generated in Example 8-1-3. This dual code is a $(7, 3)$ cyclic code associated with the parity polynomial

$$\begin{aligned} h_1(p) &= (p+1)(p^3+p+1) \\ &= p^4+p^3+p^2+1 \qquad (8\text{-}1\text{-}33) \end{aligned}$$

The reciprocal polynomial is

$$p^4 h_1(p^{-1}) = 1 + p + p^2 + p^4$$

This polynomial generates the $(7,3)$ dual code given in Table 8-1-3. The reader can verify that the code words in the $(7,3)$ dual code are orthogonal to the code words in the $(7,4)$ cyclic code of Example 8-1-3. Note that neither the $(7,4)$ nor the $(7,3)$ codes are systematic.

It is desirable to show how a generator matrix can be obtained from the generator polynomial of a cyclic (n, k) code. As previously indicated, the generator matrix for an (n, k) code can be constructed from any set of k

TABLE 8-1-3 $(7,3)$ DUAL CODE
Generator Polynomial $p^4h_1(p^{-1}) = p^4 + p^2 + p + 1$

Information bits			**Code words**						
p^2	p^1	p^0	p^6	p^5	p^4	p^3	p^2	p^1	p^0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1	0
0	1	1	0	1	1	1	0	0	1
1	0	0	1	0	1	1	1	0	0
1	0	1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0	1	0
1	1	1	1	1	0	0	1	0	1

linearly independent code words. Hence, given the generator polynomial $g(p)$, an easily generated set of k linearly independent code words is the code words corresponding to the set of k linearly independent polynomials

$$p^{k-1}g(p),\quad p^{k-2}g(p),\quad \ldots,\quad pg(p),\quad g(p)$$

Since any polynomial of degree less than or equal to $n-1$ and divisible by $g(p)$ can be expressed as a linear combination of this set of polynomials, the set forms a basis of dimension k. Consequently, the code words associated with these polynomials form a basis of dimension k for the (n, k) cyclic code.

Example 8-1-5

The four rows of the generator matrix for the $(7, 4)$ cyclic code with generator polynomial $g_1(p) = p^3 + p^2 + 1$ are obtained from the polynomials

$$p^i g_1(p) = p^{3+i} + p^{2+i} + p^i, \qquad i = 3, 2, 1, 0$$

It is easy to see that the generator matrix is

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{8-1-34}$$

Similarly, the generator matrix for the $(7, 4)$ cyclic code generated by the polynomial $g_2(p) = p^3 + p + 1$ is

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{8-1-35}$$

The parity check matrices corresponding to $\mathbf{G}_1$ and $\mathbf{G}_2$ can be constructed in the same manner by using the respective reciprocal polynomials (Problem 8-8).

Note that the generator matrix obtained by this construction is not in systematic form. We can construct the generator matrix of a cyclic code in the systematic form $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$ from the generator polynomial as follows. First, we observe that the lth row of $\mathbf{G}$ corresponds to a polynomial of the form $p^{n-l} + R_l(p)$, $l = 1, 2, \ldots, k$, where $R_l(p)$ is a polynomial of degree less than $n-k$. This form can be obtained by dividing p^{n-l} by $g(p)$. Thus, we have

$$\frac{p^{n-l}}{g(p)} = Q_l(p) + \frac{R_l(p)}{g(p)}, \qquad l = 1, 2, \ldots, k$$

or, equivalently,

$$p^{n-l} = Q_l(p)g(p) + R_l(p), \qquad l = 1, 2, \ldots, k \tag{8-1-36}$$

where $Q_l(p)$ is the quotient. But $p^{n-l} + R_l(p)$ is a code word of the cyclic code since $p^{n-l} + R_l(p) = Q_l(p)g(p)$. Therefore the desired polynomial corresponding to the lth row of $\mathbf{G}$ is $p^{n-l} + R_l(p)$.

Example 8-1-6

For the (7, 4) cyclic code with generator polynomial $g_2(p) = p^3 + p + 1$, previously discussed in Example 8-1-5, we have

$$\begin{aligned}
p^6 &= (p^3 + p + 1)g_2(p) + p^2 + 1\\
p^5 &= (p^2 + 1)g_2(p) + p^2 + p + 1\\
p^4 &= pg_2(p) + p^2 + p\\
p^3 &= g_2(p) + p + 1
\end{aligned}$$

Hence, the generator matrix of the code in systematic form is

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1\\ 0 & 1 & 0 & 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{8-1-37}$$

and the corresponding parity check matrix is

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0\\ 0 & 1 & 1 & 1 & 0 & 1 & 0\\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{8-1-38}$$

It is left as an exercise for the reader to demonstrate that the generator matrix $\mathbf{G}_2$ given by (8-1-35) and the systematic form given by (8-1-37) generate the same set of code words (Problem 8-2).

The method for constructing the generator matrix $\mathbf{G}$ in systematic form according to (8-1-36) also implies that a systematic code can be generated directly from the generator polynomial $g(p)$. Suppose that we multiply the message polynomial $X(p)$ by p^{n-k}. Thus, we obtain

$$p^{n-k}X(p) = x_{k-1}p^{n-1} + x_{k-2}p^{n-2} + \ldots + x_1p^{n-k+1} + x_0p^{n-k}$$

In a systematic code, this polynomial represents the first k bits in the code word $C(p)$. To this polynomial we must add a polynomial of degree less than $n-k$ representing the parity check bits. Now, if $p^{n-k}X(p)$ is divided by $g(p)$, the result is

$$\frac{p^{n-k}X(p)}{g(p)} = Q(p) + \frac{r(p)}{g(p)}$$

or, equivalently,

$$p^{n-k}X(p) = Q(p)g(p) + r(p) \tag{8-1-39}$$

where $r(p)$ has degree less than $n - k$. Clearly, $Q(p)g(p)$ is a code word of the cyclic code. Hence, by adding (modulo-2) $r(p)$ to both sides of (8-1-39), we obtain the desired systematic code.

To summarize, the systematic code may be generated by

1 multiplying the message polynomial $X(p)$ by p^{n-k};
2 dividing $p^{n-k}X(p)$ by $g(p)$ to obtain the remainder $r(p)$; and
3 adding $r(p)$ to $p^{n-k}X(p)$.

Below we demonstrate how these computations can be performed by using shift registers with feedback.

Since $p^n + 1 = g(p)h(p)$ or, equivalently, $g(p)h(p) = 0 \bmod (p^n + 1)$, we say that the polynomials $g(p)$ and $h(p)$ are *orthogonal*. Furthermore, the polynomials $p^i g(p)$ and $p^j h(p)$ are also orthogonal for all i and j. However, the vectors corresponding to the polynomials $g(p)$ and $h(p)$ are orthogonal only if the ordered elements of one of these vectors are reversed. The same statement applies to the vectors corresponding to $p^i g(p)$ and $p^j h(p)$. In fact, if the parity polynomial $h(p)$ is used as a generator for the $(n, n - k)$ dual code, the set of code words obtained just comprises the same code words generated by the reciprocal polynomial except that the code vectors are reversed. This implies that the generator matrix for the dual code obtained from the reciprocal polynomial $p^k h(p^{-1})$ can also be obtained indirectly from $h(p)$. Since the parity check matrix **H** for the (n, k) cyclic code is the generator matrix for the dual code, it follows that **H** can also be obtained from $h(p)$. The following example illustrates these relationships.

Example 8-1-7

The dual code to the (7, 4) cyclic code generated by $g_1(p) = p^3 + p^2 + 1$ is the (7, 3) dual code that is generated by the reciprocal polynomial $p^4 h_1(p^{-1}) = p^4 + p^2 + p + 1$. However, we may also use $h_1(p)$ to obtain the generator matrix for the dual code. Then, the matrix corresponding to the polynomials $p^i h_1(p)$, $i = 2, 1, 0$, is

$$\mathbf{G}_{h1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

The generator matrix for the (7, 3) dual code, which is the parity check matrix for the (7, 4) cyclic code, consists of the rows of $\mathbf{G}_{h1}$ taken in reverse order. Thus,

$$\mathbf{H}_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

The reader may verify that $\mathbf{G}_1\mathbf{H}_1' = 0$.

Note that the column vectors of $\mathbf{H}_1$ consist of all seven binary vectors of length 3, except the all-zero vector. But this is just the description of the parity check matrix for a (7, 4) Hamming code. Therefore, the (7, 4) cyclic code is equivalent to the (7, 4) Hamming code discussed previously in Examples 8-1-1 and 8-1-2.

Encoders for Cyclic Codes The encoding operations for generating a cyclic code may be performed by a linear feedback shift register based on the use of either the generator polynomial or the parity polynomial. First, let us consider the use of $g(p)$.

As indicated above, the generation of a systematic cyclic code involves three steps, namely multiplying the message polynomial $X(p)$ by p^{n-k}, dividing the product by $g(p)$, and, finally, adding the remainder to $p^{n-k}X(p)$. Of these three steps, only the division is nontrivial.

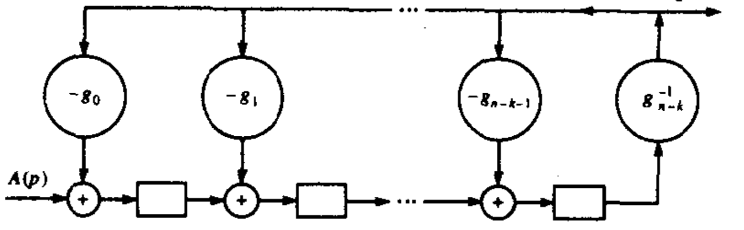
The division of the polynomial $A(p) = p^{n-k}X(p)$ of degree $n-1$ by the polynomial

$$g(p) = g_{n-k}p^{n-k} + g_{n-k-1}p^{n-k-1} + \dots + g_1 p + g_0$$

may be accomplished by the $(n-k)$ stage feedback shift register illustrated in Fig. 8-1-2. Initially, the shift register contains all zeros. The coefficients of $A(p)$ are clocked into the shift register one (bit) coefficient at a time, beginning with the higher-order coefficients, i.e., with a_{n-1}, followed by a_{n-2}, and so on. After the kth shift, the first nonzero output of the quotient is $q_1 = g_{n-k}a_n$. Subsequent outputs are generated as illustrated in Fig. 8-1-2. For each output coefficient in the quotient, we must subtract the polynomial $g(p)$ multiplied by that coefficient, as in ordinary long division. This subtraction is performed by means of the feedback part of the shift register. Thus, the feedback shift register in Fig. 8-1-2 performs division of two polynomials.

In our case, $g_{n-k} = g_0 = 1$, and, for binary codes, the arithmetic operations are performed in modulo-2 arithmetic. Consequently, the subtraction operations reduce to modulo-2 addition. Furthermore, we are only interested in

FIGURE 8-1-2 A feedback shift register for dividing the polynomial $A(p)$ by $g(p)$.



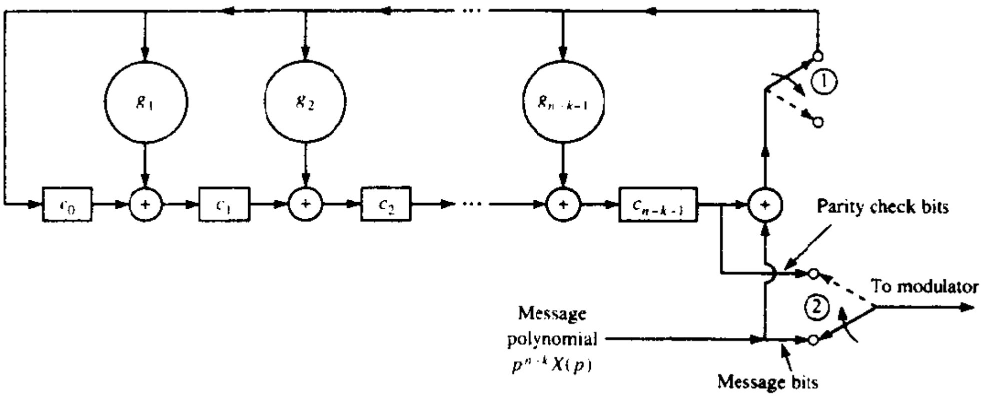


FIGURE 8-1-3 Encoding of a cyclic code by use of the generator polynomial $g(p)$.

generating the parity check bits for each code word, since the code is systematic. Consequently, the encoder for the cyclic code takes the form illustrated in Fig. 8-1-3. The first k bits at the output of the encoder are simply the k information bits. These k bits are also clocked simultaneously into the shift register, since the switch 1 is in the closed position. Note that the polynomial multiplication of p^{n-k} with $X(p)$ is not performed explicitly. After the k information bits are all clocked into the encoder, the positions of the two switches are reversed. At this time, the contents of the shift register are simply the $n-k$ parity check bits, which correspond to the coefficients of the remainder polynomial. These $n-k$ bits are clocked out one at a time and sent to the modulator.

Example 8-1-8

The shift register for encoding the (7, 4) cyclic code with generator polynomial $g(p) = p^3 + p + 1$ is illustrated in Fig. 8-1-4. Suppose the input

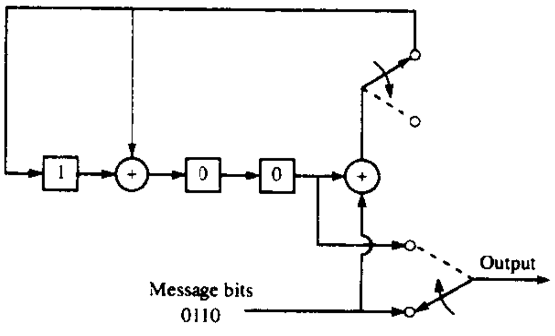


FIGURE 8-1-4 The encoder for the (7, 4) cyclic code with generator polynomial $g(p) = p^3 + p + 1$.

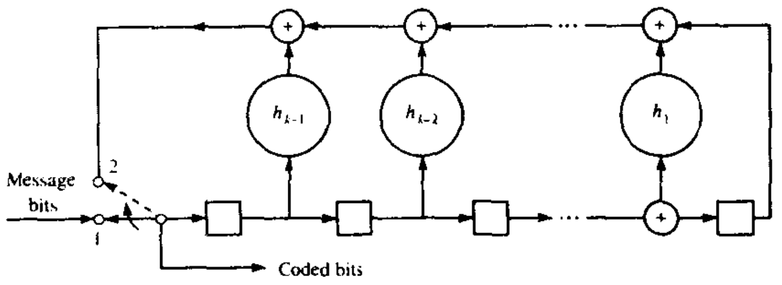


FIGURE 8-1-5 Encoder for an (n, k) cyclic code based on parity polynomial $h(p)$.

message bits are 0110. The contents of the shift register are as follows:

Input	Shift	Shift register contents
	0	0 0 0
0	1	0 0 0
1	2	1 1 0
1	3	1 0 1
0	4	1 0 0

Hence, the three parity check bits are 100, which correspond to the code bits $c_5 = 0$, $c_6 = 0$, and $c_7 = 1$.

Instead of using the generator polynomial, we may implement the encoder for the cyclic code by making use of the parity polynomial

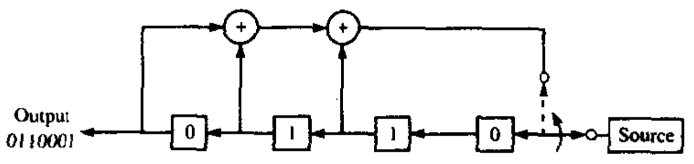
$$h(p) = p^k + h_{k-1}p^{k-1} + \ldots + h_1 p + 1$$

The encoder is shown in Fig. 8-1-5. Initially, the k information bits are shifted into the shift register and simultaneously fed to the modulator. After all k information bits are in the shift register, the switch is thrown into position 2 and the shift register is clocked $n - k$ times to generate the $n - k$ parity check bits as illustrated in Fig. 8-1-5.

Example 8-1-9

The parity polynomial for the (7, 4) cyclic code generated by $g(p) = p^3 + p + 1$ is $h(p) = p^4 + p^2 + p + 1$. The encoder for this code based on the parity polynomial is illustrated in Fig. 8-1-6. If the input to the encoder is

FIGURE 8-1-6 The encoder for the (7, 4) cyclic code based on the parity polynomial $h(p) = p^4 + p^2 + 1$.



the message bits 0110, the parity check bits are $c_5 = 0$, $c_6 = 0$, and $c_7 = 1$, as is easily verified.

It should be noted that the encoder based on the generator polynomial is simpler when $n - k < k$ $(k > \frac{1}{2}n)$, i.e., for high rate codes $(R_c > \frac{1}{2})$, while the encoder based on the parity polynomial is simpler when $k < n - k$ $(k < \frac{1}{2}n)$, which corresponds to low rate codes $(R_c < \frac{1}{2})$.

Cyclic Hamming Codes The class of cyclic codes include the Hamming codes, which have a block length $n = 2^m - 1$ and $n - k = m$ parity check bits, where m is any positive integer. The cyclic Hamming codes are equivalent to the Hamming codes described in Section 8-1-2.

Cyclic (23, 12) Golay Code The linear (23, 12) Golay code described in Section 8-1-2 can be generated as a cyclic code by means of the generator polynomial

$$g(p) = p^{11} + p^9 + p^7 + p^6 + p^5 + p + 1 \tag{8-1-40}$$

The code words have a minimum distance $d_{\min} = 7$.

Maximum-Length Shift-Register Codes Maximum-length shift-register codes are a class of cyclic codes with

$$(n, k) = (2^m - 1, m) \tag{8-1-41}$$

where m is a positive integer. The code words are usually generated by means of an m-stage digital shift register with feedback, based on the parity polynomial. For each code word to be transmitted, the m information bits are loaded into the shift register, and the switch is thrown from position 1 to position 2. The contents of the shift register are shifted to the left one bit at a time for a total of $2^m - 1$ shifts. This operation generates a systematic code with the desired output length $n = 2^m - 1$. For example, the code words generated by the $m = 3$ stage shift register in Fig. 8-1-7 are listed in Table 8-1-4.

Note that, with the exception of the all-zero code word, all the code words generated by the shift register are different cyclic shifts of a single code word. The reason for this structure is easily seen from the state diagram of the shift register, which is illustrated in Fig. 8-1-8 for $m = 3$. When the shift register is loaded initially and shifted $2^m - 1$ times, it will cycle through all possible $2^m - 1$ states. Hence, the shift resgister is back to its original state in $2^m - 1$ shifts.

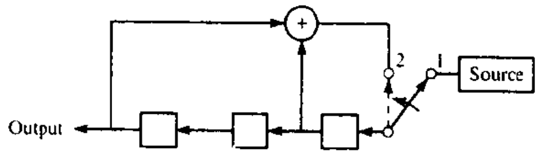


FIGURE 8-1-7 Three-stage $(m = 3)$ shift register with feedback.

TABLE 8-1-4 MAXIMUM-LENGTH SHIFT-REGISTER CODE FOR $m = 3$

Information bits			Code words						
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1	0	1
0	1	0	0	1	0	0	1	1	1
0	1	1	0	1	1	1	0	1	0
1	0	0	1	0	0	1	1	1	0
1	0	1	1	0	1	0	0	1	1
1	1	0	1	1	0	1	0	0	1
1	1	1	1	1	1	0	1	0	0

Consequently, the output sequence is periodic with length $n = 2^m - 1$. Since there are $2^m - 1$ possible states, this length corresponds to the largest possible period. This explains why the $2^m - 1$ code words are different cyclic shifts of a single code word.

Maximum-length shift-register codes exist for any positive value of m.

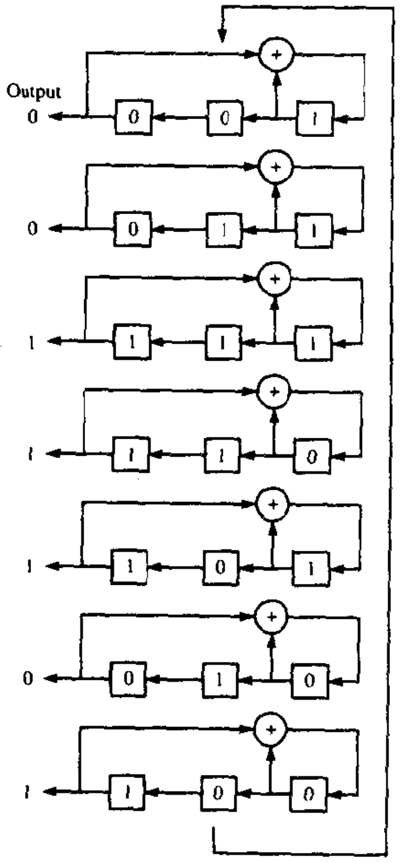


FIGURE 8-1-8 The seven states for the $m = 3$ maximum length shift register.

TABLE 8-1-5 SHIFT-REGISTER CONNECTIONS FOR GENERATING MAXIMUM-LENGTH SEQUENCES

m	Stages connected to modulo-2 adder	m	Stages connected to modulo-2 adder	m	Stages connected to modulo-2 adder
2	1, 2	13	1, 10, 11, 13	24	1, 18, 23, 24
3	1, 3	14	1, 5, 9, 14	25	1, 23
4	1, 4	15	1, 15	26	1, 21, 25, 26
5	1, 4	16	1, 5, 14, 16	27	1, 23, 26, 27
6	1, 6	17	1, 15	28	1, 26
7	1, 7	18	1, 12	29	1, 28
8	1, 5, 6, 7	19	1, 15, 18, 19	30	1, 8, 29, 30
9	1, 6	20	1, 18	31	1, 29
10	1, 8	21	1, 20	32	1, 11, 31, 32
11	1, 10	22	1, 22	33	1, 21
12	1, 7, 9, 12	23	1, 19	34	1, 8, 33, 34

Source: Forney (1970).

Table 8-1-5 lists the stages connected to the modulo-2 adder that result in a maximum-length shift register for $2 \leq m \leq 34$.

Another characteristic of the code words in a maximum-length shift-register code is that each code word, with the exception of the all-zero code word, contains 2^{m-1} ones and 2^{m-1} zeros. Hence all these code words have identical weights, namely, $w = 2^{m-1}$. Since the code is linear, this weight is also the minimum distance of the code, i.e.,

$$d_{\min} = 2^{m-1}$$

Finally, note that the (7, 3) maximum-length shift-register code shown in Table 8-1-4 is identical to the (7, 3) code given in Table 8-1-3, which is the dual of the (7, 4) Hamming code given in Table 8-1-2. This is not a coincidence. The maximum-length shift-register codes are the dual codes of the cyclic Hamming $(2^m - 1, 2^m - 1 - m)$ codes.

The shift register for generating the maximum-length code may also be used to generate a periodic binary sequence with period $n = 2^m - 1$. The binary periodic sequence exhibits a periodic autocorrelation $\phi(m)$ with values $\phi(m) = n$ for $m = 0, \pm n, \pm 2n, \ldots$, and $\phi(m) = -1$ for all other shifts as described in Section 13-2-4. This impulse-like autocorrelation implies that the power spectrum is nearly white and, hence, the sequence resembles white noise. As a consequence, maximum-length sequences are called pseudo-noise (PN) sequences and find use in the scrambling of data and in the generation of spread spectrum signals.

Bose–Chaudhuri–Hocquenghem (BCH) Codes BCH codes comprise a large class of cyclic codes that include both binary and nonbinary alphabets.

Binary BCH codes may be constructed with parameters

$$\begin{aligned} n &= 2^m - 1 \\ n - k &\leq mt \\ d_{\min} &= 2t + 1 \end{aligned} \tag{8-1-42}$$

where m ($m \geq 3$) and t are arbitrary positive integers. Hence, this class of binary codes provides the communications system designer with a large selection of block lengths and code rates. Nonbinary BCH codes include the powerful Reed–Solomon codes that are described later.

The generator polynomials for BCH codes can be constructed from factors of $p^{2^m-1} + 1$. Table 8-1-6 lists the coefficients of generator polynomials for BCH codes of block lengths $7 \leq n \leq 255$, corresponding to $3 \leq m \leq 8$. The coefficients are given in octal form, with the left-most digit corresponding to the highest-degree term of the generator polynomial. Thus, the coefficients of the generator polynomial for the (15, 5) code are 2467, which in binary form is 10 100 110 111. Consequently, the generator polynomial is $g(p) = p^{10} + p^8 + p^5 + p^4 + p^2 + p + 1$.

A more extensive list of generator polynomials for BCH codes is given by Peterson and Weldon (1972), who tabulate the polynomial factors of $p^{2^m-1} + 1$ for $m \leq 34$.

8-1-4 Optimum Soft-Decision Decoding of Linear Block Codes

In this subsection, we derive the performance of linear binary block codes on an AWGN channel when optimum (unquantized) soft-decision decoding is employed at the receiver. The bits of a code word may be transmitted by any one of the binary signaling methods described in Chapter 5. For our purposes, we consider binary (or quaternary) coherent PSK, which is the most efficient method, and binary orthogonal FSK either with coherent detection or noncoherent detection.

Let $\mathscr{E}$ denote the transmitted signal energy per code word and let $\mathscr{E}_c$ denote the signal energy required to transmit a single element (bit) in the code word. Since there are n bits per code word, $\mathscr{E} = n\mathscr{E}_c$, and since each code word conveys k bits of information, the energy per information bit is

$$\mathscr{E}_b = \frac{\mathscr{E}}{k} = \frac{n}{k}\mathscr{E}_c = \frac{\mathscr{E}_c}{R_c} \tag{8-1-43}$$

The code words are assumed to be equally likely a priori with prior probability $1/M$.

Suppose the bits of a code word are transmitted by binary PSK. Thus each code word results in one of M signaling waveforms. From Chapter 5, we know that the optimum receiver, in the sense of minimizing the average probability

TABLE 8-1-6 COEFFICIENTS OF GENERATOR POLYNOMIALS (IN OCTAL FORM) FOR BCH CODES OF LENGTHS $7 \leq n \leq 255$

n	k	t	$g(p)$
7	4	1	13
15	11	1	23
	7	2	721
	5	3	2467
31	26	1	45
	21	2	3551
	16	3	107657
	11	5	5423325
	6	7	313365047
63	57	1	103
	51	2	12471
	45	3	1701317
	39	4	166623567
	36	5	1033500423
	30	6	157464165547
	24	7	17323260404441
	18	10	1363026512351725
	16	11	6331141367235453
	10	13	472622305527250155
	7	15	5231045543503271737
127	120	1	211
	113	2	41567
	106	3	11554743
	99	4	3447023271
	92	5	624730022327
	85	6	130704476322273
	78	7	26230002166130115
	71	9	6255010713253127753
	64	10	1206534025570773100045
	57	11	335265252505705053517721
	50	13	54446512523314012421501421
	43	14	17721772213651227521220574343
	36	15	3146074666522075044764574721735
	29	21	403114461367670603667530141176155
	22	23	123376070404722522435445626637647043
	15	27	22057042445604554770523013762217604353
	8	31	7047264052751030651476224271567733130217
255	247	1	435
	239	2	267543
	231	3	156720665
	223	4	75626641375
	215	5	23157564726421
	207	6	16176560567636227
	199	7	7633031270420722341
	191	8	2663470176115333714567
	187	9	52755313540001322236351
	179	10	22624710717340432416300455
	171	11	15416214212342356077061630 67

TABLE 8-1-6 *(Continued)*

n	*k*	*t*	*g(p)*
	163	12	75004155100756025515747245146011
	155	13	37575130054076650157225064646677633
	147	14	16421301735371655253041653054410117111
	139	15	461401732060175561570722730247453567445
	131	18	215713331471510151261250277442142024165471
	123	19	120614052242066003717210326516141226272506267
	115	21	60526665572100247263636404600276352556313472737
	107	22	22205772322066256312417300235347420176574750154441
	99	23	10656667253473174222741416201574332252411076432303431
	91	25	6750265030327444172723631724732511075550762720724344561
	87	26	110136763414743236435231634307172046206722545273311721317
	79	27	66700035637657500020270344207366174621015326711766541342355
	71	29	2402471052064432151555417211233116320544425036255764322170603 5
	63	30	10754475055163544325315217357707003666111726455267613656702543301
	55	31	73154252035011001330152753060320543254143267550105570444260354736 17
	47	42	253354201706264656303304137740623317512333414544604500506602455254317 3
	45	43	152020560552341611311013463764237015636700244707623730332021570250515 41
	37	45	513633025506700741417744724543753042073570617432343234764435473740304 4003
	29	47	302571553667307146552706401236137711534224232420117411406025475741040 3565037
	21	55	125621525706033265600177315360761210322734140565307454252115312161446 6513473725
	13	59	464173200505256454442657371425006600433067744547656140317467721357026 134460500547
	9	63	157260252174724632010310432553551346141623672120440745451127661155477 05561677516057

Source: Stenbit (1964), © 1964 IEEE.

of a code word error, for the AWGN channel, can be realized as a parallel bank of M filters matched to the M possible transmitted waveforms. The outputs of the M matched filters at the end of each signaling interval, which encompasses the transmission of n bits in the code word, are compared and the code word corresponding to the largest matched filter output is selected. Alternatively, M cross-correlators can be employed. In either case, the receiver implementation can be simplified. That is, an equivalent optimum receiver can be realized by use of a single filter (or cross-correlator) matched to the binary PSK waveform used to transmit each bit in the code word, followed by a decoder that forms the M decision variables corresponding to the M code words.

To be specific, let r_j, $j = 1, 2, \ldots, n$, represent the n sampled outputs of the matched filter for any particular code word. Since the signaling is binary coherent PSK, the output r_j may be expressed either as

$$r_j = \sqrt{\mathscr{E}_c} + n_j \tag{8-1-44}$$

when the jth bit of a code word is a 1, or as

$$r_j = -\sqrt{\mathscr{E}_c} + n_j \tag{8-1-45}$$

when the jth bit is a 0. The variables $\{n_j\}$ represent additive white gaussian noise at the sampling instants. Each n_j has zero mean and variance $\frac{1}{2}N_0$. From knowledge of the M possible transmitted code words and upon reception of $\{r_j\}$, the optimum decoder forms the M correlation metrics

$$CM_i = C(\mathbf{r}, \mathbf{C}_i) = \sum_{j=1}^{n} (2c_{ij} - 1)r_j, \qquad i = 1, 2, \ldots, M \tag{8-1-46}$$

where c_{ij} denotes the bit in the jth position of the ith code word. Thus, if $c_{ij} = 1$, the weighting factor $2c_{ij} - 1 = 1$, and if $c_{ij} = 0$, the weighting factor $2c_{ij} - 1 = -1$. In this manner, the weighting $2c_{ij} - 1$ aligns the signal components in $\{r_j\}$ such that the correlation metric corresponding to the actual transmitted code word will have a mean value $\sqrt{\mathscr{E}_c}n$, while the other $M - 1$ metrics will have smaller mean values.

Although the computations involved in forming the correlation metrics for soft-decision decoding according to (8-1-46) are relatively simple, it may still be impractical to compute (8-1-46) for all the possible code words when the number of code words is large, e.g., $M > 2^{10}$. In such a case it is still possible to implement soft-decision decoding using algorithms which employ techniques for discarding improbable code words without computing their entire correlation metrics as given by (8-1-46). Several different types of soft-decision decoding algorithms have been described in the technical literature. The interested reader is referred to the papers by Forney (1966b), Weldon (1971), Chase (1972), Wainberg and Wolf (1973), Wolf (1978), and Matis and Modestino (1982).

In determining the probability of error for a linear block code, note that when such a code is employed on a binary-input, symmetric channel such as the AWGN channel with optimum soft-decision decoding, the error probability for the transmission of the mth code word is the same for all m. Hence, we assume for simplicity that the all-zero code word $\mathbf{C}_1$ is transmitted. For correct decoding of $\mathbf{C}_1$, the correlation metric CM_1 must exceed all the other $M-1$ correlation metrics CM_m, $m=2, \ldots, M$. All the CM are gaussian distributed. The mean value of CM_1 is $\sqrt{\mathscr{E}_c}n$, while the mean values of CM_m, $m=2, \ldots, M$ is $\sqrt{\mathscr{E}_c}n(1-2w_m/n)$. The variance of each decision variable is $\frac{1}{2}N_0$. The derivation of the exact expression for the probability of correct decoding or, equivalently, the probability of a code word error is complicated by the correlations among the M correlation metrics. The cross-correlation coefficients between $\mathbf{C}_1$ and the other $M-1$ code words are

$$\rho_m = 1 - 2w_m/n, \qquad m=2, \ldots, M \tag{8-1-47}$$

where w_m denotes the weight of the mth code word.

Instead of attempting to derive the exact error probability, we resort to a union bound. The probability that $CM_m > CM_1$ is

$$P_2(m) = Q\left(\sqrt{\frac{\mathscr{E}}{N_0}(1-\rho_m)}\right) \tag{8-1-48}$$

where $\mathscr{E} = k\mathscr{E}_b$ is the transmitted energy per waveform. Substitution for ρ_m from (8-1-47) and for $\mathscr{E}$ yields

$$\begin{aligned} P_2(m) &= Q\left(\sqrt{\frac{2\mathscr{E}_b}{N_0}R_c w_m}\right) \\ &= Q(\sqrt{2\gamma_b R_c w_m}) \end{aligned} \tag{8-1-49}$$

where γ_b is the SNR per bit and R_c is the code rate. Then the average probability of a code word error is bounded from above by the sum of the binary error events given by (8-1-49). Thus,

$$\begin{aligned} P_M &\leq \sum_{m=2}^{M} P_2(m) \\ &\leq \sum_{m=2}^{M} Q(\sqrt{2\gamma_b R_c w_m}) \end{aligned} \tag{8-1-50}$$

The computation of the probability of error for soft-decision decoding according to (8-1-50) requires knowledge of the weight distribution of the code. Weight distributions of many codes are given in a number of texts on coding theory, e.g., Berlekamp (1968) and MacWilliams and Sloane (1977).

A somewhat looser bound is obtained by noting that

$$Q(\sqrt{2\gamma_b R_c w_m}) \leq Q(\sqrt{2\gamma_b R_c d_{\min}}) < \exp(-\gamma_b R_c d_{\min}) \tag{8-1-51}$$

Consequently,

$$P_M \leq (M-1)Q(\sqrt{2\gamma_b R_c d_{\min}}) < \exp(-\gamma_b R_c d_{\min} + k \ln 2) \tag{8-1-52}$$

This bound is particularly useful since it does not require knowledge of the weight distribution of the code. When the upper bound in (8-1-52) is compared with the performance of an uncoded binary PSK system, which is upper-bounded as $\frac{1}{2}\exp(-\gamma_b)$, we find that coding yields a gain of approximately $10\log(R_c d_{\min} - k\ln 2/\gamma_b)$ dB. We may call this the *coding gain*. We note that its value depends on the code parameters and also on the SNR per bit γ_b.

The expression for the probability of error for equicorrelated waveforms that can be obtained for the simplex signals described in Section 5-2 gives us yet a third approximation to the error probabilities for coded waveforms. We know that the maximum cross-correlation coefficient between a pair of coded waveforms is

$$\rho_{\max} = 1 - \frac{2}{n} d_{\min} \tag{8-1-53}$$

If we assume as a worst case that all the M code words have a cross-correlation coefficient equal to $\rho_{\max}$ then the code word error probability can easily be manipulated. Since some code words are separated by more than the minimum distance, the error probability evaluated for $\rho_r = \rho_{\max}$ is actually an upper bound. Thus,

$$P_M \leq 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-v^2/2} \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{v+\sqrt{4\gamma_b R_c d_{\min}}} e^{-x^2/2}\, dx \right)^{M-1} dv \tag{8-1-54}$$

The bounds on the performance of linear block codes given above are in terms of the block error or code word error probability. The evaluation of the equivalent bit error probability P_b is much more complicated. In general, when a block error is made, some of the k information bits in the block will be correct and some will be in error. For orthogonal waveforms, the conversion factor that multiplies P_M to yield P_b is $2^{k-1}/(2^k - 1)$. This factor is unity for $k = 1$ and approaches $\frac{1}{2}$ as k increases, which is equivalent to assuming that, on the average, half of the k bits will be in error when a block error occurs. The conversion factor for coded waveforms depends in a complicated way on the distance properties of the code, but is certainly no worse than assuming that, on the average, half of the k bits will be in error when a block error occurs. Consequently, $P_b \leq \frac{1}{2}P_M$.

The bounds on performance given by (8-1-50), (8-1-52), and (8-1-54) also apply to the case in which a pair of bits of a code word are transmitted by quaternary PSK, since quaternary PSK may be viewed as being equivalent to two independent binary PSK waveforms transmitted in phase quadrature. Furthermore, the bounds in (8-1-52) and (8-1-54), which depend only on the minimum distance of the code, apply also to nonlinear binary block codes.

If binary orthogonal FSK is used to transmit each bit of a code word on the AWGN channel, the optimum receiver can be realized by means of two matched filters, one matched to the frequency corresponding to a transmission of a 0, and the other to the frequency corresponding to a transmission of a 1, followed by a decoder that forms the M correlation metrics corresponding to the M possible code words. The detection at the receiver may be coherent or noncoherent. In either case, let r_{0j} and r_{1j} denote the input samples to the combiner. The correlation metrics formed by the decoder may be expressed as

$$CM_i = \sum_{j=1}^{n} [c_{ij}r_{1j} + (1 - c_{ij})r_{0j}], \qquad i = 1, 2, \ldots, M \tag{8-1-55}$$

where c_{ij} represents the jth bit in the ith code word. The code word corresponding to the largest of the $\{CM_i\}$ is selected as the transmitted code word.

If the detection of the binary FSK waveforms is coherent, the random variables $\{r_{0j}\}$ and $\{r_{1j}\}$ are gaussian and, hence, the correlation metrics $\{CM_i\}$ are also gaussian. In this case, bounds on the performance of the code are easily obtained. To be specific, suppose that the all-zero code word $\mathbf{C}_1$ is transmitted. Then,

$$\left.\begin{aligned} r_{0j} &= \sqrt{\mathscr{E}_c} + n_{0j} \\ r_{1j} &= n_{1j} \end{aligned}\right\} \quad j = 1, 2, \ldots, n \tag{8-1-56}$$

where the $\{n_{ij}\}$, $i = 0, 1$, $j = 1, 2, \ldots, n$, are mutually statistically independent gaussian random variables with zero mean and variance $\frac{1}{2}N_0$. Consequently CM_1 is gaussian with mean $\sqrt{\mathscr{E}_c}n$ and variance $\frac{1}{2}N_0$. On the other hand, the correlation metric CM_m, corresponding to the code word having weight w_m, is gaussian with mean $\sqrt{\mathscr{E}_c}n(1 - w_m/n)$ and variance $\frac{1}{2}nN_0$. Since the $\{CM_m\}$ are correlated, we again resort to a union bound. The correlation coefficients are given by

$$\rho_m = 1 - w_m/n \tag{8-1-57}$$

Hence, the probability that $CM_m > CM_1$ is

$$P_2(m) = Q(\sqrt{\gamma_b R_c w_m}) \tag{8-1-58}$$

Comparison of this result with that given in (8-1-49) for coherent PSK reveals that coherent PSK requires 3 dB less SNR to achieve the same performance. This is not surprising in view of the fact that uncoded PSK is 3 dB better than binary orthogonal FSK with coherent detection. Hence, the advantage of PSK over FSK is maintained in the coded waveforms. We conclude, then, that the bounds given in (8-1-50), (8-1-52), and (8-1-54) apply to coded waveforms transmitted by binary orthogonal coherent FSK with γ_b replaced by $\frac{1}{2}\gamma_b$.

If square-law detection of the binary orthogonal FSK signal is employed at the receiver, the performance is further degraded by the noncoherent combining loss, as shown in Chapter 12. Suppose again that the all-zero code word is transmitted. Then the correlation metrics are given by (8-1-55), where the input variables to the decoder are now

$$\left.\begin{aligned} r_{0j} &= |\sqrt{\mathscr{E}_c} + N_{0j}|^2 \\ r_{1j} &= |N_{1j}|^2 \end{aligned}\right\} \quad j = 1, 2, \ldots, n \tag{8-1-59}$$

where $\{N_{0j}\}$ and $\{N_{1j}\}$ represent complex-valued mutually statistically independent gaussian random variables with zero mean and variance N_0. The correlation metric CM_1 is given as

$$CM_1 = \sum_{j=1}^{n} r_{0j} \tag{8-1-60}$$

while the correlation metric corresponding to the code word having weight w_m is statistically equivalent to the correlation metric of a code word in which $c_{mj} = 1$ for $1 \leq j \leq w_m$ and $c_{mj} = 0$ for $w_m + 1 \leq j \leq n$. Hence, CM_m may be expressed as

$$CM_m = \sum_{j=1}^{w_m} r_{1j} + \sum_{j=w_m+1}^{n} r_{0j} \tag{8-1-61}$$

The difference between CM_1 and CM_m is

$$CM_1 - CM_m = \sum_{j=1}^{w_m} (r_{0j} - r_{1j}) \tag{8-1-62}$$

and the probability of error is simply the probability that $CM_1 - CM_m < 0$. But this difference is a special case of the general quadratic form in complex-valued gaussian random variables considered in Chapter 12 and Appendix B. The expression for the probability of error in deciding between CM_1 and CM_m is (see Section 12-1-1)

$$P_2(m) = \frac{1}{2^{2w_m - 1}} \exp\left(-\tfrac{1}{2}\gamma_b R_c w_m\right) \sum_{i=0}^{w_m - 1} K_i \left(\tfrac{1}{2}\gamma_b R_c w_m\right)^i \tag{8-1-63}$$

where, by definition,

$$K_i = \frac{1}{i!} \sum_{r=0}^{w_m - 1 - i} \binom{2w_m - 1}{r} \tag{8-1-64}$$

The union bound obtained by summing $P_2(m)$ over $2 \leq m \leq M$ provides us with an upper bound on the probability of a code word error.

As an alternative, we may use the *minimum distance* instead of the weight distribution to obtain the looser upper bound

$$P_M \leq \frac{M - 1}{2^{2d_{\min} - 1}} \exp\left(-\tfrac{1}{2}\gamma_b R_c d_{\min}\right) \sum_{i=0}^{d_{\min} - 1} K_i \left(\tfrac{1}{2}\gamma_b R_c d_{\min}\right)^i \tag{8-1-65}$$

A measure of the noncoherent combining loss inherent in the square-law detection and combining of the n elementary binary FSK waveforms in a code word can be obtained from Fig. 12-1-1, where $d_{\min}$ is used in place of L. The loss obtained is relative to the case in which the n elementary binary FSK waveforms are first detected coherently and combined as in (8-1-55) and then the sums are square-law-detected or envelope-detected to yield the M decision variables. The binary error probability for the latter case is

$$P_2(m) = \tfrac{1}{2} \exp\left(-\tfrac{1}{2}\gamma_b R_c w_m\right) \tag{8-1-66}$$

and, hence,

$$P_M \leq \sum_{m=2}^{M} P_2(m)$$

If $d_{\min}$ is used instead of the weight distribution, the union bound for the code word error probability in the latter case is

$$P_M \leq \tfrac{1}{2}(M-1)\exp\left(-\tfrac{1}{2}\gamma_b R_c d_{\min}\right) \tag{8-1-67}$$

The channel bandwidth required to transmit the coded waveforms can be determined as follows. If binary PSK is used to transmit each bit in a code word, the required bandwidth is approximately equal to the reciprocal of the time interval devoted to the transmission of each bit. For an information rate of R bits/s, the time available to transmit k information bits and $n-k$ redundant (parity) bits (n total bits) is $T=k/R$. Hence,

$$W=\frac{1}{T/n}=\frac{n}{k/R}=\frac{R}{R_c} \tag{8-1-68}$$

Therefore, the bandwidth expansion factor B_e for the coded waveform is

$$\begin{aligned} B_e &= \frac{W}{R} \\ &= \frac{n}{k}=\frac{1}{R_c} \end{aligned} \tag{8-1-69}$$

On the other hand, if binary FSK with noncoherent detection is employed for transmitting the bits in a code word, $W \approx 2n/T$, and, hence, the bandwidth expansion factor increases by approximately a factor of 2 relative to binary PSK. In any case, B_e increases inversely with the code rate, or, equivalently, it increases linearly with the block size n.

We are now in a position to compare the performance characteristics and bandwidth requirements of coded signaling waveforms with orthogonal signaling waveforms. A comparison of the expression for P_M given in (5-2-21) for orthogonal waveforms and in (8-1-54) for coded waveforms with coherent PSK indicates that the coded waveforms result in a loss of at most $10\log(n/2d_{\min})$ dB relative to orthogonal waveforms having the same number of waveforms. On the other hand, if we compensate for the loss in SNR due to coding by increasing the number of code words so that coded transmission requires $M_c=2^{k_c}$ waveforms and orthogonal signaling requires $M_o=2^{k_o}$ waveforms then [from the union bounds in (5-2-27) and (8-1-52)], the performance obtained with the two sets of signaling waveforms at high SNR is about equal if

$$k_o = 2R_c d_{\min} \tag{8-1-70}$$

Under this condition, the bandwidth expansion factor for orthogonal signaling can be expressed as

$$B_{eo} = \frac{M_o}{2\log_2 M_o} = \frac{2^{k_o}}{2k_o} = \frac{2^{2R_c d_{\min}}}{4R_c d_{\min}} \tag{8-1-71}$$

while, for coded signaling waveforms, we have $B_{ec} = 1/R_c$. The ratio of B_{eo} given in (8-1-71) to B_{ec}, which is

$$\frac{B_{eo}}{B_{ec}} = \frac{2^{2R_c d_{\min}}}{4d_{\min}} \tag{8-1-72}$$

provides a measure of the relative bandwidth between orthogonal signaling and signaling with coded coherent PSK waveforms.

For example, suppose we use a (63, 33) binary cyclic code that has a minimum distance $d_{\min} = 12$. The bandwidth ratio for orthogonal signaling relative to this code, given by (8-1-72), is 127. This is indicative of the bandwidth efficiency obtained through coding relative to orthogonal signaling.

8-1-5 Hard-Decision Decoding

The bounds given in Section 8-1-4 on the performance of coded signaling waveforms on the AWGN channel are based on the premise that the samples from the matched filter or cross correlator are not quantized. Although this processing yields the best performance, the basic limitation is the computational burden of forming M correlation metrics and comparing these to obtain the largest. The amount of computation becomes excessive when the number M of code words is large.

To reduce the computational burden, the analog samples can be quantized and the decoding operations are then performed digitally. In this subsection, we consider the extreme situation in which each sample corresponding to a single bit of a code word is quantized to two levels: zero and one. That is, a (hard) decision is made as to whether each transmitted bit in a code word is a 0 or a 1. The resulting discrete-time channel (consisting of the modulator, the AWGN channel, and the demodulator) constitutes a BSC with crossover probability p. If coherent PSK is employed in transmitting and receiving the bits in each code word then

$$\begin{aligned} p &= Q\left(\sqrt{\frac{2\mathscr{E}_c}{N_0}}\right) \\ &= Q(\sqrt{2\gamma_b R_c}) \end{aligned} \tag{8-1-73}$$

On the other hand, if FSK is used to transmit the bits in each code word then

$$p = Q(\sqrt{\gamma_b R_c}) \tag{8-1-74}$$

for coherent detection and

$$p = \tfrac{1}{2}\exp\left(-\tfrac{1}{2}\gamma_b R_c\right) \tag{8-1-75}$$

for noncoherent detection.

Minimum-Distance (Maximum-Likelihood) Decoding The n bits from the demodulator corresponding to a received code word are passed to the decoder, which compares the received code word with the M possible transmitted code words and decides in favor of the code word that is closest in Hamming distance (number of bit positions in which two code words differ) to the received code word. This minimum distance decoding rule is optimum in the sense that it results in a minimum probability of a code word error for the binary symmetric channel.

A conceptually simple, albeit computationally inefficient, method for decoding is to first add (modulo 2) the received code word vector to all the M possible transmitted code words $\mathbf{C}_i$ to obtain the error vectors $\mathbf{e}_i$. Hence, $\mathbf{e}_i$ represents the error event that must have occurred on the channel in order to transform the code word $\mathbf{C}_i$ into the particular received code word. The number of errors in transforming $\mathbf{C}_i$ into the received code word is just equal to the number of 1s in $\mathbf{e}_i$. Thus, if we simply compute the weight of each of the M error vectors $\{\mathbf{e}_i\}$ and decide in favor of the code word that results in the smallest weight error vector, we have, in effect, a realization of the minimum distance decoding rule.

A more efficient method for hard-decision decoding makes use of the parity check matrix $\mathbf{H}$. To elaborate, suppose that $\mathbf{C}_m$ is the transmitted code word and $\mathbf{Y}$ is the received code word at the output of the demodulator. In general, $\mathbf{Y}$ may be expressed as

$$\mathbf{Y} = \mathbf{C}_m + \mathbf{e}$$

where $\mathbf{e}$ denotes an arbitrary binary error vector. The product $\mathbf{YH}'$ yields

$$\begin{aligned} \mathbf{YH}' &= (\mathbf{C}_m + \mathbf{e})\mathbf{H}' \\ &= \mathbf{C}_m\mathbf{H}' + \mathbf{eH}' \\ &= \mathbf{eH}' = \mathbf{S} \end{aligned} \tag{8-1-76}$$

where the $(n-k)$-dimensional vector $\mathbf{S}$ is called the *syndrome of the error pattern.* In other words, the vector $\mathbf{S}$ has components that are zero for all parity check equations that are satisfied and nonzero for all parity check equations that are not satisfied. Thus, $\mathbf{S}$ contains the pattern of failures in the parity checks.

We emphasize that the syndrome $\mathbf{S}$ is a characteristic of the error pattern and not of the transmitted code word. Furthermore, we observe that there are 2^n possible error patterns and only 2^{n-k} syndromes. Consequently, different error patterns result in the same syndrome.

Suppose we construct a decoding table in which we list all the 2^k possible code words in the first row, beginning with the all-zero code word in the first (left-most) column. This all-zero code word also represents the all-zero error pattern. We fill in the first column by listing first all $n-1$ error patterns $\{\mathbf{e}_i\}$ of weight 1. If $n < 2^{n-k}$, we may then list all double error patterns, then all triple error patterns, etc., until we have a total of 2^{n-k} entries in the first column. Thus, the number of rows that we can have is 2^{n-k}, which is equal to the number of syndromes. Next, we add each error pattern in the first column to the corresponding code words. Thus, we fill in the remainder of the $n \times (n-k)$ table as follows:

$$\begin{array}{lllll}
\mathbf{C}_1 & \mathbf{C}_2 & \mathbf{C}_3 & \ldots & \mathbf{C}_{2^k} \\
\mathbf{e}_2 & \mathbf{C}_2+\mathbf{e}_2 & \mathbf{C}_3+\mathbf{e}_2 & \ldots & \mathbf{C}_{2^k}+\mathbf{e}_2 \\
\mathbf{e}_3 & \mathbf{C}_2+\mathbf{e}_3 & \mathbf{C}_3+\mathbf{e}_3 & \ldots & \mathbf{C}_{2^k}+\mathbf{e}_3 \\
\vdots & \vdots & \vdots & & \vdots \\
\mathbf{e}_{2^{n-k}} & \mathbf{C}_2+\mathbf{e}_{2^{n-k}} & \mathbf{C}_3+\mathbf{e}_{2^{n-k}} & \ldots & \mathbf{C}_{2^k}+\mathbf{e}_{2^{n-k}}
\end{array}$$

This table is called a *standard array*. Each row, including the first, consists of k possible received code words that would result from the corresponding error pattern in the first column. Each row is called a *coset* and the first (left-most) code word (or error pattern) is called a *coset leader*. Therefore, a coset consists of all the possible received code words resulting from a particular error pattern (coset leader).

Example 8-1-10

Let us construct the standard array for the (5, 2), systematic code with generator matrix given by

$$\mathbf{G}=\begin{bmatrix}1 & 0 & 1 & 0 & 1\\ 0 & 1 & 0 & 1 & 1\end{bmatrix}$$

Thic code has a minimum distance $d_{\min}=3$. The standard array is given in Table 8-1-7. Note that in this code, the coset leaders consist of the all-zero error pattern, five error patterns of weight 1, and two error patterns of

TABLE 8-1-7 STANDARD ARRAY FOR THE (5, 2) CODE

Code words			
0 0 0 0 0	**0 1 0 1 1**	**1 0 1 0 1**	**1 1 1 1 0**
0 0 0 0 1	0 1 0 1 0	1 0 1 0 0	1 1 1 1 1
0 0 0 1 0	0 1 0 0 1	1 0 1 1 1	1 1 1 0 0
0 0 1 0 0	0 1 1 1 1	1 0 0 0 1	1 1 0 1 0
0 1 0 0 0	0 0 0 1 1	1 1 1 0 1	1 0 1 1 0
1 0 0 0 0	1 1 0 1 1	0 0 1 0 1	0 1 1 1 0
1 1 0 0 0	1 0 0 1 1	0 1 1 0 1	0 0 1 1 0
1 0 0 1 0	1 1 0 0 1	0 0 1 1 1	0 1 1 0 0

weight 2. Although many more double error patterns exist, there is only room for two to complete the table. These were selected such that their corresponding syndromes are distinct from those of the single error patterns.

Now, suppose that $\mathbf{e}_i$ is a coset leader and that $\mathbf{C}_m$ was the transmitted code word. Then, the error pattern $\mathbf{e}_i$ would result in the received code word

$$\mathbf{Y} = \mathbf{C}_m + \mathbf{e}_i$$

The syndrome is

$$\mathbf{S} = (\mathbf{C}_m + \mathbf{e}_i)\mathbf{H}' = \mathbf{C}_m\mathbf{H}' + \mathbf{e}_i\mathbf{H}' = \mathbf{e}_i\mathbf{H}'$$

Clearly, all received code words in the same coset have the same syndrome, since the latter depends only on the error pattern. Furthermore, each coset has a different syndrome. Having established this characteristic of the standard array, we may simply construct a syndrome decoding table in which we list the 2^{n-k} syndromes and the corresponding 2^{n-k} coset leaders that represent the minimum weight error patterns. Then, given a received code vector $\mathbf{Y}$, we compute the syndrome

$$\mathbf{S} = \mathbf{Y}\mathbf{H}'$$

For the computed $\mathbf{S}$, we find the corresponding (most likely) error vector, say $\hat{\mathbf{e}}_m$. This error vector is added to $\mathbf{Y}$ to yield the decoded word

$$\hat{\mathbf{C}}_m = \mathbf{Y} \oplus \hat{\mathbf{e}}_m$$

Example 8-1-11

Consider the (5, 2) code with the standard array given in Table 8-1-7. The syndromes versus the most likely error patterns are given in Table 8-1-8. Now suppose the actual error vector on the channel is

$$\mathbf{e} = [1 \quad 0 \quad 1 \quad 0 \quad 0]$$

TABLE 8-1-8 SYNDROME TABLE FOR THE (5, 2) CODE

Syndrome	Error pattern
0 0 0	0 0 0 0 0
0 0 1	0 0 0 0 1
0 1 0	0 0 0 1 0
1 0 0	0 0 1 0 0
0 1 1	0 1 0 0 0
1 0 1	1 0 0 0 0
1 1 0	1 1 0 0 0
1 1 1	1 0 0 1 0

The syndrome computed for the error is $\mathbf{S} = [0 \ 0 \ 1]$. Hence, the error determined from the table is $\hat{\mathbf{e}} = [0 \ 0 \ 0 \ 0 \ 1]$. When $\hat{\mathbf{e}}$ is added to $\mathbf{Y}$, the result is a decoding error. In other words the (5, 2) code corrects all single *errors and only two double errors, namely* $[1 \ 1 \ 0 \ 0 \ 0]$ *and* $[1 \ 0 \ 0 \ 1 \ 0]$.

Syndrome Decoding of Cyclic Codes As described above, hard-decision decoding of a linear block code may be accomplished by first computing the syndrome $\mathbf{S} = \mathbf{YH}'$, then using a table lookup to find the most probable error pattern $\mathbf{e}$ corresponding to the computed syndrome $\mathbf{S}$, and, finally, adding the error pattern $\mathbf{e}$ to the received vector $\mathbf{Y}$ to obtain the most probable code word $\hat{\mathbf{C}}_m$. When the code is cyclic, the syndrome computation may be performed by a shift register similar in form to that used for encoding.

To elaborate, let us consider a systematic cyclic code and let us represent the received code vector $\mathbf{Y}$ by the polynomial $Y(p)$. In general, $\mathbf{Y} = \mathbf{C} + \mathbf{e}$, where $\mathbf{C}$ is the transmitted code word and $\mathbf{e}$ is the error vector. Hence, we have

$$\begin{aligned} Y(p) &= C(p) + e(p) \\ &= X(p)g(p) + e(p) \end{aligned} \tag{8-1-77}$$

Now, suppose we divide $Y(p)$ by the generator polynomial $g(p)$. This division will yield

$$\frac{Y(p)}{g(p)} = Q(p) + \frac{R(p)}{g(p)}$$

or, equivalently,

$$Y(p) = Q(p)g(p) + R(p) \tag{8-1-78}$$

The remainder $R(p)$ is a polynomial of degree less than or equal to $n - k - 1$. If we combine (8-1-77) with (8-1-78), we obtain

$$e(p) = [X(p) + Q(p)]g(p) + R(p) \tag{8-1-79}$$

This relationship illustrates that the remainder $R(p)$ obtained from dividing $Y(p)$ by $g(p)$ depends only on the error polynomial $e(p)$, and, hence, $R(p)$ is simply the syndrome associated with the error pattern $\mathbf{e}$. Therefore,

$$Y(p) = Q(p)g(p) + S(p) \tag{8-1-80}$$

where $S(p)$ is the syndrome polynomial of degree less than or equal to $n - k - 1$. If $g(p)$ divides $Y(p)$ exactly then $S(p) = 0$ and the received decoded word is $\hat{\mathbf{C}}_m = \mathbf{Y}$.

The division of $Y(p)$ by the generator polynomial $g(p)$ may be carried out by means of a shift register which performs division as described previously. First the received vector $\mathbf{Y}$ is shifted into an $(n - k)$-stage shift register as

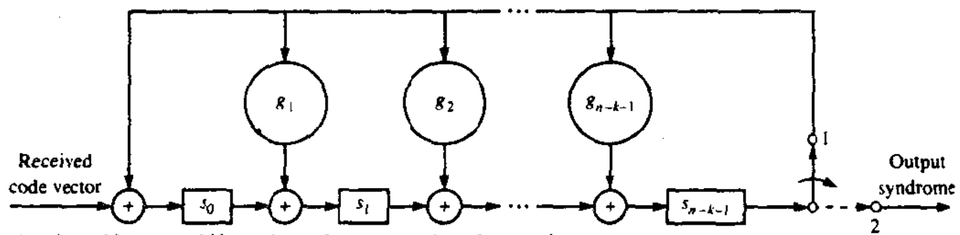


FIGURE 8-1-9 An $(n-k)$-stage shift register for computing the syndrome.

illustrated in Fig. 8-1-9. Initially, all the shift-register contents are zero and the switch is closed in position 1. After the entire n-bit received vector has been shifted into the register, the contents of the $n-k$ stages constitute the syndrome with the order of the bits numbered as shown in Fig. 8-1-9. These bits may be clocked out by throwing the switch into position 2. Given the syndrome from the $(n-k)$-stage shift register, a table lookup may be performed to identify the most probable error vector.

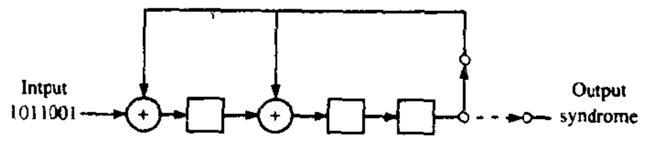
Example 8-1-12

Let us consider the syndrome computation for the (7, 4) cyclic Hamming code generated by the polynomial $g(p) = p^3 + p + 1$. Suppose that the received vector is $\mathbf{Y} = [1\ 0\ 0\ 1\ 1\ 0\ 1]$. This is fed into the three-stage register shown in Fig. 8-1-10. After seven shifts the contents of the shift register are 110, which corresponds to the syndrome $\mathbf{S} = [0\ 1\ 1]$. The most probable error vector corresponding to this syndrome is $\mathbf{e} = [0\ 0\ 0\ 1\ 0\ 0\ 0]$ and, hence,

$$\hat{\mathbf{C}}_m = \mathbf{Y} + \mathbf{e} = [1\quad 0\quad 0\quad 0\quad 1\quad 0\quad 1]$$

The information bits are 1 0 0 0.

FIGURE 8-1-10 Syndrome computation for the (7, 4) cyclic code with generator polynomial $g(p) = p^3 + p + 1$ and received vector $\mathbf{Y} = [1\ 0\ 0\ 1\ 1\ 0\ 1]$.



Shift	Register contents
0	000
1	100
2	010
3	001
4	010
5	101
5	100
7	110

The table lookup decoding method using the syndrome is practical only when $n-k$ is small, e.g., $n-k<10$. This method is impractical for many interesting and powerful codes. For example, if $n-k=20$, the table has 2^{20} (approximately 1 million) entries. Such a large amount of storage and the time required to locate an entry in such a large table renders the table lookup decoding method impractical for long codes having large numbers of check bits.

More efficient and practical hard-decision decoding algorithms have been devised for the class of cyclic codes and, more specifically, the BCH codes. A description of these algorithms requires further development of computational methods with finite fields, which is beyond the scope of our treatment of coding theory. It suffices to indicate that efficient decoding algorithms exist which make it possible to implement long BCH codes with high redundancy in practical digital communications systems. The interested reader is referred to the texts of Peterson and Weldon (1972), Lin and Costello (1983), Blahut (1983), and Berlekamp (1968), and to the paper by Forney (1965).

Error Detection and Error Correction Capability It is clear from the discussion above that when the syndrome consists of all zeros, the received code word is one of the 2^k possible transmitted code words. Since the minimum separation between a pair of code words is $d_{\min}$, it is possible for an error pattern of weight $d_{\min}$ to transform one of these 2^k code words in the code into another code word. When this happens we have an *undetected error*. On the other hand, if the actual number of errors is less than $d_{\min}$, the syndrome will have a nonzero weight. When this occurs, we have detected the presence of one or more errors on the channel. Clearly, the (n, k) block code is capable of *detecting* $d_{\min}-1$ errors. Error detection may be used in conjunction with an automatic repeat-request (ARQ) scheme for retransmission of the code word.

The *error correction capability* of a code also depends on the minimum distance. However, the number of correctable error patterns is limited by the number of possible syndromes or coset leaders in the standard array. To determine the error correction capability of an (n, k) code, it is convenient to view the 2^k code words as points in an n-dimensional space. If each code word is viewed as the center of a sphere of radius (Hamming distance) t, the largest value that t may have without intersection (or tangency) of any pair of the 2^k spheres is $t=\lfloor\frac{1}{2}(d_{\min}-1)\rfloor$, where $\lfloor x\rfloor$ denotes the largest integer contained in x. Within each sphere lie all the possible received code words of distance less than or equal to t from the valid code word. Consequently, any received code vector that falls within a sphere is decoded into the valid code word at the center of the sphere. This implies that an (n, k) code with minimum distance $d_{\min}$ is capable of correcting $t=\lfloor\frac{1}{2}(d_{\min}-1)\rfloor$ errors. Figure 8-1-11 is a two-dimensional representation of the code words and the spheres.

As described above, a code may be used to detect $d_{\min}-1$ errors or to correct $t=\lfloor\frac{1}{2}(d_{\min}-1)\rfloor$ errors. Clearly, to correct t error implies that we have

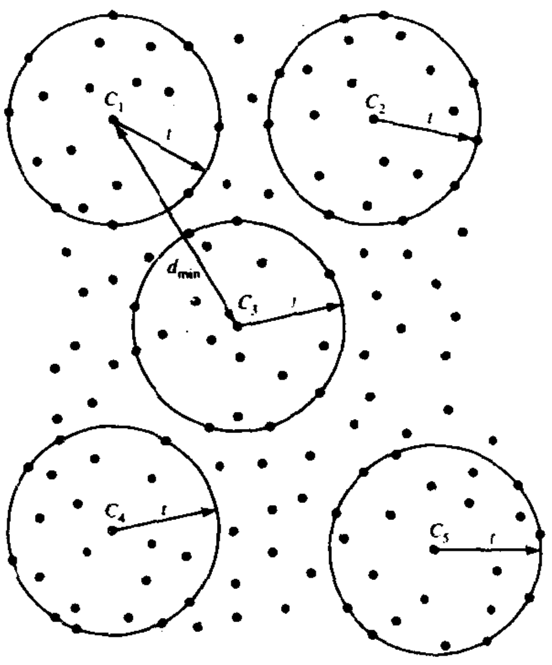


FIGURE 8-1-11 A representation of code words as centers of spheres of radius $t = \lfloor \frac{1}{2}(d_{\min} - 1) \rfloor$.

detected t errors. However, it is also possible to detect more than t errors if we compromise in the error correction capability of the code. For example, a code with $d_{\min} = 7$ can correct $t = 3$ errors. If we wish to detect four errors, we can do so by reducing the radius of the sphere around each code word from 3 to 2. Thus, patterns with four errors are detectable but only patterns of two errors are correctable. In other words, when only two errors occur, these are corrected, and when three or four errors occur, the receiver may ask for a retransmission. If more than four errors occur, they will go undetected if the code word falls within a sphere of radius 2. Similarly, for $d_{\min} = 7$, five errors can be detected and one error corrected. In general, a code with minimum distance $d_{\min}$ can detect e_d errors and correct e_c errors, where

$$e_d + e_c \leq d_{\min} - 1$$

and

$$e_c \leq e_d$$

Probability of Error Based on Error Correction We conclude this section with the derivation of the probability of error for hard-decision decoding of linear binary block codes based on error correction only.

From the above discussion, it is clear that the optimum decoder for a binary symmetric channel will decode correctly if (but not necessarily only if) the number of errors in a code word is less than half the minimum distance $d_{\min}$ of the code. That is, any number of errors up to

$$t = \lfloor \tfrac{1}{2}(d_{\min} - 1) \rfloor$$

are always correctable. Since the binary symmetric channel is memoryless, the bit errors occur independently. Hence, the probability of m errors in a block of n bits is

$$P(m, n) = \binom{n}{m} p^m (1-p)^{n-m} \tag{8-1-81}$$

and, therefore, the probability of a code word error is upper-bounded by the expression

$$P_M \leq \sum_{m=t+1}^{n} P(m, n) \tag{8-1-82}$$

Equality holds in (8-1-82) if the linear block code is a perfect code. In order to describe the basic characteristics of a perfect code, suppose we place a sphere of radius t around each of the possible transmitted code words. Each sphere around a code word contains the set of all code words of Hamming distance less than or equal to t from the code word. Now, the number of code words in a sphere of radius $t = \lfloor \frac{1}{2}(d_{\min} - 1) \rfloor$ is

$$1 + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{t} = \sum_{i=0}^{t} \binom{n}{i}$$

Since there are $M = 2^k$ possible transmitted code words, there are 2^k nonoverlapping spheres each having a radius t. The total number of code words enclosed in the 2^k spheres cannot exceed the 2^n possible received code words. Thus, a t-error correcting code must satisfy the inequality

$$2^k \sum_{i=0}^{t} \binom{n}{i} \leq 2^n$$

or, equivalently,

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i} \tag{8-1-83}$$

A perfect code has the property that all spheres of Hamming distance $t = \lfloor \frac{1}{2}(d_{\min} - 1) \rfloor$ around the $M = 2^k$ possible transmitted code words are disjoint and every received code word falls in one of the spheres. Thus, every received code word is at most, at distance t from one of the possible transmitted code words and (8-1-83) holds with equality. For such a code, all error patterns of weight less than or equal to t are corrected by the optimum (minimum distance) decoder. On the other hand, any error pattern of weight $t+1$ or greater cannot be corrected. Consequently, the expression for the error probability given in (8-1-82) holds with equality.The Golay (23, 12) code, having $d_{\min}=7$ and $t=3$, is a perfect code. The Hamming codes, which have the parameters $n=2^{n-k}-1$, $d_{min}=3$, and $t=1$, are also perfect codes. These two nontrivial codes and the trivial code consisting of two code words of odd length n and $d_{\min}=n$ are the only perfect binary block codes. These codes are optimum on the BSC in the sense that they result in a minimum error probability among all codes having the same block length and the same number of information bits.

The optimality property defined above also holds for quasiperfect codes. A quasiperfect code is characterized by the property that all spheres of Hamming radius t around the M possible transmitted code words are disjoint and every received code word is at most at distance $t+1$ from one of the possible transmitted code words. For such a code, all error patterns of weight less than or equal to t and some error patterns of weight $t+1$ are correctable, but any error pattern of weight $t+2$ or greater leads to incorrect decoding of the code word. Clearly, (8-1-82) is an upper bound on the error probability and

$$P_M \geqslant \sum_{m=t+2}^{n} P(m, n) \tag{8-1-84}$$

is a lower bound.

A more precise measure of the performance for quasiperfect codes can be obtained by making use of the inequality in (8-1-83). That is, the total number of code words outside the 2^k spheres of radius t is

$$N_{t+1}=2^n-2^k\sum_{i=0}^{t}\binom{n}{i}$$

If these code words are equally subdivided into 2^k sets and each set is associated with one of the 2^k spheres then each sphere is enlarged by the addition of

$$\beta_{t+1}=2^{n-k}-\sum_{i=0}^{t}\binom{n}{i} \tag{8-1-85}$$

code words having distance $t+1$ from the transmitted code word. Consequently, of the $\binom{n}{t+1}$ error patterns of distance $t+1$ from each code word, we can correct β_{t+1} error patterns. Thus, the error probability for decoding the quasiperfect code may be expressed as

$$P_M=\sum_{m=t+2}^{n} P(m,n)+\left[\binom{n}{t+1}-\beta_{t+1}\right]p^{t+1}(1-p)^{n-t-1} \tag{8-1-86}$$

There are many known quasiperfect codes, although they do not exist for all choices of n and k. Since such codes are optimum for the binary symmetric channel, any (n, k) linear block code must have an error probability that is at least as large as (8-1-86). Consequently, (8-1-86) is a lower bound on the probability of error for any (n, k) linear block code, where t is the largest integer such that $\beta_{t+1} \geq 0$.

Another pair of upper and lower bounds is obtained by considering two code words that differ by the minimum distance. First, we note that P_M cannot be less than the probability of erroneously decoding the transmitted code word as its nearest neighbor, which is at distance $d_{\min}$ from the transmitted code word. That is,

$$P_M \geq \sum_{m=[d_{\min}/2]+1}^{d_{\min}} \binom{d_{\min}}{m} p^m (1-p)^{d_{\min}-m} \tag{8-1-87}$$

On the other hand, P_M cannot be greater than $M - 1$ times the probability of erroneously decoding the transmitted code word as its nearest neighbor, which is at distance $d_{\min}$ from the transmitted code word. That is a union bound, which is expressed as

$$P_M \leq (M-1) \sum_{m=[d_{\min}/2]+1}^{d_{\min}} \binom{d_{\min}}{m} p^m (1-p)^{d_{\min}-m} \tag{8-1-88}$$

When M is large, the lower bound in (8-1-87) and the upper bound in (8-1-88) are very loose.

A tight upper bound on P_M can be obtained by applying the Chernoff bound presented earlier in Section 2-1-6. We assume again that the all-zero code was transmitted. In comparing the received code word to the all-zero code word and to a code word of weight w_m, the probability of a decoding error, obtained from the Chernoff bound (Problem 8-22), is upper-bounded by the expression

$$P_2(w_m) \leq [4p(1-p)]^{w_m/2} \tag{8-1-89}$$

The union of these binary decisions yields the upper bound

$$P_M \leq \sum_{m=2}^{M} [4p(1-p)]^{w_m/2} \tag{8-1-90}$$

A simpler version of (8-1-90) is obtained if we employ $d_{\min}$ in place of the weight distribution. That is,

$$P_M \leq (M-1)[4p(1-p)]^{d_{\min}/2} \tag{8-1-91}$$

Of course (8-1-90) is a tighter upper bound than (8-1-91).

In Section 8-1-6, we compare the various bounds given above for a specific code, namely, the Golay (23, 12) code. In addition, we compare the error rate performance of hard-decision and soft-decision decoding.

8-1-6 Comparison of Performance between Hard-Decision and Soft-Decision Decoding

It is both interesting and instructive to compare the bounds on the error rate performance of linear block codes for soft-decision decoding and hard-decision decoding on an AWGN channel. For illustrative purposes, we shall use the Golay (23, 12) code, which has the relatively simple weight distribution given in Table 8-1-1. As stated previously, this code has a minimum distance $d_{\min} = 7$.

First we compute and compare the bounds on the error probability for hard-decision decoding. Since the Golay (23, 12) code is a perfect code, the exact error probability for hard-decision decoding is

$$P_M = \sum_{m=4}^{23} \binom{23}{m} p^m (1-p)^{23-m}$$
$$= 1 - \sum_{m=0}^{3} \binom{23}{m} p^m (1-p)^{23-m} \qquad (8\text{-}1\text{-}92)$$

where p is the probability of a binary digit error for the binary symmetric channel. Binary (or four-phase) coherent PSK is assumed to be the modulation/demodulation technique for the transmission and reception of the binary digits contained in each code word. Thus, the appropriate expression for p is given by (8-1-73). In addition to the exact error probability given by (8-1-92), we have the lower bound given by (8-1-87) and the three upper bounds given by (8-1-88), (8-1-90), and (8-1-91).

Numerical results obtained from these bounds are compared with the exact error probability in Fig. 8-1-12. We observe that the lower bound is very loose.

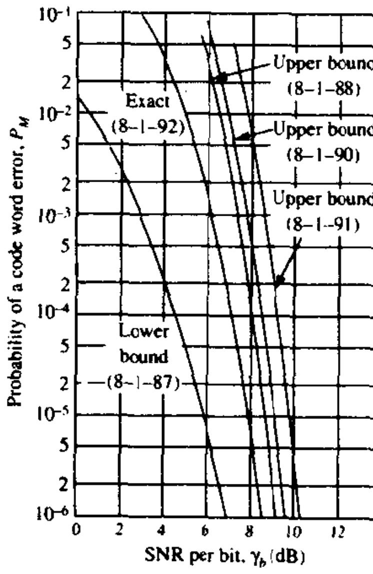


FIGURE 8-1-12 Comparison of bounds with exact error probability for hard-decision decoding of Golay (23, 12) code.

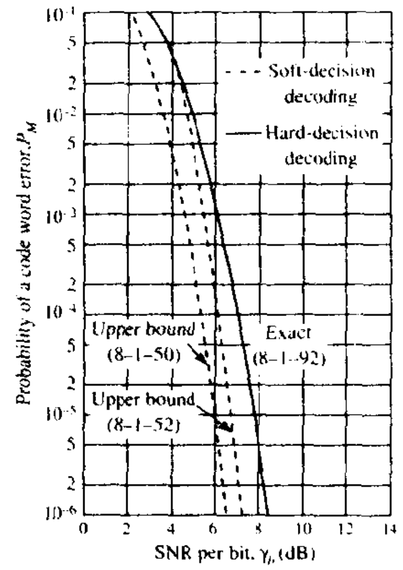


FIGURE 8-1-13 Comparison of soft-decision decoding with hard-decision decoding for the Golay (23, 12) code.

At $P_M = 10^{-5}$, the lower bound is off by approximately 2 dB from the exact error probability. At $P_M = 10^{-2}$, the difference increases to approximately 4 dB. Of the three upper bounds, the one given by (8-1-88) is the tightest; it differs by less than 1 dB from the exact error probability at $P_M = 10^{-5}$. The Chernoff bound in (8-1-90), which employs the weight distribution, is also relatively tight. Finally, the Chernoff bound that employs only the minimum distance of the code is the poorest of the three. At $P_M = 10^{-5}$, it differs from the exact error probability by approximately 2 dB. All three upper bounds are very loose for error rates above $P_M = 10^{-2}$.

It is also interesting to compare the performance between soft- and hard-decision decoding. For this comparison, we use the upper bounds on the error probability for soft-decision decoding given by (8-1-52) and the exact error probability for hard-decision decoding given by (8-1-92). Figure 8-1-13 illustrates these performance characteristics. We observe that the two bounds for soft-decision decoding differ by approximately 0.5 dB at $P_M = 10^{-6}$ and by approximately 1 dB at $P_M = 10^{-2}$. We also observe that the difference in performance between hard- and soft-decision decoding is approximately 2 dB in the range $10^{-2} < P_M < 10^{-6}$. In the range $P_M > 10^{-2}$, the curve of the error probability for hard-decision decoding crosses the curves for the bounds. This behavior indicates that the bounds for soft-decision decoding are loose when $P_M > 10^{-2}$.

The 2 dB difference between hard- and soft-decision decoding is a characteristic that applies not only to the Golay code, but is a fundamental result that applies in general to coded digital communications over the AWGN channel. This result is derived below by computing the capacity of the AWGN channel with hard- and soft-decision decoding.

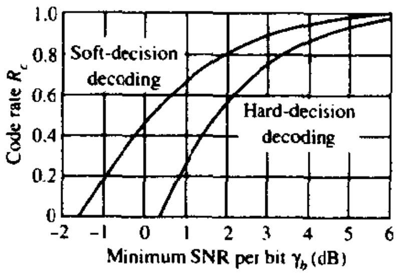


FIGURE 8-1-14 Code rate as a function of the minimum SNR per bit for soft- and hard-decision decoding.

The channel capacity of the BSC in bits per code symbol, derived in Section 7-1-2, is

$$C = 1 + p \log_2 p + (1-p) \log_2 (1-p) \tag{8-1-93}$$

where the probability of a bit error for binary, coherent PSK on an AWGN channel is given by (8-1-73). Suppose we use (8-1-73) for p, let $C = R_c$ in (8-1-93), and then determine the value of γ_b that satisfies this equation. The result is shown in Fig. 8-1-14 as a graph of R_c versus γ_b. For example, suppose that we are interested in using a code with rate $R_c = \frac{1}{2}$. For this code rate, note that the minimum SNR per bit required to achieve capacity with hard-decision decoding is approximately 1.6 dB.

What is the limit on the minimum SNR as the code rate approaches zero? For small values of R_c, the probability p can be approximated as

$$p \approx \tfrac{1}{2} - \sqrt{\gamma_b R_c / \pi} \tag{8-1-94}$$

When the expression for p is substituted into (8-1-93) and the logarithms in (8-1-93) are approximated by

$$\log_2 (1 + x) \approx (x - \tfrac{1}{2}x^2)/\ln 2$$

the channel capacity formula reduces to

$$C = \frac{2}{\pi \ln 2} \gamma_b R_c \tag{8-1-95}$$

Now we set $C = R_c$. Thus, in the limit as R_c approaches zero, we obtain the result

$$\gamma_b = \tfrac{1}{2}\pi \ln 2 \quad (0.37\,\text{dB}) \tag{8-1-96}$$

The capacity of the binary-input AWGN channel with soft-decision decoding can be computed in a similar manner. The expression for the capacity in bits per code symbol, derived in Section 7-1-2, is

$$C = \tfrac{1}{2} \sum_{k=0}^{1} \int_{-\infty}^{\infty} p(y \mid k) \log_2 \frac{p(y \mid k)}{p(y)} dy \tag{8-1-97}$$

where $p(y \mid k)$, $k = 0, 1$, denote the probability density functions of the demodulator output conditioned on the transmitted bit being a 0 and a 1, respectively. For the AWGN channel, we have

$$p(y \mid k) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-(y-m_k)^2/2\sigma^2}, \qquad k = 0, 1 \tag{8-1-98}$$

where $m_0 = -\sqrt{\mathscr{E}_c}$, $m_1 = \sqrt{\mathscr{E}_c}$, $\sigma^2 = \frac{1}{2}N_0$, and $\mathscr{E}_c = R_c \mathscr{E}_b$. The unconditional probability density $p(y)$ is simply one-half of the sum of $p(y \mid 1)$ and $p(y \mid 0)$. As R_c approaches zero, the expression (8-1-97) for the channel capacity can be approximated as

$$C \approx \gamma_b R_c / \ln 2 \tag{8-1-99}$$

Again, we set $C = R_c$. Thus, as $R_c \to 0$, the minimum SNR per bit to achieve capacity is

$$\gamma_b = \ln 2 \quad (-1.6\,\text{dB}) \tag{8-1-100}$$

By using (8-1-98) in (8-1-97) and setting $C = R_c$, a numerical solution can be obtained for code rates in the range $0 \leq R_c \leq 1$. The result of this solution is also shown in Fig. 8-1-14.

From the above, we observe that in the limit as R_c approaches zero, the difference in SNR γ_b between hard- and soft-decision decoding is $\frac{1}{2}\pi$, which is approximately 2 dB. On the other hand, as R_c increases toward unity, the difference in γ_b between these two decoding techniques decreases. For example, at $R_c = 0.8$, the difference is about 1.5 dB.

The curves in Fig. 8-1-14 provide more information than just the difference in performance between soft- and hard-decision decoding. These curves also specify the minimum SNR per bit that is required for a given code rate. For example, a code rate of $R_c = 0.8$ can provide arbitrarily small error probability at an SNR per bit of 2 dB, when soft-decision decoding is used. By comparison, an uncoded binary PSK requires 9.6 dB to achieve an error probability of 10^{-5} Hence, a 7.6 dB gain is possible by employing a rate $R_c = \frac{4}{5}$ code. Unfortunately, to achieve such a large coding gain usually implies the use of an extremely long block length code, which leads to a very complex receiver. Nevertheless, the curves in Fig. 8-1-14 provide a benchmark for comparing the coding gains achieved by practically implementable codes with the ultimate limits for either soft- or hard-decision decoding.

Instead of comparing the difference between hard- and soft-decision decoding based on the channel capacity relations, we may perform similar comparisons based on the random coding rate parameters. In Chapter 7, we demonstrated that the ensemble average probability of error for randomly selected binary code words is upper-bounded as

$$\bar{P}_e < 2^{-n(R_0 - R_c)} \tag{8-1-101}$$

where $R_c = k/n$ is the code rate and the cutoff rate R_0 represents the upper bound on R_c such that $\bar{P}_e \to 0$ as $n \to \infty$. For unquantized (soft-decision) decoding, R_0 is given as

$$R_0 = \log_2 \frac{2}{1 - e^{-\mathscr{E}_c/N_0}} \tag{8-1-102}$$

where $\mathscr{E}_c/N_0 = R_c \gamma_b$ is the SNR per dimension. This result was derived in Section 7-2.

On the other hand, if the output of the demodulator is quantized to Q levels prior to decoding, the Chernoff bound may be used to upper-bound the ensemble average binary error probability $\overline{P_2(\mathbf{s}_i, \mathbf{s}_m)}$ defined in Section 7-2. The result of this derivation is the same upper bound on $\bar{P}_e$ given in (8-1-101) but with R_0 replaced by R_Q, where

$$R_Q = \max_{\{p_j\}} \left\{ -\log_2 \sum_{i=0}^{Q-1} \left[\sum_{j=0}^{1} p_j \sqrt{P(i \mid j)} \right]^2 \right\} \tag{8-1-103}$$

In (8-1-103), $\{p_j\}$ are the prior probabilities of the two signals at the input to the channel and $\{P(i \mid j)\}$ denote the transition probabilities of the channel. For example, in the case of a binary symmetric channel, we have $p_1 = p_0 = \frac{1}{2}$, $P(0 \mid 0) = P(1 \mid 1) = 1 - p$, and $P(0 \mid 1) = P(1 \mid 0) = p$. Hence,

$$R_Q = \log_2 \frac{2}{1 + \sqrt{4p(1-p)}} \qquad Q = 2 \tag{8-1-104}$$

where

$$p = Q(\sqrt{2\gamma_b R_c}) \tag{8-1-105}$$

A plot of R_Q versus $10 \log (\mathscr{E}_c/N_0)$ is illustrated in Fig. 8-1-15 for $Q = 2$ and $Q = \infty$ (soft-decision decoding). Note that the difference in decoder performance between unquantized soft-decision decoding and hard-decision decoding is approximately 2 dB. In fact, it is easily demonstrated again that as $\mathscr{E}_c/N_0 \to 0$, the loss in performance due to hard-decision decoding is

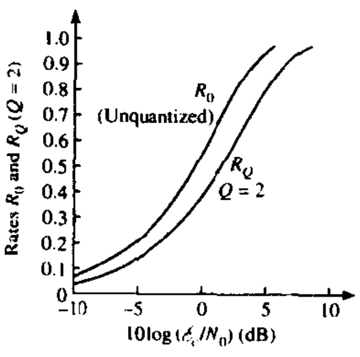


FIGURE 8-1-15 Comparison of R_0 (soft-decision decoding) with R_Q (hard-decision decoding) as a function of the SNR per dimension.

$10\log_{10}\frac{1}{2}\pi \approx 2\,\text{dB}$, which is the same decibel difference that was obtained in our comparison of the channel capacity relations. We mention that about 1 dB of this loss can be recovered by quantizing the output of the demodulator to three levels instead of two (see Problem 7-11). Additional improvements are possible by quantizing the output into more than three levels, as shown in Section 7-3.

8-1-7 Bounds on Minimum Distance of Linear Block Codes

The expressions for the probability of error derived in this chapter for soft-decision and hard-decision decoding of linear binary block codes clearly indicate the importance that the minimum distance parameter plays in the performance of the code. If we consider soft-decision decoding, for example, the upper bound on the error probability given by (8-1-52) indicates that, for a given code rate $R_c = k/n$, the probability of error in an AWGN channel decreases exponentially with $d_{\min}$. When this bound is used in conjunction with the lower bound on $d_{\min}$ given below, we obtain an upper bound on P_M that can be achieved by many known codes. Similarly, we may use the upper bound given by (8-1-82) for the probability of error for hard-decision decoding in conjunction with the lower bound on $d_{\min}$ to obtain an upper bound on the error probability for linear binary block codes on the binary symmetric channel.

On the other hand, an upper bound on $d_{\min}$ can be used to determine a lower bound on the probability of error achieved by the best code. For example, suppose that hard-decision decoding is employed. In this case, we have the two lower bounds on P_M given by (8-1-86) and (8-1-87), with the former being the tighter. When either one of these two bounds is used in conjunction with an upper bound on $d_{\min}$ the result is a lower bound on P_M for the best (n, k) code. Thus, upper and lower bounds on $d_{\min}$ are important in assessing the capabilities of codes.

A simple upper bound on the minimum distance of an (n, k) binary or non-binary linear block code was given in (8-1-14) as $d_{\min} \leq n - k + 1$. It is convenient to normalize this expression by the block size n. That is,

$$\frac{d_{\min}}{n} \leq (1 - R_c) + \frac{1}{n} \tag{8-1-106}$$

where R_c is the code rate. For large n, the factor $1/n$ can be neglected.

If a code has the largest possible distance, i.e., $d_{\min} = n - k + 1$, it is called a *maximum-distance-separable code.* Except for the trivial repetition-type codes, there are no binary maximum-separable codes. In fact, the upper bound in (8-1-106) is extremely loose for binary codes. On the other hand, nonbinary codes with $d_{\min} = n - k + 1$ do exist. For example, the Reed–Solomon codes, which comprise a subclass of BCH codes, are maximum-distance-separable.

In addition to the upper bound given above, there are several relatively tight bounds on the minimum distance of linear block codes. We shall briefly describe four important bounds, three of which are upper bounds and the other a lower bound. The derivations of these bounds are lengthy and are not of particular interest in our subsequent discussion. The interested reader may refer to Chapter 4 of the book by Peterson and Weldon (1972) for those derivations.

One upper bound on the minimum distance can be obtained from the inequality in (8-1-83). By taking the logarithm of both sides of (8-1-83) and dividing by n, we obtain

$$1 - R_c \geq \frac{1}{n} \log_2 \sum_{i=0}^{t} \binom{n}{i} \tag{8-1-107}$$

Since the error-correcting capability of the code, measured by t, is related to the minimum distance, the above relation is an upper bound on the minimum distance. It is called the *Hamming upper bound.*

The asymptotic form of (8-1-107) is obtained by letting $n \to \infty$. Now, for any n, let t_0 be the largest integer t for which (8-1-107) holds. Then, it can be shown (Peterson and Weldon, 1972) that as $n \to \infty$, the ratio t/n for any (n, k) block code cannot exceed t_0/n, where t_0/n satisfies the equation

$$1 - R_c = H(t_0/n) \tag{8-1-108}$$

and $H(x)$ is the binary entropy function defined by (3-2-10).

The generalization of the Hamming bound to nonbinary codes is simply

$$1 - R_c \geq \frac{1}{n} \log_q \left[\sum_{i=0}^{t} \binom{n}{i} (q-1)^i \right] \tag{8-1-109}$$

Another upper bound, developed by Plotkin (1960), may be stated as follows. The number of check digits required to achieve a minimum distance $d_{\min}$ in an (n, k) linear block code satisfies the inequality

$$n - k \geq \frac{q d_{\min} - 1}{q - 1} - 1 - \log_q d_{\min} \tag{8-1-110}$$

where q is the alphabet size. When the code is binary, (8-1-110) may be expressed as

$$\frac{d_{\min}}{n} \left(1 - \frac{1}{2 d_{\min}} \log_2 d_{\min} \right) \leq \frac{1}{2} \left(1 - R_c + \frac{2}{n} \right)$$

In the limit as $n \to \infty$ with $d_{\min}/n \leq \frac{1}{2}$, (8-1-110) reduces to

$$d_{\min}/n \leq \tfrac{1}{2}(1 - R_c) \tag{8-1-111}$$

Finally, there is another tight upper bound on the minimum distance obtained by Elias (Berlekamp, 1968). It may be expressed in its asymptotic form as

$$d_{\min}/n \leq 2A(1-A) \tag{8-1-112}$$

where the parameter A is related to the code rate through the equation

$$R_c = 1 + A\log_2 A + (1-A)\log_2(1-A), \qquad 0 \leq A \leq \tfrac{1}{2} \tag{8-1-113}$$

Lower bounds on the minimum distance of (n, k) block codes also exist. In particular, binary block codes exist that have a normalized minimum distance that asymptotically satisfies the inequality

$$d_{\min}/n \geq \alpha \tag{8-1-114}$$

where α is related to the code rate through the equation

$$\begin{aligned} R_c &= 1 - H(\alpha) \\ &= 1 + \alpha\log_2\alpha + (1-\alpha)\log_2(1-\alpha), \qquad 0 \leq \alpha \leq \tfrac{1}{2} \end{aligned} \tag{8-1-115}$$

This lower bound is a special case of a lower bound developed by Gilbert (1952) and Varsharmov (1957), which applies to nonbinary and binary block codes.

The asymptotic bounds given above are plotted in Fig. 8-1-16 for binary codes. Also plotted in the figure for purposes of comparison are curves of the minimum distance as a function of code rate for BCH codes of block lengths $n = 31$ and 63. We observe that for $n = 31$ and 63, the normalized minimum distance falls well above the Varsharmov–Gilbert lower bound. As the block length n increases, the efficiency of the BCH codes diminishes. For example, when $n = 1023$, the curve for the normalized minimum distance falls close to

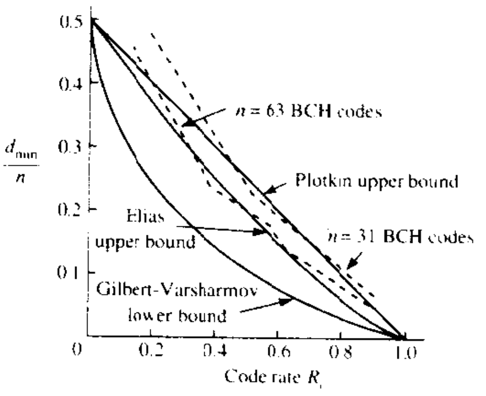


FIGURE 8-1-16 Upper and lower bounds on normalized minimum distance as a function of code rate.

the Varsharmov–Gilbert bound. As n increases beyond $n = 1023$, the normalized minimum distance of the BCH codes continues to decrease and falls below the Varsharmov–Gilbert bound. That is, $d_{\min}/n$ approaches zero as n tends to infinity. Consequently the BCH codes, which are the most important class of cyclic codes, are not very efficient at large block lengths.

8-1-8 Nonbinary Block Codes and Concatenated Block Codes

A nonbinary block code consists of a set of fixed-length code words in which the elements of the code words are selected from an alphabet of q symbols, denoted by $\{0, 1, 2, \ldots, q-1\}$. Usually, $q = 2^k$, so that k information bits are mapped into one of the q symbols. The length of the nonbinary code word is denoted by N and the number of information symbols encoded into a block of N symbols is denoted by K. The minimum distance of the nonbinary code is denoted by $D_{\min}$. A systematic (N, K) block code consists of K information symbols and $N - K$ parity check symbols.

Among the various types of *nonbinary linear block codes*, the Reed–Solomon codes are some of the most important for practical applications. As indicated previously, they comprise a subset of the BCH codes, which in turn are a class of cyclic codes. These codes are described by the parameters

$$\begin{aligned} N &= q - 1 = 2^k - 1 \\ K &= 1, 2, 3, \ldots, N-1 \\ D_{\min} &= N - K + 1 \\ R_c &= K/N \end{aligned} \qquad (8\text{-}1\text{-}116)$$

Such a code is guaranteed to correct up to

$$\begin{aligned} t &= \lfloor \tfrac{1}{2}(D_{\min} - 1) \rfloor \\ &= \lfloor \tfrac{1}{2}(N - K) \rfloor \end{aligned} \qquad (8\text{-}1\text{-}117)$$

symbol errors. Of course, these codes may be extended or shortened in the manner described previously for binary block codes.

The weight distribution $\{A_i\}$ of the class of Reed–Solomon codes is known. The coefficients in the weight enumerating polynomial are given as

$$A_i = \binom{N}{i}(q-1)\sum_{j=0}^{i-D}(-1)^j\binom{i-1}{j}q^{i-j-D}, \qquad i \geq D_{\min} \qquad (8\text{-}1\text{-}118)$$

where $D \equiv D_{\min}$ and $q = 2^k$.

One reason for the importance of the Reed–Solomon codes is their good distance properties. A second reason for their importance is the existence of efficient hard-decision decoding algorithms, which make it possible to implement relatively long codes in many practical applications where coding is desirable.

A nonbinary code is particularly matched to an M-ary modulation technique for transmitting the 2^k possible symbols. Specifically, M-ary orthogonal signaling, e.g., M-ary FSK, is frequently used. Each of the 2^k symbols in the q-ary alphabet is mapped to one of the $M = 2^k$ orthogonal signals. Thus, the transmission of a code word is accomplished by transmitting N orthogonal signals, where each signal is selected from the set of $M = 2^k$ possible signals.

The optimum demodulator for such a signal corrupted by AWGN consists of M matched filters (or cross-correlators) whose outputs are passed to the decoder, either in the form of soft decisions or in the form of hard decisions. If hard decisions are made by the demodulator, the symbol error probability P_M and the code parameters are sufficient to characterize the performance of the decoder. In fact, the modulator, the AWGN channel, and the demodulator form an equivalent discrete (M-ary) input, discrete (M-ary) output, symmetric memoryless channel characterized by the transition probabilities $P_c = 1 - P_M$ and $P_M/(M-1)$. This channel model, which is illustrated in Fig. 8-1-17, is a generalization of the BSC.

The performance of the hard-decision decoder may be characterized by the following upper bound on the code word error probability:

$$P_e \le \sum_{i=t+1}^{N} \binom{N}{i} P_M^i (1-P_M)^{N-i} \tag{8-1-119}$$

where t is the number of errors guaranteed to be corrected by the code.

When a code word error is made, the corresponding symbol error probability is

$$P_{es} = \frac{1}{N} \sum_{i=t+1}^{N} i \binom{N}{i} P_M^i (1-P_M)^{N-i} \tag{8-1-120}$$

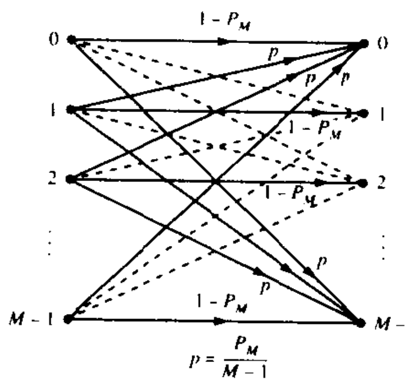


FIGURE 8-1-17 M-ary input, M-ary output, symmetric memoryless channel.

Furthermore, if the symbols are converted to binary digits, the bit error probability corresponding to (8-1-120) is

$$P_{eb} = \frac{2^{k-1}}{2^k - 1} P_{es} \tag{8-1-121}$$

Example 8-1-13

Let us evaluate the performance of an $N = 2^5 - 1 = 31$ Reed–Solomon code with $D_{\min} = 3$, 5, 9, and 17. The corresponding values of K are 29, 27, 23, and 15. The modulation is $M = 32$ orthogonal FSK with noncoherent detection at the receiver.

The probability of a symbol error is given by (5-4-46), and may be expressed as

$$P_M = \frac{1}{M} e^{-\gamma} \sum_{n=2}^{M} (-1)^n \binom{M}{n} e^{\gamma/n} \tag{8-1-122}$$

where γ is the SNR per code symbol. By using (8-1-122) in (8-1-120) and combining the result with (8-1-121), we obtain the bit error probability. The results of these computations are plotted in Fig. 8-1-18. Note that the more powerful codes (large $D_{\min}$) give poorer performance at low SNR per bit than the weaker codes. On the other hand, at high SNR, the more powerful codes give better performance. Hence, there are crossovers among the various codes, as illustrated for example in Fig. 8-1-18 for the $t = 1$ and $t = 8$ codes. Crossovers also occur among the $t = 1$, 2, and 4 codes at smaller values of SNR per bit. Similarly, the curves for $t = 4$ and 8 and for $t = 8$ and 2 cross in the region of high SNR. This is the characteristic behavior for noncoherent detection of the coded waveforms.

If the demodulator does not make a hard decision on each symbol, but

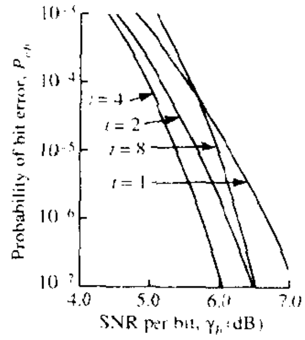


FIGURE 8-1-18 Performance of several $N = 31$, t-error correcting Reed–Solomon codes with 32-ary FSK modulation on an AWGN channel (noncoherent demodulation).

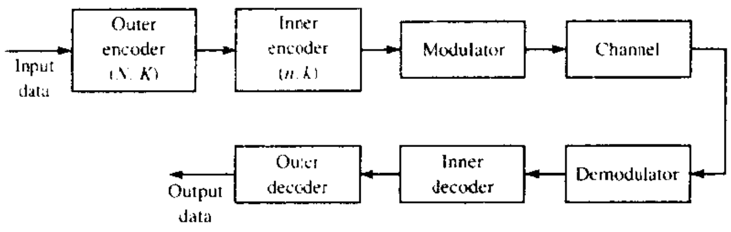


FIGURE 8-1-19 Block diagram of a communications system employing a concatenated code.

instead, passes the unquantized matched filter outputs to the decoder, soft-decision decoding can be performed. This decoding involves the formation of $q^K = 2^{kK}$ correlation metrics, where each metric corresponds to one of the q^K code words and consists of a sum of N matched filter outputs corresponding to the N code symbols. The matched filter outputs may be added coherently, or they may be envelope-detected and then added, or they may be square-law detected and then added. If coherent detection is used and the channel noise is AWGN, the computation of the probability of error is a straightforward extension of the binary case considered in Section 8-1-4. On the other hand, when envelope detection or square-law detection and noncoherent combining are used to form the decision variables, the computation of the decoder performance is considerably more complicated.

Concatenated Block Codes A concatenated code consists of two separate codes which are combined to form a larger code. Usually one code is selected to be nonbinary and the other is binary. The two codes are concatenated as illustrated in Fig. 8-1-19. The nonbinary (N, K) code forms the outer code and the binary code forms the inner code. Code words are formed by subdividing a block of kK information bits into K groups, called *symbols*, where each symbol consists of k bits. The K k-bit symbols are encoded into N k-bit symbols by the outer encoder, as is usually done with a nonbinary code. The inner encoder takes each k-bit symbol and encodes it into a binary block code of length n. Thus we obtain a concatenated block code having a block length of Nn bits and containing kK information bits. That is, we have created an equivalent (Nn, Kk) long binary code. The bits in each code word are transmitted over the channel by means of PSK or, perhaps, by FSK.

We also indicate that the minimum distance of the concatenated code is $d_{\min}D_{\min}$, where $D_{\min}$ is the minimum distance of the outer code and $d_{\min}$ is the minimum distance of the inner code. Furthermore, the rate of the concatenated code is Kk/Nn, which is equal to the product of the two code rates.

A hard-decision decoder for a concatenated code is conveniently separated into an inner decoder and an outer decoder. The inner decoder takes the hard decisions on each group of n bits, corresponding to a code word of the inner code, and makes a decision on the k information bits based on maximum-likelihood (minimum-distance) decoding. These k bits represent one symbol of the outer code. When a block of N k-bit symbols are received from the inner decoder, the outer decoder makes a hard decision on the K k-bit symbols based on maximum-likelihood decoding.

Soft-decision decoding is also a possible alternative with a concatenated code. Usually, the soft-decision decoding is performed on the inner code, if it is selected to have relatively few code words, i.e., if 2^k is not too large. The outer code is usually decoded by means of hard-decision decoding, especially if the block length is long and there are many code words. On the other hand, there may be a significant gain in performance when soft-decision decoding is used on both the outer and inner codes, to justify the additional decoding complexity. This is the case in digital communications over fading channels, as we shall demonstrate in Chapter 14.

We conclude this subsection with the following example.

Example 8-1-14

Suppose that the (7, 4) Hamming code described in Examples 8-1-1 and 8-1-2 is used as the inner code in a concatenated code in which the outer code is a Reed–Solomon code. Since $k = 4$, we select the length of the Reed–Solomon code to be $N = 2^4 - 1 = 15$. The number of information symbols K per outer code word may be selected over the range $1 \leq K \leq 14$ in order to achieve a desired code rate.

8-1-9 Interleaving of Coded Data for Channels with Burst Errors

Most of the well-known codes that have been devised for increasing the reliability in the transmission of information are effective when the errors caused by the channel are statistically independent. This is the case for the AWGN channel. However, there are channels that exhibit bursty error characteristics. One example is the class of channels characterized by multipath and fading, which is described in detail in Chapter 14. Signal fading due to time-variant multipath propagation often causes the signal to fall below the noise level, thus resulting in a large number of errors. A second example is the class of magnetic recording channels (tape or disk) in which defects in the recording media result in clusters of errors. Such error clusters are not usually corrected by codes that are optimally designed for statistically independent errors.

Considerable work has been done on the construction of codes that are capable of correcting burst errors. Probably the best known burst error correcting codes are the subclass of cyclic codes called Fire codes, named after P. Fire (1959), who discovered them. Another class of cyclic codes for burst error correction were subsequently discovered by Burton (1969).

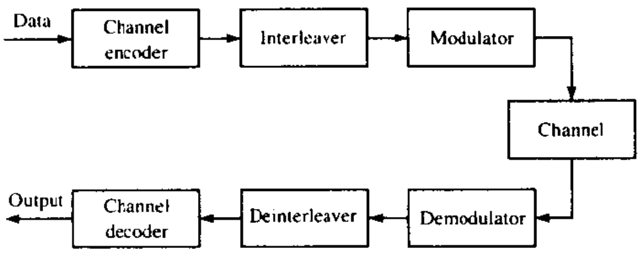


FIGURE 8-1-20 Block diagram of system employing interleaving for burst-error channel.

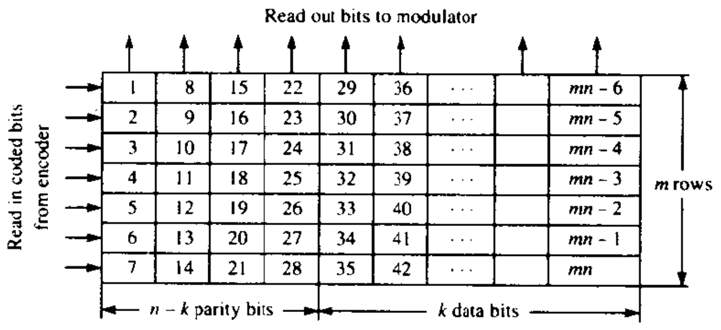
A *burst* of errors of length b is defined as a sequence of b-bit errors, the first and last of which are 1's. The *burst error correction capability* of a code is defined as one less than the length of the shortest uncorrectable burst. It is relatively easy to show that a systematic (n, k) code, which has $n - k$ parity check bits, can correct bursts of length $b \leq \lfloor \frac{1}{2}(n-k) \rfloor$.

An effective method for dealing with burst error channels is to interleave the coded data in such a way that the bursty channel is transformed into a channel having independent errors. Thus, a code designed for independent channel errors (short bursts) is used.

A block diagram of a system that employs interleaving is shown in Fig. 8-1-20. The encoded data are reordered by the interleaver and transmitted over the channel. At the receiver, after (either hard- or soft-decision) demodulation, the deinterleaver puts the data in proper sequence and passes it to the decoder. As a result of the interleaving/deinterleaving, error bursts are spread out in time so that errors within a code word appear to be independent.

The interleaver can take one of two forms: a block structure or a convolutional structure. A block *interleaver* formats the encoded data in a rectangular array of m rows and n columns. Usually, each row of the array constitutes a code word of length n. An *interleaver of degree m* consists of m rows (m code words) as illustrated in Fig. 8-1-21. The bits are read out

FIGURE 8-1-21 A block interleaver for coded data.


Read out bits to modulator
Read in coded bits from encoder
1 8 15 22 29 36 ··· mn − 6
2 9 16 23 30 37 ··· mn − 5
3 10 17 24 31 38 ··· mn − 4
4 11 18 25 32 39 ··· mn − 3
5 12 19 26 33 40 ··· mn − 2
6 13 20 27 34 41 ··· mn − 1
7 14 21 28 35 42 ··· mn
m rows
n − k parity bits
k data bits

column-wise and transmitted over the channel. At the receiver, the deinterleaver stores the data in the same rectangular array format, but it is read out row-wise, one code word at a time. As a result of this reordering of the data during transmission, a burst of errors of length $l = mb$ is broken up into m bursts of length b. Thus, an (n, k) code that can handle burst errors of length $b \le \lfloor \frac{1}{2}(n-k) \rfloor$ can be combined with an interleaver of degree m to create an interleaved (mn, mk) block code that can handle bursts of length mb.

A *convolutional interleaver* can be used in place of a block interleaver in much the same way. Convolutional interleavers are better matched for use with the class of convolutional codes that is described in the following section. Convolutional interleaver structures have been described by Ramsey (1970) and Forney (1971).

8-2 CONVOLUTIONAL CODES

A convolutional code is generated by passing the information sequence to be transmitted through a linear finite-state shift register. In general, the shift register consists of K (k-bit) stages and n linear algebraic function generators, as shown in Fig. 8-2-1. The input data to the encoder, which is assumed to be binary, is shifted into and along the shift register k bits at a time. The number of output bits for each k-bit input sequence is n bits. Consequently, the code rate is defined as $R_c = k/n$, consistent with the definition of the code rate for a block code. The parameter K is called the *constraint length* of the convolutional code.†

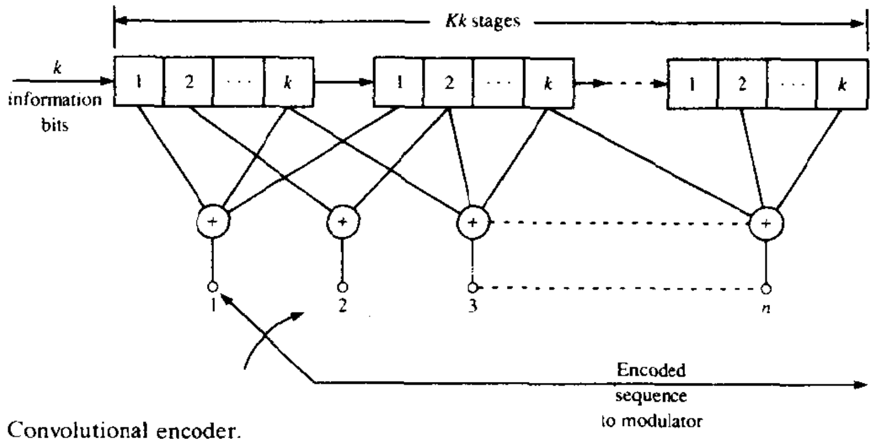


FIGURE 8-2-1 Convolutional encoder.

† In many cases, the constraint length of the code is given in bits rather than k-bit bytes. Hence the shift register may be called a *L-stage shift register*, where $L = Kk$. Furthermore, L may not be a multiple of k, in general.

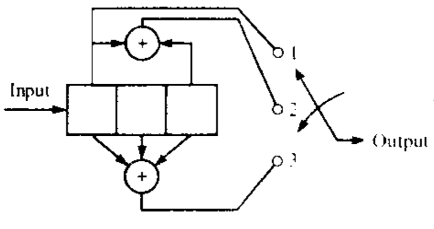


FIGURE 8-2-2 $K = 3$, $k = 1$, $n = 3$ convolutional encoder.

One method for describing a convolutional code is to give its generator matrix, just as we did for block codes. In general, the generator matrix for a convolutional code is semi-infinite since the input sequence is semi-infinite in length. As an alternative to specifying the generator matrix, we shall use a functionally equivalent representation in which we specify a set of n vectors, one vector for each of the n modulo-2 adders. Each vector has Kk dimensions and contains the connections of the encoder to that modulo-2 adder. A 1 in the ith position of the vector indicates that the corresponding stage in the shift register is connected to the modulo-2 adder and a 0 in a given position indicates that no connection exists between that stage and the modulo-2 adder.

To be specific, let us consider the binary convolutional encoder with constraint length $K = 3$, $k = 1$, and $n = 3$, which is shown in Fig. 8-2-2. Initially, the shift register is assumed to be in the all-zero state. Suppose the first input bit is a 1. Then the output sequence of 3 bits is 111. Suppose the second bit is a 0. The output sequence will then be 001. If the third bit is a 1, the output will be 100, and so on. Now, suppose we number the outputs of the function generators that generate each three-bit output sequence as 1, 2, and 3, from top to bottom, and similarly number each corresponding function generator. Then, since only the first stage is connected to the first function generator (no modulo-2 adder is needed), the generator is

$$\mathbf{g}_1 = [100]$$

The second function generator is connected to stages 1 and 3. Hence

$$\mathbf{g}_2 = [101]$$

Finally,

$$\mathbf{g}_3 = [111]$$

The generators for this code are more conveniently given in octal form as $(4, 5, 7)$. We conclude that, when $k = 1$, we require n generators, each of dimension K to specify the encoder.

For a rate k/n binary convolutional code with $k > 1$ and constraint length K, the n generators are Kk-dimensional vectors, as stated above. The following example illustrates the case in which $k = 2$ and $n = 3$.

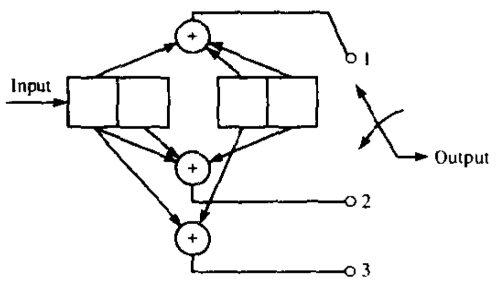


FIGURE 8-2-3 $K = 2$, $k = 2$, $n = 3$ convolutional encoder.

Example 8-2-1

Consider the rate 2/3 convolutional encoder illustrated in Fig. 8-2-3. In this encoder, two bits at a time are shifted into it and three output bits are generated. The generators are

$$\mathbf{g}_1 = [1011], \quad \mathbf{g}_2 = [1101], \quad \mathbf{g}_3 = [1010]$$

In octal form, these generators are (13, 15, 12).

There are three alternative methods that are often used to describe a convolutional code. These are the tree diagram, the trellis diagram, and the state diagram. For example, the tree diagram for the convolutional encoder shown in Fig. 8-2-2 is illustrated in Fig. 8-2-4. Assuming that the encoder is in the all-zero state initially, the diagram shows that, if the first input bit is a 0, the output sequence is 000 and, if the first bit is a 1, the output sequence is 111. Now, if the first input bit is a 1 and the second bit is a 0, the second set of three output bits is 001. Continuing through the tree, we see that if the third bit is a

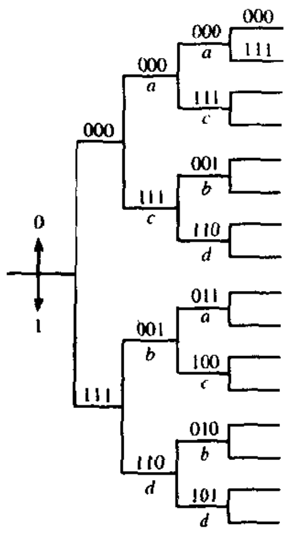
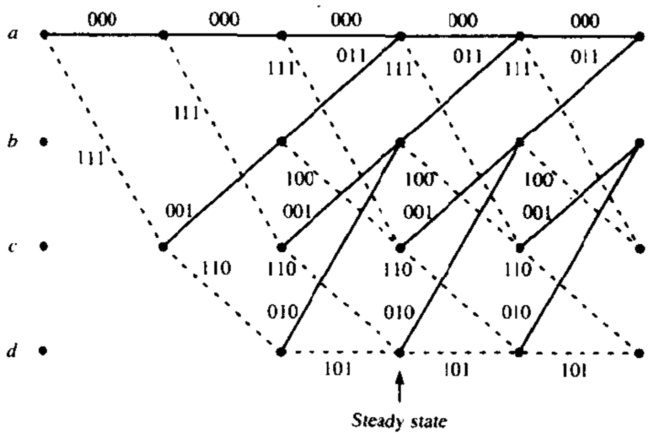


FIGURE 8-2-4 Tree diagram for rate 1/3, $K = 3$ convolutional code.

0 then the output is 011, while if the third bit is a 1 then the output is 100. Given that a particular sequence has taken us to a particular node in the tree, the branching rule is to follow the upper branch if the next input bit is a 0 and the lower branch if the bit is a 1. Thus, we trace a particular path through the tree that is determined by the input sequence.

Close observation of the tree that is generated by the convolutional encoder shown in Fig. 8-2-2 reveals that the structure repeats itself after the third stage. This behavior is consistent with the fact that the constraint length $K = 3$. That is, the three-bit output sequence at each stage is determined by the input bit and the two previous input bits, i.e., the two bits contained in the first two stages of the shift register. The bit in the last stage of the shift register is shifted out at the right and does not affect the output. Thus we may say that the three-bit output sequence for each input bit is determined by the input bit and the four possible states of the shift register, denoted as $a = 00$, $b = 01$, $c = 10$, $d = 11$. If we label each node in the tree to correspond to the four possible states in the shift register, we find that at the third stage there are two nodes with the label a, two with the label b, two with the label c, and two with the label d. Now we observe that all branches emanating from two nodes having the same label (same state) are identical in the sense that they generate identical output sequences. This means that the two nodes having the same label can be merged. If we do this to the tree shown in Fig. 8-2-4, we obtain another diagram, which is more compact, namely, a *trellis*. For example, the trellis diagram for the convolutional encoder of Fig. 8-2-2 is shown in Fig. 8-2-5. In drawing this diagram, we use the convention that a solid line denotes the output generated by the input bit 0 and a dotted line the output generated by the input bit 1. In the example being considered, we observe that, after the initial transient, the trellis contains four nodes at each stage, corresponding to the four states of the shift register, a, b, c, and d. After the second stage, each node in the trellis has two incoming paths and two outgoing paths. Of the two

FIGURE 8-2-5 Trellis diagram for rate 1/3, $K = 3$ convolutional code.



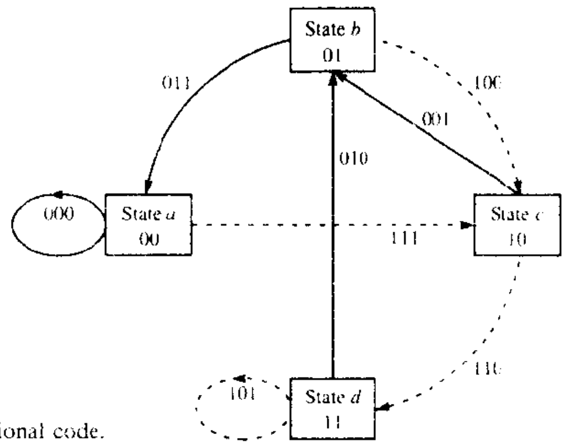


FIGURE 8-2-6 State diagram for rate 1/3, $K = 3$ convolutional code.

outgoing paths, one corresponds to the input bit 0 and the other to the path followed if the input bit is a 1.

Since the output of the encoder is determined by the input and the state of the encoder, an even more compact diagram than the trellis is the state diagram. The state diagram is simply a graph of the possible states of the encoder and the possible transitions from one state to another. For example the state diagram for the encoder shown in Fig. 8-2-2 is illustrated in Fig. 8-2-6. This diagram shows that the possible transitions are

$$a \xrightarrow{0} a,\quad a \xrightarrow{1} c,\quad b \xrightarrow{0} a,\quad b \xrightarrow{1} c,\quad c \xrightarrow{0} b,\quad c \xrightarrow{1} d,\quad d \xrightarrow{0} b,\quad d \xrightarrow{1} d,$$

where $\alpha \xrightarrow{1} \beta$ denotes the transition from state α to β when the input bit is a 1. The three bits shown next to each branch in the state diagram represent the output bits. A dotted line in the graph indicates that the input bit is a 1, while the solid line indicates that the input bit is a 0.

Example 8-2-2

Let us consider the $k = 2$, rate 2/3 convolutional code described in Example 8-2-1 and shown in Fig. 8-2-3. The first two input bits may be 00, 01, 10, or 11. The corresponding output bits are 000, 010, 111, 101. When the next pair of input bits enter the encoder, the first pair is shifted to the second stage. The corresponding output bits depend on the pair of bits shifted into the second stage and the new pair of input bits. Hence, the tree diagram for this code, shown in Fig. 8-2-7, has four branches per node, corresponding to the four possible pairs of input symbols. Since the constraint length of the code is $K = 2$, the tree begins to repeat after the second stage. As illustrated in Fig. 8-2-7, all the branches emanating from nodes labeled a (state a) yield identical outputs. By merging the nodes having identical labels, we obtain the trellis, which is shown in Fig. 8-2-8. Finally, the state diagram for this code is shown in Fig. 8-2-9.

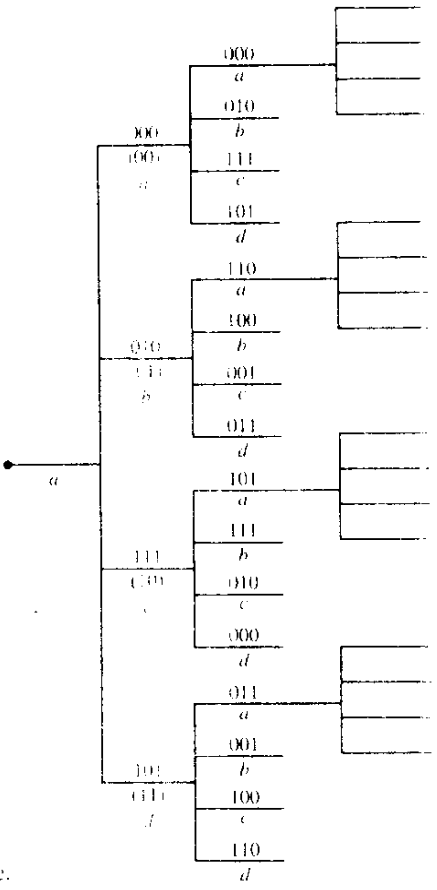


FIGURE 8-2-7 Tree diagram for $K = 2$, $k = 2$, $n = 3$ convolutional code.

To generalize, we state that a rate k/n, constraint length K, convolutional code is characterized by 2^k branches emanating from each node of the tree diagram. The trellis and the state diagrams each have $2^{k(K-1)}$ possible states. There are 2^k branches entering each state and 2^k branches leaving each state (in the trellis and tree, this is true after the initial transient).

The three types of diagrams described above are also used to represent nonbinary convolutional codes. When the number of symbols in the code alphabet is $q = 2^k$, $k > 1$, the resulting nonbinary code may also be represented as an equivalent binary code. The following example considers a convolutional code of this type.

Example 8-2-3

Let us consider the convolutional code generated by the encoder shown in Fig. 8-2-10. This code may be described as a binary convolutional code with parameters $K = 2$, $k = 2$, $n = 4$, $R_c = 1/2$, and having the generators

$$\mathbf{g}_1 = [1010], \quad \mathbf{g}_2 = [0101], \quad \mathbf{g}_3 = [1110], \quad \mathbf{g}_4 = [1001]$$

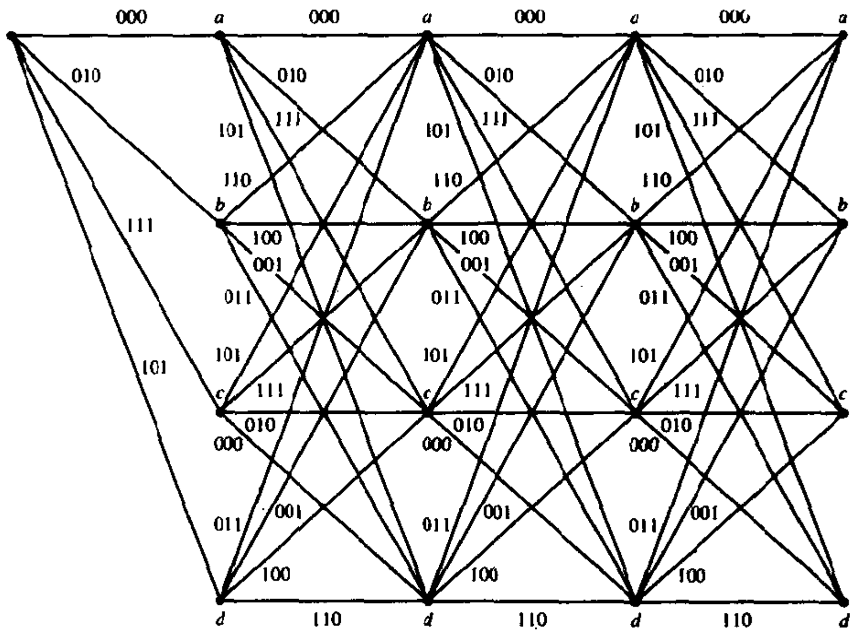
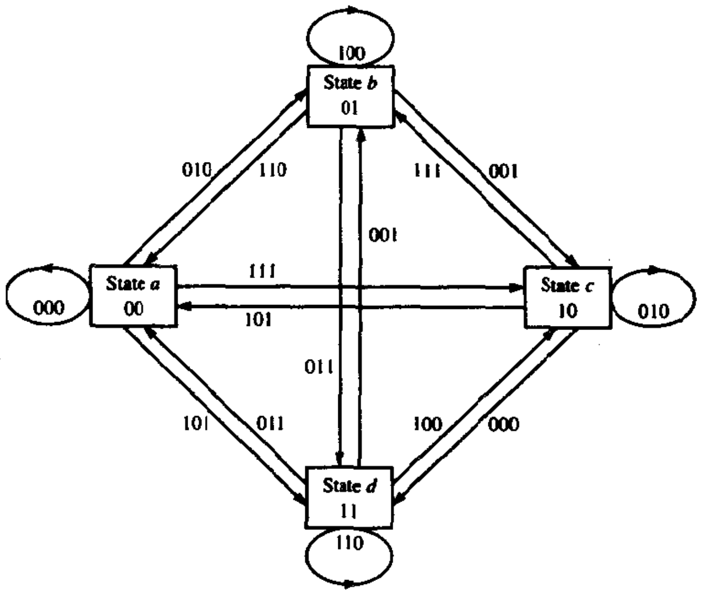


FIGURE 8-2-8 Trellis diagram for $K = 2$, $k = 2$, $n = 3$ convolutional code.

FIGURE 8-2-9 State diagram for $K = 2$, $k = 2$, $n = 3$ convolutional code.



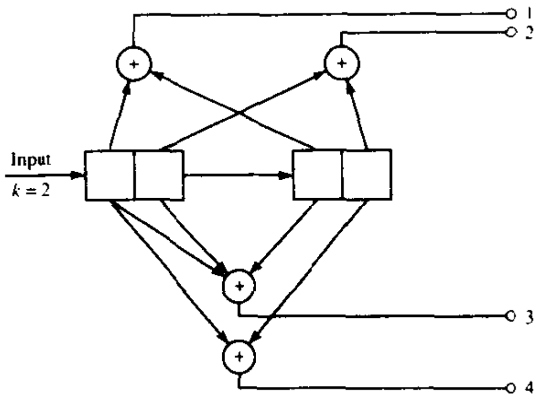


FIGURE 8-2-10 $K=2$, $k=2$, $n=4$ convolutional encoder.

Except for the difference in rate, this code is similar in form to the rate 2/3, $k=2$ convolutional code considered in Example 8-2-1.

Alternatively, the code generated by the encoder in Fig. 8-2-10 may be described as a nonbinary ($q=4$) code with one quaternary symbol as an input and two quaternary symbols as an output. In fact, if the output of the encoder is treated by the modulator and demodulator as q-ary ($q=4$) symbols that are transmitted over the channel by means of some M-ary ($M=4$) modulation technique, the code is appropriately viewed as nonbinary.

In any case, the tree, the trellis, and the state diagrams are independent of how we view the code. That is, this particular code is characterized by a tree with four branches emanating from each node, or a trellis with four possible states and four branches entering and leaving each state or, equivalently, by a state diagram having the same parameters as the trellis.

8-2-1 The Transfer Function of a Convolutional Code

The distance properties and the error rate performance of a convolutional code can be obtained from its state diagram. Since a convolutional code is linear, the set of Hamming distances of the code sequences generated up to some stage in the tree, from the all-zero code sequence, is the same as the set of distances of the code sequences with respect to any other code sequence. Consequently, we assume *without loss of generality that the all-zero code sequence is the input to* the encoder.

The state diagram shown in Fig. 8-2-6 will be used to demonstrate the method for obtaining the distance properties of a convolutional code. First, we label the branches of the state diagram as either $D^0=1$, D^1, D^2, or D^3, where the exponent of D denotes the Hamming distance of the sequence of output bits corresponding to each branch from the sequence of output bits corresponding to the all-zero branch. The self-loop at node a can be eliminated, since it contributes nothing to the distance properties of a code sequence relative to

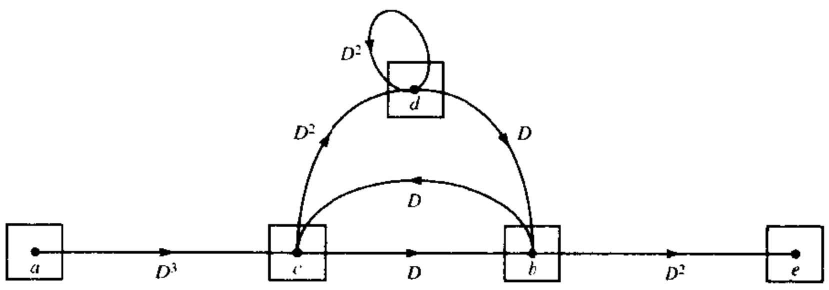


FIGURE 8-2-11 State diagram for rate 1/3, $K = 3$ convolutional code.

the all-zero code sequence. Furthermore, node a is split into two nodes, one of which represents the input and the other the output of the state diagram. Figure 8-2-11 illustrates the resulting diagram. We use this diagram, which now consists of five nodes because node a was split into two, to write the four state equations

$$\begin{aligned} X_c &= D^3X_a + DX_b \\ X_b &= DX_c + DX_d \\ X_d &= D^2X_c + D^2X_d \\ X_e &= D^2X_b \end{aligned} \tag{8-2-1}$$

The transfer function for the code is defined as $T(D) = X_e/X_a$. By solving the state equations given above, we obtain

$$\begin{aligned} T(D) &= \frac{D^6}{1-2D^2} \\ &= D^6 + 2D^8 + 4D^{10} + 8D^{12} + \dots \\ &= \sum_{d=6}^{\infty} a_d D^d \end{aligned} \tag{8-2-2}$$

where, by definition,

$$a_d = \begin{cases} 2^{(d-6)/2} & (\text{even } d) \\ 0 & (\text{odd } d) \end{cases} \tag{8-2-3}$$

The transfer function for this code indicates that there is a single path of Hamming distance $d = 6$ from the all-zero path that merges with the all-zero path at a given node. From the state diagram shown in Fig. 8-2-6 or the trellis diagram shown in Fig. 8-2-5, it is observed that the $d = 6$ path is $acbe$. There is no other path from node a to node e having a distance $d = 6$. The second term in (8-2-2) indicates that there are two paths from node a to node e having a

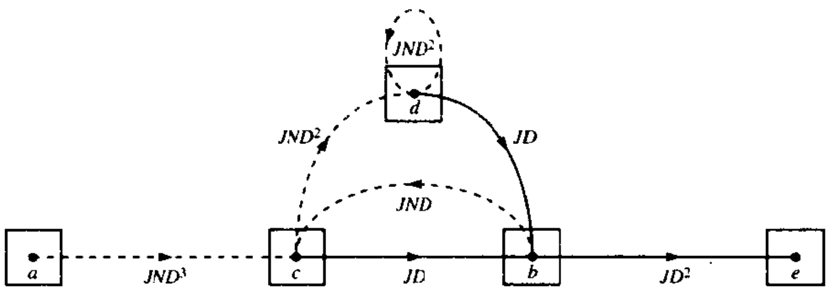


FIGURE 8-2-12 State diagram for rate 1/3, $K = 3$ convolutional code.

distance $d = 8$. Again, from the state diagram or the trellis, we observe that these paths are *acdbe* and *acbcbe*. The third term in (8-2-2) indicates that there are four paths of distance $d = 10$, and so forth. Thus the transfer function gives us the distance properties of the convolutional code. The minimum distance of the code is called the *minimum free distance* and denoted by d_{free}. In our example, $d_{\text{free}} = 6$.

The transfer function can be used to provide more detailed information than just the distance of the various paths. Suppose we introduce a factor N into all branch transitions caused by the input bit 1. Thus, as each branch is traversed, the cumulative exponent on N increases by one only if that branch transition is due to an input bit 1. Furthermore, we introduce a factor of J into each branch of the state diagram so that the exponent of J will serve as a counting variable to indicate the number of branches in any given path from node a to node e. For the rate 1/3 convolutional code in our example, the state diagram that incorporates the additional factors of J and N is shown in Fig. 8-2-12.

The state equations for the state diagram shown in Fig. 8-2-12 are

$$\begin{aligned} X_c &= JND^3X_a + JNDX_b \\ X_b &= JDX_c + JDX_d \\ X_d &= JND^2X_c + JND^2X_d \\ X_e &= JD^2X_b \end{aligned} \tag{8-2-4}$$

Upon solving these equations for the ratio X_e/X_a, we obtain the transfer function

$$\begin{aligned} T(D, N, J) &= \frac{J^3ND^6}{1 - JND^2(1 + J)} \\ &= J^3ND^6 + J^4N^2D^8 + J^5N^2D^8 + J^5N^3D^{10} \\ &\quad + 2J^6N^3D^{10} + J^7N^3D^{10} + \dots \end{aligned} \tag{8-2-5}$$

This form for the transfer functions gives the properties of all the paths in the convolutional code. That is, the first term in the expansion of $T(D, N, J)$ indicates that the distance $d = 6$ path is of length 3 and of the three information bits, one is a 1. The second and third terms in the expansion of $T(D, N, J)$ indicate that of the two $d = 8$ terms, one is of length 4 and the second has length 5. Two of the four information bits in the path having length 4 and two of the five information bits in the path having length 5 are 1s. Thus, the exponent of the factor J indicates the length of the path that merges with the all-zero path for the first time, the exponent of the factor N indicates the number of 1s in the information sequence for that path, and the exponent of D indicates the distance of the sequence of encoded bits for that path from the all-zero sequence.

The factor J is particularly important if we are transmitting a sequence of finite duration, say m bits. In such a case, the convolutional code is truncated after m nodes or m branches. This implies that the transfer function for the truncated code is obtained by truncating $T(D, N, J)$ at the term J^m. On the other hand, if we are transmitting an extremely long sequence, i.e., essentially an infinite-length sequence, we may wish to suppress the dependence of $T(D, N, J)$ on the parameter J. This is easily accomplished by setting $J = 1$. Hence, for the example given above, we have

$$\begin{aligned} T(D, N, 1) = T(D, N) &= \frac{ND^6}{1 - 2ND^2} \\ &= ND^6 + 2N^2D^8 + 4N^3D^{10} + \dots \\ &= \sum_{d=6}^{\infty} a_d N^{(d-4)/2} D^d \end{aligned} \tag{8-2-6}$$

where the coefficients $\{a_d\}$ are defined by (8-2-3).

The procedure outlined above for determining the transfer function of a binary convolutional code is easily extended to nonbinary codes. In the following example, we determine the transfer function of the nonbinary convolutional code previously introduced in Example 8-2-3.

Example 8-2-4

The convolutional code shown in Fig. 8-2-10 has the parameters $K = 2$, $k = 2$, $n = 4$. In this example, we have a choice of how we label distances and count errors, depending on whether we treat the code as binary or nonbinary. Suppose we treat the code as nonbinary. Thus, the input to the encoder and the output are treated as quaternary symbols. In particular, if we treat the input and output as quaternary symbols 00, 01, 10, and 11, the distance measured in symbols between the sequences 0111 and 0000 is 2. Furthermore, suppose that an input symbol 00 is decoded as the symbol 11; then we have made one symbol error. This convention applied to the

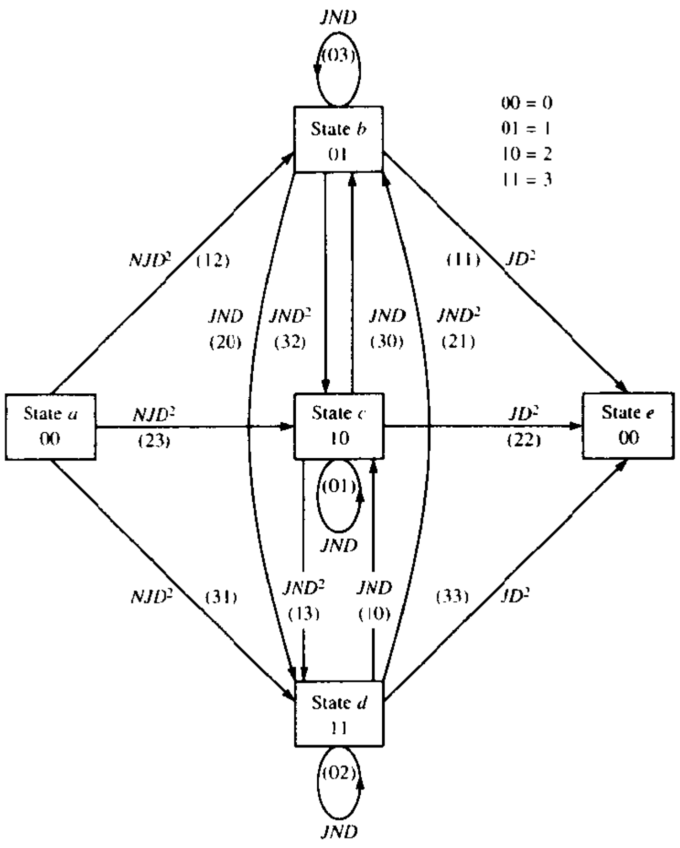


FIGURE 8-2-13 State diagram for $K = 2$, $k = 2$, rate 1/2 nonbinary code.

convolutional code shown in Fig. 8-2-10 results in the state diagram illustrated in Fig. 8-2-13, from which we obtain the state equations

$$
\begin{aligned}
X_b &= NJD^2X_a + NJDX_b + NJDX_c + NJD^2X_d \\
X_c &= NJD^2X_a + NJD^2X_b + NJDX_c + NJDX_d \\
X_d &= NJD^2X_a + NJDX_b + NJD^2X_c + NJDX_d \\
X_e &= JD^2(X_b + X_c + X_d)
\end{aligned}
\tag{8-2-7}
$$

Solution of these equations leads to the transfer function

$$T(D, N, J) = \frac{3NJ^2D^4}{1 - 2NJD - NJD^2} \tag{8-2-8}$$

This expression for the transfer function is particularly appropriate when the quaternary symbols at the output of the encoder are mapped into a

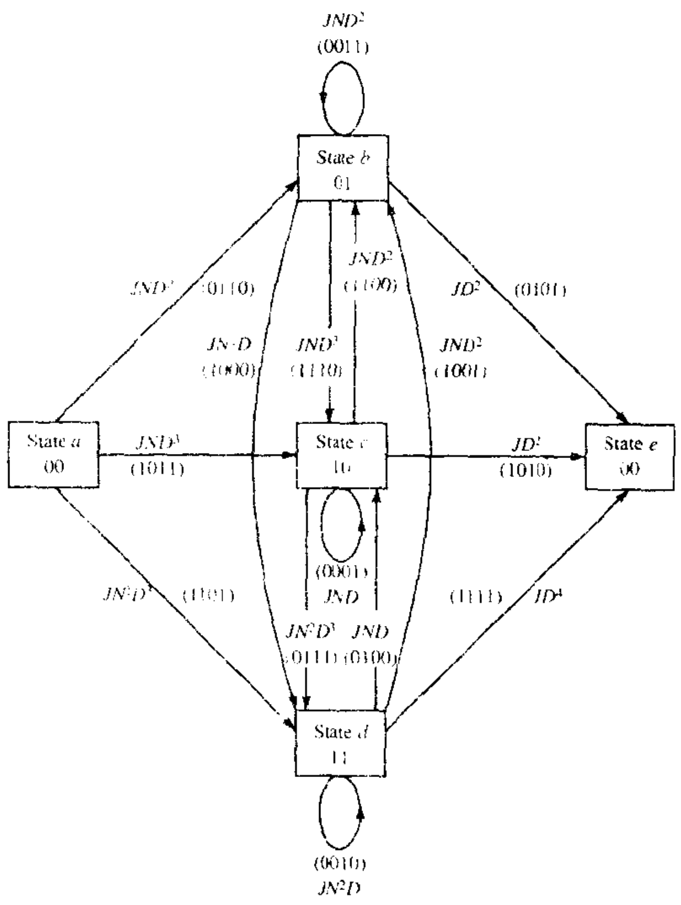


FIGURE 8-2-14 State diagram for $K = 2$, $k = 2$, rate 1/2 convolutional code with output treated as a binary sequence.

corresponding set of quaternary waveforms $s_m(t)$, $m = 1, 2, 3, 4$, e.g., four orthogonal waveforms. Thus, there is a one-to-one correspondence between code symbols and signal waveforms.

Alternatively, for example, the output of the encoder may be transmitted as a sequence of binary digits by means of binary PSK. In such a case, it is appropriate to measure distance in terms of bits. When this convention is employed, the state diagram is labeled as shown in Fig. 8-2-14. Solution of the state equations obtained from this state diagram yields a transfer function that is different from the one given in (8-2-8).

Some convolutional codes exhibit a characteristic behavior that is called *catastrophic error propagation*. When a code that has this characteristic is used on a binary symmetric channel, it is possible for a finite number of channel errors to cause an infinite number of decoding errors. Such a code can be identified from its state diagram. It will contain a zero-distance path (a path with multiplier $D^0 = 1$) from some nonzero state back to the same state. This means that one can loop around this zero-distance path an infinite number of times without increasing the distance relative to the all-zero path. But, if this self-loop corresponds to the transmission of a 1, the decoder will make an infinite number of errors. Since such codes are easily recognized, they are easily avoided in practice.

8-2-2 Optimum Decoding of Convolutional Codes—The Viterbi Algorithm

In the decoding of a block code for a memoryless channel, we computed the distances (Hamming distance for hard-decision decoding and euclidean distance for soft-decision decoding) between the received code word and the 2^k possible transmitted code words. Then we selected the code word that was closest in distance to the received code word. This decision rule, which requires the computation of 2^k *metrics*, is optimum in the sense that it results in a minimum probability of error for the binary symmetric channel with $p < \frac{1}{2}$ and the additive white gaussian noise channel.

Unlike a block code, which has a fixed length n, a convolutional encoder is basically a finite-state machine. Hence the optimum decoder is a maximum-likelihood sequence estimator (MLSE) of the type described in Section 5-1-4 for signals with memory, such as NRZI and CPM. Therefore, optimum decoding of a convolutional code involves a search through the trellis for the most probable sequence. Depending on whether the detector following the demodulator performs hard or soft decisions, the corresponding metric in the trellis search may be either a Hamming metric or a euclidean metric, respectively. We elaborate below, using the trellis in Fig. 8-2-5 for the convolutional code shown in Fig. 8-2-2.

Consider the two paths in the trellis that begin at the initial state a and remerge at state a after three state transitions (three branches), corresponding to the two information sequences 000 and 100 and the transmitted sequences 000 000 000 and 111 001 011, respectively. We denote the transmitted bits by $\{c_{jm},\ j = 1, 2, 3;\ m = 1, 2, 3\}$, where the index j indicates the jth branch and the index m the mth bit in that branch. Correspondingly, we define $\{r_{jm},\ j = 1, 2, 3;\ m = 1, 2, 3\}$ as the output of the demodulator. If the detector performs hard-decision decoding, its output for each transmitted bit is either 0 or 1. On the other hand, if soft-decision decoding is employed and the coded sequence is transmitted by binary coherent PSK, the input to the decoder is

$$r_{jm} = \sqrt{\mathscr{E}_c}(2c_{jm} - 1) + n_{jm} \qquad (8\text{-}2\text{-}9)$$

where n_{jm} represents the additive noise and $\mathscr{E}_c$ is the transmitted signal energy for each code bit.

A metric is defined for the jth branch of the ith path through the trellis as the logarithm of the joint probability of the sequence $\{r_{jm},\ m = 1, 2, 3\}$

conditioned on the transmitted sequence $\{c_{jm}^{(i)}, m = 1, 2, 3\}$ for the ith path. That is,

$$\mu_j^{(i)} = \log P(\mathbf{Y}_j \mid \mathbf{C}_j^{(i)}), \qquad j = 1, 2, 3, \ldots \tag{8-2-10}$$

Furthermore, a metric for the ith path consisting of B branches through the trellis is defined as

$$PM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)} \tag{8-2-11}$$

The criterion for deciding between two paths through the trellis is to select the one having the larger metric. This rule maximizes the probability of a correct decision or, equivalently, it minimizes the probability of error for the sequence of information bits. For example, suppose that hard-decision decoding is performed by the demodulator, yielding the received sequence {101 000 100}. Let $i = 0$ denote the three-branch all-zero path and $i = 1$ the second three-branch path that begins in the initial state a and remerges with the all-zero path at state a after three transitions. The metrics for these two paths are

$$\begin{aligned} PM^{(0)} &= 6\log(1-p) + 3\log p \\ PM^{(1)} &= 4\log(1-p) + 5\log p \end{aligned} \tag{8-2-12}$$

where p is the probability of a bit error. Assuming that $p < \frac{1}{2}$, we find that the metric $PM^{(0)}$ is larger than the metric $PM^{(1)}$. This result is consistent with the observation that the all-zero path is at Hamming distance $d = 3$ from the received sequence, while the $i = 1$ path is at Hamming distance $d = 5$ from the received path. Thus, the Hamming distance is an equivalent metric for hard-decision decoding.

Similarly, suppose that soft-decision decoding is employed and the channel adds white gaussian noise to the signal. Then the demodulator output is described statistically by the probability density function

$$p(r_{jm} \mid c_{jm}^{(i)}) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left\{ -\frac{[r_{jm} - \sqrt{\mathscr{E}_c}(2c_{jm}^{(i)} - 1)]^2}{2\sigma^2} \right\} \tag{8-2-13}$$

where $\sigma^2 = \frac{1}{2}N_0$ is the variance of the additive gaussian noise. If we neglect the terms that are common to all branch metrics, the branch metric for the jth branch of the ith path may be expressed as

$$\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1) \tag{8-2-14}$$

where, in our example, $n = 3$. Thus the correlation metrics for the two paths under consideration are

$$\begin{aligned} CM^{(0)} &= \sum_{j=1}^{3} \sum_{m=1}^{3} r_{jm}(2c_{jm}^{(0)} - 1) \\ CM^{(1)} &= \sum_{j=1}^{3} \sum_{m=1}^{3} r_{jm}(2c_{jm}^{(1)} - 1) \end{aligned} \tag{8-2-15}$$

Having defined the branch metrics and path metrics computed by the decoder, we now consider the use of the Viterbi algorithm for optimum decoding of the convolutionally encoded information sequence. We consider the two paths described above, which merge at state a after three transitions. Note that any particular path through the trellis that stems from this node will add identical terms to the path metrics $CM^{(0)}$ and $CM^{(1)}$. Consequently, if $CM^{(0)} > CM^{(1)}$ at the merged node a after three transitions $CM^{(0)}$ will continue to be larger than $CM^{(1)}$ for any path that stems from node a. This means that the path corresponding to $CM^{(1)}$ can be discarded from further consideration. The path corresponding to the metric $CM^{(0)}$ is the *survivor*. Similarly, one of the two paths that merge at state b can be elminated on the basis of the two corresponding metrics. This procedure is repeated at state c and state d. As a result, after the first three transitions, there are four surviving paths, one terminating at each state, and a corresponding metric for each survivor. This procedure is repeated at each stage of the trellis as new signals are received in subsequent time intervals.

In general, when a binary convolutional code with $k = 1$ and constraint length K is decoded by means of the Viterbi algorithm, there are 2^{K-1} states. Hence, there are 2^{K-1} surviving paths at each stage and 2^{K-1} metrics, one for each surviving path. Furthermore, a binary convolutional code in which k bits at a time are shifted into an encoder that consists of K (k-bit) shift-register stages generates a trellis that has $2^{k(K-1)}$ states. Consequently, the decoding of such a code by means of the Viterbi algorithm requires keeping track of $2^{k(K-1)}$ surviving paths and $2^{k(K-1)}$ metrics. At each stage of the trellis, there are 2^k paths that merge at each node. Since each path that converges at a common node requires the computation of a metric, there are 2^k metrics computed for each node. Of the 2^k paths that merge at each node, only one survives, and this is the most-probable (minimum-distance) path. Thus the number of computations in decoding performed at each stage increases exponentially with k and K. The exponential increase in computational burden limits the use of the Viterbi algorithm to relatively small values of K and k.

The decoding delay in decoding a long information sequence that has been convolutionally encoded is usually too long for most practical applications. Moreover, the memory required to store the entire length of surviving sequences is large and expensive. As indicated in Section 5-1-4, a solution to this problem is to modify the Viterbi algorithm in a way which results in a fixed decoding delay without significantly affecting the optimal performance of the algorithm. Recall that the modification is to retain at any given time t only the most recent δ decoded information bits (symbols) in each surviving sequence. As each new information bit (symbol) is received, a final decision is made on the bit (symbol) received δ branches back in the trellis, by comparing the metrics in the surviving sequences and deciding in favor of the bit in the sequence having the largest metric. If δ is chosen sufficiently large, all surviving sequences will contain the identical decoded bit (symbol) δ branches back in time. That is, with high probability, all surviving sequences at time t stem from the same node at $t-\delta$. It has been found experimentally (computer simulation) that a delay $\delta \geqslant 5K$ results in a negligible degradation in the performance relative to the optimum Viterbi algorithm.

8-2-3 Probability of Error for Soft-Decision Decoding

The topic of this subsection is the error rate performance of the Viterbi algorithm on an additive white gaussian noise channel with soft-decision decoding.

In deriving the probability of error for convolutional codes, the linearity property for this class of codes is employed to simplify the derivation. That is, we assume that the all-zero sequence is transmitted and we determine the probability of error in deciding in favor of another sequence. The coded binary digits for the jth branch of the convolutional code, denoted as $\{c_{jm}, m=1,2,\ldots,n\}$ and defined in Section 8-2-2, are assumed to be transmitted by binary PSK (or four-phase PSK) and detected coherently at the demodulator. The output of the demodulator, which is the input to the Viterbi decoder, is the sequence $\{r_{jm}, m=1,2,\ldots,n; j=1,2,\ldots\}$ where r_{jm} is defined in (8-2-9).

The Viterbi soft-decision decoder forms the branch metrics defined by (8-2-14) and from these computes the path metrics

$$CM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)} = \sum_{j=1}^{B} \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1) \tag{8-2-16}$$

where i denotes any one of the competing paths at each node and B is the number of branches (information symbols) in a path. For example, the all-zero path, denoted as $i=0$, has a path metric

$$\begin{aligned} CM^{(0)} &= \sum_{j=1}^{B} \sum_{m=1}^{n} (-\sqrt{\mathscr{E}_c} + n_{jm})(-1) \\ &= \sqrt{\mathscr{E}_c}\, Bn + \sum_{j=1}^{B} \sum_{m=1}^{n} n_{jm} \end{aligned} \tag{8-2-17}$$

Since the convolutional code does not necessarily have a fixed length, we derive its performance from the probability of error for sequences that merge with the all-zero sequence for the first time at a given node in the trellis. In particular, we define the first-event error probability as the probability that another path that merges with the all-zero path at node B has a metric that exceeds the metric of the all-zero path for the first time. Suppose the incorrect path, call it $i=1$, that merges with the all-zero path differs from the all-zero path in d bits, i.e., there are d 1s in the path $i=1$ and the rest are 0s. The probability of error in the pairwise comparison of the metrics $CM^{(0)}$ and $CM^{(1)}$ is

$$\begin{aligned} P_2(d) &= P(CM^{(1)} \geqslant CM^{(0)}) = P(CM^{(1)} - CM^{(0)} \geqslant 0) \\ P_2(d) &= P\left[2\sum_{j=1}^{B} \sum_{m=1}^{n} r_{jm}(c_{jm}^{(1)} - c_{jm}^{(0)}) \geqslant 0\right] \end{aligned} \tag{8-2-18}$$

Since the coded bits in the two paths are identical except in the d positions, (8-2-18) can be written in the simpler form

$$P_2(d) = P\left(\sum_{l=1}^{d} r_l' \geq 0\right) \tag{8-2-19}$$

where the index l runs over the set of d bits in which the two paths differ and the set $\{r_l'\}$ represents the input to the decoder for these d bits.

The $\{r_l'\}$ are independent and identically distributed gaussian random variables with mean $-\sqrt{\mathscr{E}_c}$ and variance $\frac{1}{2}N_0$. Consequently the probability of error in the pairwise comparison of these two paths that differ in d bits is

$$\begin{aligned} P_2(d) &= Q\left(\sqrt{\frac{2\mathscr{E}_c}{N_0}d}\right) \\ &= Q(\sqrt{2\gamma_b R_c d}) \end{aligned} \tag{8-2-20}$$

where $\gamma_b = \mathscr{E}_b/N_0$ is the received SNR per bit and R_c is the code rate.

Although we have derived the first-event error probability for a path of distance d from the all-zero path, there are many possible paths with different distances that merge with the all-zero path at a given node B. In fact, the transfer function $T(D)$ provides a complete description of all the possible paths that merge with the all-zero path at node B and their distances. Thus we can sum the error probability in (8-2-20) over all possible path distances. Upon performing this summation, we obtain an upper bound on the first-event error probability in the form

$$\begin{aligned} P_e &\leq \sum_{d=d_{\text{free}}}^{\infty} a_d P_2(d) \\ &\leq \sum_{d=d_{\text{free}}}^{\infty} a_d Q(\sqrt{2\gamma_b R_c d}) \end{aligned} \tag{8-2-21}$$

where a_d denotes the number of paths of distance d from the all-zero path that merge with the all-zero path for the first time.

There are two reasons why (8-2-21) is an upper bound on the first-event error probability. One is that the events that result in the error probabilities $\{P_2(d)\}$ are not disjoint. This can be seen from observation of the trellis. Second, by summing over all possible $d \geq d_{\text{free}}$, we have implicitly assumed that the convolutional code has infinite length. If the code is truncated periodically after B nodes, the upper bound in (8-2-21) can be improved by summing the error events for $d_{\text{free}} \leq d \leq B$. This refinement has some merit in determining the performance of short convolutional codes, but the effect on performance is negligible when B is large.

The upper bound in (8-2-21) can be expressed in a slightly different form if the Q function is upper-bounded by an exponential. That is,

$$Q(\sqrt{2\gamma_b R_c d}) \leq e^{-\gamma_b R_c d} = D^d\big|_{D=e^{-\gamma_b R_c}} \tag{8-2-22}$$

If we use (8-2-22) in (8-2-21), the upper bound on the first-event error probability can be expressed as

$$P_e < T(D)\big|_{D=e^{-\gamma_b R_c}} \tag{8-2-23}$$

Although the first-event error probability provides a measure of the performance of a convolutional code, a more useful measure of performance is the bit error probability. This probability can be upper-bounded by the procedure used in bounding the first-event error probability. Specifically, we know that when an incorrect path is selected, the information bits in which the selected path differs from the correct path will be decoded incorrectly. We also know that the exponents in the factor N contained in the transfer function $T(D, N)$ indicate the number of information bit errors (number of 1s) in selecting an incorrect path that merges with the all-zero path at some node B. If we multiply the pairwise error probability $P_2(d)$ by the number of incorrectly decoded information bits for the incorrect path at the node where they merge, we obtain the bit error rate for that path. The average bit error probability is upper-bounded by multiplying each pairwise error probability $P_2(d)$ by the corresponding number of incorrectly decoded information bits, for each possible incorrect path that merges with the correct path at the Bth node, and summing over all d.

The appropriate multiplication factors corresponding to the number of information bit errors for each incorrectly selected path may be obtained by differentiating $T(D, N)$ with respect to N. In general, $T(D, N)$ can be expressed as

$$T(D, N) = \sum_{d=d_{\text{free}}}^{\infty} a_d D^d N^{f(d)} \tag{8-2-24}$$

where $f(d)$ denotes the exponent of N as a function of d. Taking the derivative of $T(D, N)$ with respect to N and setting $N = 1$, we obtain

$$\begin{aligned}\frac{dT(D, N)}{dN}\bigg|_{N=1} &= \sum_{d=d_{\text{free}}}^{\infty} a_d f(d) D^d \\ &= \sum_{d=d_{\text{free}}}^{\infty} \beta_d D^d \end{aligned} \tag{8-2-25}$$

where $\beta_d = a_d f(d)$. Thus the bit error probability for $k = 1$ is upper-bounded by

$$\begin{aligned} P_b &< \sum_{d=d_{\text{free}}}^{\infty} \beta_d P_2(d) \\ &< \sum_{d=d_{\text{free}}}^{\infty} \beta_d Q(\sqrt{2\gamma_b R_c d}) \end{aligned} \tag{8-2-26}$$

If the Q function is upper-bounded by an exponential as indicated in (8-2-22) then (8-2-26) can be expressed in the simple form

$$\begin{aligned} P_b &< \sum_{d=d_{\text{free}}}^{\infty} \beta_d D^d \Big|_{D=e^{-\gamma_b R_c}} \\ &< \frac{dT(D,N)}{dN}\Big|_{N=1,\, D=e^{-\gamma_b R_c}} \end{aligned} \tag{8-2-27}$$

If $k > 1$, the equivalent bit error probability is obtained by dividing (8-2-26) and (8-2-27) by k.

The expressions for the probability of error given above are based on the assumption that the code bits are transmitted by binary coherent PSK. The results also hold for four-phase coherent PSK, since this modulation/demodulation technique is equivalent to two independent (phase-quadrature) binary PSK systems. Other modulation and demodulation techniques, such as coherent and noncoherent binary FSK, can be accommodated by recomputing the pairwise error probability $P_2(d)$. That is, a change in the modulation and demodulation technique used to transmit the coded information sequence affects only the computation of $P_2(d)$. Otherwise, the derivation for P_b remains the same.

Although the above derivation of the error probability for Viterbi decoding of a convolutional code applies to binary convolutional codes, it is relatively easy to generalize it to nonbinary convolutional codes in which each nonbinary symbol is mapped into a distinct waveform. In particular, the coefficients $\{\beta_d\}$ in the expansion of the derivative of $T(D, N)$, given in (8-2-25), represent the number of symbol errors in two paths separated in distance (measured in terms of symbols) by d symbols. Again, we denote the probability of error in a pairwise comparison of two paths that are separated in distance by d as $P_2(d)$. Then the symbol error probability, for a k-bit symbol, is upper-bounded by

$$P_M \leq \sum_{d=d_{\text{free}}}^{\infty} \beta_d P_2(d)$$

The symbol error probability can be converted into an equivalent bit error probability. For example, if 2^k orthogonal waveforms are used to transmit the k-bit symbols, the equivalent bit error probability is P_M multiplied by a factor $2^{k-1}/(2^k - 1)$, as shown in Chapter 5.

8-2-4 Probability of Error for Hard-Decision Decoding

We now consider the performance achieved by the Viterbi decoding algorithm on a binary symmetric channel. For hard-decision decoding of the convolutional code, the metrics in the Viterbi algorithm are the Hamming distances between the received sequence and the $2^{k(K-1)}$ surviving sequences at each node of the trellis.

As in our treatment of soft-decision decoding, we begin by determining the first-event error probability. The all-zero path is assumed to be transmitted. Suppose that the path being compared with the all-zero path at some node B has distance d from the all-zero path. If d is odd, the all-zero path will be correctly selected if the number of errors in the received sequence is less than $\frac{1}{2}(d+1)$; otherwise, the incorrect path will be selected. Consequently, the probability of selecting the incorrect path is

$$P_2(d) = \sum_{k=(d+1)/2}^{d} \binom{d}{k} p^k (1-p)^{d-k} \tag{8-2-28}$$

where p is the probability of a bit error for the binary symmetric channel. If d is even, the incorrect path is selected when the number of errors exceeds $\frac{1}{2}d$. If the number of errors equals $\frac{1}{2}d$, there is a tie between the metrics in the two paths, which may be resolved by randomly selecting one of the paths; thus, an error occurs half the time. Consequently, the probability of selecting the incorrect path is

$$P_2(d) = \sum_{k=d/2+1}^{d} \binom{d}{k} p^k (1-p)^{d-k} + \frac{1}{2}\binom{d}{\frac{1}{2}d} p^{d/2}(1-p)^{d/2} \tag{8-2-29}$$

As indicated in Section 8-2-3, there are many possible paths with different distances that merge with the all-zero path at a given node. Therefore, there is no simple exact expression for the first-event error probability. However, we can overbound this error probability by the sum of the pairwise error probabilities $P_2(d)$ over all possible paths that merge with the all-zero path at the given node. Thus, we obtain the union bound

$$P_e < \sum_{d=d_{\text{free}}}^{\infty} a_d P_2(d) \tag{8-2-30}$$

where the coefficients $\{a_d\}$ represent the number of paths corresponding to the set of distances $\{d\}$. These coefficients are the coefficients in the expansion of the transfer function $T(D)$ or $T(D, N)$.

Instead of using the expressions for $P_2(d)$ given in (8-2-28) and (8-2-29), we can use the upper bound

$$P_2(d) < [4p(1-p)]^{d/2} \tag{8-2-31}$$

which was given in Section 8-1-5. Use of this bound in (8-2-30) yields a looser upper bound on the first-event error probability, in the form

$$\begin{aligned} P_e &< \sum_{d=d_{\text{free}}}^{\infty} a_d [4p(1-p)]^{d/2} \\ &< T(D)\big|_{D=\sqrt{4p(1-p)}} \end{aligned} \tag{8-2-32}$$

Let us now determine the probability of a bit error. As in the case of soft-decision decoding, we make use of the fact that the exponents in the factors of N that appear in the transfer function $T(D, N)$ indicate the number of nonzero information bits that are in error when an incorrect path is selected over the all-zero path. By differentiating $T(D, N)$ with respect to N and setting $N = 1$, the exponents of N become multiplication factors of the corresponding error-event probabilities $P_2(d)$. Thus, we obtain the expression for the upper bound on the bit error probability, in the form

$$P_b < \sum_{d=d_{\text{free}}}^{\infty} \beta_d P_2(d) \tag{8-2-33}$$

where the $\{\beta_d\}$ are the coefficients in the expansion of the derivative of $T(D, N)$, evaluated at $N = 1$. For $P_2(d)$, we may use either the expressions given in (8-2-28) and (8-2-29) or the upper bound in (8-2-31). If the latter is used, the upper bound on P_b can be expressed as

$$P_b < \left.\frac{dT(D, N)}{dN}\right|_{N=1,\, D=\sqrt{4p(1-p)}} \tag{8-2-34}$$

When $k > 1$, the results given in (8-2-33) and (8-2-34) for P_b should be divided by k.

A comparison of the error probability for the rate 1/3, $K = 3$ convolutional code with soft-decision decoding and hard-decision decoding is made in Fig. 8-2-15. Note that the Chernoff upper bound given by (8-2-34) is less than 1 dB above the tighter upper bound given by (8-2-33) in conjunction with (8-2-28) and (8-2-29). The advantage of the Chernoff bound is its computational

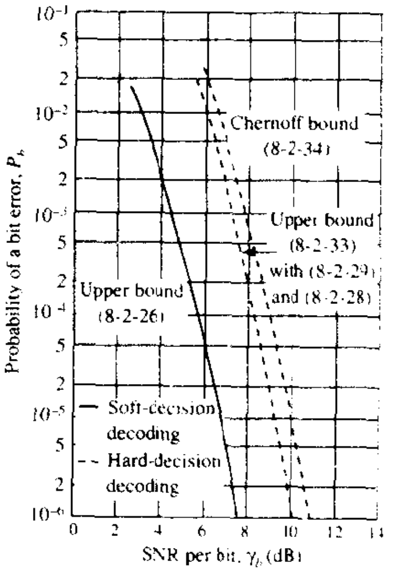


FIGURE 8-2-15 Comparison of soft-decision and hard-decision decoding for $K = 3$, $k = 1$, $n = 3$ convolutional code.

simplicity. In comparing the performance between soft-decision and hard-decision decoding, note that the difference obtained from the upper bounds is approximately 2.5 dB for $10^{-6} \leq P_b \leq 10^{-2}$.

Finally, we should mention that the ensemble average error rate performance of a convolutional code on a discrete memoryless channel, just as in the case of a block code, can be expressed in terms of the cutoff rate parameter R_0 as (for the derivation, see Viterbi and Omura, 1979).

$$\bar{P}_b < \frac{(q-1)q^{-KR_0/R_c}}{[1 - q^{-(R_0-R_c)/R_c}]^2}, \qquad R_c \leq R_0$$

where q is the number of channel input symbols, K is the constraint length of the code, R_c is the code rate, and R_0 is the cutoff rate defined in Sections 7-2 and 8-1. Therefore, conclusions reached by computing R_0 for various channel conditions apply to both block codes and convolutional codes.

8-2-5 Distance Properties of Binary Convolutional Codes

In this subsection, we shall tabulate the minimum free distance and the generators for several binary, short-constraint-length convolutional codes for several code rates. These binary codes are optimal in the sense that, for a given rate and a given constraint length, they have the largest possible d_{free}. The generators and the corresponding values of d_{free} tabulated below have been obtained by Odenwalder (1970), Larsen (1973), Paaske (1974), and Daut *et al.* (1982) using computer search methods.

Heller (1968) has derived a relatively simple upper bound on the minimum free distance of a rate $1/n$ convolutional code. It is given by

$$d_{\text{free}} \leq \min_{l \geq 1} \left\lfloor \frac{2^{l-1}}{2^l - 1}(K + l - 1)n \right\rfloor \tag{8-2-35}$$

where $\lfloor x \rfloor$ denotes the largest integer contained in x. For purposes of comparison, this upper bound is also given in the tables for the rate $1/n$ codes. For rate k/n convolutional codes, Daut *et al.* (1982) has given a modification to Heller's bound. The values obtained from this upper bound for k/n codes are also tabulated.

Tables 8-2-1 to 8-2-7 list the parameter of rate $1/n$ convolutional codes for $n = 2, 3, \ldots, 8$. Tables 8-2-8 to 8-2-11 list the parameters of several rate k/n convolutional codes for $k \leq 4$ and $n \leq 8$.

8-2-6 Nonbinary Dual-*k* Codes and Concatenated Codes

Our treatment of convolutional codes thus far has been focused primarily on binary codes. Binary codes are particularly suitable for channels in which binary or quaternary PSK modulation and coherent demodulation is possible.

TABLE 8-2-1 RATE 1/2 MAXIMUM FREE DISTANCE CODE

Constraint length K	Generators in octal		d_{free}	Upper bound on d_{free}
3	5	7	5	5
4	15	17	6	6
5	23	35	7	8
6	53	75	8	8
7	133	171	10	10
8	247	371	10	11
9	561	753	12	12
10	1,167	1,545	12	13
11	2,335	3,661	14	14
12	4,335	5,723	15	15
13	10,533	17,661	16	16
14	21,675	27,123	16	17

Source: Odenwalder (1970) and Larsen (1973).

However, there are many applications in which PSK modulation and coherent demodulation is not suitable or possible. In such cases, other modulation techniques, e.g., M-ary FSK, are employed in conjunction with noncoherent demodulation. Nonbinary codes are particularly matched to M-ary signals that are demodulated noncoherently.

In this subsection, we describe a class of nonbinary convolutional codes, called *dual-k codes,* that are easily decoded by means of the Viterbi algorithm using either soft-decision or hard-decision decoding. They are also suitable either as an outer code or as an inner code in a concatenated code, as will also be described below.

TABLE 8-2-2 RATE 1/3 MAXIMUM FREE DISTANCE CODES

Constraint length K	Generators in octal			d_{free}	Upper bound on d_{free}
3	5	7	7	8	8
4	13	15	17	10	10
5	25	33	37	12	12
6	47	53	75	13	13
7	133	145	175	15	15
8	225	331	367	16	16
9	557	663	711	18	18
10	1,117	1,365	1,633	20	20
11	2,353	2,671	3,175	22	22
12	4,767	5,723	6,265	24	24
13	10,533	10,675	17,661	24	24
14	21,645	35,661	37,133	26	26

Sources: Odenwalder (1970) and Larsen (1973).

TABLE 8-2-3 RATE 1/4 MAXIMUM FREE DISTANCE CODES

Constraint length K	Generators in octal				d_{free}	Upper bound on d_{free}
3	5	7	7	7	10	10
4	13	15	15	17	13	15
5	25	27	33	37	16	16
6	53	67	71	75	18	18
7	135	135	147	163	20	20
8	235	275	313	357	22	22
9	463	535	733	745	24	24
10	1,117	1,365	1,633	1,653	27	27
11	2,387	2,353	2,671	3,175	29	29
12	4,767	5,723	6,265	7,455	32	32
13	11,145	12,477	15,537	16,727	33	33
14	21,113	23,175	35,527	35,537	36	36

Source: Larsen (1973).

TABLE 8-2-4 RATE 1/5 MAXIMUM FREE DISTANCE CODES

Constraint length K	Generators in octal					d_{free}	Upper bound on d_{free}
3	7	7	7	5	5	13	13
4	17	17	13	15	15	16	16
5	37	27	33	25	35	20	20
6	75	71	73	65	57	22	22
7	175	131	135	135	147	25	25
8	257	233	323	271	357	28	28

Source: Daut *et al.* (1982).

TABLE 8-2-5 RATE 1/6 MAXIMUM FREE DISTANCE CODES

Constraint length K	Generators in octal			d_{free}	Upper bound on d_{free}
3	7	7	7	16	16
	7	5	5		
4	17	17	13	20	20
	13	15	15		
5	37	35	27	24	24
	33	25	35		
6	73	75	55	27	27
	65	47	57		
7	173	151	135	30	30
	135	163	137		
8	253	375	331	34	34
	235	313	357		

Source: Daut *et al.* (1982).

TABLE 8-2-6 RATE 1/7 MAXIMUM FREE DISTANCE CODES

Constraint length K	Generators in octal				d_{free}	Upper bound on d_{free}
3	7	7	7	7	18	18
	5	5	5			
4	17	17	13	13	23	23
	13	15	15			
5	35	27	25	27	28	28
	33	35	37			
6	53	75	65	75	32	32
	47	67	57			
7	165	145	173	135	36	36
	135	147	137			
8	275	253	375	331	40	40
	235	313	357			

Source: Daut *et al.* (1982).

TABLE 8-2-7 RATE 1/8 MAXIMUM FREE DISTANCE CODES

Constraint length K	Generators in octal				d_{free}	Upper bound on d_{free}
3	7	7	5	5	21	21
	5	7	7	7		
4	17	17	13	13	26	26
	13	15	15	17		
5	37	33	25	25	32	32
	35	33	27	37		
6	57	73	51	65	36	36
	75	47	67	57		
7	153	111	165	173	40	40
	135	135	147	137		
8	275	275	253	371	45	45
	331	235	313	357		

Source: Daut *et al.* (1982).

TABLE 8-2-8 RATE 2/3 MAXIMUM FREE DISTANCE CODES

Constraint length K	Generators in octal			d_{free}	Upper bound on d_{free}
2	17	06	15	3	4
3	27	75	72	5	6
4	236	155	337	7	7

Source: Duat *et al.* (1982).

TABLE 8-2-9 RATE $k/5$ MAXIMUM FREE DISTANCE CODES

Rate	Constraint length K	Generators in octal					d_{free}	Upper bound on d_{free}
2/5	2	17	07	11	12	04	6	6
	3	27	71	52	65	57	10	10
	4	247	366	171	266	373	12	12
3/5	2	35	23	75	61	47	5	5
4/5	2	237	274	156	255	337	3	4

Source: Daut *et al.* (1982).

TABLE 8-2-10 RATE $k/7$ MAXIMUM FREE DISTANCE CODES

Rate	Constraint length K	Generators in octal				d_{free}	Upper bound on d_{free}
2/7	2	05	06	12	15	9	9
		15	13	17			
	3	33	55	72	47	14	14
		25	53	75			
	4	312	125	247	366	18	18
		171	266	373			
3/7	2	45	21	36	62	8	8
		57	43	71			
4/7	2	130	067	237	274	6	7
		156	255	337			

Source: Daut *et al.* (1982).

TABLE 8-2-11 RATES 3/4 AND 3/8 MAXIMUM FREE DISTANCE CODES

Rate	Constraint length K	Generators in octal				d_{free}	Upper bound on d_{free}
3/4	2	13	25	61	47	4	4
3/8	2	15	42	23	61	8	8
		51	36	75	47		

Source: Daut *et al.* (1982).

A dual-k rate 1/2 convolutional encoder may be represented as shown in Fig. 8-2-16. It consists of two $(K = 2)$ k-bit shift-register stages and $n = 2k$ function generators. Its output is two k-bit symbols. We note that the code considered in Example 8-2-3 is a dual-2 convolutional code.

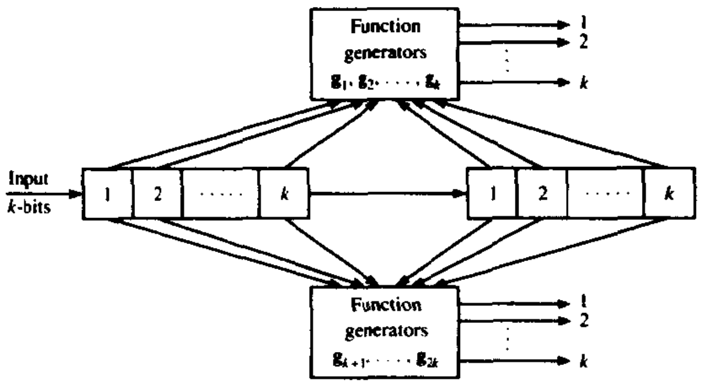

Function generators $g_1, g_2, \dots, g_k$
1
2
k
Input k-bits
1
2
k
1
2
k
Function generators $g_{k+1}, \dots, g_{2k}$
1
2
k

FIGURE 8-2-16 Encoder for rate 1/2 dual-k codes.

The $2k$ function generators for the dual-k codes have been given by Viterbi and Jacobs (1975). These may be expressed in the form

$$\begin{bmatrix} \leftarrow \mathbf{g}_1 \rightarrow \\ \leftarrow \mathbf{g}_2 \rightarrow \\ \vdots \\ \leftarrow \mathbf{g}_k \rightarrow \end{bmatrix} = \left[\begin{array}{ccccccc|cccc} 1 & 0 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 & & \dots & 0 & 1 \end{array}\right] = [\mathbf{I}_k \quad \mathbf{I}_k]$$

$$\begin{bmatrix} \leftarrow \mathbf{g}_{k+1} \rightarrow \\ \leftarrow \mathbf{g}_{k+2} \rightarrow \\ \vdots \\ \\ \leftarrow \mathbf{g}_{2k} \rightarrow \end{bmatrix} = \left[\begin{array}{cccccccc|ccccccc} 1 & 1 & 0 & 0 & & \dots & & 0 & 1 & 0 & 0 & & \dots & & 0 \\ 0 & 0 & 1 & 0 & & \dots & & 0 & 0 & 1 & 0 & & \dots & & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots & & 0 & 0 & 0 & 1 & 0 & \dots & & 0 \\ 0 & 0 & 0 & & & \dots & 0 & 1 & & & & & & & \\ 1 & 0 & 0 & & & \dots & 0 & 0 & 0 & 0 & & & \dots & 0 & 1 \end{array}\right]$$

$$= \left[\begin{array}{cccccccc|c} 1 & 1 & 0 & 0 & & \dots & & 0 & \\ 0 & 0 & 1 & 0 & & \dots & & 0 & \\ 0 & 0 & 0 & 1 & 0 & \dots & & 0 & \ddots \qquad \mathbf{I}_k \\ \hline 0 & 0 & 0 & & & \dots & 0 & 1 & \\ 1 & 0 & 0 & & & \dots & 0 & 0 & \end{array}\right] \tag{8-2-36}$$

where $\mathbf{I}_k$ denotes the $k \times k$ identity matrix.

The general form for the transfer function of a rate 1/2 dual-k code has been derived by Odenwalder (1976). It is expressed as

$$\begin{aligned} T(D, N, J) &= \frac{(2^k - 1)D^4J^2N}{1 - NJ[2D + (2^k - 3)D^2]} \\ &= \sum_{i=4}^{\infty} a_i D^i N^{f(i)} J^{h(i)} \end{aligned} \tag{8-2-37}$$

where D represents the Hamming distance for the q-ary ($q = 2^k$) symbols, the $f(i)$ exponent on N represents the number of information symbol errors that are produced in selecting a branch in the tree or trellis other than a corresponding branch on the all-zero path, and the $h(i)$ exponent on J is equal to the number of branches in a given path. Note that the minimum free distance is $d_{\text{free}} = 4$ symbols ($4k$ bits).

Lower-rate dual-k convolutional codes can be generated in a number of ways, the simplest of which is to repeat each symbol generated by the rate 1/2 code r times, where $r = 1, 2, \ldots, m$ ($r = 1$ corresponds to each symbol appearing once). If each symbol in any particular branch of the tree or trellis or state diagram is repeated r times, the effect is to increase the distance parameter from D to D^r. Consequently the transfer function for a rate $1/2r$ dual-k code is

$$T(D, N, J) = \frac{(2^k - 1)D^{4r}J^2N}{1 - NJ[2D^r + (2^k - 3)D^{2r}]} \tag{8-2-38}$$

In the transmission of long information sequences, the path length parameter J in the transfer function may be suppressed by setting $J = 1$. The resulting transfer function $T(D, N)$ may be differentiated with respect to N, and N is set to unity. This yields

$$\begin{aligned} \left.\frac{dT(D,N)}{dN}\right|_{N=1} &= \frac{(2^k - 1)D^{4r}}{[1 - 2D^r - (2^k - 3)D^{2r}]^2} \\ &= \sum_{i=4r}^{\infty} \beta_i D^i \end{aligned} \tag{8-2-39}$$

where β_i represents the number of symbol errors associated with a path having distance D^i from the all-zero path, as described previously in Section 8-2-3. The expression in (8-2-39) may be used to evaluate the error probability for dual-k codes under various channel conditions.

Performance of Dual-*k* Codes with *M*-ary Modulation Suppose that a dual-k code is used in conjunction with M-ary orthogonal signaling at the modulator, where $M = 2^k$. Each symbol from the encoder is mapped into one of the M possible orthogonal waveforms. The channel is assumed to add white gaussian noise. The demodulator consists of M matched filters.

If the decoder performs hard-decision decoding, the performance of the code is determined by the symbol error probability P_M. This error probability has been computed in Chapter 5 for both coherent and noncoherent detection. From P_M, we can determine $P_2(d)$ according to (8-2-28) or (8-2-29), which is the probability of error in a pairwise comparison of the all-zero path with a path that differs in d symbols. The probability of a bit error is upper-bounded as

$$P_b < \frac{2^{k-1}}{2^k - 1} \sum_{d=4r}^{\infty} \beta_d P_2(d) \tag{8-2-40}$$

The factor $2^{k-1}/(2^k - 1)$ is used to convert the symbol error probability to the bit error probability.

Instead of hard-decision decoding, suppose that the decoder performs soft-decision decoding using the output of a demodulator that employs a square-law detector. The expression for the bit error probability given by (8-2-40) still applies, but now $P_2(d)$ is given by (see Section 12-1-1)

$$P_2(d) = \frac{1}{2^{2d-1}} \exp\left(-\tfrac{1}{2}\gamma_b R_c d\right) \sum_{i=0}^{d-1} K_i\left(\tfrac{1}{2}\gamma_b R_c d\right) \qquad (8\text{-}2\text{-}41)$$

where

$$K_i = \frac{1}{i!} \sum_{l=0}^{d-1-i} \binom{2d-1}{l}$$

and $R_c = 1/2r$ is the code rate. This expression follows from the result (8-1-63).

Concatenated Codes In Section 8-1-8, we considered the concatenation of two block codes to form a long block code. Now that we have described convolutional codes, we broaden our viewpoint and consider the concatenation of a block code with a convolutional code or the concatenation of two convolutional codes.

As described previously, the outer code is usually chosen to be nonbinary, with each symbol selected from an alphabet of $q = 2^k$ symbols. This code may be a block code, such as a Reed–Solomon code, or a convolutional code, such as a dual-k code. The inner code may be either binary or nonbinary, and either a block or a convolutional code. For example, a Reed–Solomon code may be selected as the outer code and a dual-k code may be selected as the inner code. In such a concatenation scheme, the number of symbols in the outer (Reed–Solomon) code q equals 2^k, so that each symbol of the outer code maps into a k-bit symbol of the inner dual-k code. M-ary orthogonal signals may be used to transmit the symbols.

The decoding of such concatenated codes may also take a variety of different forms. If the inner code is a convolutional code having a short constraint length, the Viterbi algorithm provides an efficient means for decoding, using either soft-decision or hard-decision decoding.

If the inner code is a block code, and the decoder for this code performs soft-decision decoding, the outer decoder may also perform soft-decision decoding using as inputs the metrics corresponding to each word of the inner code. On the other hand, the inner decoder may make a hard decision after receipt of the code word and feed the hard decisions to the outer decoder. Then the outer decoder must perform hard-decision decoding.

The following example describes a concatenation code in which the outer code is a convolutional code and the inner code is a block code.

Example 8-2-5

Suppose we construct a concatenated code by selecting a dual-k code as the outer code and a Hadamard code as the inner code. To be specific, we select a rate 1/2 dual-5 code and a Hadamard (16, 5) inner code. The dual-5 rate

1/2 code has a minimum free distance $D_{\text{free}} = 4$ and the Hadamard code has a minimum distance $d_{min} = 8$. Hence, the concatenated code has an effective minimum distance of 32. Since there are 32 code words in the Hadamard code and 32 possible symbols in the outer code, in effect, each symbol from the outer code is mapped into one of the 32 Hadamard code words.

The probability of a symbol error in decoding the inner code may be determined from the results of the performance of block codes given in Sections 8-1-4 and 8-1-5 for soft-decision and hard-decision decoding, respectively. First, suppose that hard-decision decoding is performed in the inner decoder with the probability of a code word (symbol of outer code) error denoted as P_{32}, since $M = 32$. Then the performance of the outer code and, hence, the performance of the concatenated code is obtained by using this error probability in conjunction with the transfer function for the dual-5 code given by (8-2-37).

On the other hand, if soft-decision decoding is used on both the outer and the inner codes, the soft-decision metric from each received Hadamard code word is passed to the Viterbi algorithm, which computes the accumulated metrics for the competing paths through the trellis. We shall give numerical results on the performance of concatenated codes of this type in our discussion of coding for Rayleigh fading channels.

8-2-7 Other Decoding Algorithms for Convolutional Codes

The Viterbi algorithm described in Section 8-2-2 is the optimum decoding algorithm (in the sense of maximum-likelihood decoding of the entire sequence) for convolutional codes. However, it requires the computation of 2^{kK} metrics at each node of the trellis and the storage of $2^{k(K-1)}$ metrics and $2^{k(K-1)}$ surviving sequences, each of which may be about $5kK$ bits long. The computational burden and the storage required to implement the Viterbi algorithm make it impractical for convolutional codes with large constraint length.

Prior to the discovery of the optimum algorithm by Viterbi, a number of other algorithms had been proposed for decoding convolutional codes. The earliest was the sequential decoding algorithm originally proposed by Wozencraft (1957, 1961), and subsequently modified by Fano (1963).

The Fano sequential decoding algorithm searches for the most probable path through the tree or trellis by examining one path at a time. The increment added to the metric along each branch is proportional to the probability of the received signal for that branch, just as in Viterbi decoding, with the exception that an additional negative constant is added to each branch metric. The value of this constant is selected such that the metric for the correct path will increase on the average, while the metric for any incorrect path will decrease on the average. By comparing the metric of a candidate path with a moving (increasing) threshold, Fano's algorithm detects and discards incorrect paths.

To be more specific, let us consider a memoryless channel. The metric for the ith path through the tree or trellis from the first branch to branch B may be expressed as

$$CM^{(i)} = \sum_{j=1}^{B} \sum_{m=1}^{n} \mu_{jm}^{(i)} \tag{8-2-42}$$

where

$$\mu_{jm}^{(i)} = \log_2 \frac{p(r_{jm} \mid c_{jm}^{(i)})}{p(r_{jm})} - \mathcal{K} \tag{8-2-43}$$

In (8-2-43), r_{jm} is the demodulator output sequence, $p(r_{jm} \mid c_{jm}^{(i)})$ denotes the pdf of r_{jm} conditional on the code bit $c_{jm}^{(i)}$ for the mth bit of the jth branch of the ith path, and $\mathcal{K}$ is a positive constant. $\mathcal{K}$ is selected as indicated above so that the incorrect paths will have a decreasing metric while the correct path will have an increasing metric on the average. Note that the term $p(r_{jm})$ in the denominator is independent of the code sequence, and, hence, may be subsumed in the constant factor.

The metric given by (8-2-43) is generally applicable for either hard- or soft-decision decoding. However, it can be considerably simplified when hard-decision decoding is employed. Specifically, if we have a BSC with transition (error) probability p, the metric for each received bit, consistent with the form in (8-2-43) is given by

$$\mu_{jm}^{(i)} = \begin{cases} \log_2 [2(1-p)] - R_c & \text{if } \tilde{r}_{jm} = c_{jm}^{(i)} \\ \log_2 2p - R_c & \text{if } \tilde{r}_{jm} \neq c_{jm}^{(i)} \end{cases} \tag{8-2-44}$$

where $\tilde{r}_{jm}$ is the hard-decision output from the demodulator and $c_{jm}^{(i)}$ is the mth code bit in the jth branch of the ith path in the tree and R_c is the code rate. Note that this metric requires some (approximate) knowledge of the error probability.

Example 8-2-6

Suppose we have a rate $R_c = 1/3$ binary convolutional code for transmitting information over a BSC with $p = 0.1$. By evaluating (8-2-44) we find that

$$\mu_{jm}^{(r)} = \begin{cases} 0.52 & \text{if } \tilde{r}_{jm} = c_{jm}^{(i)} \\ -2.65 & \text{if } \tilde{r}_{jm} \neq c_{jm}^{(i)} \end{cases} \tag{8-2-45}$$

To simplify the computations, the metric in (8-2-45) may be normalized. It is well approximated as

$$\mu_{jm}^{(i)} = \begin{cases} 1 & \text{if } \tilde{r}_{jm} = c_{jm}^{(i)} \\ -5 & \text{if } \tilde{r}_{jm} \neq c_{jm}^{(i)} \end{cases} \tag{8-2-46}$$

Since the code rate is 1/3, there are three output bits from the encoder for each input bit. Hence, the branch metric consistent with (8-2-46) is

$$\mu_j^{(i)} = 3 - 6d$$

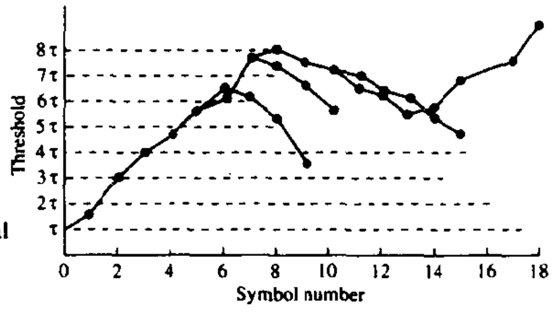


FIGURE 8-2-17 An example of the path search in sequential decoding. [*From Jordan (1966),* © *1966 IEEE.*]

or, equivalently,

$$\mu_j^{(i)} = 1 - 2d \tag{8-2-47}$$

where d is the Hamming distance of the three received bits from the three branch bits. Thus, the metric $\mu_j^{(i)}$ is simply related to the Hamming distance of the received bits to the code bits in the jth branch of the ith path.

Initially, the decoder may be forced to start on the correct path by the transmission of a few known bits of data. Then it proceeds forward from node to node, taking the most probable (largest metric) branch at each node and increasing the threshold such that the threshold is never more than some preselected value, say τ, below the metric. Now suppose that the additive noise (for soft-decision decoding) or demodulation errors resulting from noise on the channel (for hard-decision decoding) cause the decoder to take an incorrect path because it appears more probable than the correct path. This is illustrated in Fig. 8-2-17. Since the metrics of an incorrect path decrease on the average, the metric will fall below the current threshold, say τ_0. When this occurs, the decoder backs up and takes alternative paths through the tree or trellis, in order of decreasing branch metrics, in an attempt to find another path that exceeds the threshold τ_0. If it is successful in finding an alternative path, it continues along that path, always selecting the most probable branch at each node. On the other hand, if no path exists that exceeds the threshold τ_0, the threshold is reduced by an amount τ and the original path is retraced. If the original path does not stay above the new threshold, the decoder resumes its backward search for other paths. This procedure is repeated, with the threshold reduced by τ for each repetition, until the decoder finds a path that remains above the adjusted threshold. A simplified flow diagram of Fano's algorithm is shown in Fig. 8-2-18.

The sequential decoding algorithm requires a buffer memory in the decoder to store incoming demodulated data during periods when the decoder is searching for alternate paths. When a search terminates, the decoder must be capable of processing demodulated bits sufficiently fast to empty the buffer prior to commencing a new search. Occasionally, during extremely long searches, the buffer may overflow. This causes loss of data, a condition that

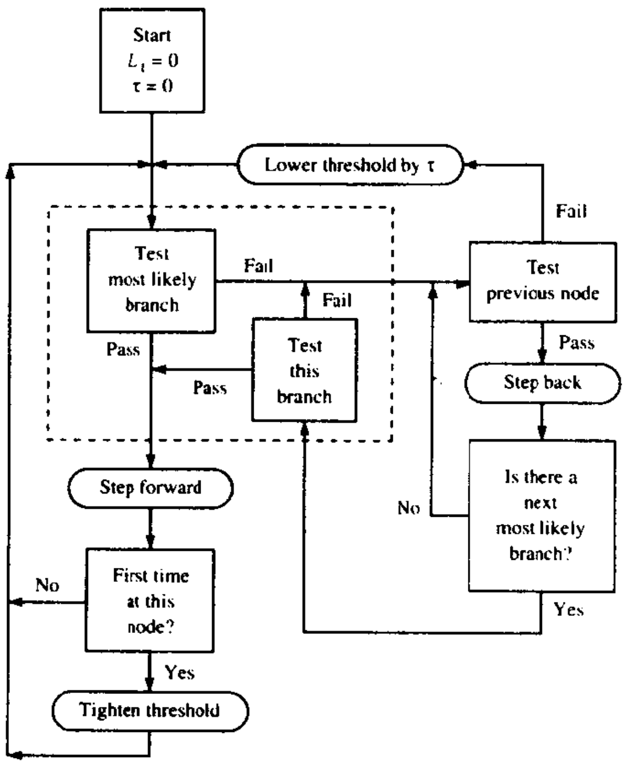
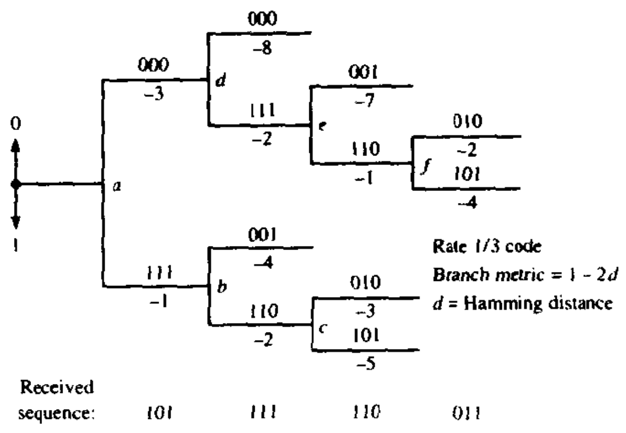


FIGURE 8-2-18 A simplified flow diagram of Fano's algorithm. [*From Jordan (1966),* © *1966 IEEE.*]

can be remedied by retransmission of the lost information. In this regard, we should mention that the cutoff rate R_0 has special meaning in sequential decoding. It is the rate above which the average number of decoding operations per decoded digit becomes infinite, and it is termed the *computational cutoff rate* R_{comp}. In practice, sequential decoders usually operate at rates near R_0.

The Fano sequential decoding algorithm has been successfully implemented in several communication systems. Its error rate performance is comparable to that of Viterbi decoding. However, in comparison with Viterbi decoding, sequential decoding has a significantly larger decoding delay. On the positive side, sequential decoding requires less storage than Viterbi decoding and, hence, it appears attractive for convolutional codes with a large constraint length. The issues of computational complexity and storage requirements for sequential decoding are interesting and have been thoroughly investigated. For an analysis of these topics and other characteristics of the Fano algorithm, the interested reader may refer to Gallager (1968), Wozencraft and Jacobs (1965), Savage (1966), and Forney (1974).

Another type of sequential decoding algorithm, called a *stack algorithm*, has been proposed independently by Jelinek (1969) and Zigangirov (1966). In contrast to the Viterbi algorithm, which keeps track of $2^{(K-1)k}$ paths and



Stack with accumulated path metrics

Step *a*	Step *b*	Step *c*	Step *d*	Step *e*	Step *f*
-1	-2	-3	-2	-1	-2
-3	-3	-3	-3	-3	-3
	-4	-4	-4	-4	-4
		-5	-5	-5	-4
			-8	-7	-5
				-8	-7
					-8

FIGURE 8-2-19 A example of the stack algorithm for decoding a rate 1/3 convolutional code.

corresponding metrics, the stack sequential decoding algorithm deals with fewer paths and their corresponding metrics. In a stack algorithm, the more probable paths are ordered according to their metrics, with the path at the top of the stack having the largest metric. At each step of the algorithm, only the path at the top of the stack is extended by one branch. This yields 2^k successors and their corresponding metrics. These 2^k successors along with the other paths are then reordered according to the values of the metrics and all paths with metrics that fall below some preselected amount from the metric of the top path may be discarded. Then the process of extending the path with the largest metric is repeated. Figure 8-2-19 illustrates the first few steps in a stack algorithm.

It is apparent that when none of the 2^k extensions of the path with the largest metric remains at the top of the stack, the next step in the search involves the extension of another path that has climbed to the top of the stack. It follows that the algorithm does not necessarily advance by one branch through the trellis in every iteration. Consequently, some amount of storage must be provided for newly received signals and previously received signals in order to allow the algorithm to extend the search along one of the shorter paths, when such a path reaches the top of the stack.

In a comparison of the stack algorithm with the Viterbi algorithm, the stack algorithm requires fewer metric computations, but this computational saving is offset to a large extent by the computations involved in reordering the stack after every iteration. In comparison with the Fano algorithm, the stack algorithm is computationally simpler, since there is no retracing over the same path as is done in the Fano algorithm. On the other hand, the stack algorithm requires more storage than the Fano algorithm.

A third alternative to the optimum Viterbi decoder is a method called *feedback decoding* (Heller, 1975), which has been applied to decoding for a BSC (hard-decision decoding). In feedback decoding, the decoder makes a hard decision on the information bit at stage j based on metrics computed from stage j to stage $j+m$, where m is a preselected positive integer. Thus, the decision on the information bit is either 0 or 1 depending on whether the minimum Hamming distance path that begins at stage j and ends at stage $j+m$ contains a 0 or 1 in the branch emanating from stage j. Once a decision is made on the information bit at stage j, only that part of the tree that stems from the bit selected at stage j is kept (half the paths emanating from node j) and the remaining part is discarded. This is the feedback feature of the decoder.

The next step is to extend the part of the tree that has survived to stage $j+1+m$ and consider the paths from stage $j+1$ to $j+1+m$ in deciding on the bit at stage $j+1$. Thus, this procedure is repeated at every stage. The parameter m is simply the number of stages in the tree that the decoder looks ahead before making a hard decision. Since a large value of m results in a large amount of storage, it is desirable to select m as small as possible. On the other hand, m must be sufficiently large to avoid a severe degradation in performance. To balance these two conflicting requirements, m is usually selected in the range $K \leq m \leq 2K$, where K is the constraint length. Note that this decoding delay is significantly smaller than the decoding delay in a Viterbi decoder, which is usually about $5K$.

Example 8-2-7

Let us consider the use of a feedback decoder for the rate 1/3 convolutional code shown in Fig. 8-2-2. Figure 8-2-20 illustrates the tree diagram and the operation of the feedback decoder for $m=2$. That is, in decoding the bit at branch j, the decoder considers the paths at branches j, $j+1$, and $j+2$. Beginning with the first branch, the decoder computes eight metrics (Hamming distances), and decides that the bit for the first branch is 0 if the minimum distance path is contained in the upper part of the tree, and 1 if the minimum distance path is contained in the lower part of the tree. In this example, the received sequence for the first three branches is assumed to be 101111110, so that the minimum distance path is in the upper part of the tree. Hence, the first output bit is 0.

The next step is to extend the upper part of the tree (the part of the tree that has survived) by one branch, and to compute the eight metrics for

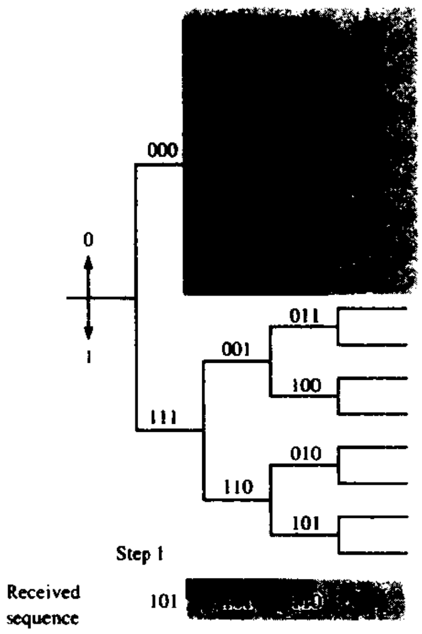


FIGURE 8-2-20 An example of feedback decoding for a rate 1/3 convolutional code.

Step 1 : Upper-tree metrics : 7,6,5,2* ; lower-tree metrics : 5,4,3,4 → 0
Step 2 : Upper-tree metrics : 7,6,5,6 ; lower-tree metrics : 3,6,1* ,2 → 1

branches 2, 3, and 4. For the assumed received sequence 111110011, the minimum-distance path is contained in the lower part of the section of the tree that survived from the first step. Hence, the second output bit is 1. The third step is to extend this lower part of the tree and to repeat the procedure described for the first two steps.

Instead of computing metrics as described above, a feedback decoder for the BSC may be efficiently implemented by computing the syndrome from the received sequence and using a table lookup method for correcting errors. This method is similar to the one described for decoding block codes. For some convolutional codes, the feedback decoder simplifies to a form called a *majority logic decoder* or a *threshold decoder* (Massey, 1963; Heller, 1975).

8-2-8 Practical Considerations in the Application of Convolutional Codes

Convolutional codes are widely used in many practical applications of communications system design. Viterbi decoding is predominantly used for short constraint lengths ($K \leq 10$), while sequential decoding is used for long constraint length codes, where the complexity of Viterbi decoding becomes prohibitive. The choice of constraint length is dictated by the desired coding gain.

From the error probability results for soft-decision decoding given by

TABLE 8-2-12 UPPER BOUNDS ON CODING GAIN FOR SOFT-DECISION DECODING OF SOME CONVOLUTION CODES

Rate 1/2 codes			Rate 1/3 codes		
Constraint length K	d_{free}	Upper bound (db)	Constraint length K	d_{free}	Upper bound (dB)
3	5	3.98	3	8	4.26
4	6	4.77	4	10	5.23
5	7	5.44	5	12	6.02
6	8	6.02	6	13	6.37
7	10	6.99	7	15	6.99
8	10	6.99	8	16	7.27
9	12	7.78	9	18	7.78
10	12	7.78	10	20	8.24

(8-2-26) it is apparent that the coding gain achieved by a convolutional code over an uncoded binary PSK or QPSK system is

$$\text{coding gain} \leq 10 \log_{10} (R_c d_{\text{free}})$$

We also know that the minimum free distance d_{free} can be increased either by decreasing the code rate or by increasing the constraint length, or both. Table 8-2-12 provides a list of upper bounds on the coding gain for several convolutional codes. For purposes of comparison, Table 8-2-13 lists the actual coding gains and the upper bounds for several short constraint length convolutional codes with Viterbi decoding. It should be noted that the coding gain increases toward the asymptotic limit as the SNR per bit increases.

These results are based on soft-decision Viterbi decoding. If hard-decision decoding is used, the coding gains are reduced by approximately 2 dB for the AWGN channel.

Larger coding gains than those listed in the above tables are achieved by

TABLE 8-2-13 CODING GAIN (dB) FOR SOFT-DECISION VITERBI DECODING

P_b	$\mathscr{E}_b/N_0$ uncoded (dB)	$R_c = 1/3$		$R_c = 1/2$			$R_c = 2/3$		$R_c = 3/4$	
		$K=7$	$K=8$	$K=5$	$K=6$	$K=7$	$K=6$	$K=8$	$K=6$	$K=9$
10^{-3}	6.8	4.2	4.4	3.3	3.5	3.8	2.9	3.1	2.6	2.6
10^{-5}	9.6	5.7	5.9	4.3	4.6	5.1	4.2	4.6	3.6	4.2
10^{-7}	11.3	6.2	6.5	4.9	5.3	5.8	4.7	5.2	3.9	4.8

Source: Jacobs (1974); © IEEE.

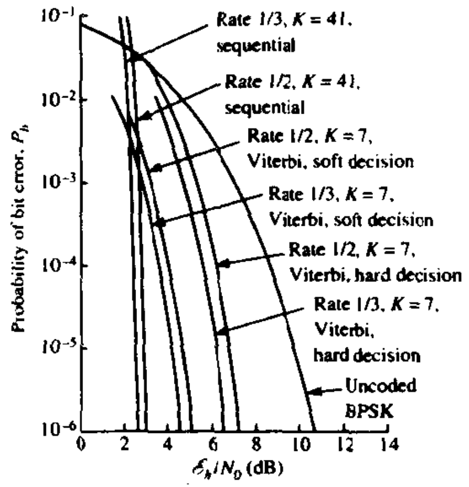


FIGURE 8-2-21 Performance of rate 1/2 and rate 1/3 Viterbi and sequential decoding. [*From Omura and Levitt (1982).* © *1982 IEEE.*]

employing long constraint length convolutional codes, e.g., $K = 50$, and decoding such codes by sequential decoding. Invariably, sequential decoders are implemented for hard-decision decoding to reduce complexity. Figure 8-2-21 illustrates the error rate performance of several constraint-length $K = 7$ convolutional codes for rates 1/2 and 1/3 and for sequential decoding (with hard decisions) of a rate 1/2 and a rate 1/3 constraint-length $K = 41$ convolutional codes. Note that the $K = 41$ codes achieve an error rate of 10^{-6} at 2.5 and 3 dB, which are within 4–4.5 dB of the channel capacity limit, i.e., in vicinity of the cutoff rate limit. However, the rate 1/2 and rate 1/3, $K = 7$ codes with soft-decision Viterbi decoding operate at about 5 and 4.4 dB at 10^{-6}, respectively. These short-constraint-length codes achieve a coding gain of about 6 dB at 10^{-6}, while the long constraint codes gain about 7.5–8 dB.

Two important issues in the implementation of Viterbi decoding are

1 the effect of path memory truncation, which is a desirable feature that ensures a fixed decoding delay, and

2 the degree of quantization of the input signal to the Viterbi decoder.

As a rule of thumb, we stated that path memory truncation to about five constraint lengths has been found to result in negligible performance loss. Figure 8-2-22 illustrates the performance obtained by simulation for rate 1/2, constraint-lengths $K = 3$, 5, and 7 codes with memory path length of 32 bits. In addition to path memory truncation, the computations were performed with eight-level (three bits) quantized input signals from the demodulator. The broken curves are performance results obtained from the upper bound in the bit error rate given by (8-2-26). Note that the simulation results are close to the theoretical upper bounds, which indicate that the degradation due to path memory truncation and quantization of the input signal has a minor effect on performance (0.20–0.30 dB).

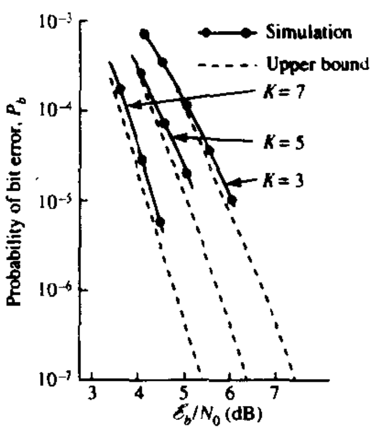


FIGURE 8-2-22 Bit error probability for rate 1/2 Viterbi decoding with eight-level quantized inputs to the decoder and 32-bit path memory. [*From Heller and Jacobs (1971).* © *1971 IEEE.*]

Figure 8-2-23 illustrates the bit error rate performance obtained via simulation for hard-decision decoding of convolutional codes with $K = 3$–8. Note that with the $K = 8$ code, an error rate of 10^{-5} requires about 6 dB, which represents a coding gain of nearly 4 dB relative to uncoded QPSK.

The effect of input signal quantization is further illustrated in Fig. 8-2-24 for a rate 1/2, $K = 5$ code. Note that three-bit quantization (eight levels) is about 2 dB better than hard-decision decoding, which is the ultimate limit between soft-decision decoding and hard-decision decoding on the AWGN channel. The combined effect of signal quantization and path memory trunction for the rate 1/2, $K = 5$ code with 8-, 16-, and 32-bit path memories and either one- or three-bit quantization is shown in Fig. 8-2-25. It is apparent from these results that a path memory as short as three constraint lengths does not seriously degrade performance.

When the signal from the demodulator is quantized to more than two levels, another problem that must be considered is the spacing between quantization levels. Figure 8-2-26 illustrates the simulation results for an eight-level uniform quantizer as a function of the quantizer threshold spacing. We observe that

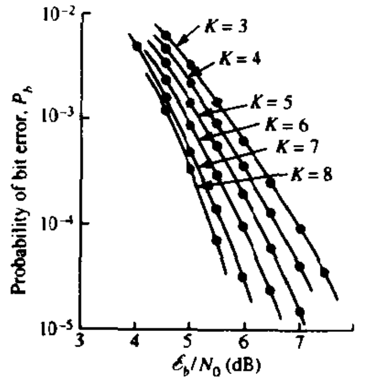


FIGURE 8-2-23 Performance of rate 1/2 codes with hard-decision Viterbi decoding and 32-bit path memory truncation. [*From Heller and Jacobs (1971).* © *1971 IEEE.*]

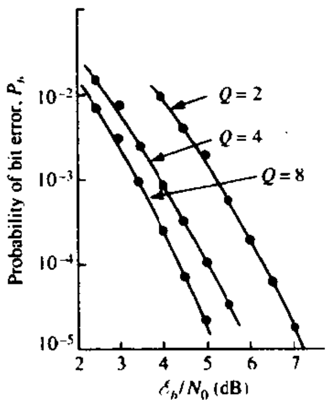


FIGURE 8-2-24 Performance of rate 1/2, $K = 5$ code with eight-, four-, and two-level quantization at the input to the Viterbi decoder. Path truncation length = 32 bits. [*From Heller and Jacobs (1971).* © *1971 IEEE.*]

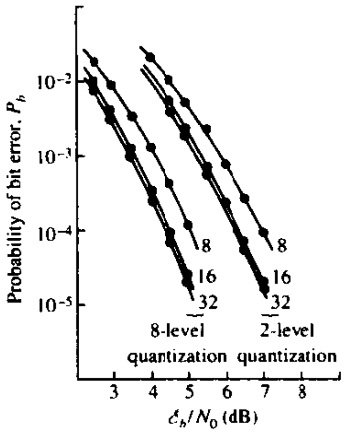


FIGURE 8-2-25 Performance of rate 1/2, $K = 5$ code with 32-, 16-, and 8-bit path memory truncation and eight- and two-level quantization. [*From Heller and Jacobs (1971).* © *1971 IEEE.*]

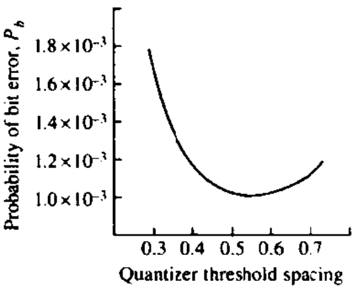


FIGURE 8-2-26 Error rate performance of rate 1/2, $K = 5$ Viterbi decoder for $\mathscr{E}_b/N_0 = 3.5$ dB and eight-level quantization as a function of quantizer threshold level spacing for equally spaced thresholds [*From Heller and Jacobs (1971).* © *1971 IEEE.*]

there is an optimum spacing between thresholds (approximately equal to 0.5). However, the optimum is sufficiently broad (0.4–0.7) so that, once it is set, there is little degradation resulting from variations in the AGC level of the order of ±20%.

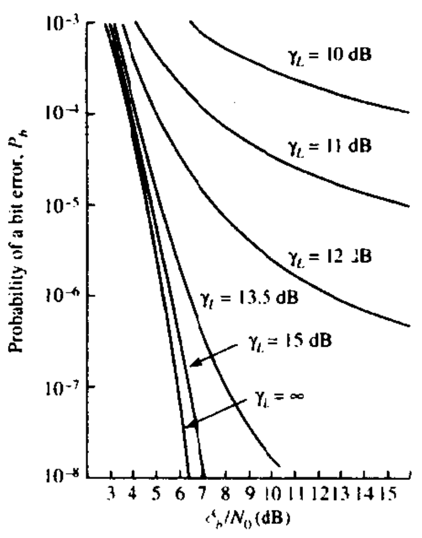


FIGURE 8-2-27 Performance of a rate 1/2, $K = 7$ code with Viterbi decoding and eight-level quantization as a function of the carrier phase tracking loop SNR γ_L. [*From Heller and Jacobs (1971).* © *1971 IEEE.*]

Finally, we should point out some important results in the performance degradation due to carrier phase variations. Figure 8-2-27 illustrates the performance of a rate 1/2, $K = 7$ code with eight-level quantization and a carrier phase tracking loop SNR γ_L. Recall that in a PLL, the phase error has a variance that is inversely proportional to γ_L. The results in Fig. 8-2-27 indicate that the degradation is large when the loop SNR is small ($\gamma_L < 12$ dB), and causes the error rate performance to bottom out at relatively high error rate.

8-3 CODED MODULATION FOR BANDWIDTH-CONSTRAINED CHANNELS

In the treatment of block and convolutional codes in Sections 8-1 and 8-2, respectively, performance improvement was achieved by expanding the bandwidth of the transmitted signal by an amount equal to the reciprocal of the code rate. Recall for example that the improvement in performance achieved by an (n, k) binary block code with soft-decision decoding is approximately $10\log_{10}(R_c d_{\min} - k \ln 2/\gamma_b)$ compared with uncoded binary or quaternary PSK. For example, when $\gamma_b = 10$ the (24, 12) extended Golay code gives a coding gain of 5 dB. This coding gain is achieved at a cost of doubling the bandwidth of the transmitted signal and, of course, at the additional cost in receiver implementation complexity. Thus, coding provides an effective method for trading bandwidth and implementation complexity against transmitter power. This situation applies to digital communications systems that are designed to operate in the power-limited region where $R/W < 1$.

In this section, we consider the use of coded signals for bandwidth-constrained channels. For such channels, the digital communications system is designed to use bandwidth-efficient multilevel/phase modulation, such as PAM, PSK, DPSK, or QAM, and operates in the region where $R/W > 1$. When coding is applied to the bandwidth-constrained channel, a performance gain is desired without expanding the signal bandwidth. This goal can be achieved by increasing the number of signals over the corresponding uncoded system to compensate for the redundancy introduced by the code.

For example, suppose that a system employing uncoded four-phase PSK modulation achieves an $R/W = 2$ (bits/s)/Hz at an error probability of 10^{-6} For this error rate the SNR per bit is $\gamma_b = 10.5$ dB. We may try to reduce the SNR per bit by use of coded signals, but this must be done without expanding the bandwidth. If we choose a rate $R_c = 2/3$ code, it must be accompanied by an increase in the number of signal points from four (two bits per symbol) to eight (three bits per symbol). Thus, the rate 2/3 code used in conjunction with eight-phase PSK, for example, yields the same data throughput as uncoded four-phase PSK. However, we recall that an increase in the number of signal phases from four to eight requires an additional 4 dB approximately in signal power to maintain the same error rate. Hence, if coding is to provide a benefit, the performance gain of the rate 2/3 code must overcome this 4 dB penalty.

If the modulation is treated as a separate operation independent of the encoding, the use of very powerful codes (large-constraint-length convolutional codes or large-block-length block codes) is required to offset the loss and provide some significant coding gain. On the other hand, if the modulation is an integral part of the encoding process and is designed in conjunction with the code to increase the minimum euclidean distance between pairs of coded signals, the loss from the expansion of the signal set is easily overcome and a significant coding gain is achieved with relatively simple codes. The key to this integrated modulation and coding approach is to devise an effective method for mapping the coded bits into signal points such that the minimum euclidean distance is maximized. Such a method was developed by Ungerboeck (1982), based on the principle of *mapping by set partitioning*. We describe this principle by means of two examples.

Example 8-3-1: An 8-PSK Signal Constellation

Let us partition the eight-phase signal constellation shown in Fig. 8-3-1 into subsets of increasing minimum euclidean distance. In the eight-phase signal set, the signal points are located on a circle of radius $\sqrt{\mathscr{E}}$ and have a minimum distance separation of

$$d_0 = 2\sqrt{\mathscr{E}} \sin \tfrac{1}{8}\pi = \sqrt{(2-\sqrt{2})\mathscr{E}} = 0.765\sqrt{\mathscr{E}}$$

In the first partitioning, the eight points are subdivided into two subsets of four points each, such that the minimum distance between points increases to $d_1 = \sqrt{2\mathscr{E}}$. In the second level of partitioning, each of the two subsets is subdivided into two subsets of two points, such that the minimum distance increases to $d_2 = 2\sqrt{\mathscr{E}}$. This results in four subsets of two points each.

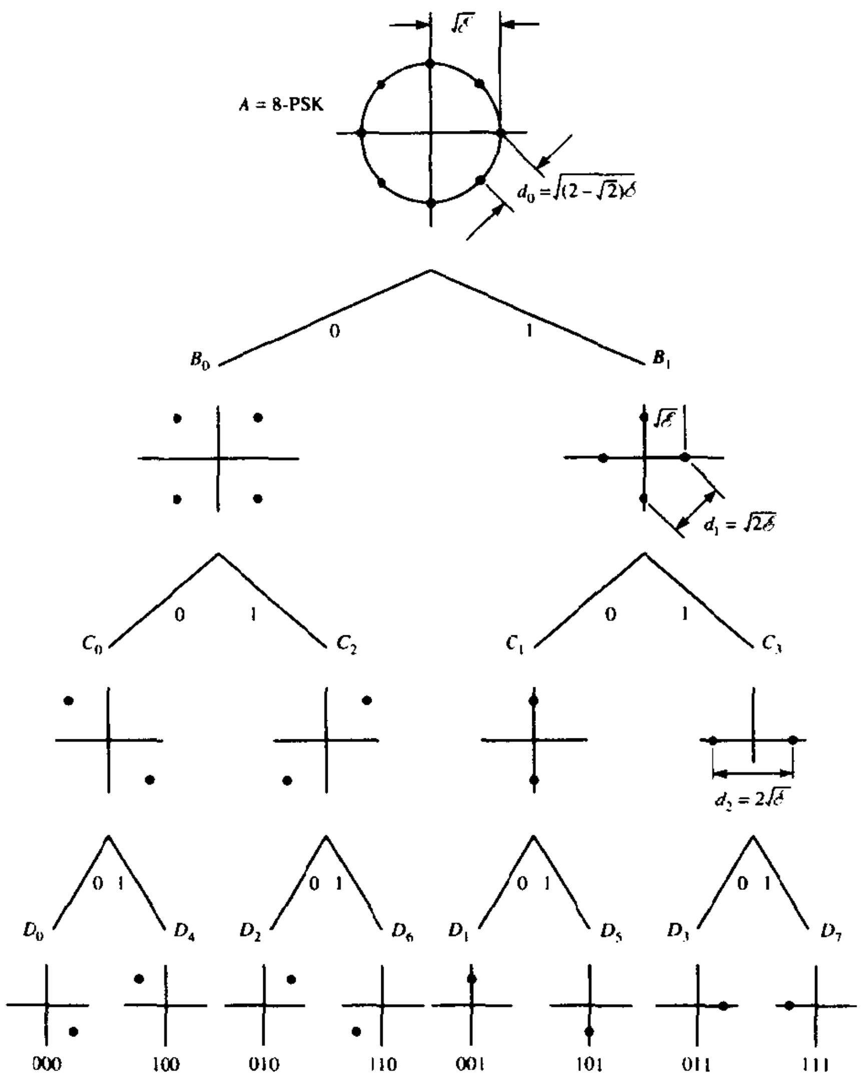


FIGURE 8-3-1 Set partitioning of an 8-PSK signal set.

Finally, the last stage of partitioning leads to eight subsets, where each subset contains a single point. Note that each level of partitioning increases the minimum euclidean distance between signal points. The results of these three stages of partitioning are illustrated in Fig. 8-3-1. The way in which the coded bits are mapped into the partitioned signal points is described below.

Example 8-3-2: A 16-QAM Signal Constellation

The 16-point rectangular signal constellation shown in Fig. 8-3-2 is first divided into two subsets by assigning alternate points to each subset as

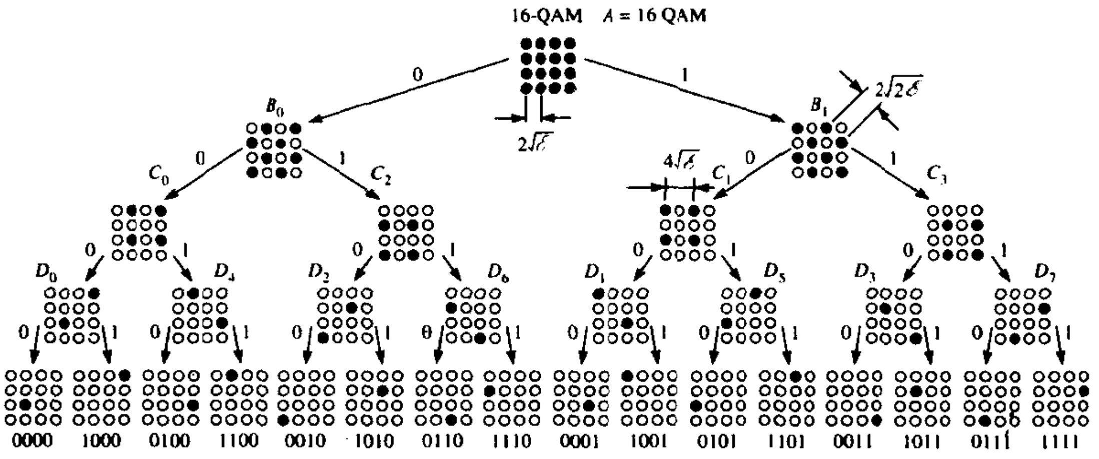


FIGURE 8-3-2 Set partitioning of 16-QAM signal.

illustrated in the figure. Thus, the distance between points is increased from $2\sqrt{\mathscr{E}}$ to $2\sqrt{2\mathscr{E}}$ by the first partitioning. Further partitioning of the two subsets leads to greater separation in euclidean distance between signal points as illustrated in Fig. 8-3-2. It is interesting to note that for the rectangular signal constellations, each level of partitioning increases the minimum euclidean distance by $\sqrt{2}$, i.e., $d_{i+1}/d_i = \sqrt{2}$ for all i.

In these two examples, the partitioning was carried out to the limit where each subset contains only a single point. In general, this may not be necessary. For example, the 16-point QAM signal constellation may be partitioned only twice, to yield four subsets of four points each. Similarly, the eight-phase PSK signal constellation can be partitioned twice, to yield four subsets of two points each.

The degree to which the signal is partitioned depends on the characteristics of the code. In general, the encoding process is performed as illustrated in Fig. 8-3-3. A block of m information bits is separated into two groups of length k_1

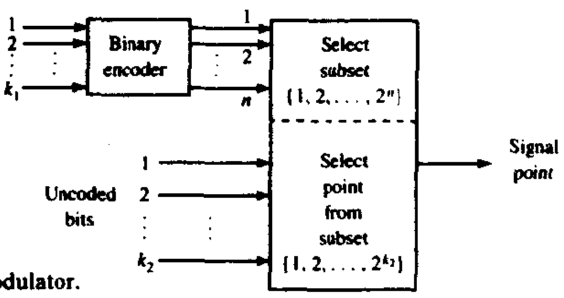

1
2
k_1
Binary encoder
1
2
n
Select subset $\{1, 2, \ldots, 2^n\}$
Uncoded bits
1
2
k_2
Select point from subset $\{1, 2, \ldots, 2^{k_2}\}$
Signal point

FIGURE 8-3-3 General structure of combined encoder/modulator.

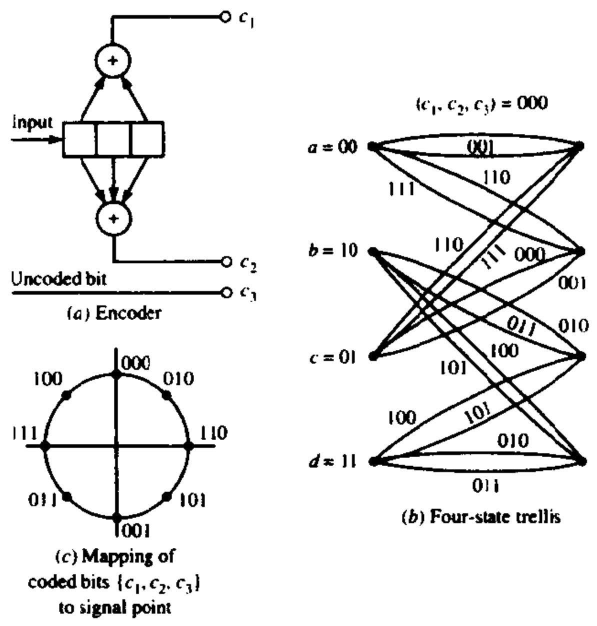


FIGURE 8-3-4 Four-state trellis-coded 8-PSK modulation.

and k_2. The k_1 bits are encoded into n bits while the k_2 bits are left uncoded. Then, the n bits from the encoder are used to select one of the 2^n possible subsets in the partitioned signal set while the k_2 bits are used to select one of the 2^{k_2} signal points in each subset. When $k_2 = 0$, all m information bits are encoded.

Example 8-3-3

Consider the use of the rate 1/2 convolutional code shown in Fig. 8-3-4 to encode one information bit while the second information bit is left uncoded. When used in conjunction with an eight-point signal constellation, e.g., eight-phase PSK or eight-point QAM, the two encoded bits are used to select one of the four subsets in the signal constellation, while the remaining information bit is used to select one of the two points within each subset. In this case, $k_1 = 1$ and $k_2 = 1$. The four-state trellis, which is shown in Fig. 8-3-4(b), is basically the trellis for the rate 1/2 convolution encoder with the addition of parallel paths in each transition to accommodate the uncoded bit c_3. Thus, the coded bits (c_1, c_2) are used to select one of the four subsets that contain two signal points each, while the uncoded bit is used to select one of the two signal points within each subset. Note that signal points within a subset are separated in distance by $d_2 = 2\sqrt{\mathscr{E}}$. Hence, the euclidean distance between parallel paths is d_2. The mapping of coded bits (c_1, c_2, c_3) to signal points is illustrated in Fig. 8-3-4(c). As an alternative coding scheme, we may use a rate 2/3 convolutional encoder, and, thus, encode

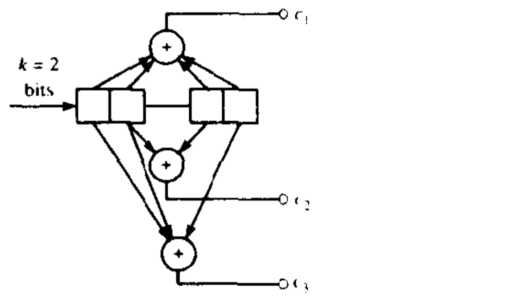


FIGURE 8-3-5 Rate 2/3 convolutional encoder for encoding both information bits.

both information bits as shown in Fig. 8-3-5. This encoding leads to an eight-state trellis and results in better performance, but also requires a more complex implementation of the decoder as described below.

Either block codes or convolutional codes may be used in conjunction with the partitioned signal constellation. In general, convolutional codes provide comparable coding gains to block codes and the availability of the Viterbi algorithm results in a simpler implementation for soft-decision decoding. For this reason, we limit our discussion to convolutional codes (linear trellis codes) and more generally to (nonlinear) trellis codes.

Trellis-Coded Modulation Let us consider the use of the 8-PSK signal constellation in conjunction with trellis codes. Uncoded four-phase PSK (4-PSK) is used as a reference in measuring coding gain. Uncoded 4-PSK employs the signal points in either subset B_0 or B_1 of Fig. 8-3-1, for which the minimum distance of the signal points is $\sqrt{2\mathscr{E}}$. Note that this signal corresponds to a trivial one-state trellis with four parallel state transitions as shown in Fig. 8-3-6(a). The subsets D_0, D_2, D_4, and D_6 are used as the signal points for the purpose of illustration.

For the coded 8-PSK modulation, we may use the four-state trellis shown in Fig. 8-3-6(b). Note that each branch in the trellis corresponds to one of the four subsets C_0, C_1, C_2, or C_3. For the eight-point constellation, each of the subsets C_0, C_1, C_2, and C_3, contains two signal points. Hence, the state transition C_0 contains the two signal points corresponding to the bits (000, 100) or (0, 4) in octal representation. Similarly, C_2 contains the two signal points corresponding to (010, 110), or to (2, 6) in octal, C_1 contains the points corresponding to (001, 101), or (1, 5) in octal, and C_3 contains the points corresponding to (011, 111), or (3, 7) in octal. Thus, each transition in the four-state trellis contains two parallel paths, as shown in more detail in Fig. 8-3-6(c). Note that any two signal paths that diverge from one state and remerge at the same state after more than one transition have a squared euclidean distance of $d_0^2 + 2d_1^2 = d_0^2 + d_2^2$ between them. For example, the

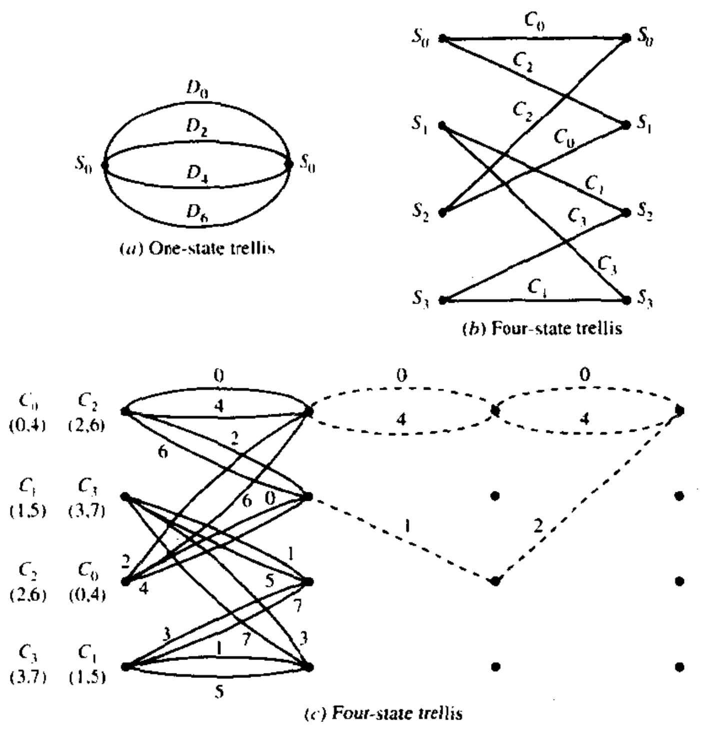


FIGURE 8-3-6 Uncoded 4-PSK and trellis-coded 8-PSK modulation.

signal paths 0, 0, 0 and 2, 1, 2 are separated by $d_0^2 + d_2^2 = [(0.765)^2 + 4]\mathscr{E} = 4.585\mathscr{E}$. On the other hand, the squared euclidean distance between parallel transitions is $d_2^2 = 4\mathscr{E}$. Hence, the minimum euclidean distance separation between paths that diverge from any state and remerge at the same state in the four-state trellis is $d_2 = 2\sqrt{\mathscr{E}}$. This minimum distance in the trellis code is called the *free euclidean distance* and denoted by D_{fed}.

In the four-state trellis of Fig. 8-3-6(*b*), $D_{\text{fed}} = 2\sqrt{\mathscr{E}}$. When compared with the euclidean distance $d_0 = \sqrt{2\mathscr{E}}$ for the uncoded 4-PSK modulation, we observe that the four-state trellis code gives a coding gain of 3 dB.

We should emphasize that the four-state trellis code illustrated in Fig. 8-3-6(*b*) is optimum in the sense that it provides the largest free euclidean distance. Clearly, many other four-state trellis codes can be constructed, including the one shown in Fig. 8-3-7, which consists of four distinct transitions from each state to all other states. However, neither this code nor any of the other possible four-state trellis codes gives a larger D_{fed}.

The construction of the optimum four-state trellis code for the eight-point constellation was performed on the basis of the following heuristic rules:

(**a**) Parallel transitions (when they occur) are assigned to signal points separated by the maximum euclidean distance, e.g., $d_2 = 2\sqrt{\mathscr{E}}$ for 8-PSK in the four subsets C_0, C_1, C_2, C_3.

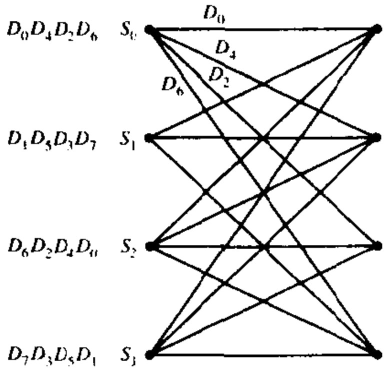


FIGURE 8-3-7 An alternative four-state trellis code.

(b) The transition originating from and merging into any state is assigned the subsets (C_0, C_2) or (C_1, C_3), which have a maximum distance $d_1 = \sqrt{2\mathscr{E}}$.
(c) The signal points should occur with equal frequency.

Note that rules (a) and (b) guarantee that the euclidean distance associated with single and multiple paths that diverge from any state and remerge in that state exceeds the euclidean distance of uncoded 4-PSK. Rule (c) guarantees that the trellis code will have a regular structure.

We should indicate that the specific mapping of coded bits into signal points, as illustrated in Fig. 8-3-1, where the eight signal points are represented in an equivalent binary form, is not important. Other mappings can be devised by permuting subsets in a way that preserves the main property of increased minimum distance among the subsets.

In the four-state trellis code, the parallel transitions were separated by the euclidean distance $2\sqrt{\mathscr{E}}$, which is also D_{fed}. Hence, the coding gain of 3 dB is limited by the distance of the parallel transitions. Larger gains in performance relative to uncoded 4-PSK can be achieved by using trellis codes with more states, which allow for the elimination of the parallel transitions. Thus, trellis codes with eight or more states would use distinct transitions to obtain a larger D_{fed}.

For example, in Fig. 8-3-8, we illustrate an eight-state trellis code due to Ungerboeck (1982) for the 8-PSK signal constellation. The state transitions for maximizing the free euclidean distance were determined from application of the three basic rules given above. In this case, note that the minimum squared euclidean distance is

$$D_{\text{fed}}^2 = d_0^2 + 2d_1^2 = 4.585\mathscr{E}$$

which, when compared with $d_0^2 = 2\mathscr{E}$ for uncoded 4-PSK, represents a gain of 3.6 dB. Ungerboeck (1982, 1987) has also found rate 2/3 trellis codes with 16, 32, 64, 128, and 256 states that achieve coding gains ranging from 4 to 5.75 dB for 8-PSK modulation.

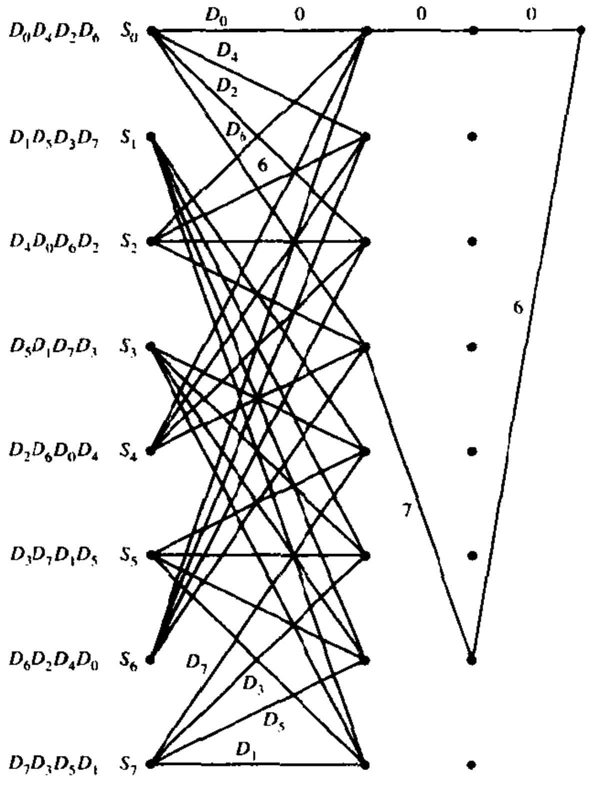


FIGURE 8-3-8 Eight-state trellis code for coded 8-PSK modulation.

The basic principle of set partitioning is easily extended to larger PSK signal constellations that yield greater bandwidth efficiency. For example, 3 (bits/s)/Hz can be achieved with either uncoded 8-PSK or with trellis-coded 16-PSK modulation. Ungerboeck (1987) has devised trellis codes and has evaluated the coding gains achieved by simple rate 1/2 and rate 2/3 convolutional codes for the 16-PSK signal constellations. The results are summarized below.

Soft-decision Viterbi decoding for trellis-coded modulation is accomplished in two steps. Since each branch in the trellis corresponds to a signal subset, the first step in decoding is to determine the best signal point within each subset, i.e., the point in each subset that is closest in distance to the received point. We may call this *subset decoding*. In the second step, the signal point selected from each subset and its squared distance metric are used for the corresponding branch in the Viterbi algorithm to determine the signal path through the code trellis that has the minimum sum of squared distances from the sequence of received (noisy channel output) signals.

The error rate performance of the trellis-coded signals in the presence of additive gaussian noise can be evaluated by following the procedure described in Section 8-2 for convolutional codes. Recall that this procedure involves the computation of the probability of error for all different error events and summing these error event probabilities to obtain a union bound on the first-event error probability. Note, however, that, at high SNR, the first-event error probability is dominated by the leading term, which has the minimum distance D_{fed}. Consequently, at high SNR, the first-event error probability is well approximated as

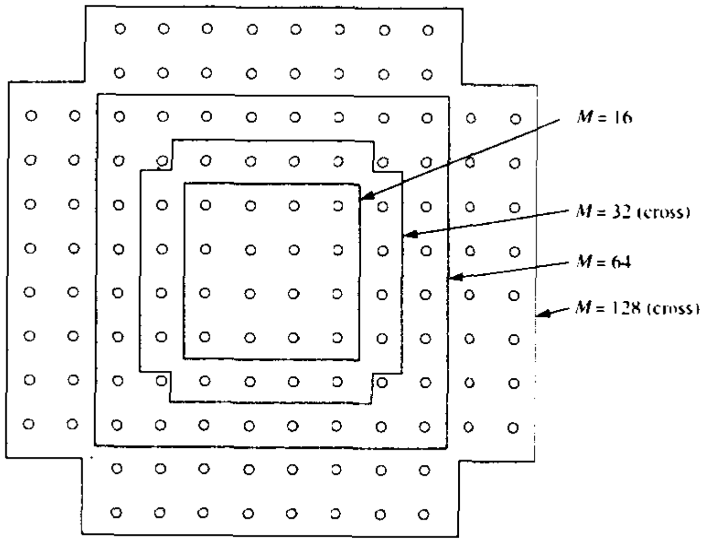
$$P_e \approx N_{\text{fed}} Q\left(\sqrt{\frac{D_{\text{fed}}^2}{2N_0}}\right) \tag{8-3-1}$$

where N_{fed} denotes the number of signal sequences with distance D_{fed} that diverge at any state and remerge at that state after one or more transitions.

In computing the coding gain achieved by trellis-coded modulation, we usually focus on the gain achieved by increasing D_{fed} and neglect the effect of N_{fed}. However, trellis codes with a large number of states may result in a large N_{fed} that cannot be ignored in assessing the overall coding gain.

In addition to the trellis-coded PSK modulations described above, powerful trellis codes have also been developed for PAM and QAM signal constellations. Of particular practical importance is the class of trellis-coded two-dimensional rectangular signal constellations. Figure 8-3-9 illustrates these signal constellations for M-QAM where $M = 16$, 32, 64, and 128. The $M = 32$ and 128 constellations have a cross pattern and are sometimes called *cross-constellations*. The underlying rectangular grid containing the signal points in M-QAM is called a *lattice of type* Z_2 (the subscript indicates the dimensionality of the space). When set partitioning is applied to this class of signal constellations, the minimum euclidean distance between successive partitions is $d_{i+1}/d_i = \sqrt{2}$ for all i, as previously observed in Example 8-3-2.

FIGURE 8-3-9 Rectangular two-dimensional (QAM) signal constellations.



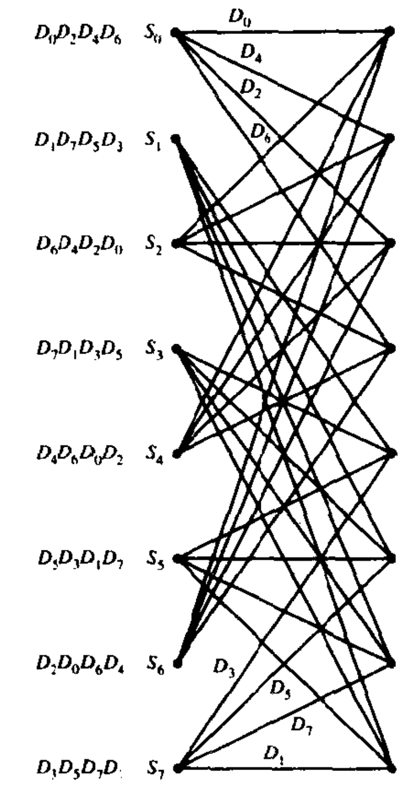


FIGURE 8-3-10 Eight-state trellis for rectangular QAM signal constellations.

Figure 8-3-10 illustrates an eight-state trellis code that can be used with any of the M-QAM rectangular signal constellations for which $M = 2^k$, where $k = 4, 5, 6, \ldots$, etc. With the eight-state trellis, we associate eight signal subsets, so that any of the M-QAM signals sets for $M \geqslant 16$ are suitable. For $M = 2^{m+1}$, two input bits ($k_1 = 2$) are encoded into $n = 3$ ($n = k_1 + 1$) bits that are used to select one of the eight states corresponding to the eight subsets. The additional $k_2 = m - k_1$ input bits are used to select signal points within a subset, and result in parallel transitions in the eight-state trellis. Hence, 16-QAM involves two parallel transitions in each branch of the trellis. More generally, the choice of an $M = 2^{m+1}$-point QAM signal constellation implies that the eight-state trellis contains 2^{m-2} parallel transitions in each branch.

The assignment of signal subsets to transitions is based on the same set of basic (heuristic) rules described above for the 8-PSK signal constellation. Thus, the four (branches) transitions originating from or leading to the same state are assigned either the subsets D_0, D_2, D_4, D_6 or D_1, D_3, D_5, D_7. Parallel transitions are assigned signal points contained within the corresponding subsets. This eight-state trellis code provides a coding gain of 4 dB. The euclidean distance of parallel transitions exceeds the free euclidean distance, and, hence, the code performance is not limited by parallel transitions.

Larger size trellis codes for M-QAM provide even larger coding gains. For

TABLE 8-3-1 CODING GAINS FOR TRELLIS-CODED PAM SIGNALS

Number of states	k_1	Code rate $\frac{k_1}{k_1+1}$	$m=1$ coding gain (dB) of 4-PAM versus uncoded 2-PAM	$m=2$ coding gain (dB) of 8-PAM versus uncoded 4-PAM	$m\to\infty$ asymptotic coding gain (dB)	$m\to\infty$ N_{fed}
4	1	1/2	2.55	3.31	3.52	4
8	1	1/2	3.01	3.77	3.97	4
16	1	1/2	3.42	4.18	4.39	8
32	1	1/2	4.15	4.91	5.11	12
64	1	1/2	4.47	5.23	5.44	36
128	1	1/2	5.05	5.81	6.02	66

Source: Ungerboeck (1987).

example, trellis codes with 2^v states for an $M = 2^{m+1}$ QAM signal constellation can be constructed by convolutionally encoding k_1 input bits into $k_1 + 1$ output bits. Thus, a rate $R_c = k_1/(k_1 + 1)$ convolutional code is employed for this purpose. Usually, the choice of $k_1 = 2$ provides a significant fraction of the total coding gain that is achievable. The additional $k_2 = m - k_1$ input bits are uncoded, and are transmitted in each signal interval by selecting signal points within a subset.

Tables 8-3-1 to 8-3-3, taken from the paper by Ungerboeck (1987), provide a summary of coding gains achievable with trellis-coded modulation. Table 8-3-1 summarizes the coding gains achieved for trellis-coded (one-dimensional) PAM modulation with rate 1/2 trellis codes. Note that the coding gain with a 128-state trellis code is 5.8 dB for octal PAM, which is close to the channel cutoff rate R_0 and less than 4 dB from the channel capacity limit for error rates in the range of 10^{-6}–10^{-8}. We should also observe that the number of paths

TABLE 8-3-2 CODING GAINS FOR TRELLIS-CODED 16-PSK MODULATION

Number of states	k_1	Code rate $\frac{k_1}{k_1+1}$	$m=3$ coding gain (dB) of 16-PSK versus uncoded 8-PSK	$m\to\infty$ N_{fed}
4	1	1/2	3.54	4
8	1	1/2	4.01	4
16	1	1/2	4.44	8
32	1	1/2	5.13	8
64	1	1/2	5.33	2
128	1	1/2	5.33	2
256	2	2/3	5.51	8

Source: Ungeboeck (1987).

TABLE 8-3-3 CODING GAINS FOR TRELLIS-CODED QAM MODULATION

Number of states	k_1	Code rate $\frac{k_1}{k_1+1}$	$m=3$ gain (dB) of 16-QAM versus uncoded 8-QAM	$m=4$ gain (dB) of 32-QAM versus uncoded 16-QAM	$m=5$ gain (dB) of 64-QAM versus uncoded 32-QAM	$m=\infty$ asymptotic coding gain (dB)	N_{fed}
4	1	1/2	3.01	3.01	2.80	3.01	4
8	2	2/3	3.98	3.98	3.77	3.98	16
16	2	2/3	4.77	4.77	4.56	4.77	56
32	2	2/3	4.77	4.77	4.56	4.77	16
64	2	2/3	5.44	5.44	4.23	5.44	56
128	2	2/3	6.02	6.02	5.81	6.02	344
256	2	2/3	6.02	6.02	5.81	6.02	44

Source: Ungerboeck (1987).

N_{fed} with free euclidean distance D_{fed} becomes large with an increase in the number of states.

Table 8-3-2 lists the coding gain for trellis-coded 16-PSK. Again, we observe that the coding gain for eight or more trellis stages exceeds 4 dB, relative to uncoded 8-PSK. A simple rate 1/2 code yields 5.33 dB gain with a 128-stage trellis.

Table 8-3-3 contains the coding gains obtained with trellis-coded QAM signals. Relatively simple rate 2/3 trellis codes yield a gain of 6 dB with 128 trellis stages for $m = 3$ and 4.

The results in these tables clearly illustrate the significant coding gains that are achievable with relatively simple trellis codes. A 6 dB coding gain is close to the cutoff rate R_0 for the signal sets under consideration. Additional gains that would lead to transmission in the vicinity of the channel capacity bound are difficult to attain without a significant increase in coding/decoding complexity.

Since the channel capacity provides the ultimate limit on code performance, we should emphasize that continued partitioning of large signal sets quickly leads to signal point separation within any subset that exceeds the free euclidean distance of the code. In such cases, parallel transitions are no longer the limiting factor on D_{fed}. Usually, a partition to eight subsets is sufficient to obtain a coding gain of 5–6 dB with simple rate 1/2 or rate 2/3 trellis codes with either 64 or 128 trellis stages, as indicated in Tables 8-3-1 to 8-3-3.

Convolutional encoders for the linear trellis codes listed in Tables 8-3-1 to 8-3-3 for the M-PAM, M-PSK, and M-QAM signal constellations are given in the papers by Ungerboeck (1982, 1987). The encoders may be realized either with feedback or without feedback. For example Fig. 8-3-11 illustrates three feedback-free convolutional encoders corresponding to 4-, 8-, and 16-state trellis codes for 8-PSK and 16-QAM signal constellations. Equivalent realizations of these trellis codes based on systematic convolutional encoders with

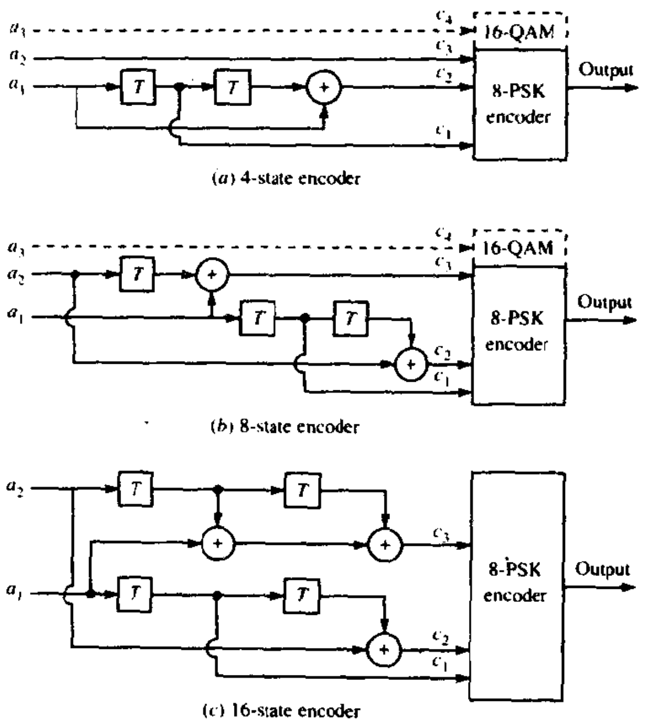


FIGURE 8-3-11 Minimal feedback-free convolutional encoders for 8-PSK and 16-QAM signals. [*From Ungerboeck (1982). © 1982 IEEE.*]

feedback are shown in Fig. 8-3-12. Usually, the systematic convolutional encoders are preferred in practical applications.

A potential problem with linear trellis codes is that the modulated signal sets are not usually invariant to phase rotations. This poses a problem in practical applications where differential encoding is usually employed to avoid phase ambiguities when a receiver must recover the carrier phase after a temporary loss of signal. The problem of phase invariance and differential encoding/decoding was solved by Wei (1984a, b), who devised linear and nonlinear trellis codes that are rotationally invariant under either 180° or 90° phase rotations, respectively. For example, Fig. 8-3-13 illustrates a nonlinear eight-state convolutional encoder for a 32-QAM rectangular signal constellation that is invariant under 90° phase rotations. This trellis code has been adopted as an international standard for 9600 and 14,000 bits/s (high-speed) telephone line modems.

Trellis-coded modulation schemes have also been developed for multidimensional signals. In practical systems, multidimensional signals are transmitted as a sequence of either one-dimensional (PAM) or two-dimensional (QAM) signals. Trellis codes based on 4-, 8-, and 16-dimensional signal constellations have been constructed, and some of these codes have been

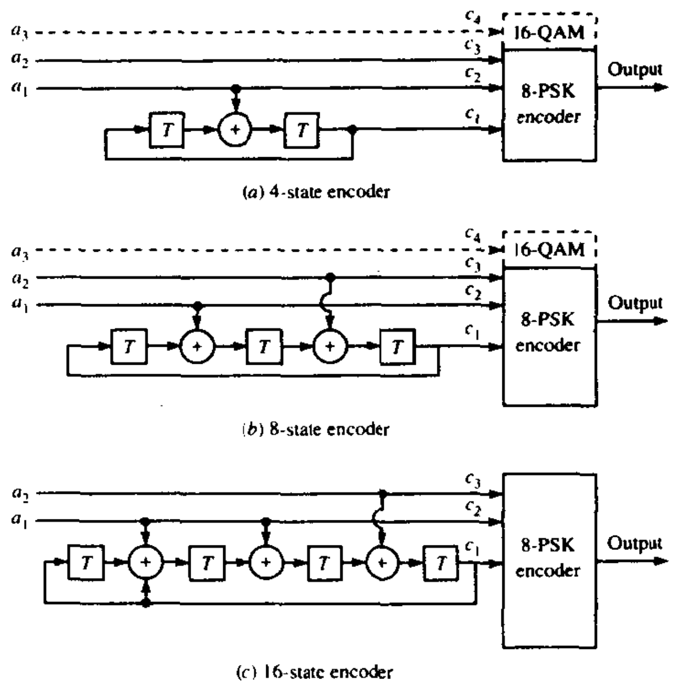
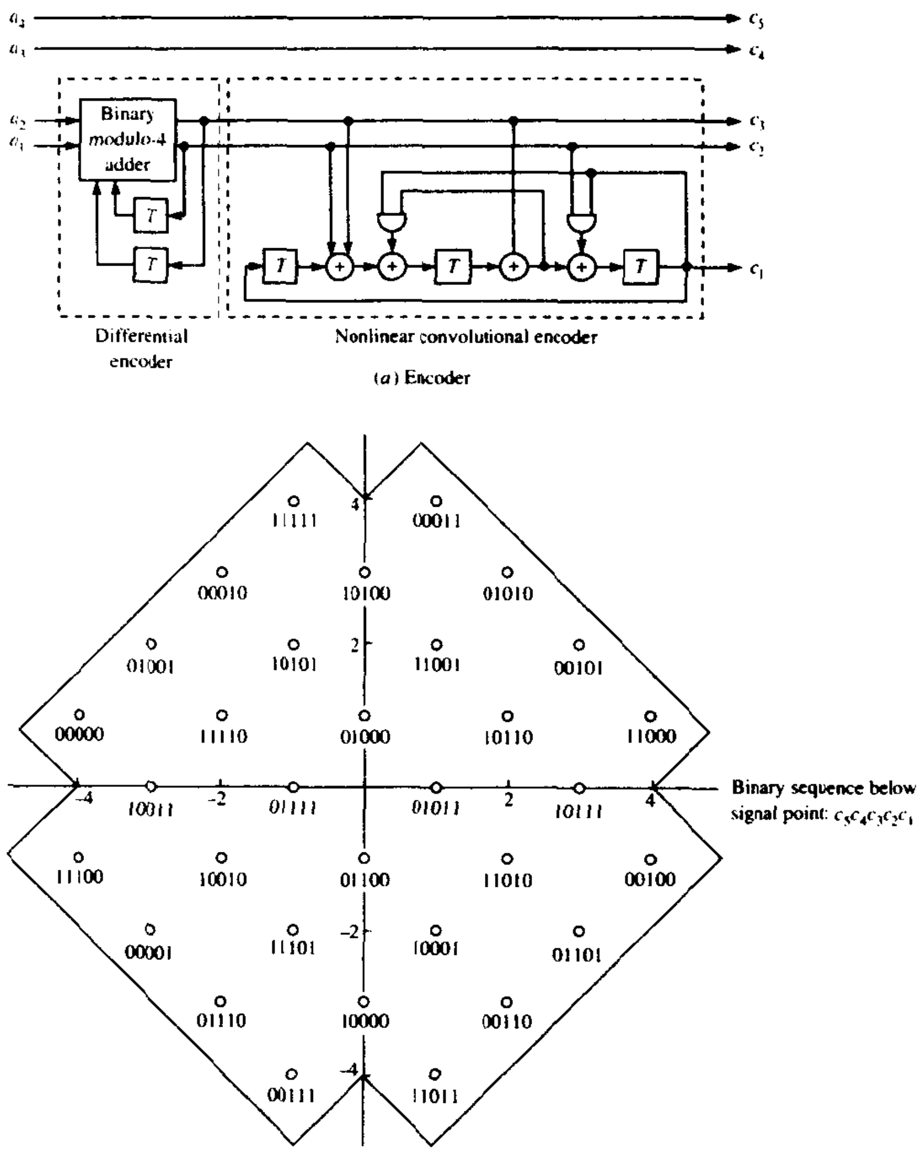


FIGURE 8-3-12 Equivalent realizations of systematic convolutional encoders with feedback for 8-PSK and 16-QAM. [*From Ungerboeck (1982). © 1982 IEEE.*]

implemented in commercially available modems. A potential advantage of trellis-coded multidimensional signals is that we can use smaller constituent two-dimensional signal constellations that allow for a trade-off between coding gain and implementation complexity. The papers by Wei (1987), Ungerboeck (1987), Gersho and Lawrence (1984), and Forney *et al.* (1984) treat multidimensional signal constellations for trellis-coded modulation.

Finally, we should mention that a new design technique for trellis-coded modulation based on lattices and cosets of a sublattice has been described by Calderbank and Sloane (1987) and Forney (1988). This method for constructing trellis codes provides an alternative to the set partitioning method described above. However, the two methods are closely related. In this alternative method, a block of k_1 bits is fed to a convolutional encoder. Each block of k_1 input bits produces an output symbol that is a coset of the sublattice Λ', which is a subset of the chosen lattice. A second block of k_2 input bits is used to select one of the points in the coset at the output of the convolutional encoder. It is apparent that the cosets of the sublattice are akin to the subsets in set partitioning and the elements of the cosets are akin to the signal points within a subset. This new method has led to the discovery of new powerful trellis codes involving larger signal constellations, many of which are listed in the paper by Calderbank and Sloane (1987).



(b) 32-point QAM (cross) signal

FIGURE 8-3-13 Eight-state nonlinear convolutional encoder for 32-QAM signal set that exhibits invariance under 90° phase rotations.

8-4 BIBLIOGRAPHICAL NOTES AND REFERENCES

The pioneering work on coding and coded waveforms for digital communications was done by Shannon (1948a, b), Hamming (1950), and Golay (1949). These works were rapidly followed with papers on code performance by

Gilbert (1952), new codes by Muller (1954) and Reed (1954), and coding techniques for noisy channels by Elias (1954, 1955) and Slepian (1956).

During the period 1960–1970, there were a number of significant contributions in the development of coding theory and decoding algorithms. In particular, we cite the papers by Reed and Solomon (1960) on Reed–Solomon codes, the papers by Hocquenghem (1959) and Bose and Ray-Chaudhuri (1960a, b) on BCH codes, and the Ph.D dissertation of Forney (1966a) on concatenated codes. These works were followed by the papers of Goppa (1970, 1971) on the construction of a new class of linear cyclic codes, now called Goppa codes (see also Berlekamp, 1973), and the paper of Justesen (1972) on a constructive technique for asymptotically good codes. During this period, work on decoding algorithms was primarily focused on BCH codes. The first decoding algorithm for binary BCH codes was developed by Peterson (1960). A number of refinements and generalizations by Chien (1964), Forney (1965), Massey (1965), and Berlekamp (1968) led to the development of a computationally efficient algorithm for BCH codes, which is described in detail by Lin and Costello (1983).

In parallel with these developments on block codes are the developments in convolutional codes, which were invented by Elias (1955). The major problem in convolutional coding was decoding. Wozencraft and Reiffen (1961) described a sequential decoding algorithm for convolutional codes. This algorithm was later modified and refined by Fano (1963), and it is now called the *Fano algorithm*. Subsequently, the stack algorithm was devised by Ziganzirov (1966) and Jelinek (1969), and the Viterbi algorithm was devised by Viterbi (1967). The optimality and the relatively modest complexity for small constraint lengths have served to make the Viterbi algorithm the most popular in decoding of convolutional codes with $K \leq 10$.

One of the most important contributions in coding during the 1970s was the work of Ungerboeck and Csajka (1976) on coding for bandwidth-constrained channels. In this paper, it was demonstrated that a significant coding gain can be achieved through the introduction of redundancy in a bandwidth-constrained channel and trellis codes were described for achieving coding gains of 3–4 dB. This work has generated much interest among researchers and has led to a large number of publications over the past 10 years. A number of references can be found in the papers by Ungerboeck (1982, 1987) and Forney *et al.* (1984). Additional papers on coded modulation for bandwidth-constrained channels may also be found in the Special Issue on Voiceband Telephone Data Transmission, *IEEE Journal on Selected Areas in Communication* (September 1984). A comprehensive treatment of trellis-coded modulation is given in the book by Biglieri *et al.* (1991).

In addition to the references given above on coding, decoding, and coded signal design, we should mention the collection of papers published by the IEEE Press entitled *Key Papers in the Development of Coding Theory*, edited by Berlekamp (1974). This book contains important papers that were published in the first 25 years of coding theory. We should also cite the Special

Issue on Error-Correcting Codes, *IEEE Transactions on Communications* (October 1971).

PROBLEMS

8-1 The generator matrix for a linear binary code is

$$\mathbf{G}=\begin{bmatrix}0&0&1&1&1&0&1\\0&1&0&0&1&1&1\\1&0&0&1&1&1&0\end{bmatrix}$$

a Express **G** in systematic $[\mathbf{I}\mid\mathbf{P}]$ form.
b Determine the parity check matrix **H** for the code.
c Construct the table of syndromes for the code.
d Determine the minimum distance of the code.
e Demonstrate that the code word corresponding to the informatin sequence 101 is orthogonal to **H**.

8-2 List the code words generated by the matrices given in (8-1-35) and (8-1-37), and, thus, demonstrate that these matrices generate the same set of code words.

8-3 The weight distribution of Hamming codes is known. Expressed as a polynomial in powers of x, the weight distribution for the binary Hamming codes of block length n is

$$\begin{aligned}A(x)&=\sum_{i=0}^{n}A_i x^i\\&=\frac{1}{n+1}\left[(1+x)^n+n(1+x)^{(n-1)/2}(1-x)^{(n+1)/2}\right]\end{aligned}$$

where A_i is the number of code words of weight i. Use this formula to determine the weight distribution of the (7, 4) Hamming code and check your result with the list of code words given in Table 8-1-2.

8-4 The polynomial

$$g(p)=p^4+p+1$$

is the generator for the (15, 11) Hamming binary code.
a Determine a generator matrix **G** for this code in systematic form.
b Determine the generator polynomial for the dual code.

8-5 For the (7, 4) cyclic Hamming code with generator polynomial $g(p)=p^3+p^2+1$, construct an (8, 4) extended Hamming code and list all the code words. What is $d_{\min}$ for the extended code?

8-6 An (8, 4) linear block code is constructed by shortening a (15, 11) Hamming code generated by the generator polynomial $g(p)=p^4+p+1$.
a Construct the code words of the (8, 4) code and list them.
b What is the minimum distance of the (8, 4) code?

8-7 The polynomial $p^{15}+1$ when factored yields

$$\begin{aligned}p^{15}+1&=(p^4+p^3+1)(p^4+p^3+p^2+p+1)\\&\quad\times(p^4+p+1)(p^2+p+1)(p+1)\end{aligned}$$

a Construct a systematic (15, 5) code using the generator polynomial

$$g(p)=(p^4+p^3+p^2+p+1)(p^4+p+1)(p^2+p+1)$$

b What is the minimum distance of the code?
c How many random errors per code word can be corrected?
d How many errors can be detected by this code?
e List the code words of a (15, 2) code constructed from the generator polynomial

$$g(p) = (p^{15} + 1)/(p^2 + p + 1)$$

and determine the minimum distance.

8-8 Construct the parity check matrices $\mathbf{H}_1$ and $\mathbf{H}_2$ corresponding to the generator matrices $\mathbf{G}_1$ and $\mathbf{G}_2$ given by (8-1-34) and (8-1-35), respectively.

8-9 Construct an extended (8, 4) code from the (7, 4) Hamming code by specifying the generator matrix and the parity check matrix.

8-10 A systematic (6, 3) code has the generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Construct the standard array and determine the correctable error patterns and their corresponding syndromes.

8-11 Construct the standard array for the (7, 3) code with generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and determine the correctable patterns and their corresponding syndromes.

8-12 Determine the correctable error patterns (of least weight) and their syndromes for the systematic (7, 4) cyclic Hamming code.

8-13 Prove that if the sum of two error patterns $\mathbf{e}_1$ and $\mathbf{e}_2$ is a valid code word $\mathbf{C}_j$ then each pattern has the same syndrome.

8-14 Let $g(p) = p^8 + p^6 + p^4 + p^2 + 1$ be a polynomial over the binary field.
a Find the lowest-rate cyclic code whose generator polynomial is $g(p)$. What is the rate of this code?
b Find the minimum distance of the code found in (a).
c What is the coding gain for the code found in (a).

8-15 The polynomial $g(p) = p + 1$ over the binary field is considered.
a Show that this polynomial can generate a cyclic code for any choice of n. Find the corresponding k.
b Find the systematic form of $\mathbf{G}$ and $\mathbf{H}$ for the code generated by $g(p)$.
c Can you say what type of code this generator polynomial generates?

8-16 Design a (6, 2) cyclic code by choosing the shortest possible generator polynomial.
a Determine the generator matrix $\mathbf{G}$ (in the systematic form) for this code and find all possible code words.
b How many errors can be corrected by this code?

8-17 Prove that any two n-tuples in the same row of a standard array add to produce a valid code word.

8-18 Beginning with a (15, 7) BCH code, construct a shortened (12, 4) code. Give the generator matrix for the shortened code.

8-19 In Section 8-1-2, it was indicated that when an (n, k) Hadamard code is mapped into waveforms by means of binary PSK, the corresponding $M = 2^k$ waveforms are orthogonal. Determine the bandwidth expansion factor for the M orthogonal waveforms and compare this with the bandwidth requirements of orthogonal FSK detected coherently.

8-20 Show that the signaling waveforms generated from a maximum-length shift-register code by mapping each bit in a code word into a binary PSK signal are equicorrelated with correlation coefficient $\rho_r = -1/(M-1)$, i.e., the M waveforms form a simplex set.

8-21 Compute the error probability obtained with a (7, 4) Hamming code on an AWGN channel, both for hard-decision and soft-decision decoding. Use (8-1-50), (8-1-52), (8-1-82), (8-1-90), and (8-1-91).

8-22 Use the results in Section 2-1-6 to obtain the Chernoff bound for hard-decision decoding given by (8-1-89) and (8-1-90). Assume that the all-zero code word is transmitted and determine an upper bound on the probability that code word C_m, having weight w_m, is selected. This occurs if $\frac{1}{2}w_m$ or more bits are in error. To apply the Chernoff bound, define a sequence of w_m random variables as

$$X_i = \begin{cases} 1 & \text{with probability } p \\ -1 & \text{with probability } 1-p \end{cases}$$

where $i = 1, 2, \ldots, w_m$, and p is the probability of error. For the BSC, the $\{X_i\}$ are statistically independent.

8-23 A convolutional code is described by

$$\mathbf{g}_1 = [1 \quad 0 \quad 0], \qquad \mathbf{g}_2 = [1 \quad 0 \quad 1], \qquad \mathbf{g}_3 = [1 \quad 1 \quad 1]$$

a Draw the encoder corresponding to this code.
b Draw the state-transition diagram for this code.
c Draw the trellis diagram for this code.
d Find the transfer function and the free distance of this code.
e Verify whether or not this code is catastrophic.

8-24 The convolutional code of Problem 8-23 is used for transmission over a AWGN channel with hard-decision decoding. The output of the demodulator detector is (101001011110111 . . .). Using the Viterbi algorithm, find the transmitted sequence.

8-25 Repeat Problem 8-23 for a code with

$$\mathbf{g}_1 = [1 \quad 1 \quad 0], \qquad \mathbf{g}_2 = [1 \quad 0 \quad 1], \qquad \mathbf{g}_3 = [1 \quad 1 \quad 1]$$

8-26 The block diagram of a binary convolutional code is shown in Fig. P8-26.
a Draw the state diagram for the code.
b Find the transfer function of the code, $T(D)$.
c What is d_{free}, the minimum free distance of the code?

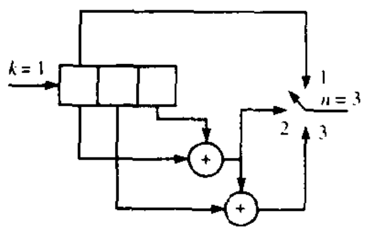


FIGURE P8-26

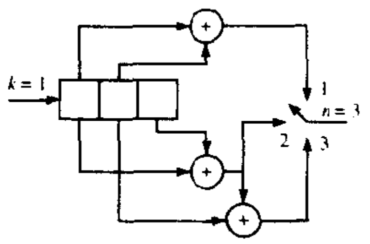


FIGURE P8-27

d Assume that a message has been encoded by this code and transmitted over a binary-symmetric channel with an error probability of $p = 10^{-5}$. If the received sequence is $\mathbf{r} = (110, 110, 110, 111, 010, 101, 101)$, using the Viterbi algorithm, find the transmitted bit sequence.

e Find an upper bound to the bit error probability of the code when the above binary-symmetric channel is employed. Make any reasonable approximation.

8-27 The block diagram of a (3, 1) convolutional code is shown in Fig. P8-27.

a Draw the state diagram of the code.

b Find the transfer function $T(D)$ of the code.

c Find the minimum free distance (d_{free}) of the code and show the corresponding path (at distance d_{free} from the all-zero code word) on the trellis.

d Assume that four information bits (x_1, x_2, x_3, x_4), followed by two zero bits, have been encoded and sent via a binary-symmetric channel with crossover probability equal to 0.1. The received sequence is (111, 111, 111, 111, 111, 111). Use the Viterbi decoding algorithm to find the most likely data sequence.

8-28 In the convolutional code generated by the encoder shown in Fig. P8-28.

a Find the transfer function of the code in the form $T(N, D)$.

b Find d_{free} of the code.

c If the code is used on a channel using hard-decision Viterbi decoding, assuming the crossover probability of the channel is $p = 10^{-6}$, use the hard-decision bound to find an upper bound on the average bit error probability of the code.

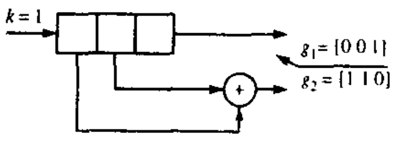


FIGURE P8-28

8-29 Figure P8-29 depicts a rate 1/2, constraint length $K = 2$, convolutional code.

a Sketch the tree diagram, the trellis diagram, and the state diagram.

b Solve for the transfer function $T(D, N, J)$ and, from this, specify the minimum free distance.

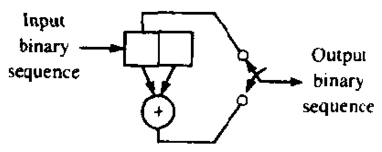


FIGURE P8-29

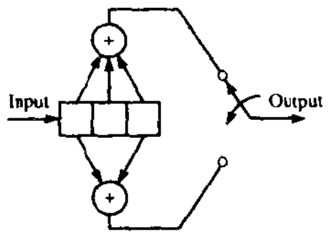


FIGURE P8-30

8-30 A rate 1/2, $K = 3$, binary convolutional encoder is shown in Fig. P8-30.
a Draw the tree diagram, the trellis diagram, and the state diagram.
b Determine the transfer function $T(D, N, J)$ and, from this, specify the minimum free distance.

8-31 Sketch the convolutional encoders for the following codes:
a rate 1/2, $K = 5$, maximum free distance code (Table 8-2-1);
b rate 1/3, $K = 5$, maximum free distance code (Table 8-2-2);
c rate 2/3, $K = 2$, maximum free distance code (Table 8-2-8).

8-32 Draw the state diagram for the rate 2/3, $K = 2$, convolutional code indicated in Problem 8-31(c) and, for each transition, show the output sequence and the distance of the output sequence from the all-zero sequence.

8-33 Consider the $K = 3$, rate 1/2, convolutional code shown in Fig. P8-30. Suppose that the code is used on a binary symmetric channel and the received sequence for the first eight branches is 0001100000001001. Trace the decisions on a trellis diagram and label the survivors' Hamming distance metric at each node level. If a tie occurs in the metrics required for a decision, always choose the upper path (arbitrary choice).

8-34 Use the transfer function derived in Problem 8-30 for the $R_c = 1/2$, $K = 3$, convolutional code to compute the probability of a bit error for an AWGN channel with (a) hard-decision and (b) soft-decision decoding. Compare the performance by plotting the results of the computation on the same graph.

8-35 Use the generators given by (8-2-36) to obtain the encoder for a dual-3, rate 1/2 convolutional code. Determine the state diagram and derive the transfer function $T(D, N, J)$.

8-36 Draw the state diagram for the convolutional code generated by the encoder shown in Fig. P8-36 and, thus, determine if the code is catastrophic or noncatastrophic. Also, give an example of a rate 1/2, $K = 4$, convolutional encoder that exhibits catastrophic error propagation.

8-37 A trellis coded signal is formed as shown in Fig. P8-37 by encoding one bit by use

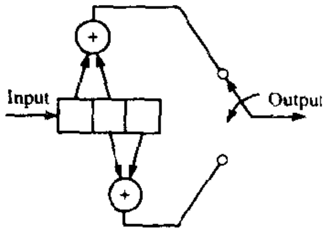


FIGURE P8-36

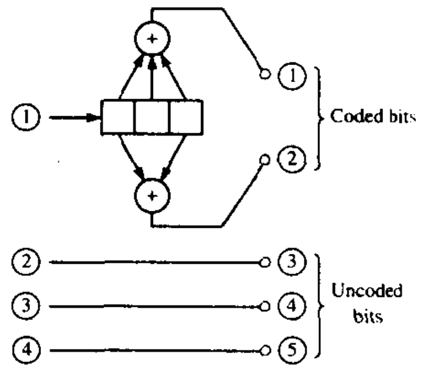


FIGURE P8-37

of a rate 1/2 convolutional code, while three additional information bits are left uncoded. Perform the set partitioning of a 32-QAM (cross) constellation and indicate the subsets in the partition. By how much is the distance between adjacent signal points increased as a result of partitioning?

8-38 Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two code words of length n with distance d and assume that these two code words are transmitted via a binary-symmetric channel with crossover probability p. Let $P(d)$ denote the error probability in transmission of these two code words.

a Show that

$$P(d) \leq \sum_{i=1}^{2^n} \sqrt{p(\mathbf{y}_i \mid \mathbf{x}_1)p(\mathbf{y}_i \mid \mathbf{x}_2)}$$

where the summation is over all binary sequences $\mathbf{y}_i$.

b From the above, conclude that

$$P(d) \leq [4p(1-p)]^{d/2}$$