CHAPTER 17

Low-Density Parity-Check Codes

Besides turbo codes, low-density parity-check (LDPC) codes form another class of Shannon limit (or *channel capacity*)–approaching codes. LDPC codes were first discovered by Gallager [1, 2] in the early 1960s. Unfortunately, Gallager's remarkable discovery was mostly ignored by coding researchers for almost 20 years, until Tanner's work in 1981, in which he provided a new interpretation of LDPC codes from a graphical point of view. Tanner's work was also ignored by coding theorists for another 14 years, until the late 1990s when some coding researchers began to investigate codes on graphs and iterative decoding. Their research resulted in the rediscovery of Gallager's LDPC codes and further generalizations. Long LDPC codes with iterative decoding based on belief propagation have been shown to achieve an error performance only a fraction of a decibel away from the Shannon limit [4, 5, 9, 10, 12–14, 18–20]. This discovery makes the LDPC codes strong competitors with turbo codes for error control in many communication and digital storage systems where high reliability is required. LDPC codes have some advantages over turbo codes: (1) they do not require a long interleaver to achieve good error performance; (2) they have better block error performance; (3) their error floor occurs at a much lower BER; and (4) their decoding is not trellis based.

Although Gallager proposed LDPC codes for error control, he did not provide a specific method for constructing good LDPC codes algebraically and systematically, as the BCH, RS, and finite geometry codes are constructed; however, he did propose a general method for constructing a class of pseudorandom LDPC codes. Good LDPC codes that have been found are largely computer generated, especially long codes, and their encoding is very complex owing to the lack of structure. Kou, Lin, and Fossorier [15–20] introduced the first algebraic and systematic construction of LDPC codes based on finite geometries. The large classes of finite-geometry LDPC codes have relatively good minimum distances, and their Tanner graphs do not contain short cycles. These codes can be decoded in various ways ranging from low to high decoding complexity and from reasonably good error performance to very good error performance. Furthermore, these codes are either cyclic or quasi-cyclic. Consequently, their encoding is simple and can be implemented with linear shift registers. Some long finite-geometry LDPC codes have error performance only a few tenths of a decibel away from the Shannon limit.

This chapter presents LDPC codes and their construction based on various methods. Also included are various methods for decoding these codes, ranging from low to high decoding complexities and reasonable to large coding gains.

17.1 INTRODUCTION TO LDPC CODES

As described in Chapter 3, a linear block code C of length n is uniquely specified by either a generator matrix $\mathbb{G}$ or a parity-check matrix $\mathbb{H}$. If it is specified by a parity-check matrix $\mathbb{H}$, the code C is simply the null space of $\mathbb{H}$. An n-tuple $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ over $GF(2)$ is a codeword if and only if $\mathbf{v}\mathbb{H}^T = \mathbb{0}$. This simply says that the code bits of a codeword in C must satisfy a set of parity-check equations specified by the rows of $\mathbb{H}$. LDPC codes are specified in terms of their parity-check matrices.

Definition 17.1 An LDPC code is defined as the null space of a parity-check matrix $\mathbb{H}$ that has the following structural properties: (1) each row consists of ρ 1's; (2) each column consists of γ 1's; (3) the number of 1's in common between any two columns, denoted by λ, is no greater than 1; that is, $\lambda = 0$ or 1; and (4) both ρ and γ are small compared with the length of the code and the number of rows in $\mathbb{H}$ [1, 2].

Properties (1) and (2) simply say that the parity-check matrix $\mathbb{H}$ has constant row and column weights ρ and γ, respectively. Property (3) implies that no two rows of $\mathbb{H}$ have more than one 1 in common. Because both ρ and γ are small compared with the code length and the number of rows in the matrix, $\mathbb{H}$ therefore has a small density of 1's. For this reason, $\mathbb{H}$ is said to be a *low-density parity-check matrix*, and the code specified by $\mathbb{H}$ is hence called an LDPC code. We define the *density* r of the parity-check matrix $\mathbb{H}$ as the ratio of the total number of 1's in $\mathbb{H}$ to the total number of entries in $\mathbb{H}$. Then, we readily see that

$$r = \rho/n = \gamma/J,$$

where J is the number of rows in $\mathbb{H}$. The low density of $\mathbb{H}$ simply implies that $\mathbb{H}$ is a sparse matrix. The LDPC code given by Definition 17.1 is called a (γ, ρ)-*regular* LDPC code. If all the columns or all the rows of the parity check matrix $\mathbb{H}$ do not have the same weight, an LDPC code is then said to be *irregular*. In this chapter we are mainly concerned with regular LDPC codes; irregular codes are considered in Section 17.15. Note that the rows of $\mathbb{H}$ are not necessarily linearly independent over $GF(2)$. In this case, to determine the dimension of the code, it is necessary to determine the rank of $\mathbb{H}$.

EXAMPLE 17.1

Consider the matrix $\mathbb{H}$ given in Figure 17.1. Each column and each row of this matrix consist of four 1's, respectively. It can be checked easily that no two columns (or two rows) have more than one 1 in common. The density of this matrix is 0.267. Therefore, it is a low-density matrix. The null space of this matrix gives a (15, 7) LDPC code with a minimum distance of 5. It will be shown in a later section that this code is cyclic and is a BCH code.

Let k be a positive integer greater than 1. For a given choice of ρ and γ, Gallager gave the following construction of a class of linear codes specified by their parity-check matrices. Form a $k\gamma \times k\rho$ matrix $\mathbb{H}$ that consists of γ $k \times k\rho$ submatrices,

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

FIGURE 17.1: The parity-check matrix of a (15, 7) LDPC code.

$\mathbb{H}_1, \mathbb{H}_2, \cdots, \mathbb{H}_\gamma$. Each row of a submatrix has ρ 1's, and each column of a submatrix contains a single 1. Therefore, each submatrix has a total of $k\rho$ 1's. For $1 \leq i \leq k$, the ith row of $\mathbb{H}_1$ contains all its ρ 1's in columns $(i-1)\rho + 1$ to $i\rho$. The other submatrices are merely *column permutations* of $\mathbb{H}_1$. Then,

$$\mathbb{H} = \begin{bmatrix} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \vdots \\ \mathbb{H}_\gamma \end{bmatrix}. \tag{17.1}$$

From this construction of $\mathbb{H}$, it is clear that (1) no two rows in a submatrix of $\mathbb{H}$ have any 1-component in common; and (2) no two columns of a submatrix of $\mathbb{H}$ have more that one 1 in common. Because the total number of ones in $\mathbb{H}$ is $k\rho\gamma$ and the total number of entries in $\mathbb{H}$ is $k^2\rho\gamma$, the density of $\mathbb{H}$ is $1/k$. If k is chosen much greater than 1, $\mathbb{H}$ has a very small density and is a sparse matrix. Now, whether $\mathbb{H}$ possesses the third property of the parity-check matrix of an LDPC code defined in Definition 17.1 (that is, no two columns of $\mathbb{H}$ have more than one 1 in common) depends on the choice of the $\gamma - 1$ column permutations of the submatrix $\mathbb{H}_1$. Therefore, the null space of $\mathbb{H}$ gives a linear block code of length $n = k\rho$, but the code is not necessarily an LDPC code with $\lambda = 0$ or 1. Random permutations of columns of $\mathbb{H}_1$ to form the other submatrices result in a class of linear block codes that contains a subclass of LDPC codes defined by Definition 17.1.

Gallager did not provide a method for choosing column permutations of the submatrix $\mathbb{H}_1$ to form the other submatrices, $\mathbb{H}_2, \mathbb{H}_3, \cdots, \mathbb{H}_\gamma$, such that the overall matrix $\mathbb{H}$ gives an LDPC code with good minimum distance and the structural properties required in Definition 17.1. Computer searches are needed to find good

LDPC codes, especially for long codes. Recently, such long LDPC codes have been constructed that achieve an error performance only a few tenths (or a hundredth) of a decibel away from the Shannon limit [4, 9–14, 21, 22].

EXAMPLE 17.2

Let $k = 5$, $\rho = 4$, and $\gamma = 3$. Using Gallager's construction, we can form a 15×20 matrix $\mathbb{H}$ as shown in Figure 17.2, which consists of three 5×20 submatrices, $\mathbb{H}_1$, $\mathbb{H}_2$, and $\mathbb{H}_3$. Each row of $\mathbb{H}_1$ consists of four consecutive 1's, and no two rows have any 1-component in common. Submatrices $\mathbb{H}_2$ and $\mathbb{H}_3$ are obtained by two different permutations of columns of $\mathbb{H}_1$ such that no two columns (or two rows) of $\mathbb{H}$ have more than one 1 in common. The density of $\mathbb{H}$ is $r = 0.20$. Consequently, the null space of $\mathbb{H}$ gives an LDPC code. It is a (20, 7) linear code with a minimum distance of 6 (see Problem 17.11).

Consider an LDPC code C of length n specified by a $J \times n$ parity-check matrix $\mathbb{H}$. Let $\mathbb{h}_1, \mathbb{h}_2, \cdots, \mathbb{h}_J$ denote the rows of $\mathbb{H}$ where

$$\mathbb{h}_j = (h_{j,0}, h_{j,1}, \cdots, h_{j,n-1}),$$

for $1 \leq j \leq J$. Let $\mathbb{v} = (v_0, v_1, \cdots, v_{n-1})$ be a codeword in C. Then, the inner product

$$\mathbb{s}_j = \mathbb{v} \cdot \mathbb{h}_j = \sum_{l=0}^{n-1} v_l h_{j,l} = 0 \tag{17.2}$$

gives a parity-check sum (or parity-check equation). There are a total of J such parity-check sums specified by the J rows of $\mathbb{H}$. A code bit v_l is said to be checked by

$$\mathbb{H} = \left[\begin{array}{cccccccccccccccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}\right]$$

FIGURE 17.2: The parity-check matrix of a Gallager LDPC code.

the sum $\mathbf{v} \cdot \mathbb{h}_j$ (or by the row $\mathbb{h}_j$) if $h_{j,l} = 1$. For $0 \leq l < n$, let $A_l = \{\mathbb{h}_1^{(l)}, \mathbb{h}_2^{(l)}, \cdots, \mathbb{h}_\gamma^{(l)}\}$ denote the set of rows in $\mathbb{H}$ that check on the code bit v_l. For $1 \leq j \leq \gamma$, let

$$\mathbb{h}_j^{(l)} = (h_{j,0}^{(l)}, h_{j,1}^{(l)}, \cdots, h_{j,n-1}^{(l)}). \tag{17.3}$$

Then, $h_{1,l}^{(l)} = h_{2,l}^{(l)} = \cdots = h_{\gamma,l}^{(l)} = 1$. It follows from the third structural property of the parity-check matrix of an LDPC code that any code bit other than v_l is checked by at most one row in A_l. Therefore, the γ rows in A_l are orthogonal on the code bit v_l (see Section 8.1 for the definition of orthogonality). Let S_l denote the set of parity-check sums formed by the rows in A_l. Then, the code bit v_l is contained in every parity-check sum in S_l, and each of the other $n - 1$ code bits is contained in at most one parity-check sum in S_l. The parity-check sums in S_l are said to be orthogonal on the code bit v_l. For every code bit of a regular LDPC code C, there are γ parity-check sums that are orthogonal on it. Consequently, C can be decoded with one-step majority-logic decoding, as discussed in Chapter 8. Any error pattern with $\lfloor \gamma/2 \rfloor$ or fewer errors can be corrected. Consequently, the minimum distance d_{min} of the code is at least $\gamma + 1$; that is, $d_{min} \geq \gamma + 1$. If γ is too small, one-step majority-logic decoding of an LDPC code will give very poor error performance. Gallager proposed two algorithms for decoding LDPC codes [1, 2], one hard-decision and one soft-decision, that give much better error performance than the simple one-step majority-logic decoding.

17.2 TANNER GRAPHS FOR LINEAR BLOCK CODES

A graph $\mathcal{G}$ consists of a set of vertices, denoted by $\mathcal{V} = \{v_1, v_2, \cdots\}$, and a set of edges, denoted by $\mathcal{E} = \{e_1, e_2, \cdots\}$, such that each edge e_k is identified with an unordered pair (v_i, v_j) of vertices. Such a graph $\mathcal{G}$ is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The vertices v_i and v_j associated with edge e_k are called the *end vertices* of e_k. A graph is most commonly represented by a diagram in which the vertices are represented as points and each edge as a line joining its end vertices. With this graphical representation, the two end vertices of an edge are said to be connected by the edge, and the edge is said to be *incident* with (or on) its end vertices. The number of edges that are incident on a vertex v_i is called the *degree*, denoted by $d(v_i)$, of vertex v_i. Figure 17.3 shows a graph consisting of six vertices and 10 edges. Edge b connects vertices v_1

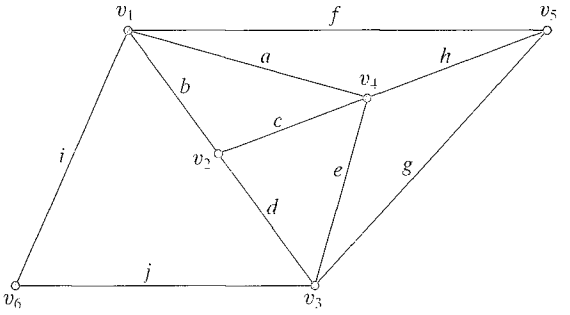


FIGURE 17.3: A graph with six vertices and 10 edges.

and v_2. Edges b, c, and d are incident on vertex v_2, and hence the degree of vertex v_2 is 3. Two edges that are incident on a common vertex are said to be *adjacent* or *connected*. Two vertices are said to be adjacent if they are connected by an edge. For example, in Figure 17.3, edges a and h are adjacent and are connected at vertex v_4. Vertices v_3 and v_5 are adjacent. A graph with a finite number of vertices as well as a finite number of edges is called a *finite graph*. There are many good books on graph theory, but reference [35] provides a good introduction to this subject.

A *path* in a graph is defined as a finite alternating sequence of vertices and edges, beginning and ending with vertices, such that each edge is incident with the vertices preceding and following it, and no vertex appears more than once. The number of edges on a path is called the *length* of the path. In Figure 17.3, for instance, $v_1, b, v_2, c, v_4, h, v_5, g, v_3$ is a path of length 4. It is possible for a path to begin and end at the same vertex. Such a closed path is called a *cycle*. No vertex on a cycle (except the initial and the final vertex) appears more than once. For example, the closed path $v_1, b, v_2, c, v_4, a, v_1$ in Figure 17.3 is a cycle of length 3. The closed path $v_1, f, v_5, g, v_3, j, v_6, i, v_1$ is a cycle of length 4. An edge for which the initial and terminal vertices are the same forms a cycle of length 1 and is called a *self-loop*. A graph without cycles is said to be *acyclic* and is called a *tree*. Figure 17.4 shows an acyclic graph. The length of the shortest cycle in a graph is called the *girth* of the graph. The girth of the graph in Figure 17.3 is 3.

A graph $\mathcal{G}$ is said to be *connected* if there is at least one path between every pair of vertices in $\mathcal{G}$. Both graphs shown in Figures 17.3 and 17.4 are connected graphs. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called a *bipartite* graph if its vertex set $\mathcal{V}$ can be partitioned into two disjoint subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ such that every edge in $\mathcal{E}$ joins a vertex in $\mathcal{V}_1$ with a vertex in $\mathcal{V}_2$, and no two vertices in either $\mathcal{V}_1$ or $\mathcal{V}_2$ are connected. It is obvious that a bipartite graph has no self-loop. Figure 17.5 shows a bipartite graph, with $\mathcal{V}_1 = \{v_1, v_2, v_3\}$ and $\mathcal{V}_2 = \{v_4, v_5, v_6, v_7, v_8\}$. If a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains cycles, then all the cycles have even lengths. To see that this is true, suppose we trace a cycle in $\mathcal{G}$ from a vertex v_i in $\mathcal{V}_1$. The first edge of this cycle must connect vertex v_i to a vertex v_j in $\mathcal{V}_2$. The second edge of the cycle must connect v_j to a vertex v_k in $\mathcal{V}_1$. The third edge of the cycle connects v_k to a vertex v_l in $\mathcal{V}_2$. Then, the fourth edge of the cycle connects v_l to a vertex in $\mathcal{V}_1$. This tracing process continues until the last edge of the cycle terminates at the starting vertex v_i. Leaving a vertex in $\mathcal{V}_1$ and returning to a vertex in $\mathcal{V}_1$ requires two edges each time. Therefore, the length of the cycle must be a multiple of 2, an even number.

The vertices of a graph are also commonly called *nodes*. In the following discussion, vertex and node are used interchangeably.

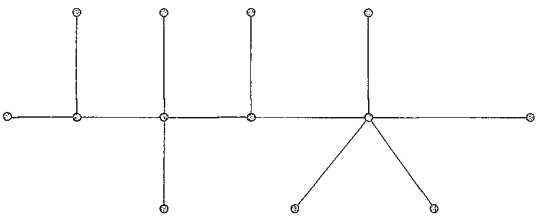


FIGURE 17.4: An acyclic graph.

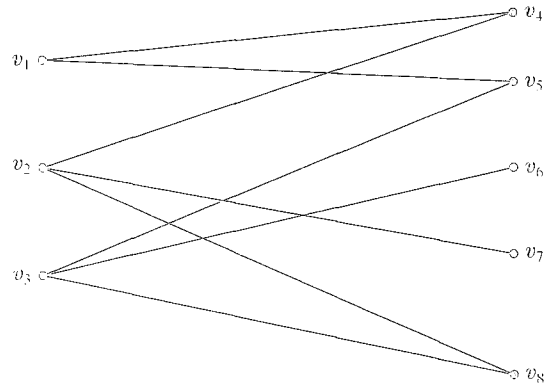


FIGURE 17.5: A bipartite graph.

Linear codes can be represented by graphs in various ways [39]. The most well known graphical representation of codes is the trellis representation, discussed in Chapter 9 for linear block codes and in Chapter 12 for convolutional codes. Trellis representation of a code makes it possible to devise trellis-based decoding algorithms to achieve MLD or near MLD with a significant reduction in decoding complexity, as we showed in Chapters 12 and 14. Another useful graphical representation of linear block codes is the representation of a linear block code by a *Tanner graph* [3], which displays the incidence relationship between the code bits of the code and the parity-check sums that check on them.

For a linear block code of length n specified by a parity-check matrix $\mathbb{H}$ with J rows, $\mathbb{h}_1, \mathbb{h}_2, \cdots, \mathbb{h}_J$, we construct a graph $\mathcal{G}_T$ that consists of two sets of vertices, $\mathcal{V}_1$ and $\mathcal{V}_2$. The first set $\mathcal{V}_1$ consists of n vertices that represent the n code bits of the code. These vertices, denoted by $v_0, v_1, \cdots, v_{n-1}$, are called the *code-bit vertices* (or *variable nodes*). The second set $\mathcal{V}_2$ consists of J vertices that represent the J parity-check sums (or equations), $s_1, s_2, \cdots, s_J$, given by (17.2) that the code bits must satisfy. These vertices are called the *check-sum vertices* (or *check nodes*). A code-bit vertex v_l is connected to a check-sum vertex s_j by an edge, denoted by (v_l, s_j), if and only if the code bit v_l is contained in (or checked by) the parity-check sum s_j. No two code-bit vertices are connected, and no two check-sum vertices are connected. It is clear that $\mathcal{G}_T$ is a bipartite graph. This graph was first proposed by Tanner [3] to study the structure of LDPC codes for iterative decoding and hence is called a Tanner graph. The degree of a code-bit vertex v_l is simply equal to the number of the parity-check sums that contain v_l, and the degree of a check-sum vertex s_j is simply equal to the number of code bits that are checked by s_j.

For a regular LDPC code, the degrees of all the code-bit vertices in the Tanner graph of the code are the same and equal to γ (the column weight of the parity-check matrix); and the degrees of all the check-sum vertices are the same and equal to ρ (the row weight of the parity-check matrix). Such a Tanner graph is said to be *regular*. Furthermore, it follows from Definition 17.1 that no two code bits of an LDPC code are checked by two different parity-check sums. This implies that the Tanner graph $\mathcal{G}_T$ of an LDPC code does not contain cycles of length 4. To decode

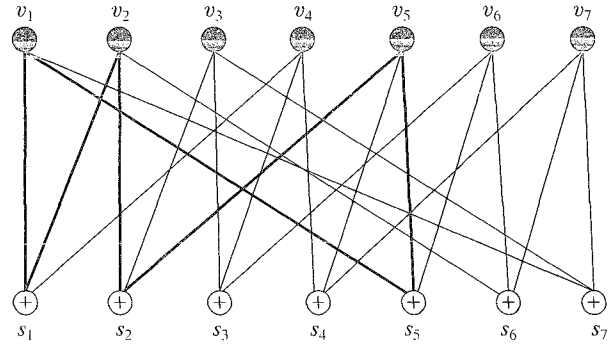


FIGURE 17.6: The Tanner graph of a (7, 3) linear block code.

an LDPC code (or any linear block code) with *iterative decoding based on belief propagation* (IDBP) [23, 26], it is important that the Tanner graph of the code does not contain cycles of short lengths, such as lengths 4 and 6—especially cycles of length 4. Short cycles limit the decoding performance and prevent the decoding to converge to MLD [10, 25, 28, 36–39].

EXAMPLE 17.3

Consider a (7, 3) linear block code which is the null space of the following parity-check matrix.

$$\mathbb{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

It is easy to check that this code is an LDPC code. Its Tanner graph is shown in Figure 17.6. The graph does not contain cycles of length 4, but it does contain cycles of length 6; one such cycle is marked by heavy lines. Therefore, the girth of this Tanner graph is 6.

17.3 A GEOMETRIC CONSTRUCTION OF LDPC CODES

LDPC codes can be constructed algebraically based on the points and lines of finite geometries, such as Euclidean and projective geometries over finite fields, presented in Chapter 8. In this section, we present geometric construction of two types of LDPC codes based on a finite geometry. These two types of LDPC codes are closely related, and their Tanner graphs are dual to each other.

Let $\mathbb{Q}$ be a finite geometry with n points and J lines that has the following fundamental structural properties: (1) every line consists of ρ points; (2) every point

lies on γ lines (i.e., every point is intersected by γ lines); (3) two points are connected by one and only one line; and (4) two lines are either disjoint (i.e., they have no point in common) or they intersect at one and only one point. We denote the points and lines in $\mathbb{Q}$ with $\{\mathbb{p}_1, \mathbb{p}_2, \cdots, \mathbb{p}_n\}$ and $\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_J\}$, respectively. Let

$$\mathbb{v} = (v_1, v_2, \cdots, v_n)$$

be an n-tuple over $GF(2)$ whose components correspond to the n points of geometry $\mathbb{Q}$, where the ith component v_i corresponds to the ith point $\mathbb{p}_i$ of $\mathbb{Q}$. Let $\mathcal{L}$ be a line in $\mathbb{Q}$. We define a vector based on the points on $\mathcal{L}$ as follows:

$$\mathbb{v}_{\mathcal{L}} = (v_1, v_2, \cdots, v_n),$$

where

$$v_i = \begin{cases} 1 & \text{if } \mathbb{p}_i \text{ is a point on } \mathcal{L}, \\ 0 & \text{otherwise.} \end{cases}$$

This vector $\mathbb{v}_{\mathcal{L}}$ is simply the incidence vector of line $\mathcal{L}$ defined in Chapter 8, which displays the points on $\mathcal{L}$. It is clear that the weight of this incidence vector is ρ.

We form a $J \times n$ matrix $\mathbb{H}_{\mathbb{Q}}^{(1)}$ whose rows are the incidence vectors of the J lines of the finite geometry $\mathbb{Q}$ and whose columns correspond to the n points of $\mathbb{Q}$. It follows from the fundamental structural properties of the finite geometry $\mathbb{Q}$ and the definition of the incidence vector of a line in $\mathbb{Q}$ that matrix $\mathbb{H}_{\mathbb{Q}}^{(1)}$ has the following properties: (1) each row has ρ 1s; (2) each column has γ 1s (this number is simply the number of lines in $\mathbb{Q}$ that intersect at the point that corresponds to the column); (3) no two rows have more than one 1 in common (this follows from the fundamental property of geometry $\mathbb{Q}$ that two lines are either disjoint or intersect at one and only one point in $\mathbb{Q}$); and (4) no two columns have more than one 1 in common; that is, $\lambda = 0$ or 1 (this follows from property 3). The density of this matrix $\mathbb{H}_{\mathbb{Q}}^{(1)}$ is $r = \rho/n = \gamma/J$. If ρ and γ are small compared with n and J, then the density r of $\mathbb{H}_{\mathbb{Q}}^{(1)}$ is small, and hence $\mathbb{H}_{\mathbb{Q}}^{(1)}$ is a low-density (or sparse) matrix.

The null space of $\mathbb{H}_{\mathbb{Q}}^{(1)}$ gives an LDPC code of length n. This code is called the *type-I geometry-*$\mathbb{Q}$ LDPC code, denoted by $C_{\mathbb{Q}}^{(1)}$, which has a minimum distance of at least $\gamma + 1$. The Tanner graph of this code is a regular bipartite graph with n code bit vertices and J check-sum vertices. Each code bit vertex has degree γ, and each check-sum vertex has degree ρ. The graph contains no cycles of length 4. It does contain cycles of length 6 (the proof of this is left as a problem).

To construct the second type of LDPC code, we need to form vectors corresponding to the points of the finite geometry $\mathbb{Q}$. Let

$$\mathbb{v} = (v_1, v_2, \cdots, v_J)$$

be a J-tuple over $GF(2)$ whose components correspond to the J lines of $\mathbb{Q}$, where the ith component v_i corresponds to the ith line $\mathcal{L}_i$ of $\mathbb{Q}$. Let $\mathbb{p}$ be a point in $\mathbb{Q}$. We define a J-tuple based on the lines in $\mathbb{Q}$ that intersect at $\mathbb{p}$ as follows:

$$\mathbb{v}_{\mathbb{p}} = (v_1, v_2, \cdots, v_J),$$

where

$$v_i = \begin{cases} 1 & \text{if the } i\text{th line } \mathcal{L}_i \text{ of } \mathbb{Q} \text{ contains the point } \mathbb{p}, \\ 0 & \text{otherwise.} \end{cases}$$

This vector $\mathbf{v}_{\mathbb{p}}$ simply displays the lines that intersect at $\mathbb{p}$, and hence it is called the *intersecting* (or incidence) vector of point $\mathbb{p}$.

We form an $n \times J$ matrix $\mathbb{H}_{\mathbb{Q}}^{(2)}$ whose rows are the intersecting vectors of the n points in the finite-geometry $\mathbb{Q}$ and whose columns correspond to the J lines of $\mathbb{Q}$. It follows from the definition of the incidence vector of a line in $\mathbb{Q}$ that the columns of $\mathbb{H}_{\mathbb{Q}}^{(2)}$ are simply the incidence vectors of the J lines in $\mathbb{Q}$. Furthermore, it follows from the definition of the intersecting vector of a point in $\mathbb{Q}$ that the columns of $\mathbb{H}_{\mathbb{Q}}^{(1)}$ are simply the intersecting vectors of the n points in $\mathbb{Q}$. Consequently, $\mathbb{H}_{\mathbb{Q}}^{(2)}$ is simply the *transpose* of $\mathbb{H}_{\mathbb{Q}}^{(1)}$ and vice versa; that is,

$$\begin{aligned} \mathbb{H}_{\mathbb{Q}}^{(2)} &= [\mathbb{H}_{\mathbb{Q}}^{(1)}]^T, \\ \mathbb{H}_{\mathbb{Q}}^{(1)} &= [\mathbb{H}_{\mathbb{Q}}^{(2)}]^T. \end{aligned} \tag{17.4}$$

Matrix $\mathbb{H}_{\mathbb{Q}}^{(2)}$ has the following properties: (1) each row has weight γ; (2) each column has weight ρ; (3) no two columns have more than one 1 in common; and (4) no two rows have more than one 1 in common. The density of $\mathbb{H}_{\mathbb{Q}}^{(2)}$ is the same as the density of $\mathbb{H}_{\mathbb{Q}}^{(1)}$. The ranks of $\mathbb{H}_{\mathbb{Q}}^{(1)}$ and $\mathbb{H}_{\mathbb{Q}}^{(2)}$ are the same.

The null space of $\mathbb{H}_{\mathbb{Q}}^{(2)}$ gives an LDPC code of length J with a minimum distance of at least $\rho + 1$. This LDPC code is called the *type-II geometry-$\mathbb{Q}$* LDPC code, denoted by $C_{\mathbb{Q}}^{(2)}$. $C_{\mathbb{Q}}^{(1)}$ and $C_{\mathbb{Q}}^{(2)}$ are called *companion* codes. Because $\mathbb{H}_{\mathbb{Q}}^{(1)}$ and $\mathbb{H}_{\mathbb{Q}}^{(2)}$ have the same rank, $C_{\mathbb{Q}}^{(1)}$ and $C_{\mathbb{Q}}^{(2)}$ have the same number of parity check bits. It follows from (17.4) and structural properties of $\mathbb{H}_{\mathbb{Q}}^{(1)}$ and $\mathbb{H}_{\mathbb{Q}}^{(2)}$ that the Tanner graphs of $C_{\mathbb{Q}}^{(1)}$ and $C_{\mathbb{Q}}^{(2)}$ are *dual* graphs; that is, the code-bit vertices of one graph become the check-sum vertices of the other graph, and the check-sum vertices of one graph become the code-bit vertices of the other graph.

There are two families of well-known finite geometries, Euclidean and projective geometries over finite fields, which were presented in Chapter 8. Based on these two families of finite geometries, four classes of finite-geometry LDPC codes can be constructed:

1. type-I Euclidean geometry (EG)-LDPC codes,
2. type-II EG-LDPC codes,
3. type-I projective geometry (PG)-LDPC codes,
4. type-II PG-LDPC codes.

17.4 EG-LDPC CODES

Consider the m-dimensional Euclidean geometry over $GF(2^s)$, EG$(m, 2^s)$, whose structure was described in Chapter 8. This geometry consists of 2^{ms} points, and each point is represented by an m-tuple over $GF(2^s)$. The point represented by the

all-zero m-tuple, $\mathbb{0} = (0, 0, \cdots, 0)$, is called the *origin*. There are

$$J = \frac{2^{(m-1)s}(2^{ms} - 1)}{2^s - 1} \tag{17.5}$$

lines in $\mathrm{EG}(m, 2^s)$, and each line consists of 2^s points. Each point in $\mathrm{EG}(m, 2^s)$ is intersected by

$$\gamma = \frac{2^{ms} - 1}{2^s - 1} \tag{17.6}$$

lines. Two lines in $\mathrm{EG}(m, 2^s)$ are either disjoint or they intersect at one and only point. Each line in $\mathrm{EG}(m, 2^s)$ has $2^{(m-1)s} - 1$ lines parallel to it. From (17.5) we can readily see that there are $(2^{ms} - 1)/(2^s - 1)$ bundles of parallel lines, and each parallel bundle consists of $2^{(m-1)s}$ parallel lines.

To construct a type-I EG-LDPC code based on $\mathrm{EG}(m, 2^s)$, we form the parity-check matrix $\mathbb{H}_{EG}^{(1)}$ whose rows are the incidence vectors of all the lines in $\mathrm{EG}(m, 2^s)$ and whose columns correspond to all the points in $\mathrm{EG}(m, 2^s)$. Therefore, $\mathbb{H}_{EG}^{(1)}$ consists of

$$J = \frac{2^{(m-1)s}(2^{ms} - 1)}{2^s - 1}$$

rows and $n = 2^{ms}$ columns. Because each line in $\mathrm{EG}(m, 2^s)$ consists of 2^s points, each row of $\mathbb{H}_{EG}^{(1)}$ has weight $\rho = 2^s$. Since each point in $\mathrm{EG}(m, 2^s)$ is intersected by $(2^{ms} - 1)/(2^s - 1)$ lines, each column of $\mathbb{H}_{EG}^{(1)}$ has weight $\gamma = (2^{ms} - 1)/(2^s - 1)$. The density r of $\mathbb{H}_{EG}^{(1)}$ is

$$r = \frac{\rho}{n} = \frac{2^s}{2^{ms}} = 2^{-(m-1)s}. \tag{17.7}$$

For $m \geq 2$ and $s \geq 2$, $r \leq 1/4$ and $\mathbb{H}_{EG}^{(1)}$ is a low-density parity-check matrix. The null space of $\mathbb{H}_{EG}^{(1)}$ hence gives an LDPC code of length $n = 2^{ms}$, which is called an m-dimensional type-I $(0, s)$th-order EG-LDPC code, denoted by $C_{EG}^{(1)}(m, 0, s)$. The minimum distance of this code is lower bounded as follows:

$$d_{min} \geq \gamma + 1 = \frac{2^{ms} - 1}{2^s - 1} + 1. \tag{17.8}$$

Because $(2^{ms} - 1)/(2^s - 1)$ orthogonal check-sums can be formed for each code bit, this code is capable of correcting $t_{ML} = \lfloor (2^{ms} - 1)/2(2^s - 1) \rfloor$ or fewer random errors with one-step majority-logic decoding, as discussed in Chapter 8.

To construct an m-dimensional type-II EG-LDPC code, we take the transpose of $\mathbb{H}_{EG}^{(1)}$, which gives the parity-check matrix

$$\mathbb{H}_{EG}^{(2)} = [\mathbb{H}_{EG}^{(1)}]^T.$$

Matrix $\mathbb{H}_{EG}^{(2)}$ consists of $J = 2^{ms}$ rows and $n = 2^{(m-1)s}(2^{ms} - 1)/(2^s - 1)$ columns. Each row of $\mathbb{H}_{EG}^{(2)}$ has weight $\rho = (2^{ms} - 1)/(2^s - 1)$, and each column of $\mathbb{H}_{EG}^{(2)}$ has weight $\gamma = 2^s$. The null space of $\mathbb{H}_{EG}^{(2)}$ gives an LDPC code of length $n =$

$2^{(m-1)s}(2^{ms}-1)/(2^s-1)$ with a minimum distance of at least 2^s+1. This code is referred to as an m-dimensional type-II $(0, s)$th-order EG-LDPC code, denoted by $C_{EG}^{(2)}(m, 0, s)$.

EXAMPLE 17.4

Let $m = 2$ and $s = 4$. The two-dimensional Euclidean geometry EG$(2, 2^4)$ consists of 256 points and 272 lines. Therefore, $\mathbb{H}_{EG}^{(1)}$ is a 272×256 matrix with the following parameters: $\rho = 16, \gamma = 17, \lambda = 0$ or 1, and density $r = 0.0624$. The null space of $\mathbb{H}_{EG}^{(1)}$ gives the two-dimensional type-I $(0, 4)$th-order EG-LDPC code $C_{EG}^{(1)}(2, 0, 4)$ of length 256 with a minimum distance of exactly 18. The dimension of this code is 175 (or the rank of $\mathbb{H}_{EG}^{(1)}$ is 81). Hence, it is a (256, 175) LDPC code. The two-dimensional type-II $(0, 4)$th-order EG-LDPC code $C_{EG}^{(2)}(2, 0, 4)$ is the null space of $\mathbb{H}_{EG}^{(2)} = [\mathbb{H}_{EG}^{(1)}]^T$, which is a 256×272 matrix with parameters $\rho = 17, \gamma = 16, \lambda = 0$ or 1, and density $r = 0.0624$. The length of this code is $n = 272$, and its minimum distance is exactly 17. $C_{EG}^{(2)}(2, 0, 4)$ has the same number of parity bits as the type-I code $C_{EG}^{(1)}(2, 0, 4)$, which is 81. Therefore, $C_{EG}^{(2)}$ is a (272, 191) LDPC code.

In constructing the type-I and type-II EG-LDPC codes based on EG$(m, 2^s)$, we can remove the origin point of the geometry and all the lines passing through the origin. In this case, the type-I EG-LDPC code can be put in *cyclic* form and the type-II EG-LDPC code can be put in *quasi-cyclic* form. This simplifies the encoding circuit.

Let $GF(2^{ms})$ be the extension field of $GF(2^s)$. Each element of $GF(2^{ms})$ can be represented as an m-tuple over $GF(2^s)$. As pointed out in Chapter 8, $GF(2^{ms})$ forms the m-dimensional Euclidean geometry EG$(m, 2^s)$ over $GF(2^s)$. The elements of $GF(2^{ms})$ give the 2^{ms} points of EG$(m, 2^s)$, and the zero element 0 of $GF(2^{ms})$ is the origin of EG$(m, 2^s)$. Let α be a primitive element of $GF(2^{ms})$. Then, $\alpha^0 = 1, \alpha, \alpha^2, \cdots, \alpha^{2^{ms}-2}$ are all the $2^{ms}-1$ nonorigin points of EG$(m, 2^s)$. Let

$$\mathbb{v} = (v_0, v_1, \cdots, v_{2^{ms}-2})$$

be a $(2^{ms}-1)$-tuple over $GF(2)$ whose components correspond to the $2^{ms}-1$ nonorigin points of EG$(m, 2^s)$, where v_i corresponds to the point α^i with $0 \le i < 2^{ms}-1$. Let $\mathcal{L}$ be a line in EG$(m, 2^s)$ that does not pass through the origin. Based on $\mathcal{L}$, we form a $(2^{ms}-1)$-tuple over $GF(2)$ as follows:

$$\mathbb{v}_{\mathcal{L}} = (v_0, v_1, \cdots, v_{2^{ms}-2})$$

whose ith component

$$v_i = \begin{cases} 1 & \text{if } \alpha^i \text{ is a point on } \mathcal{L}, \\ 0 & \text{otherwise.} \end{cases}$$

The vector $\mathbf{v}_{\mathcal{L}}$ is the incidence vector of the line $\mathcal{L}$. It follows from (17.5) and (17.6) that there are

$$J_0 = \frac{(2^{(m-1)s} - 1)(2^{ms} - 1)}{2^s - 1} \tag{17.9}$$

lines in $\mathrm{EG}(m, 2^s)$ that do not pass through the origin.

Let $\mathbb{H}_{EG,c}^{(1)}$ be a matrix whose rows are the incidence vectors of all the J_0 lines in $\mathrm{EG}(m, 2^s)$ that do not pass through the origin and whose columns correspond to the $n = 2^{ms} - 1$ nonorigin points of $\mathrm{EG}(m, 2^s)$. This matrix has the following properties: (1) each row has weight $\rho = 2^s$; (2) each column has weight $\gamma = (2^{ms} - 1)/(2^s - 1) - 1$; (3) no two columns have more than one 1 in common; that is, $\lambda = 0$ or 1; and (4) no two rows have more than one 1 in common. The density of $\mathbb{H}_{EG,c}^{(1)}$ is

$$r = \frac{2^s}{2^{ms} - 1}. \tag{17.10}$$

Again, for $m \geq 2$ and $s \geq 2$, r is relatively small compared with 1. Therefore, $\mathbb{H}_{EG,c}^{(1)}$ is a low-density matrix.

Let $C_{EG,c}^{(1)}(m, 0, s)$ be the null space of $\mathbb{H}_{EG,c}^{(1)}$. Then, $C_{EG,c}^{(1)}(m, 0, s)$ is an LDPC code of length $n = 2^{ms} - 1$ with a minimum distance of at least

$$\gamma + 1 = \frac{2^{ms} - 1}{2^s - 1}. \tag{17.11}$$

It turns out that this LDPC code is the $(0, s)$th-order EG code defined by Definition 8.3 and hence is cyclic. We call this code an m-dimensional type-I cyclic $(0, s)$th-order EG-LDPC code. Because it is cyclic, it is completely specified by its generator polynomial $g_{EG,c}^{(1)}(X)$. Let α be a primitive element of $GF(2^{ms})$. Let h be a nonnegative integer less than $2^{ms} - 1$. For a nonnegative integer l, let $h^{(l)}$ be the remainder resulting from dividing $2^l h$ by $2^{ms} - 1$. Then, it follows from Theorem 8.3 that $g_{EG,c}^{(1)}(X)$ has α^h as a root if and only if

$$0 < \max_{0 \leq l < s} W_{2^s}(h^{(l)}) \leq (m - 1)(2^s - 1), \tag{17.12}$$

where $W_{2^s}(h^{(l)})$ is the 2^s-weight of $h^{(l)}$ defined by (8.28).

Let h_0 be the smallest integer that does not satisfy the condition of (17.12). It can be shown [42] that

$$\begin{aligned} h_0 &= (2^s - 1) + (2^s - 1)2^s + \cdots + (2^s - 1)2^{(m-3)s} + 2^{(m-2)s} + 2^{(m-1)s} \\ &= 2^{(m-1)s} + 2^{(m-2)s+1} - 1. \end{aligned} \tag{17.13}$$

Therefore, $g_{EG,c}^{(1)}(X)$ has the following sequence of consecutive powers of α:

$$\alpha, \alpha^2, \cdots, \alpha^{h_0 - 1} \tag{17.14}$$

as roots [42]. It follows from the BCH bound [43–45] (see Section 7.2) that the minimum distance of the m-dimensional type-I cyclic $(0, s)$th-order EG-LDPC code $C_{EG,c}^{(1)}(m, 0, s)$ is lower bounded as follows:

$$d_{EG,c}^{(1)} \geq 2^{(m-1)s} + 2^{(m-2)s+1} - 1. \tag{17.15}$$

This BCH bound is tighter than the bound given by (17.11). The only case for which these two bounds on minimum distance are equal is $m = 2$. In this case, both bounds are $2^s + 1$. In fact, in Section 8.5, we proved that for $m = 2$ the minimum distance of the $(0,s)$th-order EG code is exactly $2^s + 1$.

The number of parity-check symbols of $C^{(1)}_{EG,c}(m, 0, s)$ is equal to the degree of its generator polynomial $g^{(1)}_{EG,c}(X)$. A combinatorial expression for this number has been derived in [46].

A special subclass of type-I cyclic EG-LDPC codes is the class of two-dimensional type-I cyclic $(0, s)$th-order EG-LDPC codes. It follows from (8.33) that the number of parity-check bits for the two-dimensional type-I cyclic $(0, s)$th-order EG-LDPC code $C^{(1)}_{EG,c}(2, 0, s)$ of length $n = 2^{2s} - 1$ constructed based on the lines of EG$(2, 2^s)$ is

$$n - k = 3^s - 1. \tag{17.16}$$

Therefore, $C^{(1)}_{EG,c}(2, 0, s)$ has the following parameters:

$$\begin{array}{ll} \text{Length} & n = 2^{2s} - 1, \\ \text{Number of parity bits} & n - k = 3^s - 1, \\ \text{Dimension} & k = 2^{2s} - 3^s, \\ \text{Minimum distance} & d_{min} = 2^s + 1, \\ \text{Density} & r = \frac{2^s}{2^{2s}-1}. \end{array} \tag{17.17}$$

For this special case, the geometry EG$(2, 2^s)$ contains $2^{2s} - 1$ lines that do not pass through the origin. Therefore, the parity-check matrix $\mathbb{H}^{(1)}_{EG,c}$ of the $C^{(1)}_{EG,c}(2, 0, s)$ code is a $(2^{2s} - 1) \times (2^{2s} - 1)$ square matrix. $\mathbb{H}^{(1)}_{EG,c}$ can be constructed easily by taking the incidence vector $\mathbf{v}_{\mathcal{L}}$ of a line $\mathcal{L}$ in EG$(2, 2^s)$ that does not pass through the origin and then cyclically shifting this incidence vector $\mathbf{v}_{\mathcal{L}}$ $2^{2s} - 2$ times. This results in $2^{2s} - 1$ incidence vectors for the $2^{2s} - 1$ distinct lines in EG$(2, 2^s)$ that do not pass through the origin. The incidence vector $\mathbf{v}_{\mathcal{L}}$ and its $2^{2s} - 2$ cyclic shifts form the rows of $\mathbb{H}^{(1)}_{EG,c}$. Therefore, $\mathbb{H}^{(1)}_{EG,c}$ is a square circulant matrix. Both the row weight ρ and the column weight γ are equal to 2^s; that is, $\rho = \gamma = 2^s$. A list of two-dimensional type-I cyclic $(0, s)$th-order EG-LDPC codes is given in Table 17.1.

TABLE 17.1: Two-dimensional type-I cyclic $(0, s)$th-order EG-LDPC codes

s	n	k	d_{min}	ρ	γ	r
2	15	7	5	4	4	0.267
3	63	37	9	8	8	0.127
4	255	175	17	16	16	0.0627
5	1023	781	33	32	32	0.0313
6	4095	3367	65	64	64	0.01563
7	16383	14197	129	128	128	0.007813

EXAMPLE 17.5

Let $m = 2$ and $s = 2$. Consider the Galois field $GF(2^{2\times 2})$ generated by the primitive polynomial $p(X) = 1 + X + X^4$, which is given in Table 2.8. Regard this field as the two-dimensional Euclidean geometry EG$(2, 2^2)$ over $GF(2^2)$. Let α be a primitive element of $GF(2^{2\times 2})$ and $\beta = \alpha^5$. Then $\{0, 1, \beta, \beta^2\}$ form the subfield $GF(2^2)$. Every line in EG$(2, 2^2)$ consists of four points. The set of four points $\{\alpha^{14} + \eta\alpha\}$ with $\eta \in GF(2^2)$ is $\{\alpha^7, \alpha^8, \alpha^{10}, \alpha^{14}\}$, which forms a line in EG$(2, 2^2)$. This line does not pass through the origin of EG$(2, 2^2)$. The incidence vector of this line (excluding the origin point) is (000000011010001). This vector and its 14 cyclic shifts form the parity-check matrix $\mathbb{H}^{(1)}_{EG,c}$ shown in Figure 17.1. The null space of this matrix is the two-dimensional type-I cyclic (0,2)th-order EG-LDPC code $C^{(1)}_{EG,c}(2, 0, 2)$ of length 15, which is the (15, 7) LDPC code given in Example 17.1 and is the first code listed in Table 17.1. It follows from (17.12) that the generator polynomial $g^{(1)}_{EG,c}(X)$ of this cyclic LDPC code has $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^6, \alpha^8, \alpha^9$, and α^{12} as all its roots. The roots $\alpha, \alpha^2, \alpha^4$, and α^8 are conjugates, and they have the same minimal polynomial, $\Phi_1(X) = 1 + X + X^4$. The roots $\alpha^3, \alpha^6, \alpha^{12}$, and α^9 are conjugates, and their minimal polynomial is $\Phi_3(X) = 1 + X + X^2 + X^3 + X^4$. Then, $g^{(1)}_{EG,c}(X) = \Phi_1(X)\Phi_3(X) = 1 + X^4 + X^6 + X^7 + X^8$. This code is actually the (15, 7) double-error-correcting BCH code considered in Examples 6.1 and 8.1.

The companion code of the m-dimensional type-I cyclic $(0, s)$th-order EG-LDPC code $C^{(1)}_{EG,c}$ is the null space of the parity-check matrix

$$\mathbb{H}^{(2)}_{EG,qc} = [\mathbb{H}^{(1)}_{EG,c}]^T. \tag{17.18}$$

This LDPC code has length

$$n = J_0 = \frac{(2^{(m-1)s} - 1)(2^{ms} - 1)}{2^s - 1} \tag{17.19}$$

and a minimum distance d_{min} of at least $2^s + 1$. It is not cyclic but can be put in quasi-cyclic form. We call this code an m-dimensional type-II quasi-cyclic $(0, s)$th-order EG-LDPC code, denoted by $C^{(2)}_{EG,qc}(m, 0, s)$. To put $C^{(2)}_{EG,qc}(m, 0, s)$ in quasi-cyclic form, we partition the J_0 incidence vectors of lines in EG$(m, 2^s)$ not passing through the origin into

$$K = \frac{2^{(m-1)s} - 1}{2^s - 1} \tag{17.20}$$

cyclic classes. Each of these K cyclic classes contains $2^{ms} - 1$ incidence vectors, which are obtained by cyclically shifting any incidence vector in the class $2^{ms} - 1$ times. For each cyclic class of incidence vectors, we choose a representative, and the rest of the incidence vectors are generated by cyclically shifting this representative. Now, we form a $(2^{ms} - 1) \times K$ matrix $\mathbb{H}_0$ whose K columns are the K representative incidence vectors of the K cyclic classes. For $1 \leq i \leq 2^{ms} - 2$, let $\mathbb{H}_i$ be a $(2^{ms} - 1) \times K$ matrix whose columns are the ith downward cyclic shifts of the columns of $\mathbb{H}_0$. We form

the matrix $\mathbb{H}^{(2)}_{EG,qc}$ as follows:

$$\mathbb{H}^{(2)}_{EG,qc} = [\mathbb{H}_0\ \mathbb{H}_1\ \mathbb{H}_2\ \cdots\ \mathbb{H}_{2^{ms}-2}]. \tag{17.21}$$

Then, the null space of $\mathbb{H}^{(2)}_{EG,qc}$ gives the type-II EG-LDPC code $C^{(2)}_{EG,qc}(m, 0, s)$ in quasi-cyclic form. Every K cyclic shifts of a codeword in $C^{(2)}_{EG,qc}(m, 0, s)$ is also a codeword.

$\mathbb{H}^{(2)}_{EG,qc}$ can also be put in circulant form, which consists of K $(2^{ms}-1)\times(2^{ms}-1)$ circulants as follows:

$$\mathbb{H}^{(2)}_{EG,qc} = [\mathbb{G}_1\ \mathbb{G}_2\ \cdots\ \mathbb{G}_K], \tag{17.22}$$

where $\mathbb{G}_i$ is formed by downward cyclically shifting any member in the ith cyclic class of incidence vectors. Circulant form and quasi-cyclic form are considered to be equivalent; one is obtained from the other by a column permutation [44].

EXAMPLE 17.6

Let $m = s = 3$. The three-dimensional type-I cyclic (0, 3)th-order EG-LDPC code $C^{(1)}_{EG,c}(3, 0, 3)$ constructed based on EG(3, 2^3) is a (511, 139) LDPC code with a minimum distance of at least 79 (using the bound given by (17.15)). Its parity-check matrix $\mathbb{H}^{(1)}_{EG,c}$ is a 4599 × 511 matrix with $\rho = 8$, $\gamma = 72$, and density $r = 0.01565$. Then, $\mathbb{H}^{(2)}_{EG,qc} = [\mathbb{H}^{(1)}_{EG,c}]^T$ is a 511 × 4599 matrix with $\rho = 72$, $\gamma = 8$, and density $r = 0.01565$. The null space of $\mathbb{H}^{(2)}_{EG,qc}$ gives the companion code $C^{(2)}_{EG,qc}(3, 0, 3)$ of the (511, 139) type-I EG-LDPC code. It is a (4599, 4227) LDPC code. Both codes have 372 parity-check bits. The (4599, 4227) code has a minimum distance of at least nine. In quasi-cyclic form, every nine cyclic shifts of a codeword result in another codeword. In circulant form, $\mathbb{H}^{(2)}_{EG,qc}$ consists of nine 511 × 511 circulants in a row.

Recall that for $m = 2$, $\mathbb{H}^{(1)}_{EG,c}$ is a $(2^{2s}-1)\times(2^{2s}-1)$ square matrix. Because the rows of $\mathbb{H}^{(1)}_{EG,c}$ are cyclic shifts of a single incidence vector of a line in EG(2, 2^s) not passing through the origin, the set of columns of $\mathbb{H}^{(1)}_{EG,c}$ (reading top-down) is identical to the set of rows (reading from right to left) of $\mathbb{H}^{(1)}_{EG,c}$. Therefore, $\mathbb{H}^{(2)}_{EG,qc} = [\mathbb{H}^{(1)}_{EG,c}]^T$ has the same set of rows as $\mathbb{H}^{(1)}_{EG,c}$ (in reverse order reading from right to left). Consequently, $C^{(2)}_{EG,qc}$ and $C^{(1)}_{EG,c}$ are equivalent.

17.5 PG-LDPC CODES

The structure of a projective geometry over a finite field was discussed in Chapter 8. Some structural properties of points and lines of this geometry are briefly reviewed here for the purpose of constructing the PG-LDPC codes.

Let α be a primitive element of the Galois field $GF(2^{(m+1)s})$, which is considered as an extension field of $GF(2^s)$. Let

$$n = \frac{2^{(m+1)s}-1}{2^s-1}. \tag{17.23}$$

Then, the n elements

$$(\alpha^0), (\alpha^1), (\alpha^2), \cdots, (\alpha^{n-1})$$

defined in Section 8.8 form an m-dimensional projective geometry over $GF(2^s)$, $\mathrm{PG}(m, 2^s)$. The elements $(\alpha^0), (\alpha^1), (\alpha^2), \cdots, (\alpha^{n-1})$ are the points of $\mathrm{PG}(m, 2^s)$. A line in $\mathrm{PG}(m, 2^s)$ consists of $2^s + 1$ points. There are

$$J = \frac{(1 + 2^s + \cdots + 2^{ms})(1 + 2^s + \cdots + 2^{(m-1)s})}{1 + 2^s} \tag{17.24}$$

lines in $\mathrm{PG}(m, 2^s)$. Every point (α^i) in $\mathrm{PG}(m, 2^s)$ is intersected by

$$\gamma = \frac{2^{ms} - 1}{2^s - 1} \tag{17.25}$$

lines. Two lines in $\mathrm{PG}(m, 2^s)$ are either disjoint or intersect at one and only one point.

Let

$$\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$$

be an n-tuple over $GF(2)$ whose components correspond to the points $(\alpha^0), (\alpha^1), \cdots, (\alpha^{n-1})$ of $\mathrm{PG}(m, 2^s)$, where v_i corresponds to the point (α^i). Let $\mathcal{L}$ be a line in $\mathrm{PG}(m, 2^s)$. The incidence vector of $\mathcal{L}$ is an n-tuple

$$\mathbf{v}_{\mathcal{L}} = (v_0, v_1, \cdots, v_{n-1})$$

whose ith component

$$v_i = \begin{cases} 1 & \text{if } \mathcal{L} \text{ contains the point } (\alpha^i), \\ 0 & \text{otherwise.} \end{cases}$$

The weight of $\mathbf{v}_{\mathcal{L}}$ is $2^s + 1$.

We form a matrix $\mathbb{H}_{PG}^{(1)}$ whose rows are the incidence vectors of the lines in $\mathrm{PG}(m, 2^s)$ and whose columns correspond to the points of $\mathrm{PG}(m, 2^s)$. Then, $\mathbb{H}_{PG}^{(1)}$ has

$$J = \frac{(2^{(m-1)s} + \cdots + 2^s + 1)(2^{ms} + \cdots + 2^s + 1)}{2^s + 1} \tag{17.26}$$

rows and $n = (2^{(m+1)s} - 1)/(2^s - 1)$ columns. It follows from the structural properties of lines and points of $\mathrm{PG}(m, 2^s)$ developed in Section 8.8 that matrix $\mathbb{H}_{PG}^{(1)}$ has the following properties: (1) every row has weight $\rho = 2^s + 1$; (2) every column has weight $\gamma = (2^{ms} - 1)/(2^s - 1)$; (3) no two columns have more than one 1 in common; that is, $\lambda = 0$ or 1; and (4) no two rows have more than one 1 in common. The density of $\mathbb{H}_{PG}^{(1)}$ is

$$r = \frac{\rho}{n} = \frac{(2^s - 1)(2^s + 1)}{2^{(m+1)s} - 1}. \tag{17.27}$$

For $m \geq 2$, r is relatively small. For large m, r is approximately $2^{-(m-1)s}$, which is very small. Therefore, $\mathbb{H}_{PG}^{(1)}$ is a very sparse matrix.

The null space of $\mathbf{H}_{PG}^{(1)}$ gives an LDPC code of length $n = (2^{(m+1)s} - 1)/(2^s - 1)$ with a minimum distance of at least $\gamma + 1 = (2^{ms} - 1)/(2^s - 1) + 1$. This LDPC code is called an m-dimensional type-I $(0, s)$th-order PG-LDPC code, denoted by $C_{PG}^{(1)}(m, 0, s)$. It follows from Definition 8.5 that $C_{PG}^{(1)}(m, 0, s)$ is simply the $(1, s)$th-order PG code. It is therefore cyclic.

Let h be a nonnegative integer less than $2^{(m+1)s} - 1$. For a nonnegative integer l, let $h^{(l)}$ be the remainder resulting from dividing $2^l h$ by $2^{(m+1)s} - 1$. Let $g_{PG}^{(1)}(X)$ be the generator polynomial of the code $C_{PG}^{(1)}(m, 0, s)$. Let α be a primitive element of $GF(2^{(m+1)s})$. Then, it follows from Theorem 8.5 that $g_{PG}^{(1)}(X)$ has α^h as a root if and only if h is divisible by $2^s - 1$, and

$$0 \leq \max_{0 \leq l < s} W_{2^s}(h^{(l)}) = j(2^s - 1), \tag{17.28}$$

with $0 \leq j \leq m - 1$. Let $\xi = \alpha^{2^s - 1}$. The order of ξ is then $n = (2^{(m+1)s} - 1)/(2^s - 1)$. From the foregoing characterization of the roots of $g_{PG}^{(1)}(X)$, it can be shown [42] that $g_{PG}^{(1)}(X)$ has the following consecutive powers of ξ:

$$\xi, \xi^2, \cdots, \xi^{(2^{ms} - 1)/(2^s - 1)}$$

as roots. It follows from the BCH bound that the minimum distance of the m-dimensional PG-LDPC code $C_{PG}^{(1)}(m, 0, s)$ is lower bounded as follows:

$$d_{min} \geq \frac{2^{ms} - 1}{2^s - 1} + 1, \tag{17.29}$$

which is the same as the bound $\gamma + 1$. We also can obtain this bound from (8.50) by setting $\mu = 1$.

We can enumerate the number of parity-check symbols of the m-dimensional type-I $(0,s)$th-order PG-LDPC code $C_{PG}^{(1)}(m, 0, s)$ by determining the roots of its generator polynomial. A combinatorial expression can be found in [46].

A special subclass of the class of type-I PG-LDPC code is the class of two-dimensional type-I PG-LDPC codes constructed based on the family of the two-dimensional projective geometries $\mathrm{PG}(2, 2^s)$ for various s. It follows from (8.51) that the number of parity-check symbols of the two-dimensional type-I $(0, s)$th-order PG-LDPC code $C_{PG}^{(1)}(2, 0, s)$ is

$$n - k = 3^s + 1. \tag{17.30}$$

$C_{PG}^{(1)}(2, 0, s)$ has the following parameters:

Length	$n = 2^{2s} + 2^s + 1$,
Number of parity bits	$n - k = 3^s + 1$,
Dimension	$k = 2^{2s} + 2^s - 3^s$,
Minimum distance	$d_{min} = 2^s + 2$,
Density	$r = \dfrac{2^s + 1}{2^{2s} + 2^s + 1}$.

It is a difference-set code [47], presented in Section 8.3. The parity-check matrix $\mathbf{H}_{PG}^{(1)}$ of this code is a $(2^{2s} + 2^s + 1) \times (2^{2s} + 2^s + 1)$ square matrix, which can

TABLE 17.2: Two-dimensional type-I PG-LDPC codes

s	n	k	d_{min}	ρ	γ	r
2	21	11	6	5	5	0.2381
3	73	45	10	9	9	0.1233
4	273	191	18	17	17	0.0623
5	1057	813	34	33	33	0.0312
6	4161	3431	66	65	65	0.0156
7	16513	14326	130	129	129	0.0078

be obtained by taking the incidence vector of a line in PG$(2, 2^s)$ and its $2^{2s} + 2^s$ cyclic shifts as rows. Some two-dimensional type-I PG-LDPC codes are listed in Table 17.2.

EXAMPLE 17.7

Let $m = s = 2$. The two-dimensional type-I (0.2)th-order PG-LDPC code constructed based on PG$(2, 2^2)$ is the (21, 11) code given in Table 17.2. It is the (1, 2)th-order PG code given in Example 8.24, whose generator polynomial $g_{PG}^{(1)}(X)$ is $1+X^2+X^6+X^7+X^{10}$. Let α be a primitive element of $GF(2^{3\times 2})$. From Example 8.23 we find that

$$\{(\alpha), (\alpha^{11}), (\alpha^{14}), (\alpha^{15}), (\alpha^{20})\}$$

is a line $\mathcal{L}$ in PG$(2, 2^2)$. The incidence vector $\mathbf{v}_{\mathcal{L}}$ of this line is

$$\mathbf{v}_{\mathcal{L}} = (0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,0\,1).$$

This incidence vector and its 20 cyclic shifts form the parity-check matrix of the (21, 11) LDPC code.

Note that the columns of the parity-check matrix $\mathbb{H}_{PG}^{(1)}$ of the type-I $(0, s)$th-order PG-LDPC code $C_{PG}^{(1)}(m, 0, s)$ are simply the intersecting vectors of the points of PG$(m, 2^s)$. Let

$$\mathbb{H}_{PG}^{(2)} = [\mathbb{H}_{PG}^{(1)}]^T. \tag{17.31}$$

Then, the null space of $\mathbb{H}_{PG}^{(2)}$ gives an m-dimensional type-II $(0, s)$th-order PG-LDPC code, denoted by $C_{PG}^{(2)}(m, 0, s)$. $\mathbb{H}_{PG}^{(2)}$ has $J = (2^{(m+1)s} - 1)/(2^s - 1)$ rows and

$$n = \frac{(2^{(m-1)s} + \cdots + 2^s + 1)(2^{ms} + \cdots + 2^s + 1)}{2^s + 1} \tag{17.32}$$

columns. The row and column weights of $\mathbb{H}_{PG}^{(2)}$ are $\rho = (2^{ms} - 1)/(2^s - 1)$ and $\gamma = 2^s + 1$, respectively. Therefore, $C_{PG}^{(2)}(m, 0, s)$ has length n and minimum distance

$$d_{min} \geq 2^s + 2. \tag{17.33}$$

The number of parity-check symbols of $C_{PG}^{(2)}(m, 0, s)$ is the same as that of the type-I code $C_{PG}^{(1)}(m, 0, s)$. For $m = 2$, $C_{PG}^{(2)}(2, 0, s)$ is equivalent to $C_{PG}^{(1)}(2, 0, s)$. $C_{PG}^{(2)}(m, 0, s)$ is, in general, not cyclic, but it can be put in quasi-cyclic form in the same manner as for the type-II EG-LDPC code.

EXAMPLE 17.8

Let $m = 3$ and $s = 2$. The three-dimensional projective geometry PG$(3, 2^2)$ consists of 85 points and 357 lines. To construct the three-dimensional type-I PG-LDPC code $C_{PG}^{(1)}(3, 0, 2)$, we form the parity-check matrix $\mathbb{H}_{PG}^{(1)}$ whose rows are the incidence vectors of the 357 lines in PG$(3, 2^2)$ and whose columns correspond to the 85 points in PG$(3, 2^2)$. The matrix $\mathbb{H}_{PG}^{(1)}$ is a 357×85 matrix and can be put in the following form:

$$\mathbb{H}_{PG}^{(1)} = \begin{bmatrix} \mathbb{I}_{17} & \mathbb{I}_{17} & \mathbb{I}_{17} & \mathbb{I}_{17} & \mathbb{I}_{17} \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{matrix} \\ \mathbb{H}_1 \\ \mathbb{H}_2 \\ \mathbb{H}_3 \\ \mathbb{H}_4 \end{matrix},$$

where $\mathbb{I}_{17}$ is the 17×17 identity matrix, and each $\mathbb{H}_i$ is an 85×85 circulant matrix. The circulant matrices $\mathbb{H}_1$, $\mathbb{H}_2$, $\mathbb{H}_3$, and $\mathbb{H}_4$ have the following incidence vectors of lines (in polynomial form) as their first rows, respectively:

$$\mathbb{h}_1(X) = 1 + X^{24} + X^{40} + X^{71} + X^{84},$$
$$\mathbb{h}_2(X) = X^1 + X^{49} + X^{58} + X^{81} + X^{84},$$
$$\mathbb{h}_3(X) = X^3 + X^{14} + X^{32} + X^{78} + X^{84},$$
$$\mathbb{h}_4(X) = X^{16} + X^{33} + X^{50} + X^{67} + X^{84}.$$

The matrix $\mathbb{H}_{PG}^{(1)}$ has row weight $\rho = 5$ and column weight $\gamma = 21$. The null space of $\mathbb{H}_{PG}^{(1)}$ gives an (85, 24) three-dimensional type-I PG-LDPC code $C_{PG}^{(1)}(3, 0, 2)$. The companion code $C_{PG}^{(2)}(3, 0, 2)$ of this code is the null space of the parity-check matrix $\mathbb{H}_{PG}^{(2)} = [\mathbb{H}_{PG}^{(1)}]^T$. The matrix $\mathbb{H}_{PG}^{(2)}$ has row weight $\rho = 21$ and column weight $\gamma = 5$. $C_{PG}^{(2)}(3, 0, 2)$ has the same number of parity-check bits as $C_{PG}^{(1)}(3, 0, 2)$. Hence, it is a (357, 296) PG-LDPC code and has a minimum distance of at least 6.

EXAMPLE 17.9

Let $m = 4$, and $s = 2$. The parity-check matrix $\mathbb{H}_{PG}^{(1)}$ of the four-dimensional type-I (0, 2)th-order PG-LDPC code $C_{PG}^{(1)}(4, 0, 2)$ is a 5797×341 matrix with row weight $\rho = 5$ and column weight $\gamma = 85$. $C_{PG}^{(1)}(4, 0, 2)$ is a (341, 45) LDPC code with a minimum distance of at least 86. Its companion code $C_{PG}^{(2)}(4, 0, 2)$ is the null space of $\mathbb{H}_{PG}^{(2)} = [\mathbb{H}_{PG}^{(1)}]^T$, which is a 341×5797 matrix. The length of $C_{PG}^{(2)}(4, 0, 2)$ is 5797. Because $C_{PG}^{(2)}(4, 0, 2)$ has the same number of parity-check symbols as $C_{PG}^{(1)}(4, 0, 2)$, which is 296, $C_{PG}^{(2)}(4, 0, 2)$ is a (5797, 5501) LDPC code with a minimum distance of at least 6.

17.6 DECODING OF LDPC CODES

An LDPC code can be decoded in various ways, namely: majority-logic (MLG) decoding, bit-flipping (BF) decoding, weighted BF decoding, a posteriori probability (APP) decoding, and iterative decoding based on belief propagation (IDBP) (commonly known as *sum-product algorithm* (SPA)). The first two types are hard-decision decoding, the last two are soft-decision decoding, and the third one is in between. MLG decoding (discussed in Chapter 8) is the simplest one in decoding complexity. BF decoding requires a little more decoding complexity but gives better error performance than the MLG decoding. APP decoding and the SPA decoding provide much better error performance but require much larger decoding complexity than the MLG and BF decodings. The weighted BF offers a good trade-off between error performance and decoding complexity. SPA decoding gives the best error performance among the five types of decoding of LDPC codes and yet is practically implementable. APP decoding also provides the best error performance; however, it is computationally intractable and hence will not be discussed. A simplified version of APP was presented in Chapter 10, which requires less decoding complexity at the expense of some performance degradation.

Suppose an LDPC code C is used for error control over an AWGN channel with zero mean and one-sided PSD N_0. Assume BPSK signaling with unit energy. A codeword $\mathbb{v} = (v_0, v_1, \cdots, v_{n-1})$ is mapped into a bipolar sequence $\mathbb{x} = (x_0, x_1, \cdots, x_{n-1})$ before its transmission, where $x_l = (2v_l - 1) = +1$ for $v_l = 1$, and $x_l = -1$ for $v_l = 0$ with $0 \leq l \leq n-1$. Let $\mathbb{y} = (y_0, y_1, \cdots, y_{n-1})$ be the soft-decision received sequence at the output of the receiver matched filter. For $0 \leq l \leq n-1$, $y_l = \pm 1 + n_l$, where n_l is a Gaussian random variable with zero mean and variance $N_0/2$. Let $\mathbb{z} = (z_0, z_1, \cdots, z_{n-1})$ be the binary hard-decision received sequence obtained from $\mathbb{y}$ as follows:

$$z_l = \begin{cases} 1, & \text{for } y_l > 0, \\ 0, & \text{for } y_l \leq 0. \end{cases}$$

Let $\mathbb{H}$ be the parity-check matrix of an LDPC code C with J rows and n columns. Let $\mathbb{h}_1, \mathbb{h}_2, \cdots, \mathbb{h}_J$, denote the rows of $\mathbb{H}$, where

$$\mathbb{h}_j = (h_{j,0}, h_{j,1}, \cdots, h_{j,n-1})$$

for $1 \leq j \leq J$. Then,

$$\mathbb{s} = (s_1, s_2, \cdots, s_J) = \mathbb{z} \cdot \mathbb{H}^T \tag{17.34}$$

gives the syndrome of the received sequence $\mathbb{z}$, where the jth syndrome component s_j is given by the check-sum

$$s_j = \mathbb{z} \cdot \mathbb{h}_j = \sum_{l=0}^{n-1} z_l h_{j,l}. \tag{17.35}$$

The received vector $\mathbb{z}$ is a codeword if and only if $\mathbb{s} = \mathbb{0}$. If $\mathbb{s} \neq \mathbb{0}$, errors in $\mathbb{z}$ are detected. A *nonzero* syndrome component s_j indicates a *parity failure*. The total

number of parity failures is equal to the number of nonzero syndrome components in $\mathbf{s}$. Let

$$\begin{aligned}\mathbf{e} &= (e_0, e_1, \cdots, e_{n-1}) \\ &= (v_0, v_1, \cdots, v_{n-1}) + (z_0, z_1, \cdots, z_{n-1}).\end{aligned} \tag{17.36}$$

Then, $\mathbf{e}$ is the error pattern in $\mathbf{z}$. This error pattern $\mathbf{e}$ and the syndrome $\mathbf{s}$ satisfy the condition

$$\mathbf{s} = (s_1, s_2, \cdots, s_J) = \mathbf{e} \cdot \mathbb{H}^T, \tag{17.37}$$

where

$$s_j = \mathbf{e} \cdot \mathbb{h}_j = \sum_{l=0}^{n-1} e_l h_{j,l} \tag{17.38}$$

for $1 \leq j \leq J$.

17.6.1 Majority-Logic Decoding

MLG decoding was discussed in detail in Chapter 8. Now, we apply one-step MLG decoding to LDPC codes.

It follows from the structural properties of the parity-check matrix $\mathbb{H}$ of a regular LDPC code that for every bit position l, there is a set

$$A_l = \{\mathbb{h}_1^{(l)}, \mathbb{h}_2^{(l)}, \cdots, \mathbb{h}_\gamma^{(l)}\} \tag{17.39}$$

of γ rows in $\mathbb{H}$ that are orthogonal on this bit position; that is, the lth component of each row in A_l is 1, and no two rows in A_l have a common 1 in any other position. We form the following set of syndrome equations based on the rows in A_l:

$$S_l = \{s_j^{(l)} = \mathbf{e} \cdot \mathbb{h}_j^{(l)} : \mathbb{h}_j^{(l)} \in A_l \quad \text{for } 1 \leq j \leq \gamma\}, \tag{17.40}$$

where

$$s_j^{(l)} = \mathbf{e} \cdot \mathbb{h}_j^{(l)} = \sum_{i=0}^{n-1} e_i h_{j,i}^{(l)}. \tag{17.41}$$

S_l gives a set of γ check-sums orthogonal on the error digit e_l. As presented in Section 8.1, they can be used for estimating the error digit e_l based on the one-step MLG decoding rule. Correct decoding of e_l for $0 \leq l < n$ is guaranteed if there are $\lfloor \gamma/2 \rfloor$ or fewer errors in the error pattern $\mathbf{e}$.

All four classes of finite-geometry LDPC codes are one-step MLG decodable.

17.6.2 Bit-Flipping Decoding Algorithm

Bit-flipping (BF) decoding of LDPC codes was devised by Gallager in the early 1960s [1, ?]. When detectable errors occur during transmission there will be parity-check failures in the syndrome $\mathbf{s} = (s_1, s_2, \cdots, s_J)$, and some of the syndrome bits will be

equal to 1. BF decoding is based on the change of the number of parity failures in $\{\mathbf{z} \cdot \mathbb{h}_j : 1 \leq j \leq J\}$ when a bit in the received sequence $\mathbf{z}$ is changed (or flipped).

First, the decoder computes all the parity-check sums based on (17.34) and (17.35) and then changes any bit in the received sequence $\mathbf{z}$ that is contained in more than some fixed number δ of failed parity-check equations (i.e., nonzero syndrome bits). Using the modified received sequence $\mathbf{z}'$, the decoder recomputes the parity-check sums, and the process is repeated until all the parity-check sums are equal to zero (i.e., no parity failure). At this point, the modified received sequence is a codeword in C. This decoding is an iterative decoding algorithm. The parameter δ, called the *threshold*, is a design parameter that should be chosen to optimize the error performance while minimizing the number of computations of parity-check sums. The value of δ depends on the code parameters ρ, γ, $d_{min}(C)$, and SNR.

If decoding fails for a given value of δ, then the value of δ should be reduced to allow further decoding iterations. For error patterns whose number of errors is less than or equal to the error-correcting capability of the code, the decoding will be completed in one or a few iterations. Otherwise, more decoding iterations are needed. Therefore, the number of decoding iterations is a random variable and is a function of the channel SNR. A limit may be set on the number of iterations. When this limit is reached the decoding process is terminated to avoid excessive computations. Owing to the nature of low-density parity checks, BF decoding algorithm corrects many error patterns whose number of errors exceeds the error-correcting capability of the code.

A very simple BF decoding algorithm is given here:

Step 1. Compute the parity-check sums (syndrome bits) based on (17.34) and (17.35). If all the parity-check sums are zero, stop the decoding.
Step 2. Find the number of failed parity-check equations for each bit, denoted by f_i, $i = 0, 1, ..., n-1$.
Step 3. Identify the set $\mathcal{S}$ of bits for which f_i is the largest.
Step 4. Flip the bits in set $\mathcal{S}$.
Step 5. Repeat steps 1 to 4 until all the parity-check sums are zero (for this case, we stop the iteration in step 1) or a preset maximum number of iterations is reached.

In step 5, if the preset maximum number of iterations is reached and not all the parity-check sums are zero, we may simply declare a decoding failure or decode the unmodified received sequence $\mathbf{z}$ with MLG decoding to obtain a decoded sequence, which may not be a codeword in C. The latter case is simply a hybrid BF/MLG decoding.

This simple BF decoding algorithm can be improved by using adaptive thresholds, δ's. Of course, this improvement is achieved at the expense of more computations.

17.6.3 Weighted Majority-Logic Decoding and Bit-Flipping Decoding

The simple hard-decision one-step MLG decoding and BF decoding can be improved to achieve better performance by including some kind of reliability information (or measure) of the received symbols in their decoding decisions, as described

in Section 10.9.2. Of course, such performance improvement requires additional decoding complexity.

Consider the soft-decision received sequence $\mathbf{y} = (y_0, y_1, \cdots, y_{n-1})$ at the output of the receiver matched filter. For an AWGN channel, a simple measure of the reliability of a received symbol y_l is its magnitude, $|y_l|$: the larger the magnitude $|y_l|$, the larger the reliability of the hard-decision digit z_l. This reliability measure has been used in many reliability-based algorithms for decoding linear block codes, as presented in Chapter 10.

The weighted MLG decoding presented in Section 10.9.2 can be used for decoding LDPC codes. Consider an LDPC code specified by a parity-check matrix $\mathbb{H}$ with J rows, $\mathbb{h}_1, \mathbb{h}_2, \cdots, \mathbb{h}_J$. For $0 \le l \le n-1$ and $1 \le j \le J$, we define

$$|y_j|_{min}^{(l)} \triangleq \{\min\{|y_i|\} : 0 \le i \le n-1, h_{j,i} = 1\} \tag{17.42}$$

and

$$E_l \triangleq \sum_{s_j^{(l)} \in S_l} (2s_j^{(l)} - 1)|y_j|_{min}^{(l)}. \tag{17.43}$$

E_l is simply a weighted check-sum that is orthogonal on the code bit position l. Let $\mathbf{e} = (e_0, e_1, \cdots, e_{n-1})$ be the error pattern to be estimated. Then, we can modify the one-step MLG decoding based on the weighted check-sum E_l as follows:

$$e_l = \begin{cases} 1, & \text{for } E_l > 0, \\ 0, & \text{for } E_l \le 0 \end{cases} \tag{17.44}$$

for $0 \le l \le n-1$. The preceding decoding algorithm is called *weighted* MLG decoding [48].

The decision rule given by (17.44) can be used in BF decoding. In this case the decoding is carried out as follows:

Step 1. Compute the parity-check sums (syndrome bits) based on (17.34) and (17.35). If all the parity-check sums are zero, stop the decoding.
Step 2. Compute E_l based on (17.43), for $0 \le l \le n-1$.
Step 3. Find the bit position l for which E_l is the largest.
Step 4. Flip the bit z_l.
Step 5. Repeat steps 1 through 4. This process of bit flipping continues until all the parity-check sums are zero or a preset maximum number of iterations is reached.

This modified BF algorithm is called a *weighted* BF decoding algorithm [17–20]. In step 5, if the preset maximum numbers of iteration is reached and not all the parity-check sums are zero, we may decode the unmodified received sequence $\mathbf{z}$ with the weighted MLG decoding based on weighted check-sums computed based on (17.43) and the decision function given by (17.44). This hybrid weighted BF/MLG decoding will result in a decoded sequence.

The foregoing weighted decoding algorithms require some real-number computations.

17.6.4 The Sum-Product Algorithm

The sum-product algorithm (SPA) is an iterative decoding algorithm based on belief propagation (IDBP) [10, 12–20, 23, 25–30] that is extremely efficient for decoding LDPC codes. Like the MAP decoding algorithm and its variations presented in Chapters 12 and 14, it is a symbol-by-symbol soft-in, soft-out decoding algorithm. It processes the received symbols iteratively to improve the reliability of each decoded code symbol based on the parity-check sums computed from the hard decisions of the received symbols and the sparse parity-check matrix $\mathbb{H}$ of an LDPC code. The reliability of a decoded symbol can be measured by its marginal posteriori probability, its log-likelihood ratio (LLR), or the value of its corresponding received symbol. The computed reliability measures of code symbols at the end of each decoding iteration are used as input for the next iteration. The decoding iteration process continues until a certain stopping condition (or criterion) is satisfied. Then, based on the computed reliability measures of code symbols, hard decisions are made.

Again, we consider an LDPC code C of length n specified by a parity-check matrix $\mathbb{H}$ with J rows, $\mathbb{h}_1, \mathbb{h}_2, \cdots, \mathbb{h}_J$, where

$$\mathbb{h}_j = (h_{j,0}, h_{j,1}, \cdots, h_{j,n-1}).$$

For $1 \leq j \leq J$, we define the following index set for $\mathbb{h}_j$:

$$B(\mathbb{h}_j) = \{l : h_{j,l} = 1, 0 \leq l < n\}, \tag{17.45}$$

which is called the *support* of $\mathbb{h}_j$.

The implementation of SPA decoding is based on the computation of the marginal a posteriori probabilities,

$$P(v_l|\mathbf{y})$$

for $0 \leq l < n$, where $\mathbf{y}$ is the soft-decision received sequence. Then, the LLR for each code bit is given by

$$L(v_l) = \log \frac{P(v_l = 1|\mathbf{y})}{P(v_l = 0|\mathbf{y})}. \tag{17.46}$$

Let $p_l^0 = P(v_l = 0)$ and $p_l^1 = P(v_l = 1)$ be the prior probabilities of $v_l = 0$ and $v_l = 1$, respectively.

For $0 \leq l < n$, $1 \leq j \leq J$, and each $\mathbb{h}_j \in A_l$, let $q_{j,l}^{x,(i)}$ be the conditional probability that the transmitted code bit v_l has value x, given the check-sums computed based on the check vectors in $A_l \backslash \mathbb{h}_j$ at the ith decoding iteration. For $0 \leq l < n$, $1 \leq j \leq J$, and $\mathbb{h}_j \in A_l$, let $\sigma_{j,l}^{x,(i)}$ be the conditional probability that the check-sum s_j is satisfied (i.e., $s_j = 0$), given $v_l = x$ (0 or 1) and the other code bits in $B(\mathbb{h}_j)$ have a separable distribution $\{q_{j,t}^{v_t,(i)} : t \in B(\mathbb{h}_j)\backslash l\}$; that is,

$$\sigma_{j,l}^{x,(i)} = \sum_{\{v_t : t \in B(\mathbb{h}_j)\backslash l\}} P(s_j = 0|v_l = x, \{v_t : t \in B(\mathbb{h}_j)\backslash l\}) \cdot \prod_{t \in B(\mathbb{h}_j)\backslash l} q_{j,t}^{v_t,(i)}. \tag{17.47}$$

The computed values of $\sigma_{j,l}^{x,(i)}$ are then used to update the values of $q_{j,l}^{x,(i+1)}$ as follows:

$$q_{j,l}^{x,(i+1)} = \alpha_{j,l}^{(i+1)} p_l^x \prod_{\mathbb{h}_t \in A_l \backslash \mathbb{h}_j} \sigma_{t,l}^{x,(i)}, \tag{17.48}$$

where $\alpha_{j,l}^{(i+1)}$ is chosen such that

$$q_{j,l}^{0,(i+1)} + q_{j,l}^{1,(i+1)} = 1.$$

The computed values of $q_{j,l}^{x,(i+1)}$ are then used to update the values of $\sigma_{j,l}^{x,(i+1)}$ based on (17.47). The updating between $q_{j,l}^{x,(i)}$ and $\sigma_{j,l}^{x,(i)}$ is carried out iteratively during the decoding process.

At the ith iteration step, the pseudo-posteriori probabilities are given by

$$P^{(i)}(v_l = x|\mathbb{y}) = \alpha_l^{(i)} p_l^x \prod_{\mathbb{h}_j \in A_l} \sigma_{j,l}^{x,(i-1)}, \tag{17.49}$$

where α_l^i is chosen such that $P^{(i)}(v_l = 0|\mathbb{y}) + P^{(i)}(v_l = 1|\mathbb{y}) = 1$. Based on these probabilities, we can form the following vector as the decoded candidate:

$$\mathbb{z}^{(i)} = (z_0^{(i)}, z_1^{(i)}, \cdots, z_{n-1}^{(i)})$$

with

$$z_l^{(i)} = \begin{cases} 1, & \text{for } P^{(i)}(v_l = 1|\mathbb{y}) > 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{17.50}$$

Then, compute $\mathbb{z}^{(i)} \cdot \mathbb{H}^T$. If $\mathbb{z}^{(i)} \cdot \mathbb{H}^T = \mathbb{0}$, stop the decoding iteration process, and output $\mathbb{z}^{(i)}$ as the decoded codeword.

The SPA decoding in terms of probability consists of the following steps:

Initialization: Set $i = 0$ and the maximum number of iterations to I_{max}. For every pair (j, l) such that $h_{j,l} = 1$ with $1 \le j \le J$ and $0 \le l < n$, set $q_{j,l}^{0,(0)} = p_l^0$ and $q_{j,l}^{1,(0)} = p_l^1$.

Step 1. For $0 \le l < n$, $1 \le j \le J$, and each $\mathbb{h}_j \in A_l$, compute the probabilities, $\sigma_{j,l}^{0,(i)}$ and $\sigma_{j,l}^{1,(i)}$. Go to step 2.

Step 2. For $0 \le l < n$, $1 \le j \le J$, and each $\mathbb{h}_j \in A_l$, compute the values of $q_{j,l}^{0,(i+1)}$ and $q_{j,l}^{1,(i+1)}$ and the values of $P^{(i+1)}(v_l = 0|\mathbb{y})$ and $P^{(i+1)}(v_l = 1|\mathbb{y})$. Form $\mathbb{z}^{(i+1)}$ and test $\mathbb{z}^{(i+1)} \cdot \mathbb{H}^T$. If $\mathbb{z}^{(i+1)} \cdot \mathbb{H}^T = \mathbb{0}$ or the maximum iteration number I_{max} is reached, go to step 3. Otherwise, set $i := i + 1$ and go to step 1.

Step 3. Output $\mathbb{z}^{(i+1)}$ as the decoded codeword and stop the decoding process.

A detail exposition of SPA decoding of LDPC codes can be found in [10].

To perform SPA decoding, real-number addition, subtraction, multiplication, and division, and exponential and logarithm operations are needed. In implementation the last four types of operations are more complex than addition and subtraction. For this reason we simply ignore the number of additions and subtractions in analyzing the computational complexity. From (17.46) through (17.49), we find that the number of multiplications needed in each iteration of the decoding is of the order $O(2J\rho + 4n\gamma)$, and the number of logarithm operations needed is of the order $O(n)$. We see that the computational complexity of SPA decoding is mainly a linear function of the number of 1-entries in the parity-check matrix $\mathbb{H}$ of the LDPC code to be decoded.

A different version of SPA decoding is based on updating extrinsic information at each decoding iteration. The improved extrinsic information at one decoding iteration is used for improving the extrinsic information and the reliability values of the code symbols at the next decoding iteration. The decoding iterations continue until a modified hard-decision received sequence is obtained that makes all the parity-check sums equal to zero.

Consider a code-bit position-l. Let $\mathbb{h}$ be a row in the set A_l that is orthogonal on bit position-l. Then, the extrinsic information provided to the transmitted code bit v_l by other code bits checked by $\mathbb{h}$ (or the check-sum $\mathbb{z} \cdot \mathbb{h}$) is [1, 2, 30, 49]

$$\varepsilon_l(\mathbb{h}) = \log \frac{1 + \prod\limits_{t \in B(\mathbb{h}) \setminus l} \tanh\left(\frac{2}{N_0} y_t\right)}{1 - \prod\limits_{t \in B(\mathbb{h}) \setminus l} \tanh\left(\frac{2}{N_0} y_t\right)}. \tag{17.51}$$

The total extrinsic information provided to code bit v_l by the rows in A_l is then

$$\begin{aligned} \varepsilon_l &= \sum_{\mathbb{h} \in A_l} \varepsilon_l(\mathbb{h}) \\ &= \sum_{\mathbb{h} \in A_l} \log \frac{1 + \prod\limits_{t \in B(\mathbb{h}) \setminus l} \tanh\left(\frac{2}{N_0} y_t\right)}{1 - \prod\limits_{t \in B(\mathbb{h}) \setminus l} \tanh\left(\frac{2}{N_0} y_t\right)}. \end{aligned} \tag{17.52}$$

We define a $J \times n$ matrix

$$\mathbb{E} = [E_{j,l}]_{1 \le j \le J}^{0 \le l < n}, \tag{17.53}$$

where

$$E_{j,l} = \begin{cases} \varepsilon_l - \varepsilon_l(\mathbb{h}_j), & \text{for } l \in B(\mathbb{h}_j), \\ 0, & \text{otherwise.} \end{cases} \tag{17.54}$$

The matrix $\mathbb{E}$ is called the *extrinsic matrix*. We define another $J \times n$ matrix

$$\mathbb{Y} = [Y_{j,l}]_{1 \le j \le J}^{0 \le l < n}, \tag{17.55}$$

where

$$Y_{j,l} = \begin{cases} \dfrac{4}{N_0} y_l, & \text{for } l \in B(\mathbb{h}_j), \\ 0, & \text{otherwise.} \end{cases} \tag{17.56}$$

This matrix is called the *channel information matrix*. In the decoding, each decoding iteration updates (or improves) the extrinsic values $\varepsilon_l(\mathbb{h}_j)$, ε_l, and $E_{j,l}$ for $0 \le l < n$ and $1 \le j \le J$. These values computed during the ith decoding iteration are denoted by $\varepsilon_l^{(i)}(\mathbb{h}_j)$, $\varepsilon_l^{(i)}$, and $E_{j,l}^{(i)}$. For $0 \le l < n$ and $1 \le j \le J$, the extrinsic value $E_{j,l}^{(i)}$ is used to modify the channel information $Y_{j,l}$ by setting

$$Z_{j,l}^{(i+1)} = Y_{j,l} + E_{j,l}^{(i)}. \tag{17.57}$$

We form the channel value matrix at the end of the ith iteration as follows:

$$\mathbb{Z}^{(i)} = [Z_{j,l}^{(i)}]_{1 \le j \le J}^{0 \le l < n}, \tag{17.58}$$

where

$$Z_{j,l}^{(i)} = Y_{j,l} + E_{j,l}^{(i-1)}. \tag{17.59}$$

For $i = 0$, $\mathbb{Z}^{(0)} = \mathbb{Y}$. The entries of $\mathbb{Z}^{(i)}$ are used to compute the extrinsic values for the $(i+1)$th decoding iteration. For $0 \le l < n$, $\varepsilon_l^{(i)}$ is used to modify the received vector $\mathbf{y} = (y_0, y_1, \cdots, y_{n-1})$. Let

$$\begin{aligned} \mathbb{r}^{(0)} &= (r_0^{(0)}, r_1^{(0)}, \cdots, r_{n-1}^{(0)}) \\ &= \left(\frac{4}{N_0} y_0, \frac{4}{N_0} y_1, \cdots, \frac{4}{N_0} y_{n-1} \right). \end{aligned}$$

Let

$$\mathbb{r}^{(i)} = (r_0^{(i)}, r_1^{(i)}, \cdots, r_{n-1}^{(i)})$$

be the modified received vector at the ith decoding iteration. Then, the modified received vector at the next iteration is

$$\mathbb{r}^{(i+1)} = (r_0^{(i+1)}, r_1^{(i+1)}, \cdots, r_{n-1}^{(i+1)}),$$

where

$$r_l^{(i+1)} = r_l^{(i)} + \varepsilon_l^{(i)}$$

and $r_l^{(0)} = \frac{4}{N_0} y_l$ for $0 \le l < n$.

Let

$$\mathbb{z}^{(i)} = (z_0^{(i)}, z_1^{(i)}, \cdots, z_{n-1}^{(i)})$$

be the hard-decision vector obtained from $\mathbb{r}^{(i)} = (r_0^{(i)}, r_1^{(i)}, \cdots, r_{n-1}^{(i)})$ with

$$z_l^{(i)} = \begin{cases} 1 & \text{for } r_l^{(i)} > 0, \\ 0 & \text{for } r_l^{(i)} \le 0. \end{cases} \tag{17.60}$$

This hard-decision vector can be tested whether it is a codeword by computing the syndrome $\mathbf{z}^{(i)} \cdot \mathbf{H}^T$. If $\mathbf{z}^{(i)} \cdot \mathbf{H}^T = \mathbf{0}$, $\mathbf{z}^{(i)}$ is a codeword and decoding iteration can be stopped. Otherwise, iteration continues until either the preceding condition is satisfied or a preset maximum number of iterations is reached.

SPA decoding in terms of extrinsic information is carried out as follows [30]:

Initialization: Set $i = 0$, the maximum number of iterations to I_{max}, $\mathbf{r}^{(0)} = \frac{4}{N_0}\mathbf{y}$, and $\mathbf{Z}^{(0)} = \mathbf{Y}$.

Step 1. For $0 \le l < n$, $1 \le j \le J$, and each $\mathbf{h}_j \in A_l$, compute

$$\varepsilon_l^{(i)}(\mathbf{h}_j) = \log \frac{1 + \prod_{t \in B(\mathbf{h}_j)\backslash l} \tanh(Z_{j,l}^{(i)}/2)}{1 - \prod_{t \in B(\mathbf{h}_j)\backslash l} \tanh(Z_{j,l}^{(i)}/2)}, \tag{17.61}$$

$$E_{j,l}^{(i)} = \sum_{\mathbf{h}_t \in A_l \backslash \mathbf{h}_j} \varepsilon_l^{(i)}(\mathbf{h}_t), \tag{17.62}$$

$$\varepsilon_l^{(i)} = E_{j,l}^{(i)} + \varepsilon_l^{(i)}(\mathbf{h}_j), \tag{17.63}$$

and form the extrinsic matrix $\mathbf{E}^{(i)}$. Go to step 2.

Step 2. Form $\mathbf{Z}^{(i+1)} = \mathbf{Y} + \mathbf{E}^{(i)}$, $\mathbf{r}^{(i+1)} = \mathbf{r}^{(0)} + \boldsymbol{\varepsilon}^{(i)}$, and $\mathbf{z}^{(i+1)}$, where $\boldsymbol{\varepsilon}^{(i)} = (\varepsilon_0^{(i)}, \varepsilon_1^{(i)}, \cdots, \varepsilon_{n-1}^{(i)})$. Test $\mathbf{z}^{(i+1)} \cdot \mathbf{H}^T$. If $\mathbf{z}^{(i+1)} \cdot \mathbf{H}^T = \mathbf{0}$ or the maximum iteration number I_{max} is reached, stop the decoding iteration and go to step 3. Otherwise set $i := i + 1$ and go to step 1.

Step 3. Output $\mathbf{z}^{(i+1)}$ as the decoded codeword.

The computational complexity and decoding delay (or decoding time) of the SPA increase as the number of decoding iterations increases. Long decoding delays are not desirable in high-speed communication and data storage systems. If an LDPC code has a large MLG error-correcting capability, such as a long finite-geometry LDPC code, it is possible to devise a hybrid SPA decoding scheme with MLG decoding to shorten the decoding iteration process and hence the decoding delay with only a small degradation in error performance, as follows. Based on the MLG error-correcting capability of the LDPC code to be decoded and the channel statistics, we choose an appropriate maximum number of decoding iterations $I_{\max}$. At the end of the I_{max}th iteration, if not all the computed parity-check sums are zero, we switch to MLG decoding to decode the hard-decision received sequence obtained at the I_{max}th iteration. If the number of residue errors left uncorrected in the hard-decision received sequence is within the MLG error-correcting capability of the code, the errors will be corrected. This is often the case for long LDPC codes with large majority-logic error-correcting capabilities, such as finite-geometry LDPC codes. Based on many experimental results, we have observed that SPA decoding of long finite-geometry LDPC codes converges very fast. After a few iterations, say

five, the number of residue errors left uncorrected is small, and the MLG decoding will correct these residue errors. In fact, MLG decoding can be implemented at the end of each decoding iteration. At the end of each decoding iteration, we check the hard-decision received sequence. If the number of unreliable positions based on a certain threshold is within the MLG error-correcting capability, we switch to MLG decoding to terminate the decoding iteration process: otherwise, we continue to the next decoding iteration until the I_{max}th iteration is finished. This combination of SPA decoding and MLG decoding is called *two-stage hybrid SPA/MLG decoding* [17].

17.6.5 Performance of Finite-Geometry LDPC Codes

To demonstrate the error performance of finite-geometry LDPC codes, we select several EG- and PG-LDPC codes of various lengths and decode them with various decoding algorithms.

EXAMPLE 17.10

Let $m = 2$ and $s = 5$. The two-dimensional type-I (0, 5)th-order cyclic EG-LDPC code is a (1023, 781) code (the fourth code in Table 17.1). This code has a minimum distance of 33 and rate $R = 0.763$. Its error performance with various decoding algorithms is shown in Figure 17.7. We see that MLG decoding provides the least coding gain over the uncoded BPSK but requires the least decoding complexity.

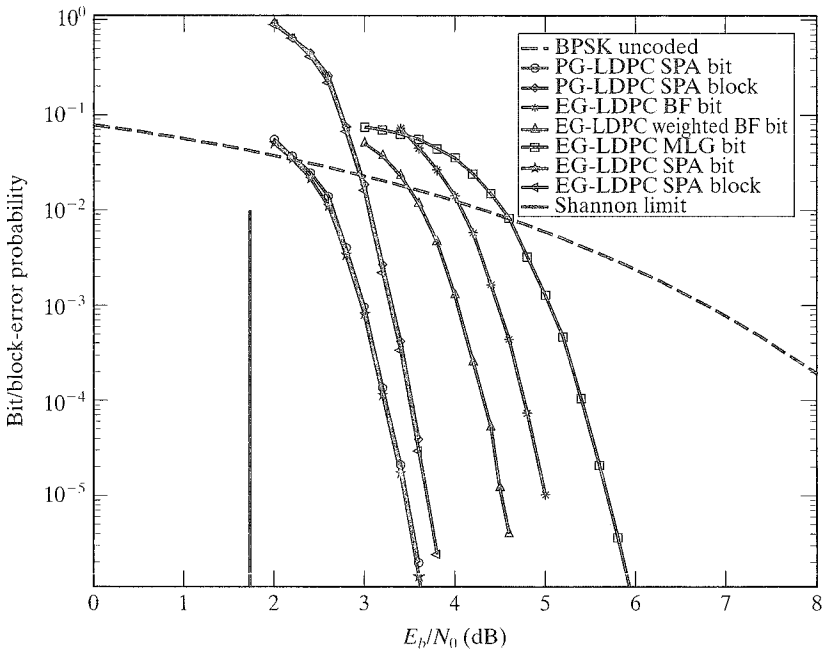


FIGURE 17.7: Bit- and block-error probabilities of the (1023, 781) two-dimensional type-I cyclic EG-LDPC code and the (1057, 813) two-dimensional type-I cyclic PG-LDPC code based on various decoding algorithms.

With MLG decoding, this code is capable of correcting 16 or fewer errors. The SPA gives the best error performance but requires the largest computational complexity. The (1023, 781) EG-LDPC code with SPA decoding achieves an error performance 1.7 dB from the Shannon limit at a BER of 10^{-5}. The BF and weighted BF decoding algorithms perform better than the MLG decoding but not as well as the SPA. They offer a good trade-off between error performance and decoding complexity. From Figure 17.7 we see that the weighted BF decoding algorithm achieves a 1.3 dB coding gain over the MLG decoding at a BER of 10^{-5} with some additional computational complexity. The two-dimensional type-I (0, 5)th-order PG-LDPC code is a (1057,813) code with a minimum distance of 34 and a rate of 0.769 (the fourth code given in Table 17.2). It is equivalent to the (1023, 781) type-I (0, 5)th-order EG-LDPC code. Its error performance with SPA decoding is also included in Figure 17.7. We see that the two codes have almost identical error performance.

EXAMPLE 17.11

Let $m = 2$ and $s = 6$. The two-dimensional type-I (0, 6)th-order cyclic EG-LDPC code is a (4095, 3367) code with a minimum distance of 65 and rate $R = 0.83$ (the fifth code given in Table 17.1). Its parity-check matrix $\mathbb{H}_{EG,c}^{(1)}$ has row weight $\rho = 64$, column weight $\gamma = 64$, and density $r = 0.01263$. This is also the (0, 6)th-order EG code given in Example 8.21. Its error performance with various decodings is shown in Figure 17.8. This code with SPA decoding achieves an error performance 1.40 dB from the Shannon limit at a BER of 10^{-5}. Again, the weighted BF decoding algorithm provides a good trade-off between error performance and decoding complexity. The equivalent two-dimensional type-I (0, 6)th-order PG-LDPC code is a (4161, 3431) code with a minimum distance of 66 and a rate of 0.825. Its error performance is almost identical to that of the (4095, 3367) EG-LDPC code, as shown in Figure 17.8.

EXAMPLE 17.12

Let $m = s = 3$. The three-dimensional type-I (0, 3)th-order cyclic EG-LDPC code $C_{EG,c}^{(1)}(3, 0, 3)$ constructed based on EG$(3, 2^3)$ is a (511, 139) code with $d_{\min} \geq 79$ and rate $R = 0.272$ given in Example 17.6. It is a low-rate code. Its parity-check matrix $\mathbb{H}_{EG,c}^{(1)}$ is a 4599×511 matrix with $\rho = 8$, $\gamma = 72$, and density $r = 0.01565$. The transpose of $\mathbb{H}_{EG,c}^{(1)}$,

$$\mathbb{H}_{EG,qc}^{(2)} = [\mathbb{H}_{EG,c}^{(1)}]^T,$$

is a 511×4599 matrix with $\rho = 72, \gamma = 8$, and $r = 0.01565$. The null space of $\mathbb{H}_{EG,qc}^{(2)}$ gives a three-dimensional type-II (0, 3)th-order quasi-cyclic EG-LDPC code $C_{EG,qc}^{(2)}(3, 0, 3)$ that is a (4599, 4227) code with a minimum distance of at least 9 and rate $R = 0.92$. It is a high-rate code. The bit- and block-error performances of both codes with SPA decoding are shown in Figure 17.9. We see that the (4599, 4227) type-II EG-LDPC code performs very well. At a $BER = 10^{-5}$, the performance of this code is only 1 dB from the Shannon limit.

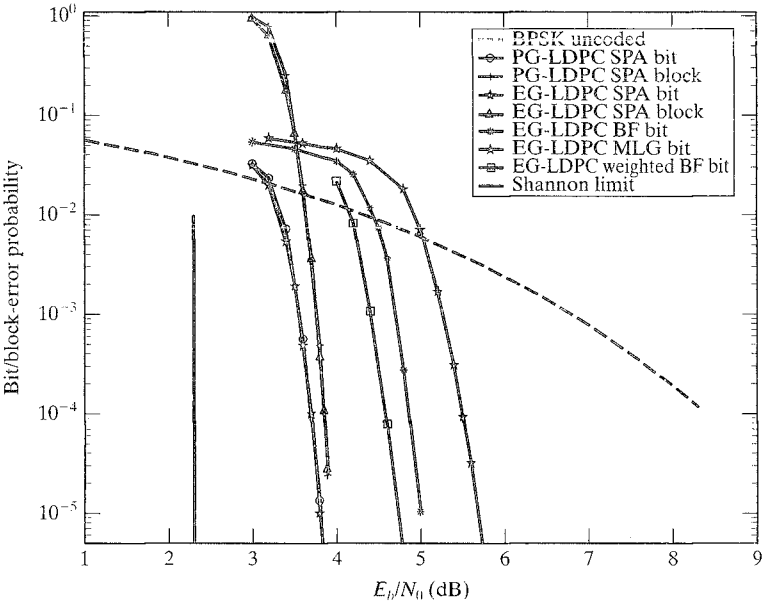


FIGURE 17.8: Bit- and block-error probabilities of the (4095, 3367) two-dimensional type-I EG-LDPC code and the (4161, 3431) two-dimensional type-I PG-LDPC code based on various decoding algorithms.

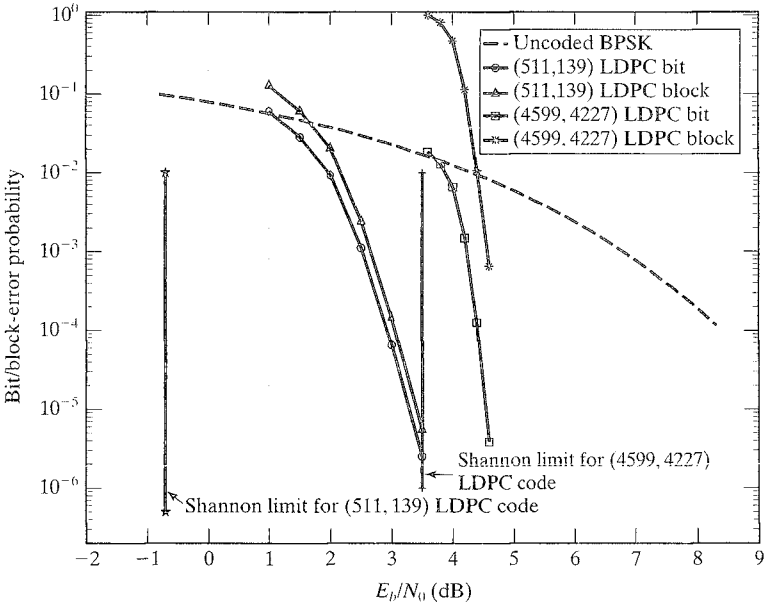


FIGURE 17.9: Error performances for the (511, 139) type-I EG-LDPC code and the (4599, 4227) type-II EG-LDPC code with SPA decoding.

Some good Gallager LDPC codes have been found with computer searches [10–13]. For lengths up to 1023 and 1057, these codes do not perform as well as their corresponding finite-geometry codes. Furthermore, they have relatively small γ, say 3 to 6, and hence they are not suitable for MLG or BF decoding.

EXAMPLE 17.13

Let $m = 2$ and $s = 4$. The two-dimensional type-I $(0, 4)$th-order PG-LDPC code is a $(273, 191)$ code (the third code given in Table 17.2) with $\rho = \gamma = 17$, rate $R = 0.699$, density $r = 0.0623$, and a minimum distance of at least 18. Its error performance with SPA decoding is shown in Figure 17.10. Also included in the figure are the SPA error performances of the two-dimensional type-I $(0, 4)$th-order cyclic $(255, 175)$ EG-LDPC code and two computer-generated $(273, 191)$ LDPC codes with $\gamma = 3$ and $\gamma = 4$, respectively. We see that the two-dimensional PG-LDPC code outperforms the two computer-generated codes significantly.

For longer lengths, computer-generated Gallager LDPC codes should perform better than finite-geometry codes with SPA decoding, but their encoding may be very complex owing to the lack of structure, such as cyclic or quasi-cyclic. Another disadvantage of computer-generated Gallager LDPC codes is that it is very hard to determine their minimum distances.

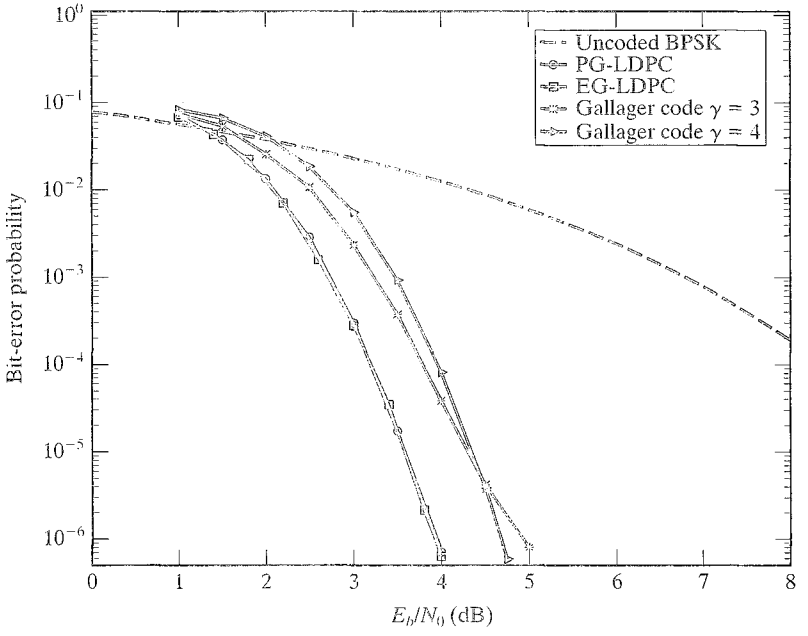


FIGURE 17.10: Bit-error probabilities of the (255, 175) type-I EG-LDPC code, (273, 191) type-I PG-LDPC code, and two computer-searched (273, 191) Gallager codes with SPA decoding.

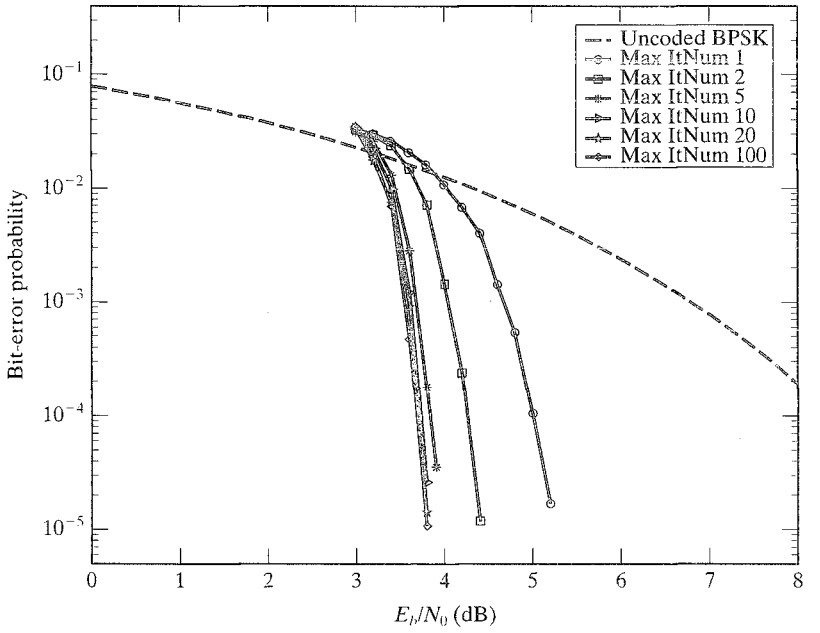


FIGURE 17.11: Convergence of SPA decoding for the (4095, 3367) type-I EG-LDPC code.

Simulation results show that the SPA decoding process of the EG- and PG-LDPC codes converges relatively fast. The difference in performance between 10 iterations and 100 iterations is very small. As an example, consider the (4095, 3367) two-dimensional type-I EG-LDPC code given in Example 17.11. The convergence of SPA decoding of this code is shown in Figure 17.11. We see that at BER $= 10^{-4}$, the performance gap between 5 iterations and 100 iterations is less than 0.1 dB. We also note that the performance gap between 2 iterations and 100 iterations is about 0.5 dB. In practice, if decoding speed (or delay) is critical, 5 iterations may be good enough, or two-stage hybrid SPA/MLG decoding may be used with the maximum number of iterations set to 2 or 3.

With SPA decoding, the error performance of an LDPC code depends on several important parameters: (1) the girth of its Tanner graph (or cycle distribution); (2) its minimum distance; (3) the column and row weights of its parity-check matrix; and (4) its error coefficient (i.e, the number of minimum-weight codewords). The girth should be large enough so that there are no short cycles, especially cycles of length 4, in its Tanner graph. Short cycles prevent SPA decoding from converging to MLD or near-MLD performance; however, girth should not be too large. An LDPC code whose Tanner graph has a large girth tends to have a relatively poor minimum distance [38]. The error floor in the error performance of an LDPC code is tied to the minimum distance of the code. Large minimum distance either removes the error floor or pushes it down to a much lower error probability. An LDPC code with a small minimum distance decoded with the SPA usually displays an error floor at a relatively high error rate and has a poor block-error performance, just like most of the turbo

codes. For example, consider the (273, 191) computer-generated code with $\gamma = 3$ whose error performance is shown in Figure 17.10. Its error performance curve displays an error floor at $\text{BER} = 5 \times 10^{-6}$. In computation of the extrinsic value or LLR of a code bit at each iteration of SPA decoding, large column and row weights (i.e., large number of orthogonal check-sums, with each check-sum consisting of many other code bits) result in a large contribution of information (or help) from a large number of other code bits. This results in a better extrinsic (or LLR) estimate, better error performance, and faster decoding convergence. The error coefficient affects the error performance of the code in the range of small SNRs or high error rates. Smaller error coefficients give better error performance in the small SNR range.

17.7 CODE CONSTRUCTION BY COLUMN AND ROW SPLITTING

A finite-geometry LDPC code C of length n can be extended by splitting each column of its parity-check matrix $\mathbb{H}$ into multiple columns. This results in a new parity-check matrix with smaller density and hence a new LDPC code. If columns are split properly, very good extended finite-geometry LDPC codes can be obtained. Some of the extended finite-geometry LDPC codes constructed perform amazingly well with SPA decoding, achieving an error performance only a few tenths of a decibel away from the Shannon limit.

Let $\mathbb{g}_0, \mathbb{g}_1, \cdots, \mathbb{g}_{n-1}$ denote the columns of the parity-check matrix $\mathbb{H}$. First, we consider splitting each column of $\mathbb{H}$ into the same number of columns. All the new columns have the same length as the original column. The weight (or 1's) of the original column is distributed among the new columns. A regular column weight distribution can be done as follows. Let q be a positive integer such that $2 \leq q \leq \gamma$. Dividing γ by q, we have

$$\gamma = q \times \gamma_{ext} + b,$$

where $0 \leq b < q$. We split each column $\mathbb{g}_i$ of $\mathbb{H}$ into q columns $\mathbb{g}_{i,1}, \mathbb{g}_{i,2}, \cdots, \mathbb{g}_{i,q}$ such that the first b columns, $\mathbb{g}_{i,1}, \mathbb{g}_{i,2}, \cdots, \mathbb{g}_{i,b}$, have weight $\gamma_{ext} + 1$, and the next $q - b$ columns, $\mathbb{g}_{i,b+1}, \mathbb{g}_{i,b+2}, \cdots, \mathbb{g}_{i,q}$, have weight γ_{ext}. The distribution of γ "ones" of $\mathbb{g}_i$ into $\mathbb{g}_{i,1}, \mathbb{g}_{i,2}, \cdots, \mathbb{g}_{i,q}$ is carried out in a rotating manner. In the first rotation, the first 1 of $\mathbb{g}_i$ is put in $\mathbb{g}_{i,1}$, the second 1 of $\mathbb{g}_i$ is put in $\mathbb{g}_{i,2}$, and so on. In the second rotation, the $(q+1)$th 1 of $\mathbb{g}_i$ is put in $\mathbb{g}_{i,1}$, the $(q+2)$th 1 of $\mathbb{g}_i$ is put in $\mathbb{g}_{i,2}$ and so on. This rotating distribution of the 1's of $\mathbb{g}_i$ continues until all the 1's of $\mathbb{g}_i$ have been distributed into the q new columns.

The column splitting results in a new parity-check matrix $\mathbb{H}_{ext}$ with qn columns that has the following structural properties: (1) each row has weight ρ; (2) each column has either weight γ_{ext} or weight $\gamma_{ext} + 1$; (3) any two columns have at most one 1 in common. If the density of $\mathbb{H}$ is r, the density of $\mathbb{H}_{ext}$ is then r/q. Therefore, the column splitting results in a new parity-check matrix with smaller density. The null space of $\mathbb{H}_{ext}$ gives an extended finite-geometry LDPC code C_{ext}. We call C_{ext} the qth extension of C. If γ is not divisible by q, then the columns of $\mathbb{H}_{ext}$ have two different weights, γ_{ext} and $\gamma_{ext} + 1$. Therefore, a code bit of the extended code C_{ext} is either checked by γ_{ext} check-sums or by $\gamma_{ext} + 1$ check-sums. In this case, the extended finite-geometry LDPC code C_{ext} is an irregular LDPC code.

EXAMPLE 17.14

Consider the (4095, 3367) two-dimensional type-I (0, 6)th-order cyclic EG-LDPC code discussed in Example 17.11. Suppose we split each column of the parity-check matrix of this code into 16 columns with rotating column weight distribution. This column splitting results in a new low-density parity-check matrix of 65520 columns with row weight $\rho = 64$, column weight $\gamma_{ext} = 4$, and density $r = 0.00098$. The null space of this matrix gives a (65520, 61425) extended two-dimensional type-I EG-LDPC code with rate 0.9375. This code decoded with SPA achieves an error performance that is only 0.42 dB away from the Shannon limit at BER $= 10^{-5}$, as shown in Figure 17.12. We see that it has a sharp waterfall error performance. This is the first algebraically constructed linear block code that performs close to the Shannon limit.

A base finite-geometry LDPC code C can be extended into codes of many different lengths. All these extended codes have different rates and different performances.

EXAMPLE 17.15

For $m = 2$ and $s = 7$, the two-dimensional type-I (0, 7)th-order cyclic EG-LDPC code is a (16383, 14197) code with minimum distance $d_{min} = 129$ whose parity-check matrix has row weight $\rho = 128$, column weight $\gamma = 128$, and density $r = 0.007813$

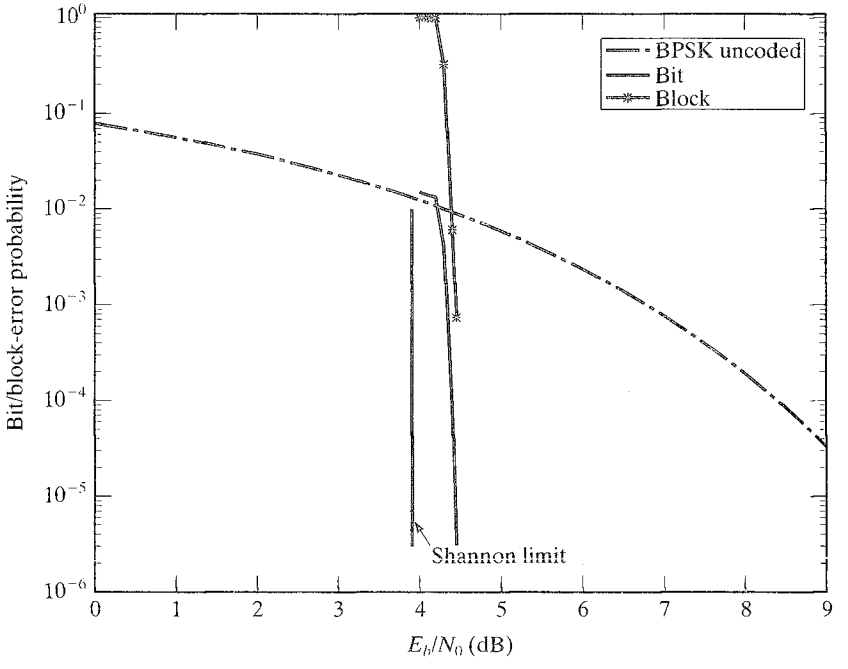


FIGURE 17.12: Bit- and block-error probabilities of the (65520, 61425) extended type-I EG-LDPC code with SPA decoding.

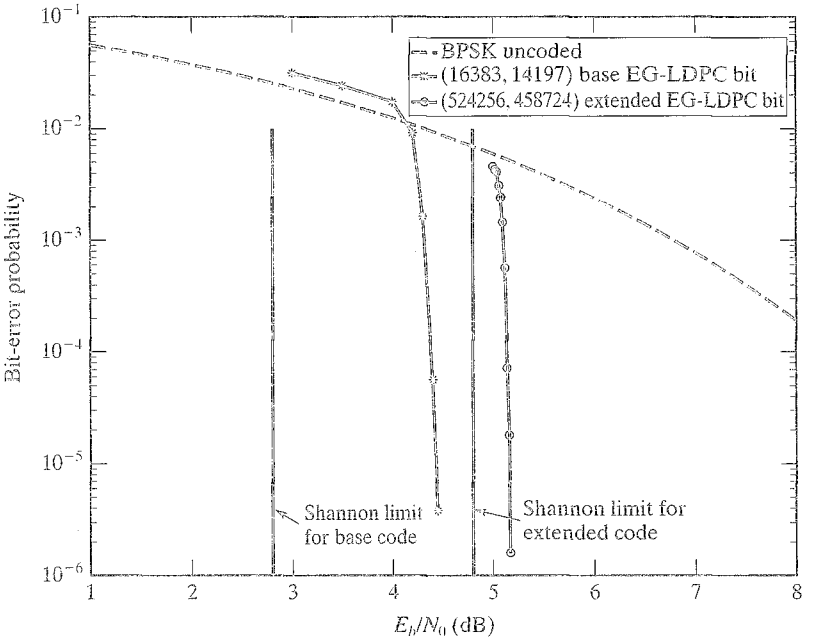


FIGURE 17.13: Error performances of the (16383, 14197) two-dimensional type-I EG-LDPC code and the (524256, 507873) extended EG-LDPC code with SPA decoding.

(the sixth code given in Table 17.1). Suppose we split each column of the parity-check matrix of this code into 32 columns. We obtain an extended EG-LDPC code of length 524,256 and rate approximately 31/32. The density of the parity-check matrix of this extended code is $r = 0.000244$. It is a very sparse matrix. The bit- and block-error performances of this extended LDPC code with SPA decoding are shown in Figure 17.13. At BER $= 10^{-4}$, the performance of the extended LDPC code is only 0.3 dB away from the Shannon limit.

EXAMPLE 17.16

Let $m = s = 3$. Consider the three-dimensional type-I (0, 3)th-order cyclic EG-LDPC code discussed in Example 17.12, which is a (511, 139) code. Its parity matrix is a 4599×511 matrix with $\rho = 8, \gamma = 72$, and density $r = 0.01565$. Suppose we extend this code by splitting each column of its parity-check matrix $\mathbb{H}$ into 24 columns. Then, the extended code C_{ext} is a (12264, 7665) LDPC code with rate $R = 0.625$. The extension results in a high-rate code. The density of the parity-check matrix $\mathbb{H}_{ext}$ of this extended code is $r_{ext} = 0.000652$, and the column weight of $\mathbb{H}_{ext}$ is $\gamma_{ext} = 3$. The bit-error performances of this extended LDPC code and its base code with SPA decoding are shown in Figure 17.14. The performance of the extended code is 1 dB from the Shannon limit at BER $= 10^{-4}$.

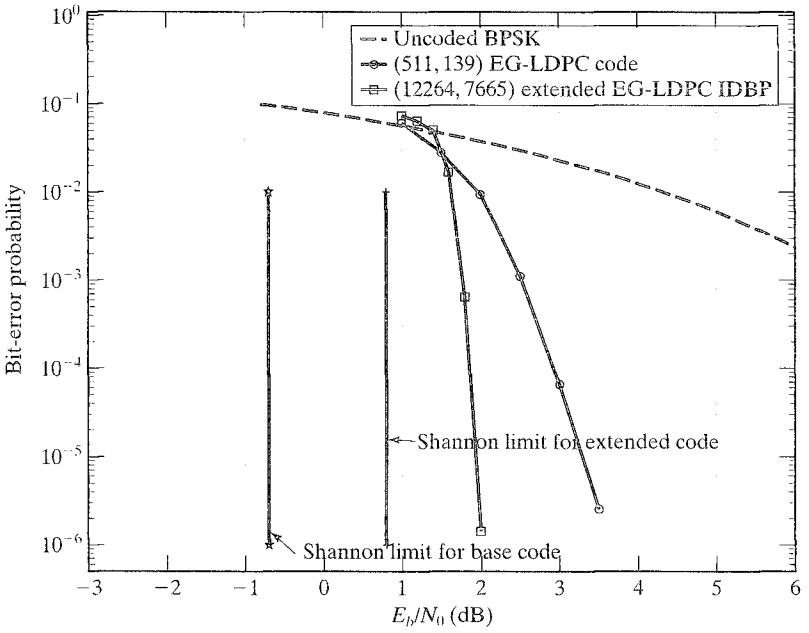


FIGURE 17.14: Error performances of the (511, 139) three-dimensional type-I (0, 3)th-order cyclic EG-LDPC code and the (12264, 7665) extended EG-LDPC code.

Column splitting of the parity-check matrix of a cyclic or quasi-cyclic finite-geometry LDPC code may result in an extended code that is neither cyclic nor quasi-cyclic; however, if we arrange the rows of the parity-check matrix into circulant submatrices and then split each column into a fixed number of new columns with column weight distributed in a rotating and circular manner, the resultant extended code can be put in quasi-cyclic form. To see this, we consider the m-dimensional type-I cyclic $(0, s)$th-order EG-LDPC code constructed based on the lines and points of the m-dimensional Euclidean geometry EG$(m, 2^s)$. The parity-check matrix $\mathbb{H}_{EG,c}^{(1)}$ of this code consists of J_0 rows and n columns, where J_0 is given by (17.9), and $n = 2^{ms} - 1$. The rows of $\mathbb{H}_{EG,c}^{(1)}$ can be grouped into $Kn \times n$ circulant submatrices, $\mathbb{H}_1, \mathbb{H}_2, \cdots, \mathbb{H}_K$, where $K = J_0/n = (2^{(m-1)s} - 1)/(2^s - 1)$ given by (17.20). Each circulant submatrix $\mathbb{H}_i$ is obtained by cyclically shifting the incidence vector of a line (not passing through the origin) n times. Therefore, $\mathbb{H}_{EG,c}^{(1)}$ can be put in the following form [17]:

$$\mathbb{H}_{EG,c}^{(1)} = \begin{bmatrix} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \vdots \\ \mathbb{H}_K \end{bmatrix}. \tag{17.64}$$

Now, we split each column of $\mathbb{H}_{EG,c}^{(1)}$ into q columns in a manner similar to that described earlier in this section; however, the 1-components in a column of $\mathbb{H}_{EG,c}^{(1)}$

must be labeled in a specific circular order [17]. For $0 \leq j < n$, let $\mathbf{g}_j^{(i)}$ be the jth column of the ith circulant submatrix $\mathbf{H}_i$. Then, the jth column $\mathbf{g}_j$ of $\mathbf{H}_{EG,c}^{(1)}$ consists of $\mathbf{g}_j^{(1)}, \mathbf{g}_j^{(2)}, \cdots, \mathbf{g}_j^{(K)}$, with one on top of the other. We label the 1-components of the jth column $\mathbf{g}_j$ of $\mathbf{H}_{EG,c}^{(1)}$ as follows. We label the first 1-component of the jth column $\mathbf{g}_j^{(1)}$ on or below the main diagonal line of circulant $\mathbf{H}_1$ and inside $\mathbf{H}_1$ as the first 1-component of the jth column $\mathbf{g}_j$ of $\mathbf{H}_{EG,c}^{(1)}$. We label the first 1-component of $\mathbf{g}_j^{(2)}$ on or below the main diagonal line of circulant $\mathbf{H}_2$ and inside $\mathbf{H}_2$ as the second 1-component of $\mathbf{g}_j$. We continue this labeling process until we label the first 1-component of $\mathbf{g}_j^{(K)}$ on or below the main diagonal line of circulant $\mathbf{H}_K$ and inside $\mathbf{H}_K$ as the Kth 1-component of column $\mathbf{g}_j$. Then, we come back to circulant $\mathbf{H}_1$ and start the second round of the labeling process. We label the second 1-component of $\mathbf{g}_j^{(1)}$ below the main diagonal line of $\mathbf{H}_1$ and inside $\mathbf{H}_1$ as the $(K+1)$th 1-component of $\mathbf{g}_j$. We label the second 1-component of $\mathbf{g}_j^{(2)}$ below the main diagonal line of circulant $\mathbf{H}_2$ and inside $\mathbf{H}_2$ as the $(K+2)$th 1-component of $\mathbf{g}_j$. We continue the second round of the labeling process until we reach the Kth circulant $\mathbf{H}_K$ again. Then, we loop back to circulant $\mathbf{H}_1$ and continue the labeling process. During the labeling process, whenever we reach the bottom of a circulant matrix $\mathbf{H}_i$, we wrap around to the top of the same column $\mathbf{g}_j^{(i)}$ of $\mathbf{H}_i$. This labeling process continues until all the 1-components of $\mathbf{g}_j$ are labeled. Once the labeling of 1-components of $\mathbf{g}_j$ is completed, we distribute the 1-components of $\mathbf{g}_j$ into q new columns in the same rotating manner as described earlier in this section. Clearly, the labeling and weight distribution can be carried out at the same time. Let $\mathbf{H}_{EG,ext}^{(1)}$ be the new matrix resulting from the column splitting. Then, $\mathbf{H}_{EG,ext}^{(1)}$ consists of K $n \times nq$ submatrices $\mathbf{H}_{ext,1}, \mathbf{H}_{ext,2}, \cdots, \mathbf{H}_{ext,K}$. For $1 \leq i \leq K$, the rows of $\mathbf{H}_{ext,i}$ are cyclic shifts of the first row q bits at a time. As a result, the null space of $\mathbf{H}_{EG,ext}^{(1)}$ gives an extended finite-geometry LDPC code in quasi-cyclic form.

To extend the m-dimensional type-II quasi-cyclic $(0, s)$th-order EG-LDPC code, we use the parity-check matrix $\mathbf{H}_{EG,ext}^{(2)}$ in the form given by (17.21) (or (17.22)). Then, we split each column of $\mathbf{H}_{EG,qc}^{(2)}$ into q columns with column weight distributed in a rotating manner. We also can take the transpose of $\mathbf{H}_{EG,c}^{(1)}$ given by (17.64) as the parity-check matrix and then split each column.

For a PG-LDPC code, the number J of rows of its parity-check matrix may not be divisible by the number n of columns of the matrix. In this case, not all the submatrices of the parity-check matrix $\mathbf{H}_{PG}^{(1)}$ of a type-I PG-LDPC code can be arranged as $n \times n$ square circulant matrices. Some of them are nonsquare circulant matrices, as shown in Example 17.8. The rows of such a nonsquare circulant matrix are still cyclic shifts of the first row, and the number of rows divides n. We can remove all the nonsquare circulant submatrices from $\mathbf{H}_{PG}^{(1)}$. The result is a new parity-check matrix

$$\mathbf{H}_{PG,d}^{(1)} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_K \end{bmatrix}.$$

in which every submatrix $\mathbb{H}_i$ is an $n \times n$ square circulant matrix. $\mathbb{H}_{PG,d}^{(1)}$ is still a low-density parity-check matrix that satisfies all the conditions given in Definition 17.1. The null space of $\mathbb{H}_{PG,d}^{(1)}$ is a cyclic supercode of the PG-LDPC code generated by $\mathbb{H}_{PG}^{(1)}$. Then, we can extend this new code by column splitting as described earlier, in a rotating and circular manner, and the resultant extended code will be in quasi-cyclic form. The null space of the transpose $[\mathbb{H}_{PG,d}^{(1)}]^T$ of $\mathbb{H}_{PG,d}^{(1)}$ gives a shortened type-II PG-LDPC code that is quasi-cyclic. Then, we can obtain extended quasi-cyclic LDPC codes by column splitting of $[\mathbb{H}_{PG,d}^{(1)}]^T$.

As an example, consider the parity-check matrix $\mathbb{H}_{PG}^{(1)}$ of the (85, 24) three-dimensional type-I PG-LDPC code $C_{PG}^{(1)}(3, 0, 2)$ given in Example 17.8. There is only one nonsquare circulant submatrix in $\mathbb{H}_{PG}^{(1)}$. Removing this nonsquare circulant submatrix from $\mathbb{H}_{PG}^{(1)}$, we obtain a new matrix

$$\mathbb{H}_{PG,d}^{(1)} = \begin{bmatrix} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \mathbb{H}_3 \\ \mathbb{H}_4 \end{bmatrix}.$$

Each submatrix in $\mathbb{H}_{PG,d}^{(1)}$ is an 85×85 circulant matrix. The null space of $\mathbb{H}_{PG,d}^{(1)}$ gives the same (85, 24) LDPC code with a minimum distance of at least 21. We can obtain extended codes by splitting the columns of $\mathbb{H}_{PG,d}^{(1)}$. The null space of $[\mathbb{H}_{PG,d}^{(1)}]^T$ gives a (340, 279) LDPC code with a minimum distance of at least 6.

We also can obtain LDPC codes by splitting each row of the parity-check matrix $\mathbb{H}$ of a base finite-geometry LDPC code into multiple rows. The resultant code has the same length as the base code but has a lower code rate. Furthermore, proper row splitting also preserves the cyclic or quasi-cyclic structure of the code. Clearly, we can obtain LDPC codes by splitting both columns and rows of the parity-check matrix of a base finite-geometry LDPC code.

EXAMPLE 17.17

For $m = 2$ and $s = 4$, the two-dimensional type-I (0, 4)th-order cyclic EG-LDPC code is a (255, 175) code (the third code in Table 17.1). Its parity-check matrix $\mathbb{H}$ has the following parameters: $\rho = 16, \gamma = 16$, and density $r = 0.0627$. Its error performance with SPA decoding is shown in Figure 17.15. If we split each column of its parity-check matrix $\mathbb{H}$ into five columns and split each row into two rows, we obtain a parity-check matrix $\mathbb{H}'$ whose columns have two different weights, 3 and 4, and whose rows have the same weight 8. The null space of this new parity-check matrix is a (1275, 765) LDPC code with rate 0.6 and density $r' = 0.00627$. The performance of this new code with SPA decoding is also shown in Figure 17.15. It is a relatively short code, but at a BER of 10^{-4}, it performs 1.8 dB from the Shannon limit.

EXAMPLE 17.18

Consider the (4095, 3367) two-dimensional type-I cyclic (0, 6)th-order EG-LDPC code $C_{EG}^{(1)}(2, 0, 6)$ given in Table 17.1. If we split each column and each row of

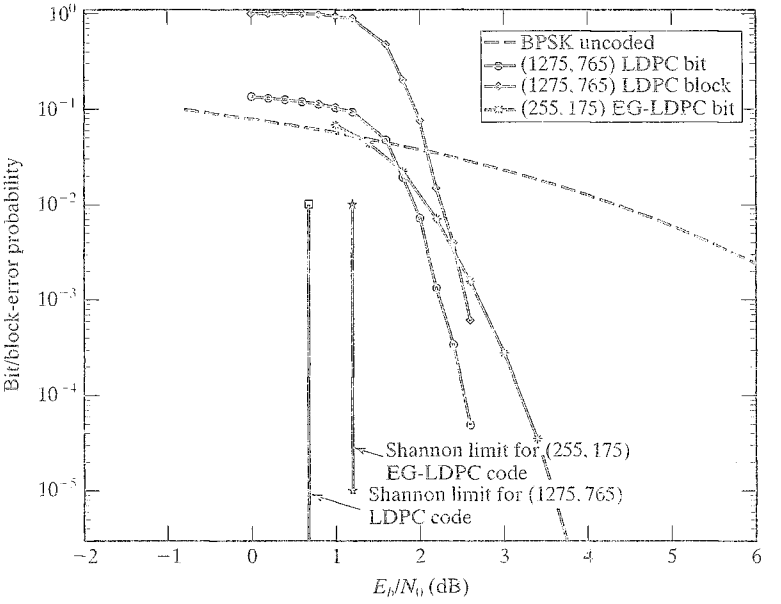


FIGURE 17.15: Bit- and block-error probabilities of the (255, 175) type-I EG-LDPC code and the (1275, 765) extended EG-LDPC code based on SPA decoding.

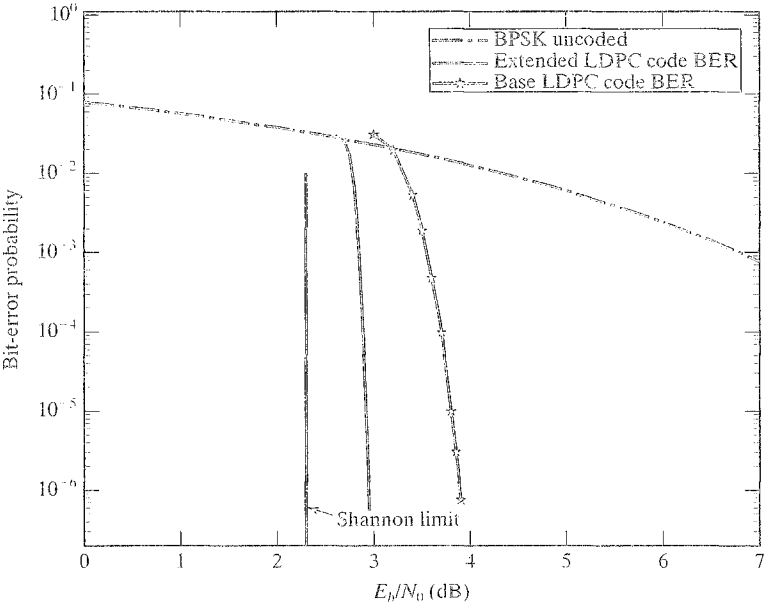


FIGURE 17.16: Bit- and block-error probabilities of the (4095, 3367) type-I EG-LDPC code and the (65520, 53235) extended EG-LDPC code based on SPA decoding.

the parity-check matrix of this code into 16 columns and 3 rows, respectively, we obtain a new parity-check matrix with column weight 4 and row weights 21 and 22. The null space of this new parity-check matrix gives a (65520, 53235) extended EG-LDPC code. This extended code and its base (4095, 3367) code have about the same rate. The error performance of the extended code is shown in Figure 17.16, and it performs 0.7 dB from the Shannon limit at a BER of 10^{-5}; however, the performance of the base code is 1.40 dB from the Shannon limit. This example shows that by a proper combination of column and row splittings of the parity-check matrix of a base finite-geometry LDPC code, we can obtain a new LDPC code that has about the same rate but better error performance.

17.8 BREAKING CYCLES IN TANNER GRAPHS

The examples given in the previous section show that properly splitting each column of the parity-check matrix $\mathbf{H}$ of a finite-geometry LDPC code C into multiple columns results in an extended LDPC code C_{ext} that performs very close to the Shannon limit with SPA decoding. A reason for this is that column splitting reduces the degree of each code-bit vertex in the Tanner graph $\mathcal{G}$ of the base code and hence reduces the number of cycles in the graph. Splitting a column of $\mathbf{H}$ into q columns results in splitting a code-bit vertex of the Tanner graph $\mathcal{G}$ of the base code into q code-bit vertices in the Tanner graph $\mathcal{G}_{ext}$ of the extended code C_{ext}. Each code-bit vertex in $\mathcal{G}_{ext}$ is connected to a smaller number of check-sum vertices than in $\mathcal{G}$. Figure 17.17(a) shows that splitting a column in $\mathbf{H}$ into two columns results in splitting a code-bit vertex in the Tanner graph $\mathcal{G}$ into two code-bit vertices in the Tanner graph $\mathcal{G}_{ext}$. The original code-bit vertex has a degree of 4, but each code-bit vertex after splitting has a degree of 2. This code-bit splitting breaks some cycles in the Tanner graph $\mathcal{G}$ of the base code C. Figures 17.18(a) and 17.19 show the breaking

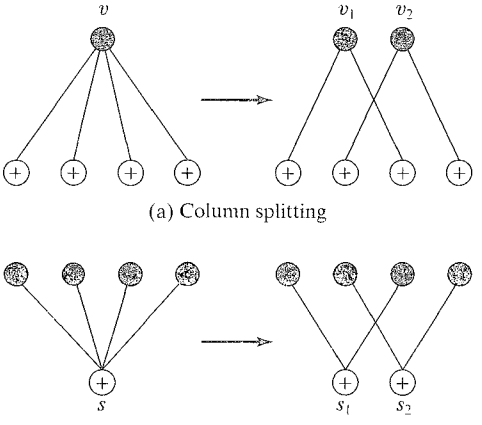
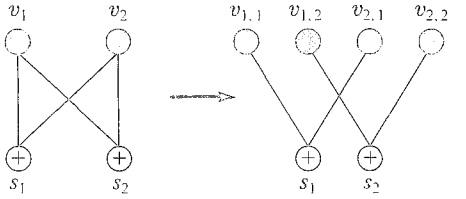
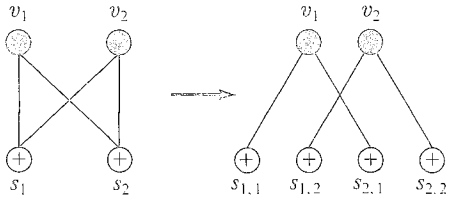


FIGURE 17.17: Graph decomposition by column/row splittings.



(a) Breaking a cycle of length 4 by column splitting operation.



(b) Breaking a cycle of length 4 by row splitting operation.

FIGURE 17.18: Cycle decomposition.

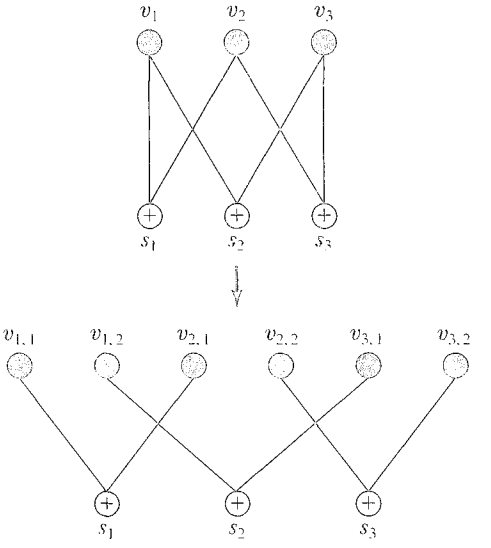


FIGURE 17.19: Decomposition of a cycle of length 6 by column splitting.

of cycles of lengths 4 and 6. Therefore, column splitting of a base finite-geometry LDPC code breaks many cycles of its Tanner graph and results in an extended LDPC code whose Tanner graph has many fewer cycles. This reduction in cycles in the Tanner graph improves the performance of the code with SPA decoding. In fact, cycles can be broken by column splitting of the parity-check matrix for any linear block code.

EXAMPLE 17.19

Consider the (7, 4) cyclic Hamming code generated by polynomial $g(X) = 1+X+X^3$. Its parity-check polynomial is $h(X) = 1+X+X^2+X^4$, and its dual code is generated by $X^4h(X^{-1}) = 1+X^2+X^3+X^4$. A parity-check matrix for the (7, 4) cyclic Hamming code can be formed by cyclic shifting the vector (1 0 1 1 1 0 0) six times, as shown in Figure 17.20.

The Tanner graph for this code is shown in Figure 17.21. It contains 21 cycles of length 4, and each code-bit vertex is on 6 of these cycles. If we split each column of $\mathbf{H}$ into two columns, we obtain a new parity-check matrix $\mathbf{H}_{ext}$, as shown in Figure 17.22. The null space of $\mathbf{H}_{ext}$ gives a (14, 8) code. From $\mathbf{H}_{ext}$, we can easily check that no four 1's appear at the four corners of a rectangle in $\mathbf{H}_{ext}$. This implies that the Tanner graph of the extended code does not have any cycle of length 4. Figure 17.23 shows the error performance of this extended Hamming code using SPA decoding, which is almost identical to MLD performance.

As another example, consider the (23, 12) Golay code. A parity-check matrix $\mathbf{H}$ for this code is shown in Figure 17.24. Based on this parity-check matrix, the Tanner graph for this code has a total of 1748 cycles of length 4, and each code-bit vertex is on 152 such cycles. The error performance of this code with SPA decoding

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

FIGURE 17.20: A parity-check matrix for the (7, 4) Hamming code generated by $g(X) = 1 + X + X^3$.

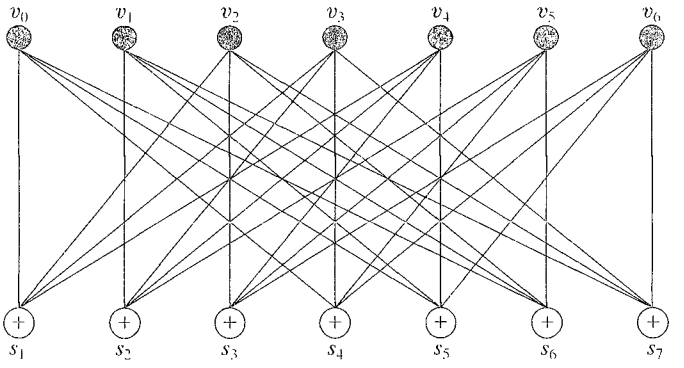


FIGURE 17.21: Tanner graph for the (7, 4) Hamming code.

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

FIGURE 17.22: Parity-check matrix for the (14, 8) extended Hamming code.

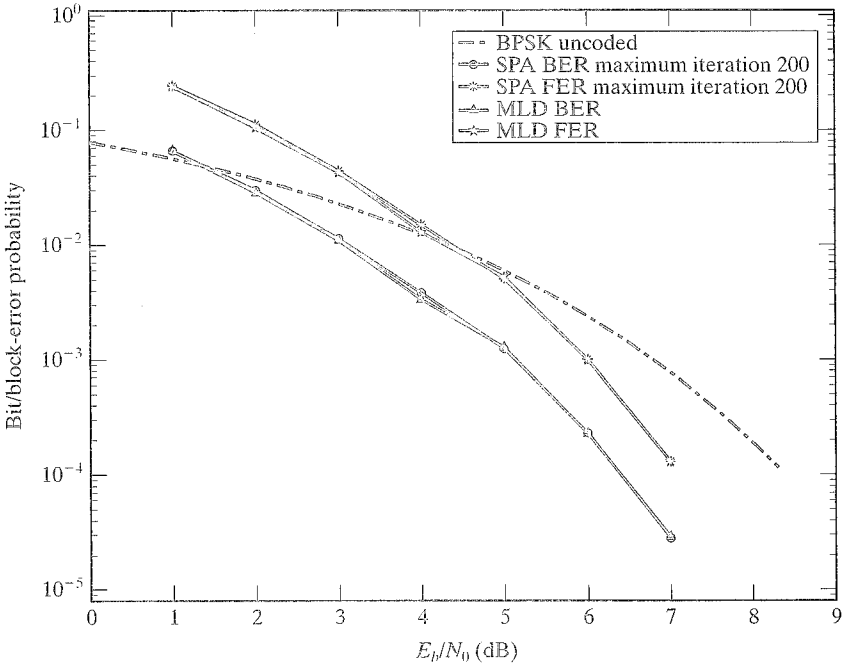


FIGURE 17.23: Error performance of the (14, 8) extended LDPC code.

with 200 maximum iterations is shown in Figure 17.25. We see that at BER = 10^{-5}, with 200 iterations, the error performance with SPA decoding for this code is 0.6 dB away from MLD, which demonstrates how cycles of length 4 prevent the SPA decoding algorithm from converging to MLD performance. Suppose we split each column of the parity-check matrix given in Figure 17.24 into two columns in a random manner. We find that this column splitting results in a (46, 24) LDPC code with a minimum distance of 5 whose Tanner graph contains only 106 cycles of length 4. The weight distribution of this code is given in Table 17.3. We see that the extended code has a very small number of codewords of low weights: 5, 6, 7, and 8. There are 82 codewords of these four low weights, which is fewer than the 253 minimum-weight-7 codewords of the (23, 12) Golay code. We see that column

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

FIGURE 17.24: A parity-check matrix for the (23, 12) Golay code.

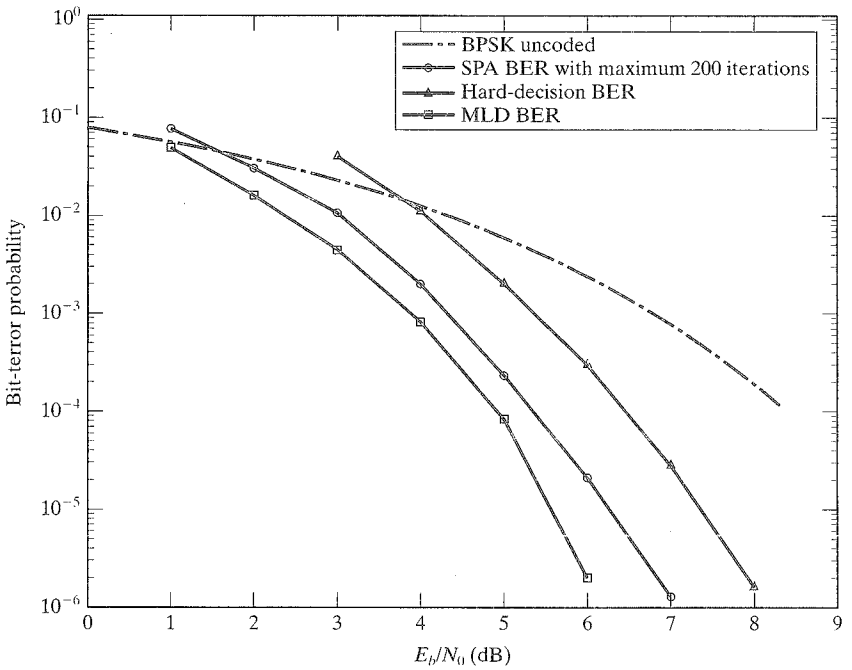


FIGURE 17.25: Bit-error performance of the (23, 12) Golay code with SPA decoding.

TABLE 17.3: Weight distribution of the extended (46, 24) Golay code

Weight	Number of codewords	Weight	Number of codewords
0	1	23	1963152
1	0	24	1882241
2	0	25	1656075
3	0	26	1336148
4	0	27	989121
5	1	28	671322
6	5	29	417751
7	18	30	237105
8	58	31	122262
9	280	32	57390
10	1000	33	24069
11	3151	34	9034
12	9034	35	3151
13	24069	36	1000
14	57390	37	280
15	122262	38	58
16	237105	39	18
17	417751	40	5
18	671322	41	1
19	989121	42	0
20	1336148	43	0
21	1656075	44	0
22	1882241	45	0
		46	1

splitting has a (weight) spectral thinning effect. The bit-error performances of this extended Golay code with SPA decoding and MLD are shown in Figure 17.26. We see that for this extended (46, 24) Golay code, SPA decoding still does not converge to the MLD performance owing to the cycles of length 4 in its Tanner graph.

Given a finite-geometry LDPC code specified by a parity-check matrix $\mathbb{H}$, we can split each column of $\mathbb{H}$ in a different manner and into different numbers of columns. Consequently, many extended finite-geometry LDPC codes can be obtained by splitting columns of the parity-check matrix $\mathbb{H}$. Of course, if columns of $\mathbb{H}$ are split differently, the resultant extended code is an irregular LDPC code.

Splitting a row in the $\mathbb{H}$ matrix is equivalent to splitting a check-sum vertex in the Tanner graph of the code and hence reduces the degree of the vertex, as shown in Figure 17.17(b). Therefore, row splitting of the parity-check matrix of a base code can also break many cycles in the Tanner graph of the base code. An example of cycle breaking by check-sum vertex splitting is shown in Figure 17.18(b). Clearly, a combination of column and row splittings will break many cycles in the Tanner graph of the base code; this may result in a very good LDPC code.

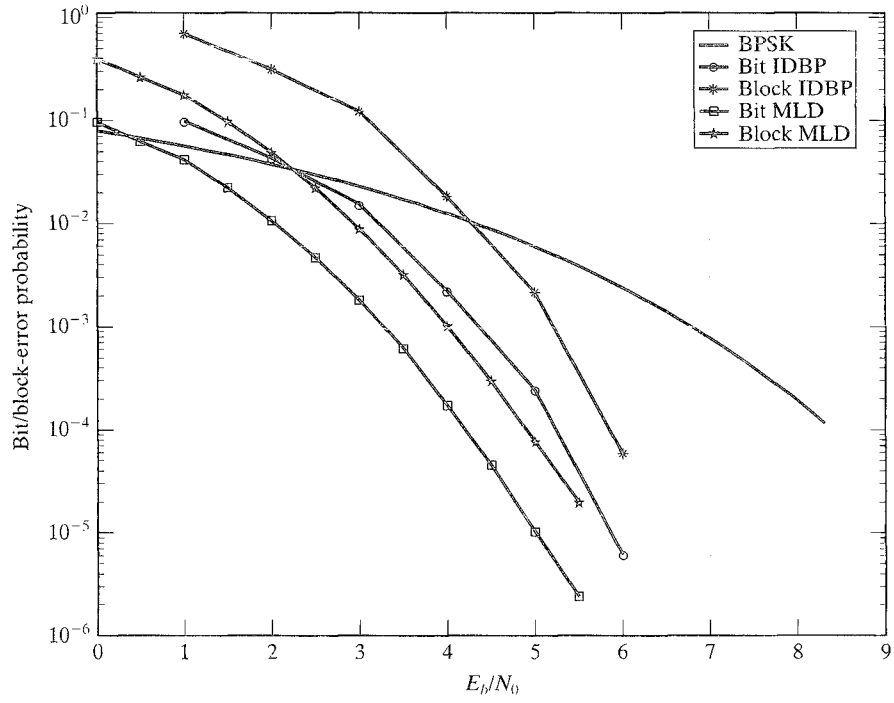


FIGURE 17.26: Error performance of the (46, 24) extended Golay code based on SPA decoding with 200 maximum iterations and MLD.

17.9 SHORTENED FINITE-GEOMETRY LDPC CODES

Finite-geometry LDPC codes can be shortened to obtain new LDPC codes, by deleting properly selected columns from their parity-check matrices. For a type-I finite-geometry LDPC code, the columns to be deleted correspond to a properly chosen set of points in the finite-geometry base on which the code is constructed. For a type-II finite-geometry LDPC code, the columns to be deleted correspond to a chosen set of lines in the finite geometry. Several shortening techniques are presented in this section.

First, we consider shortening type-I finite-geometry LDPC codes. We use a type-I EG-LDPC code to explain the shortening techniques. The same techniques can be used to shorten a type-I PG-LDPC code. Consider the m-dimensional type-I cyclic $(0, s)$th-order EG-LDPC code $C^{(1)}_{EG,c}(m, 0, s)$ constructed based on the m-dimensional Euclidean geometry $\mathrm{EG}(m, 2^s)$. Let $\mathbb{H}^{(1)}_{EG,c}$ be the parity-check matrix of this code. Then, the rows of $\mathbb{H}^{(1)}_{EG,c}$ are the incidence vectors of the lines in $\mathrm{EG}(m, 2^s)$ that do not pass through the origin of $\mathrm{EG}(m, 2^s)$, and the columns of $\mathbb{H}^{(1)}_{EG,c}$ correspond to the nonorigin points of $\mathrm{EG}(m, 2^s)$. For $1 \leq q < 2^s$, let S be a set of q parallel $(m-1)$-flats in $\mathrm{EG}(m, 2^s)$ that does not contain the origin (i.e., the origin of $\mathrm{EG}(m, 2^s)$ is not contained in any of the q parallel $(m-1)$-flats). Because each $(m-1)$-flat in $\mathrm{EG}(m, 2^s)$ contains $2^{(m-1)s}$ points, and the q $(m-1)$-flats in S

are disjoint, the total number of points in S is $q2^{(m-1)s}$. Because each $(m-1)$-flat contains $2^{(m-2)s}(2^{(m-1)s}-1)/(2^s-1)$ lines in $\mathrm{EG}(m, 2^s)$, S contains

$$J_1 = 2^{(m-2)s}(2^{(m-1)s}-1)q/(2^s-1) \tag{17.65}$$

complete lines of $\mathrm{EG}(m, 2^s)$ not passing through the origin. Now, we delete the columns of $\mathbb{H}_{EG,c}^{(1)}$ that correspond to the points in S. With this deletion of columns, the rows in $\mathbb{H}_{EG,c}^{(1)}$ that correspond to the lines contained in S become rows of zeros in the punctured matrix. Removing the rows of zeros from the punctured matrix, we obtain a new matrix $\mathbb{H}_{EG,s}^{(1)}$ that has $2^{ms} - q2^{(m-1)s} - 1$ columns and $J_0 - J_1$ rows, where J_0 is given by (17.9). Every column in this new matrix $\mathbb{H}_{EG,s}^{(1)}$ still has the same weight $(2^{ms}-1)/(2^s-1)-1$ as that of the original matrix $\mathbb{H}_{EG,c}^{(1)}$, but its rows have different weights. The weights of the rows corresponding to the lines that are completely outside of S are 2^s; however, the weights of those rows corresponding to the lines that are partially contained in S are less than 2^s. Because each row in the punctured matrix $\mathbb{H}_{EG,s}^{(1)}$ corresponds to either a complete line or a partial line in $\mathrm{EG}(m, 2^s)$, no two rows of $\mathbb{H}_{EG,s}^{(1)}$ have more than one 1-component in common, and hence no two columns have more than one 1-component in common. Therefore, the punctured matrix $\mathbb{H}_{EG,s}^{(1)}$ is an irregular low-density matrix, and its null space gives an irregular LDPC code of length $n_s = 2^{ms} - q2^{(m-1)s} - 1$, denoted by $C_{EG,s}^{(1)}(m, 0, s)$. It is a shortened code of the cyclic EG-LDPC code $C_{EG,c}^{(1)}(m, 0, s)$. The minimum distance of this shortened LDPC code is still at least $(2^{ms}-1)/(2^s-1)$.

EXAMPLE 17.20

Consider the (255, 175) two-dimensional type-I cyclic (0, 4)th-order EG-LDPC code constructed based on the two-dimensional Euclidean geometry $\mathrm{EG}(2, 2^4)$ (the third code in Table 17.1). This code has rate 0.6863 and a minimum distance of 17. The parity-check matrix $\mathbb{H}_{EG,c}^{(1)}$ of the code is a 255×255 square circulant matrix whose rows are the incidence vectors of the lines in $\mathrm{EG}(2, 2^4)$ that do not pass through the origin. The column and row weights of this parity-check matrix are both 16. Let S be a set of two parallel lines in $\mathrm{EG}(2, 2^4)$ that do not pass through the origin. Because each line in $\mathrm{EG}(2, 2^4)$ consists of 16 points, S contains 32 points. Suppose we delete those columns in $\mathbb{H}_{EG,c}^{(1)}$ that correspond to the points in S. This deletion results in a punctured matrix with two rows of zeros that correspond to the two lines in S. Removing these two rows of zeros, we obtain a 252×223 matrix $\mathbb{H}_{EG,s}^{(1)}$. All the columns of this new matrix still have the same weight 16, but its rows have two different weights. Thirteen rows of $\mathbb{H}_{EG,s}^{(1)}$ have weight 16, and all the other rows have weight 14. The null space of this new matrix gives a (223, 145) LDPC code with a minimum distance of at least 17 and rate 0.650. The puncturing removes 30 information bits and two parity bits, respectively. The error performance of this shortened EG-LDPC code is shown in Figure 17.27. We see that the shortened code and the original base code have about the same error performance.

A straightforward generalization of the foregoing shortening technique is to choose a set S of q $(m-i)$-flats in $\mathrm{EG}(m, 2^s)$ with $0 < i < m$. The $(m-i)$-flats

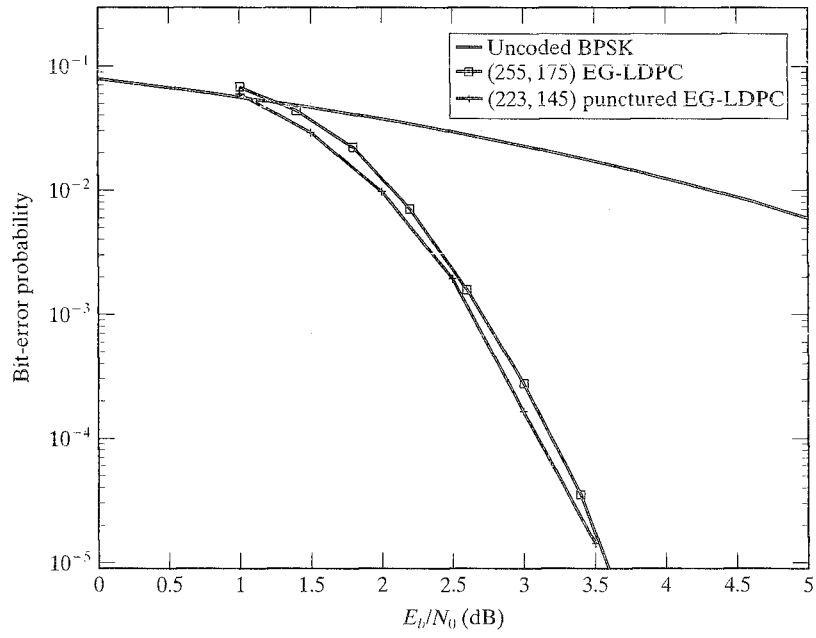


FIGURE 17.27: Bit-error probabilities of the (255, 175) EG-LDPC code and its (223, 145) punctured code.

in S are not necessarily parallel to each other; some of them may intersect. Then, the puncturing of the parity check matrix of the m-dimensional type-I EG-LDPC code is based on the points contained in S. The puncturing still does not change the column weight but results in many different row weights. The punctured matrix with the rows of zeros removed generates a shortened EG-LDPC code.

One approach to shortening the m-dimensional type-II quasi-cyclic EG-LDPC code constructed based on EG$(m, 2^s)$ is to put its parity-check matrix $\mathbb{H}^{(2)}_{EG,qc}$ in circulant form. In this case, $\mathbb{H}^{(2)}_{EG,qc}$ is simply the transpose of $\mathbb{H}^{(1)}_{EG,c}$ given by (17.64),

$$\mathbb{H}^{(2)}_{EG,qc} = [\mathbb{H}^{(1)}_{EG,c}]^T = \left[\mathbb{H}_1^T, \mathbb{H}_2^T, \cdots, \mathbb{H}_K^T\right], \tag{17.66}$$

where $K = (2^{(m-1)s} - 1)/(2^s - 1)$. Because each submatrix $\mathbb{H}_i$ is an $n \times n$ square circulant matrix, its transpose $\mathbb{H}_i^T$ is still an $n \times n$ square circulant matrix, where $n = 2^{ms} - 1$. Each column of $\mathbb{H}_i^T$ is the incidence vector of a line in EG$(m, 2^s)$ not passing through the origin and has weight 2^s. Because $\mathbb{H}_i^T$ is a square circulant, its rows and columns have the same weight 2^s. For $0 < q < K$, suppose we remove q circulant submatrices from $\mathbb{H}^{(2)}_{EG,qc}$ given by (17.66). The result is a punctured matrix, denoted by $\mathbb{H}^{(2)}_{EG,qc,s}$, with $n = 2^{ms} - 1$ rows and $(K - q)n$ columns. The column and row weights of this new matrix are 2^s and $(K - q)2^s$, respectively. The null space of this new matrix gives a regular LDPC code of length $(K - q)n$ and minimum distance of at least $2^s + 1$. The new LDPC code is still quasi-cyclic.

EXAMPLE 17.21

For $m = s = 3$, the type-II quasi-cyclic EG-LDPC code constructed based on the three-dimensional Euclidean geometry EG(3, 2^3) is a (4599, 4227) code with a minimum distance of at least 9 and rate 0.92. This code was considered in Example 17.6, and its error performance with SPA decoding is shown in Figure 17.9. The parity-check matrix $\mathbf{H}_{EG,qc}^{(2)}$ of this code is a 511×4599 matrix. In circulant form, this matrix consists of nine 511×511 circulant submatrices. Suppose we remove from this matrix 1, 3, and 6 circulant submatrices, respectively, resulting in a 511×4088, a 511×3066, and a 511×1533 punctured matrix. The null spaces of these three matrices give a (4088, 3716), a (3066, 2694), and a (1533, 1161) shortened type-II EG-LDPC code, respectively. All three codes have a minimum distance of at least 9, and their rates are 0.909, 0.878, and 0.757, respectively. The bit-error performances of these three shortened type-II EG-LDPC codes and the original base code are shown in Figure 17.28.

The foregoing techniques for shortening EG-LDPC codes can be applied to shortening PG-LDPC codes. Shortening finite-geometry LDPC codes results in a large class of regular or irregular LDPC codes.

The row-splitting technique described in Section 17.7 can be applied to splitting the rows of the parity-check matrix of a shortened finite-geometry LDPC code. The result is an expanded low-density matrix whose null space gives a new LDPC code with a lower rate.

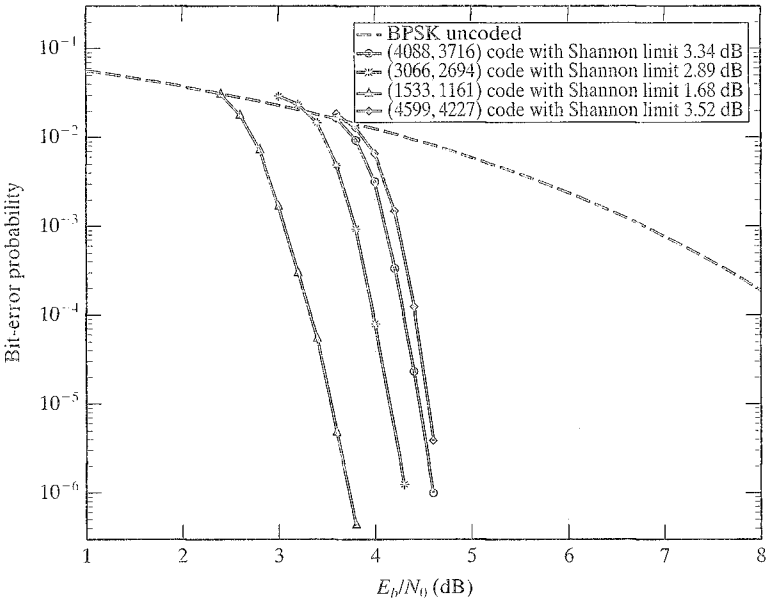


FIGURE 17.28: Bit-error probabilities of the (4088, 3716), (3066, 2694), and (1533, 1161) shortened EG-LDPC codes and the type-II three-dimensional EG-LDPC code with SPA decoding.

17.10 CONSTRUCTION OF GALLAGER LDPC CODES

To construct the parity-check matrix $\mathbb{H}_{GA}$ of a Gallager LDPC code as described in Section 17.1, we first need to construct a $k \times k\rho$ submatrix $\mathbb{H}_1$ with column weight 1 and row weight ρ, and then we need to find $\gamma - 1$ permutations to permute the columns of $\mathbb{H}_1$ to form $\gamma - 1$ other $k \times k\rho$ submatrices $\mathbb{H}_2, \mathbb{H}_3, \cdots, \mathbb{H}_\gamma$. With these γ submatrices we form the parity-check matrix $\mathbb{H}_{GA}$ as follows:

$$\mathbb{H}_{GA} = \begin{bmatrix} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \vdots \\ \mathbb{H}_\gamma \end{bmatrix}.$$

The column and row weights of $\mathbb{H}_{GA}$ are γ and ρ, respectively. The density of $\mathbb{H}_{GA}$ is $1/k$. For large k, $\mathbb{H}_{GA}$ is a sparse matrix. The null space of $\mathbb{H}_{GA}$ gives a Gallager LDPC code. The $\gamma - 1$ permutations must be chosen in such a way that the code generated by the parity-check matrix $\mathbb{H}_{GA}$ has a good minimum distance and does not contain cycles of short lengths in its Tanner graph, especially cycles of length 4. Gallager did not provide any specific way of finding these permutations, and there is no known method for constructing these permutations to guarantee that no short cycles exist in the Tanner graph of the code to be constructed. Commonly, computer searches are used to find good permutations and good Gallager codes.

In this section we present a method for constructing Gallager codes. This method is based on the parallel structure of lines in a Euclidean geometry [51, 52]. Consider the m-dimensional Euclidean geometry $\mathrm{EG}(m, 2^s)$ over $GF(2^s)$. As described in Sections 8.7 and 17.4, the lines in $\mathrm{EG}(m, 2^s)$ can be grouped into $(2^{ms} - 1)/(2^s - 1)$ parallel bundles, with each parallel bundle consisting of $2^{(m-1)s}$ lines that are parallel to each other. The parallel lines in each parallel bundle contain all the 2^{ms} points of the geometry $\mathrm{EG}(m, 2^s)$, and each point is on one and only one of these parallel lines. Let $P_1, P_2, \cdots$, denote the $(2^{ms} - 1)/(2^s - 1)$ parallel bundles. For $1 \leq i \leq (2^{ms} - 1)/(2^s - 1)$, we form a $2^{(m-1)s} \times 2^{ms}$ matrix $\mathbb{H}_i$ whose columns correspond to the 2^{ms} points in $\mathrm{EG}(m, 2^s)$ and whose rows are the incidence vectors of the lines in the parallel bundle P_i. This matrix is called the *incidence matrix* of the parallel bundle P_i. The weight of each row of $\mathbb{H}_i$ is 2^s. Because the lines in each parallel bundle are disjoint, the weight of each column of $\mathbb{H}_i$ is 1. The number of columns in $\mathbb{H}_i$ is $p = 2^s$ times the number of rows in $\mathbb{H}_i$. Therefore, the matrix $\mathbb{H}_i$ is in Gallager's form. For $1 \leq \gamma \leq (2^{ms} - 1)/(2^s - 1)$, we form the following matrix:

$$\mathbb{H}_{EG,GA} = \begin{bmatrix} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \vdots \\ \mathbb{H}_\gamma \end{bmatrix}. \tag{17.67}$$

Then, $\mathbb{H}_{EG,GA}$ is a low-density parity-check matrix in Gallager's form as presented in Section 17.1. All the columns of $\mathbb{H}_{EG,GA}$ have the same weight γ, and all the rows of $\mathbb{H}_{EG,GA}$ have the same weight 2^s. The submatrices $\mathbb{H}_2, \mathbb{H}_3, \cdots, \mathbb{H}_\gamma$ are actually column permutations of $\mathbb{H}_1$. The null space of $\mathbb{H}_{EG,GA}$ gives a regular Gallager LDPC code. Because the rows of $\mathbb{H}_{EG,GA}$ are incidence vectors of $\gamma 2^{(m-1)s}$

lines in EG($m, 2^s$), and since two lines are either disjoint or they intersect at one and only one point, no two rows (or two columns) in $\mathbb{H}_{EG,GA}$ have more than one 1-component in common. Consequently, the Gallager code generated by the parity-check matrix $\mathbb{H}_{EG,GA}$ contains no cycles of length 4 in its Tanner graph. Because there are γ rows in $\mathbb{H}_{EG,GA}$ orthogonal on each code bit, no γ columns can be added to zero. Therefore, the minimum distance of the code is at least $\gamma + 1$. In fact, the minimum distance of an EG-Gallager LDPC code must be even (see Problem 17.24). As a result, we have the following lower bounds on the minimum distance of an EG-Gallager LDPC code:

$$d_{min} \geq \begin{cases} \gamma + 1, & \text{for odd } \gamma, \\ \gamma + 2, & \text{for even } \gamma. \end{cases} \tag{17.68}$$

The preceding lower bounds may be very loose for some γ's, as we will see in Example 17.22.

For each choice of two positive integers, m and s, the foregoing construction based on the parallel bundles of lines in EG($m, 2^s$) gives a sequence of Gallager LDPC codes of length 2^{ms} for various γ's. This sequence of codes has various rates and minimum distances. Small γ's result in high rate codes, and large γ's result in codes with lower rates but larger minimum distances. For $\gamma = (2^{ms} - 1)/(2^s - 1)$, the Gallager code generated by the matrix of (17.67) is the m-dimensional type-I $(0, s)$th-order EG-LDPC code of length 2^{ms}. For $\gamma = 1$ and 2, the codes generated have minimum distances of 2 and 4, respectively. In code construction, γ is chosen greater than 1 to ensure that the resultant code will have a reasonably large minimum distance.

EXAMPLE 17.22

The two-dimensional Euclidean geometry EG($2, 2^3$) over $GF(2^3)$ consists of 72 lines that can be grouped into nine parallel bundles. Each parallel bundle consists of eight parallel lines, and each line consists of eight points. For $\gamma = 2$ to 9, we can construct eight EG-Gallager codes of length 64 with various rates and minimum distances. These codes are given in Table 17.4. The true minimum distances of these codes are determined by computing their weight distributions. Consider the code constructed by using four parallel bundles. This code is a (64, 41) code with a minimum distance of 8; however, the lower bound on the minimum distance given by (17.68) is 6.

EXAMPLE 17.23

Let $m = s = 3$. Consider the three-dimensional Euclidean geometry EG($3, 2^3$) over $GF(2^3)$. This geometry consists of 512 points and 4672 lines. The 4672 lines can be grouped into 73 parallel bundles, with each parallel bundle consisting of 64 lines parallel to each other. For each parallel bundle, we can construct a 64×512 matrix whose rows are the incidence vectors of the lines in the parallel bundle. Suppose we take 6 parallel bundles to construct a Gallager code. Then, the parity-check matrix $H_{EG,GA}$ of the code is a 384×512 matrix that consists of six 64×512 submatrices arranged in a column, with each submatrix constructed based on a parallel bundle of lines. The column and row weights of $H_{EG,GA}$ are 6 and 8, respectively. The

TABLE 17.4: EG-Gallager codes of length 64 constructed based on EG(2, 2^3)

γ	Code	Lower bound on minimum distance	True minimum distance	Number of minimum-weight codewords
2	(64, 49)	4	4	18816
3	(64, 45)	4	4	112
4	(64, 41)	6	8	5880
5	(64, 40)	6	8	3416
6	(64, 39)	8	8	1680
7	(64, 38)	8	8	560
8	(64, 37)	10	10	12544
9	(64, 37)	10	10	12544

null space of this matrix gives an EG-Gallager code of length 512. There are many possible choices of 6 parallel bundles; each choice results in an EG-Gallager LDPC code of length 512. Different choices may result in different code dimensions. One choice gives a (512, 256) EG-Gallager code with rate 0.5. Because $\gamma = 6$, the lower bound, $\gamma + 2$, gives a minimum distance of the code of at least 8; however, the true minimum distance of the code is proved to be 12. The bit-error performance of this code with SPA decoding is shown in Figure 17.29 (assuming BPSK signaling). Also included in the figure are the bit-error performances of three computer-generated LDPC codes of the same rate and the same (or almost the same) length. All three codes are constructed based on the rules given in [10] (see Section 17.14). The first two computer-generated LDPC codes have length 512. The parity-check matrix of the first computer-generated code has column and row weights of 3 and 6, respectively. The parity-check matrix of the second computer-generated code has column and row weights of 4 and 8, respectively. The third computer-generated code is a (504, 252) code whose parity-check matrix has column and row weights of 3 and 6, respectively. This (504, 252) LDPC code is the best-known computer-generated LDPC code of length 504. From Figure 17.29 we see that the (512, 256) EG-Gallager LDPC code outperforms all three computer-generated LDPC codes. Also included in the figure is the bit-error performance of the well-known NASA standard rate-1/2 64-state convolutional code with Viterbi decoding (listed in Table 12.1). This code has been widely used in commercial communication systems for error control. We see that all the LDPC codes outperform the convolutional code.

EXAMPLE 17.24

Let $m = 2$ and $s = 6$. The lines in the two-dimensional Euclidean geometry EG(2,2^6) can be grouped into 65 parallel bundles, each consisting of 64 lines parallel to each other. Each line in EG(2, 2^6) consists of 64 points. Many Gallager LDPC codes of length 4096 can be constructed based on various choices of parallel bundles. Figure 17.30 shows the bit-error performances of several Gallager LDPC codes constructed based on the choices of $\gamma = 6$, 8, 10, 32, and 65 parallel bundles. The codes are (4096, 3771), (4096, 3687), (4096, 3643), (4096, 3399) and (4096, 3367)

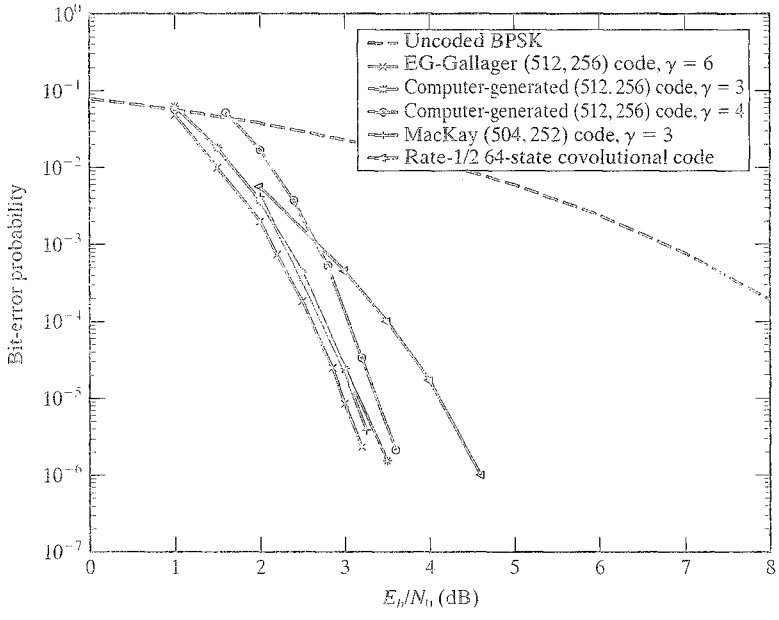


FIGURE 17.29: Bit-error probabilities of the (512, 256) EG-Gallager code, several computer-generated LDPC codes, and a rate-1/2 64-state convolutional code.

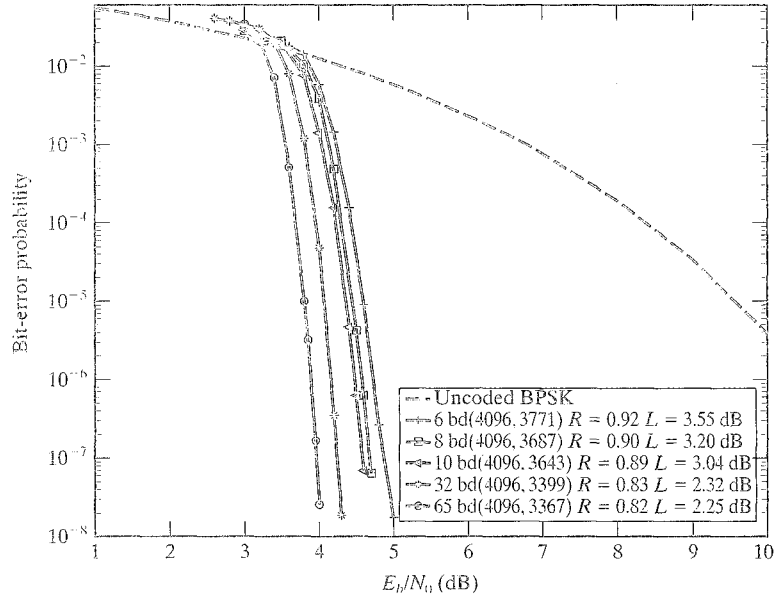


FIGURE 17.30: Bit-error probabilities of several EG-Gallager codes of length 4096, where L stands for Shannon limit, and bd stands for parallel bundle.

codes, respectively. Consider the (4096, 3687) code constructed based on 8 parallel bundles of lines in EG(2, 2^6). This code has rate 0.9 and a minimum distance of at least 10. With SPA decoding, it performs 1.2 dB from the Shannon limit at BER = 10^{-5}.

The lines in a projective geometry do not have the simple parallel structure of the lines in a Euclidean geometry; hence, LDPC codes in Gallager form cannot be constructed.

17.11 MASKED EG-GALLAGER LDPC CODES

The parity-check matrix $\mathbb{H}_{EG,GA}$ of an EG-Gallager code given by (17.67) can be masked to form a new parity-check matrix for a new LDPC code [53]. For masking, we need to decompose the incidence matrix of a parallel bundle of lines in a Euclidean geometry into a row of permutation matrices. Again, consider the m-dimensional Euclidean geometry EG(m, 2^s). For $0 \leq \mu \leq m$, a μ-flat in EG(m, 2^s) is simply a μ-dimensional subspace of the vector space $\mathbb{V}$ of all the m-tuples over $GF(2^s)$ or its coset (Section 8.5). Therefore, a μ-flat consists of $2^{\mu s}$ points. Two μ-flats are said to be parallel if they correspond to two cosets of a μ-dimensional subspace of $\mathbb{V}$. The $2^{(m-\mu)s}$ μ-flats corresponding to the $2^{(m-\mu)s}$ cosets of a μ-dimensional subspace of $\mathbb{V}$ are said to form a *parallel bundle* of μ-flats. The μ-flats in a parallel bundle contain all the points of EG(m, 2^s), and each point is on one and only one of the μ-flats in the parallel bundle.

Let $P(m, m-1)$ be a parallel bundle of $(m-1)$-flats in EG(m, 2^s). This bundle consists of 2^s parallel $(m-1)$-flats, denoted by $F_1, F_2, \cdots, F_{2^s}$. Each $(m-1)$-flat F_i in $P(m, m-1)$ consists of $2^{(m-1)s}$ points and

$$2^{(m-2)s}(2^{(m-1)s} - 1)/(2^s - 1)$$

lines. The total number of lines in $P(m, m-1)$ is

$$J_1 = 2^{(m-1)s}(2^{(m-1)s} - 1)/(2^s - 1).$$

The lines in $P(m, m-1)$ form $(2^{(m-1)s} - 1)/(2^s - 1)$ parallel bundles in EG(m, 2^s). Because there are

$$J = 2^{(m-1)s}(2^{ms} - 1)/(2^s - 1)$$

lines in EG(m, 2^s), there are

$$J - J_1 = 2^{2(m-1)s}$$

lines in EG(m, 2^s) that are not contained in $P(m, m-1)$. Let W denote the set of lines not contained in $P(m, m-1)$. A line in W contains one and only one point from each of the 2^s parallel $(m-1)$-flats in $P(m, m-1)$, because (1) $P(m, m-1)$ contains all the points of EG(m, 2^s), and (2) if a line has two points on an $(m-1)$-flat F_i in $P(m, m-1)$, the line is completely contained in F_i. The lines in W connect the parallel $(m-1)$-flats in $P(m, m-1)$ together, hence, we call them *connecting lines* with respect to the parallel bundle $P(m, m-1)$, and they form $2^{(m-1)s}$ parallel bundles of lines in EG(m, 2^s), called *connecting parallel bundles* of lines, $Q_1, Q_2, \cdots, Q_{2^{(m-1)s}}$.

For $1 \leq i \leq 2^{(m-1)s}$, let $\mathbb{A}_i$ be the $2^{(m-1)s} \times 2^{ms}$ incidence matrix of the connecting parallel bundle Q_i. For $1 \leq j \leq 2^s$, it follows from the structure of

lines in $\mathcal{W}$ that the $2^{(m-1)s}$ columns of $\mathbb{A}_i$ that correspond to the $2^{(m-1)s}$ points in the jth $(m-1)$-flat F_j in $P(m, m-1)$ form a $2^{(m-1)s} \times 2^{(m-1)s}$ permutation matrix (a column-permutated identity matrix). Therefore, $\mathbb{A}_i$ consists of 2^s permutation matrices of dimension $2^{(m-1)s}$, denoted by $\mathbb{A}_{i,1}, \mathbb{A}_{i,2}, \cdots, \mathbb{A}_{i,2^s}$. For $1 \leq \gamma \leq 2^{(m-1)s}$, we form the following $\gamma 2^{(m-1)s} \times 2^{ms}$ matrix:

$$\mathbb{H}_{EG,GA} = \begin{bmatrix} \mathbb{A}_1 \\ \mathbb{A}_2 \\ \vdots \\ \mathbb{A}_\gamma \end{bmatrix}.$$

Suppose we order the columns of $\mathbb{H}_{EG,GA}$ in such way that the first $2^{(m-1)s}$ columns correspond to the points of the first $(m-1)$-flat F_1 in $P(m, m-1)$, the second $2^{(m-1)s}$ columns correspond to the points in the second $(m-1)$-flat F_2 in $P(m, m-1)$, $\cdots$, and the last $2^{(m-1)s}$ columns correspond to the points in the last $(m-1)$-flat F_{2^s} in $P(m, m-1)$. Then, we can put $\mathbb{H}_{EG,GA}$ into following $\gamma \times 2^s$ array of $2^{(m-1)s} \times 2^{(m-1)s}$ permutation matrices:

$$\mathbb{H}_{EG,GA} = \begin{bmatrix} \mathbb{A}_{1,1} & \mathbb{A}_{1,2} & \cdots & \mathbb{A}_{1,2^s} \\ \mathbb{A}_{2,1} & \mathbb{A}_{2,2} & \cdots & \mathbb{A}_{2,2^s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{A}_{\gamma,1} & \mathbb{A}_{\gamma,2} & \cdots & \mathbb{A}_{\gamma,2^s} \end{bmatrix}. \tag{17.69}$$

which has column and row weights of γ and 2^s, respectively. For $1 \leq \gamma \leq 2^s$, the null space of $\mathbb{H}_{EG,GA}$ gives an EG-Gallager LDPC code with a minimum distance of at least $\gamma + 1$ (or $\gamma + 2$) and rate of at least $(2^s - \gamma)/2^s$.

EXAMPLE 17.25

Consider the two-dimensional Euclidean geometry EG(2,2^2) over the field $GF(2^2)$. This geometry consists of 16 points and 20 lines. Each line consists of 4 points, and each parallel bundle of lines consists of 4 parallel lines. The 20 lines of EG(2, 2^2) form five parallel bundles. In Example 8.15 we showed that the Galois field $GF(2^4)$ as an extension field of $GF(2^2)$ is a realization of EG(2, 2^2). Let α be a primitive element of $GF(2^4)$. Then, $\alpha^\infty = 0$, $\alpha^0 = 1, \alpha, \alpha^2, \cdots, \alpha^{14}$ form the 16 points of EG(2, 2^2). The following 4 lines:

$$L_1 = \{0, \alpha^4, \alpha^9, \alpha^{14}\}, \qquad L_2 = \{\alpha^0, \alpha, \alpha^3, \alpha^7\},$$
$$L_3 = \{\alpha^2, \alpha^{10}, \alpha^{11}, \alpha^{13}\}, \qquad L_4 = \{\alpha^5, \alpha^6, \alpha^8, \alpha^{12}\}$$

form a parallel bundle $P(2, 1)$. The other 16 lines in EG(2, 2^2) are connecting lines with respect to the parallel bundle $P(2, 1)$. These 16 connecting lines with respect to $P(2, 1)$ form four connecting parallel bundles, Q_1, Q_2, Q_3, and Q_4. These four connecting parallel bundles can be constructed using Table 8.5. For $1 \leq i \leq 4$, the incidence matrix $\mathbb{A}_i$ of Q_i can be decomposed into four 4×4 permutation matrices

as follows:

	L_1				L_2				L_3				L_4			
	0	α^4	α^9	α^{14}	α^0	α	α^3	α^7	α^2	α^{10}	α^{11}	α^{13}	α^5	α^6	α^8	α^{12}
	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
$\mathbb{A}_1$	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0
	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1
	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0
	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
$\mathbb{A}_2$	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1
	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0
	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0
	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
$\mathbb{A}_3$	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0
	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0
	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0
	1	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0
$\mathbb{A}_4$	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0
	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1
	0	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0

Let $\mathbb{Z} = [z_{i,j}]$ be a $\gamma \times 2^s$ matrix over $GF(2)$. We define the following product of $\mathbb{Z}$ and $\mathbb{H}_{EG,GA}$:

$$\mathbb{M} = \mathbb{Z} \otimes \mathbb{H} = \begin{bmatrix} z_{1,1}\mathbb{A}_{1,1} & z_{1,2}\mathbb{A}_{1,2} & \cdots & z_{1,2^s}\mathbb{A}_{1,2^s} \\ z_{2,1}\mathbb{A}_{2,1} & z_{2,2}\mathbb{A}_{2,2} & \cdots & z_{2,2^s}\mathbb{A}_{2,2^s} \\ \vdots & \vdots & \ddots & \vdots \\ z_{\gamma,1}\mathbb{A}_{\gamma,1} & z_{\gamma,2}\mathbb{A}_{\gamma,2} & \cdots & z_{\gamma,2^s}\mathbb{A}_{\gamma,2^s} \end{bmatrix}, \tag{17.70}$$

where

$$z_{i,j}\mathbb{A}_{i,j} = \begin{cases} \mathbb{A}_{i,j} & \text{for } z_{i,j} = 1, \\ \mathbb{O} & \text{for } z_{i,j} = 0, \end{cases} \tag{17.71}$$

and $\mathbb{O}$ is a $2^{(m-1)s} \times 2^{(m-1)s}$ zero matrix. This product defines a *masking operation* for which a set of permutation matrices in $\mathbb{H}_{EG,GA}$ given by (17.69) is masked by the zero-entries of $\mathbb{Z}$. The distribution of the permutation matrices in $\mathbb{M}$ is the same as the distribution of 1-entries in $\mathbb{Z}$. If the column and row weights of $\mathbb{Z}$ are α and β, then $\mathbb{M}$ is an (α, β)-regular matrix with column and row weights of α and β, respectively. If $\mathbb{Z}$ has different column weights or different row weights, then $\mathbb{M}$ is an irregular matrix. $\mathbb{Z}$ is called the *masking matrix*, $\mathbb{H}_{EG,GA}$ the *base matrix*, and $\mathbb{M}$ the *masked matrix*. Masking reduces the density of the base matrix and hence reduces the number of cycles in the Tanner graph of the base matrix. Consequently, the associated Tanner graph of the masked matrix has a smaller number of cycles than

the Tanner graph of the base matrix. Because the Tanner graph of the base matrix $\mathbb{H}_{EG,GA}$ has no cycles of length 4, the masked matrix $\mathbb{M}$ has no cycles of length 4. The null space of $\mathbb{M}$ gives a new regular LDPC code, called a *masked EG-Gallager* LDPC code, whose Tanner graph does not contain cycles of length 4 [53].

Masking results in a class of masked EG-Gallager LDPC codes with various rates and minimum distances. The rate of a masked EG-LDPC code is controlled by the parameters γ and 2^s, but its error performance is controlled by the pattern of the 1-entries in the masking matrix. The masking matrix $\mathbb{Z}$ can be constructed by computer or by hand if its size is small. If $2^s = k\gamma$ and $\beta = k\alpha$, it can be constructed algebraically. A γ-tuple $\mathbf{v}$ of weight α over $GF(2)$ is said to be *primitive* if γ cyclic shifts of $\mathbf{v}$ result in γ different γ-tuples. For example, the 8-tuple $\mathbf{v} = (11010000)$ is primitive. Cyclically shifting $\mathbf{v}$ eight times gives the following eight different 8-tuples:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Given a primitive γ-tuple $\mathbf{v}$ with weight α, we can form a $\gamma \times \gamma$ circulant matrix G with $\mathbf{v}$ and its $\gamma - 1$ cyclic shifts as the rows (or columns). Both the column and row weights of this circulant are α. Two primitive γ-tuples are said to be *distinct* if one cannot be obtained by cyclically shifting the other. To construct a $\gamma \times 2^s$ masking matrix $\mathbb{Z}$, we take k distinct primitive γ-tuples with weight α, $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k$ and form k circulants, $\mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_k$. Then,

$$\mathbb{Z} = [\mathbb{G}_1 \mathbb{G}_2 \cdots \mathbb{G}_k]$$

gives a $\gamma \times 2^s$ masking matrix with column and row weights of α and $\beta = k\alpha$, respectively. If possible, the k distinct primitive γ-tuples should be chosen such that the Tanner graph of $\mathbb{Z}$ is free of short cycles.

EXAMPLE 17.26

Consider the two-dimensional Euclidean geometry $\mathrm{EG}(2, 2^6)$ given in Example 17.24. Let $P(2, 1)$ be a parallel bundle of lines. With respect to this parallel bundle, there are 64 connecting parallel bundles of lines, $Q_1, Q_2, \cdots, Q_{64}$, each consisting of 64 parallel lines, and each line consisting of 64 points. The incidence matrix $\mathbb{A}_i$ of a connecting parallel bundle Q_i is a 64×4096 matrix that can be decomposed into sixty-four 64×64 permutation matrices,

$$\mathbb{A}_i = [\mathbb{A}_{i,1} \mathbb{A}_{i,2} \cdots \mathbb{A}_{i,64}].$$

Let $\gamma = 8$. We form the following 512×4096 base matrix:

$$\mathbb{H}_{EG,GA,1} = \begin{bmatrix} \mathbb{A}_1 \\ \mathbb{A}_2 \\ \vdots \\ \mathbb{A}_8 \end{bmatrix} = \begin{bmatrix} \mathbb{A}_{1,1} & \mathbb{A}_{1,2} & \cdots & \mathbb{A}_{1,64} \\ \mathbb{A}_{2,1} & \mathbb{A}_{2,2} & \cdots & \mathbb{A}_{2,64} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{A}_{8,1} & \mathbb{A}_{8,2} & \cdots & \mathbb{A}_{8,64} \end{bmatrix},$$

which is an 8×64 array of 64×64 permutation matrices. The column and row weights of $\mathbb{H}_{EG,GA,1}$ are 8 and 64, respectively. Suppose we want to mask $\mathbb{H}_{EG,GA,1}$ with an 8×64 masking matrix $\mathbb{Z}_1$ with column and row weights of 4 and 32, respectively. Among the seventy 8-tuples of weight 4, all but 6 are primitive. The 64 primitive 8-tuples can be generated by 8 distinct primitive 8-tuples, $\mathbb{v}_1, \mathbb{v}_2, \cdots, \mathbb{v}_8$, and cyclically shifting them. With these 8 distinct primitive 8-tuples, we form eight 8×8 circulants, $\mathbb{G}_1^{(1)}, \mathbb{G}_2^{(1)}, \cdots, \mathbb{G}_8^{(1)}$, with column and row weights of 4. Then,

$$\mathbb{Z}_1 = [\mathbb{G}_1^{(1)} \mathbb{G}_2^{(1)} \cdots \mathbb{G}_8^{(1)}]$$

gives an 8×64 masking matrix with column and row weights of 4 and 32. Masking $\mathbb{H}_{EG,GA,1}$ with $\mathbb{Z}_1$, we obtain a 512×4096 masked matrix

$$\mathbb{M}_1 = \mathbb{Z}_1 \otimes \mathbb{H}_{EG,GA,1}$$

with column and row weights of 4 and 32, respectively. The null space of $\mathbb{M}_1$ gives a (4096, 3585) LDPC code with rate 0.875 and a minimum distance of at least 6. The error performance of this code with SPA decoding is shown in Figure 17.31. At a BER of 10^{-6}, the code performs 1.2 dB from the Shannon limit.

To construct a lower-rate code, we need to choose a larger γ. Suppose we choose $\gamma = 16$. For the same geometry EG(2, 2^6), the base matrix $\mathbb{H}_{EG,GA,2}$ given by (17.69) is a 16×64 array of 64×64 permutation matrices. For masking, we need to construct a 16×64 masking matrix $\mathbb{Z}_2$. Suppose we choose the column and row

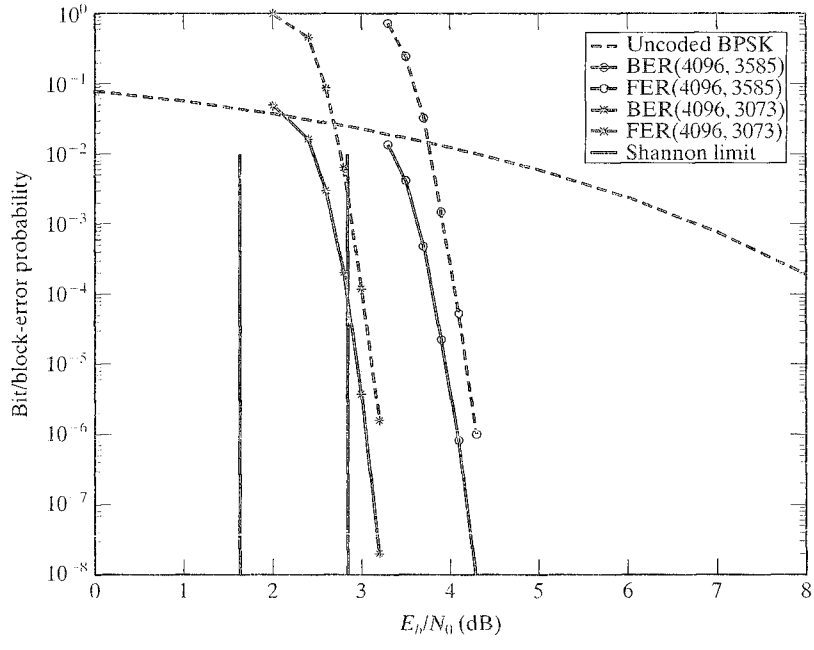


FIGURE 17.31: The error performances of the two masked Gallager-LDPC codes given in Example 17.26.

weights of $\mathbb{Z}_2$ to be 4 and 16, respectively. To construct $\mathbb{Z}_2$, we take four distinct primitive 16-tuples over $GF(2)$ with weight 4 and then form four 16×16 circulants, $\mathbb{G}_1^{(2)}, \mathbb{G}_2^{(2)}, \mathbb{G}_3^{(2)}, \mathbb{G}_4^{(2)}$, by cyclically shifting these four distinct primitive 16-tuples. Then,

$$\mathbb{Z}_2 = [\mathbb{G}_1^{(2)} \mathbb{G}_2^{(2)} \mathbb{G}_3^{(2)} \mathbb{G}_4^{(2)}]$$

gives a 16×64 masking matrix with column and row weights of 4 and 16, respectively. Masking the base matrix $\mathbb{H}_{EG.GA.2}$ with $\mathbb{Z}_2$, we obtain a 1024×4096 masked matrix $\mathbb{M}_2 = \mathbb{Z}_2 \otimes \mathbb{H}_{EG.GA.2}$ with column and row weights of 4 and 16, respectively. The null space of $\mathbb{M}_2$ gives a (4096, 3073) LDPC code with rate 0.75 and a minimum distance of at least 6. The error performance of this code with SPA decoding is also shown in Figure 17.31. At a BER of 10^{-6}, it performs 1.35 dB from the Shannon limit.

Let ρ be a positive integer such that $\gamma \leq \rho \leq 2^s$. If we take the first ρ columns of permutation matrices of the matrix $\mathbb{H}_{EG.GA}$ given by (17.69), we obtain the following $\gamma 2^{(m-1)} \times \rho 2^{(m-1)s}$ matrix:

$$\mathbb{H}_{EG.GA}(\rho) = \begin{bmatrix} \mathbb{A}_{1,1} & \mathbb{A}_{1,2} & \cdots & \mathbb{A}_{1,\rho} \\ \mathbb{A}_{2,1} & \mathbb{A}_{2,2} & \cdots & \mathbb{A}_{2,\rho} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{A}_{\gamma,1} & \mathbb{A}_{\gamma,2} & \cdots & \mathbb{A}_{\gamma,\rho} \end{bmatrix}. \tag{17.72}$$

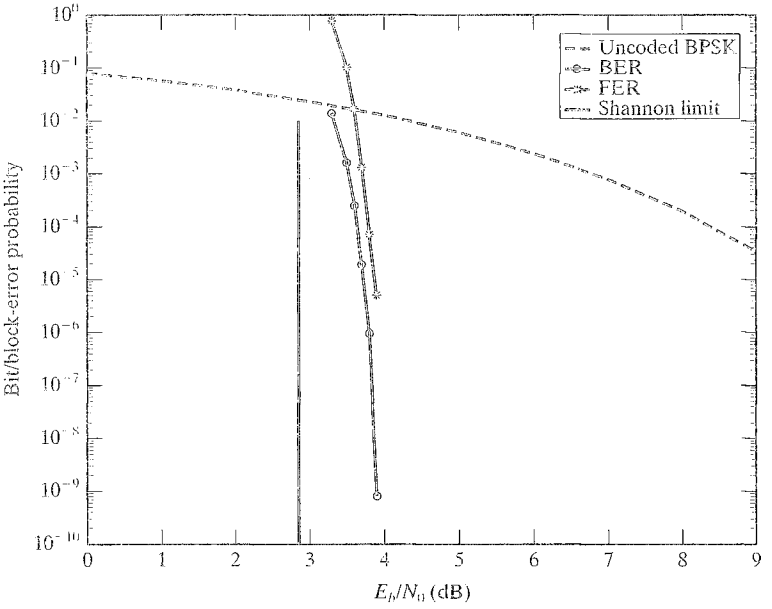


FIGURE 17.32: The error performance of the (10240, 8961) masked Gallager-LDPC code constructed based on EG(2, 2^7).

For masking construction, we can use $\mathbb{H}_{EG,GA}(\rho)$ as the base matrix. This shortening gives more flexibility for constructing codes with various lengths and rates. For example, suppose we choose EG$(2, 2^7)$ for code construction using masking. If we choose $\gamma = 10$ and $\rho = 80$, the base matrix $\mathbb{H}_{EG,GA}(80)$ is then a 10×80 array of 128×128 permutation matrices. Masking this matrix with a 10×80 matrix $\mathbb{Z}$ with column and row weights of 4 and 32, respectively, we obtain a 1280×10240 masked matrix $\mathbb{M}$. The masking matrix $\mathbb{Z}$ consists of eight circulants that are constructed by using eight distinct primitive 10-tuples over $GF(2)$ and cyclically shifting each of them 10 times. The null space of $\mathbb{M}$ gives a (10240, 8961) LDPC code with rate 0.875 whose error performance is shown in Figure 17.32. At a BER of 10^{-6}, it performs only 0.9 dB from the Shannon limit.

Masking is a very powerful techniques for constructing both regular and irregular LDPC codes.

17.12 CONSTRUCTION OF QUASI-CYCLIC CODES BY CIRCULANT DECOMPOSITION

Consider a $q \times q$ circulant $\mathbb{G}$ over $GF(2)$ with column and row weights δ. Because column and row weights of a circulant are the same, for simplicity, we say that $\mathbb{G}$ has weight δ. For $1 \leq t \leq \delta$, let $w_1, w_2, \cdots, w_t$ be a set of positive integers such that $1 \leq w_1, w_2, \cdots, w_t \leq \delta$, and $w_1 + w_2 + \cdots + w_t = \delta$. Then, we can decompose $\mathbb{G}$ into t $q \times q$ circulants with weights $w_1, w_2, \cdots, w_t$, respectively. Let $\mathbb{g}_1$ be the first column of $\mathbb{G}$. We split $\mathbb{g}_1$ into t columns of the same length q, denoted by $\mathbb{g}_1^{(1)}$, $\mathbb{g}_1^{(2)}, \cdots, \mathbb{g}_1^{(t)}$, such that the first w_1 1-components of $\mathbb{g}_1$ are put in $\mathbb{g}_1^{(1)}$, the next w_2 1-components of $\mathbb{g}_1$ are put in $\mathbb{g}_1^{(2)}, \cdots$, and the last w_t 1-components of $\mathbb{g}_1$ are put in $\mathbb{g}_1^{(t)}$. For each new column $\mathbb{g}_1^{(i)}$, we form a $q \times q$ circulant $\mathbb{G}_i$ by cyclically shifting $\mathbb{g}_1^{(i)}$ downward q times. This results in t $q \times q$ circulants, $\mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_t$, with weights $w_1, w_2, \cdots, w_t$, respectively. These circulants are called the *descendants* of $\mathbb{G}$. Such a decomposition of $\mathbb{G}$ is called *column decomposition* of $\mathbb{G}$. Column decomposition of $\mathbb{G}$ results in a $q \times tq$ matrix,

$$\mathbb{H} = [\mathbb{G}_1 \mathbb{G}_2 \cdots \mathbb{G}_t], \tag{17.73}$$

which is a row of t $q \times q$ circulants. If $w_1 = w_2 = \cdots = w_t = w$, then $\mathbb{H}$ is a regular matrix with constant column weight w and constant row weight tw. If $t = \delta$ and $w_1 = w_2 = \cdots = w(\delta) = 1$, then each descendant circulant $\mathbb{G}_i$ of $\mathbb{G}$ is a permutation matrix, and $\mathbb{H}$ is a row of δ permutation matrices. The parameter t is called the *column splitting factor*. Figure 17.33 shows a column decomposition of an 8×8 circulant of weight 3 into two descendants with weights 2 and 1, respectively.

Similarly, we can decompose G into descendants by splitting its first row into multiple rows and cyclically shifting each new row to the right q times. Let $1 \leq c \leq max\{w_i : 1 \leq i \leq t\}$. For $1 \leq i \leq t$, let $w_{i,1}, w_{i,2}, \cdots, w_{i,c}$ be a set of nonnegative integers such that $0 \leq w_{i,1}, w_{i,2}, \cdots, w_{i,c} \leq w_i$, and $w_{i,1} + w_{i,2} + \cdots + w_{i,c} = w_i$. Each descendant $\mathbb{G}_i$ in $\mathbb{H}$ of (17.73) can be decomposed into c descendant circulants with weights $w_{i,1}, w_{i,2}, \cdots, w_{i,c}$, respectively. This is done by splitting the first row $\mathbb{g}_{i,1}$ of $\mathbb{G}_i$ into c rows of the same length q, denoted by $\mathbb{g}_{i,1}^{(1)}, \mathbb{g}_{i,1}^{(2)}, \cdots, \mathbb{g}_{i,1}^{(c)}$, where $\mathbb{g}_{i,1}^{(1)}$ contains the first $w_{i,1}$ 1-components of $\mathbb{g}_{i,1}$, $\mathbb{g}_{i,1}^{(2)}$ contains the second $w_{i,2}$ 1-components of $\mathbb{g}_{i,1}, \cdots$, and $\mathbb{g}_{i,1}^{(c)}$ contains the last $w_{i,c}$ 1-components of $\mathbb{g}_{i,1}$. Cyclically shifting each

$$\begin{bmatrix} 1&0&0&0&0&1&1&0 \\ 0&1&0&0&0&0&1&1 \\ 1&0&1&0&0&0&0&1 \\ 1&1&0&1&0&0&0&0 \\ 0&1&1&0&1&0&0&0 \\ 0&0&1&1&0&1&0&0 \\ 0&0&0&1&1&0&1&0 \\ 0&0&0&0&1&1&0&1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1&0&0&0&0&0&1&0 \\ 0&1&0&0&0&0&0&1 \\ 1&0&1&0&0&0&0&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&1&0&1&0&0&0 \\ 0&0&0&1&0&1&0&0 \\ 0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&1&0&1 \end{bmatrix} \begin{bmatrix} 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \end{bmatrix}$$

FIGURE 17.33: A column decomposition of a circulant of weight 3.

new row $\mathbf{g}_{i,1}^{(k)}$ to the right q times, we obtain c circulants $\mathbb{G}_i^{(1)}, \mathbb{G}_i^{(2)}, \cdots, \mathbb{G}_i^{(c)}$ that are descendants of $\mathbb{G}_i$. This decomposition is referred to as *row decomposition*. Row decomposition of $\mathbb{G}_i$ results in the following $cq \times q$ matrix:

$$\mathbb{D}_i = \begin{bmatrix} \mathbb{G}_i^{(1)} \\ \mathbb{G}_i^{(2)} \\ \vdots \\ \mathbb{G}_i^{(c)} \end{bmatrix}, \tag{17.74}$$

which is a column of c $q \times q$ circulants. We call $\mathbb{D}_i$ the row decomposition of $\mathbb{G}_i$ and c the *row splitting factor*. In row splitting, we allow $w_{i,k} = 0$. If $w_{i,k} = 0$, $\mathbb{G}_i^{(k)}$ is a $q \times q$ zero matrix, regarded as a circulant.

If each circulant $\mathbb{G}_i$ in $\mathbb{H}$ of (17.73) is replaced by its row decomposition $\mathbb{D}_i$, we obtain the following $c \times t$ array of circulants:

$$\mathbb{D} = \begin{bmatrix} \mathbb{D}_1 & \mathbb{D}_2 & \cdots & \mathbb{D}_t \end{bmatrix} = \begin{bmatrix} \mathbb{G}_1^{(1)} & \mathbb{G}_2^{(1)} & \cdots & \mathbb{G}_t^{(1)} \\ \mathbb{G}_1^{(2)} & \mathbb{G}_2^{(2)} & \cdots & \mathbb{G}_t^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{G}_1^{(c)} & \mathbb{G}_2^{(c)} & \cdots & \mathbb{G}_t^{(c)} \end{bmatrix}, \tag{17.75}$$

which is a $cq \times tq$ matrix. If each $\mathbb{G}_i^{(k)}$ has weight 1, then $\mathbb{D}$ is an array of permutation matrices. The null space of $\mathbb{D}$ gives a code of length $n = tq$ that can be put in quasi-cyclic form (see Section 5.12).

If $\mathbb{G}$ is a sparse matrix, $\mathbb{D}$ is also a sparse matrix with smaller density than $\mathbb{G}$. If no two rows (or two columns) in $\mathbb{G}$ have more than one 1-component in common, then no two rows (or two columns) in $\mathbb{D}$ have more than one 1-component in common. In this case, the null space of $\mathbb{D}$ gives a quasi-cyclic LDPC code whose Tanner graph is free of cycles of length 4. If all the circulants in $\mathbb{D}$ have the same weight, then the code is a regular quasi-cyclic LDPC code.

As shown in Section 8.5 and Sections 17.4, 17.5, and 17.7, sparse circulants can be constructed from the incidence vectors of lines in either a Euclidean geometry or a projective geometry. Furthermore, no two rows (or two columns) either from the same or from two different circulants have more than one 1-component in common. Consequently, quasi-cyclic LDPC codes can be constructed by decomposing one or a group of these geometry circulants [54, 55].

The incidence vectors of all the lines in EG$(m, 2^s)$ not passing through the origin can be partitioned into

$$K = (2^{(m-1)s} - 1)/(2^s - 1)$$

cyclic classes, $Q_1, Q_2, \cdots, Q_K$. Each cyclic class Q_i consists of $2^{ms} - 1$ incidence vectors and can be obtained by cyclically shifting any vector in the class $2^{ms} - 1$ times. Hence, for each cyclic class Q_i, we can form a $(2^{ms} - 1) \times (2^{ms} - 1)$ circulant $\mathbb{G}_i$ using any vector in Q_i as the first row and then cyclically shifting it $2^{ms} - 2$ times. The weight of $\mathbb{G}_i$ is 2^s. Consequently, we obtain a class of K circulants,

$$\mathcal{G} = \{\mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_K\}. \tag{17.76}$$

In code construction, we can take a subset of k circulants with $1 \leq k \leq K$, say $\mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_k$, from $\mathcal{G}$ and arrange them in a row,

$$\mathbb{H} = [\mathbb{G}_1 \mathbb{G}_2 \cdots \mathbb{G}_k].$$

We choose column and row splitting factors t and c, respectively, and decompose each circulant $\mathbb{G}_i$ in $\mathbb{H}$ into a $c \times t$ array $\mathbb{D}_i$ of descendants. Replacing each $\mathbb{G}_i$ in $\mathbb{H}$ by its array decomposition $\mathbb{D}_i$, we obtain a $c \times kt$ array $\mathbb{D}$ of descendant circulants. $\mathbb{D}$ is a $c(2^{ms} - 1) \times kt(2^{ms} - 1)$ matrix. The null space of $\mathbb{D}$ gives a quasi-cyclic LDPC code C whose Tanner graph does not contain cycles of length 4 and hence the girth of its Tanner graph is at least 6. If $\mathbb{D}$ has constant column and constant row weights, then C is a regular LDPC code.

EXAMPLE 17.27

Consider the three-dimensional Euclidean geometry EG$(3, 2^3)$ over $GF(2^3)$. This geometry consists of 511 nonorigin points and 4599 lines not passing through the origin. The incidence vectors of the lines not passing through the origin form nine cyclic classes. From these cyclic classes we can form a class of nine 511×511 circulants of weight 8,

$$\mathcal{G} = \{\mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_9\}.$$

Suppose we take eight circulants from $\mathcal{G}$ and arrange them in a row to form the matrix

$$\mathbb{H} = [\mathbb{G}_1\ \mathbb{G}_2\ \mathbb{G}_3\ \mathbb{G}_4\ \mathbb{G}_5\ \mathbb{G}_6\ \mathbb{G}_7\ \mathbb{G}_8].$$

We choose column and row splitting factors $t = 2$ and $c = 4$, respectively, and decompose each circulant $\mathbb{G}_i$ into a 4×2 array $\mathbb{D}_i$ of eight 511×511 permutation matrices. Replacing each circulant $\mathbb{G}_i$ in $\mathbb{H}$ by its array decomposition $\mathbb{D}_i$, we obtain a 4×16 array $\mathbb{D}$ of 511×511 permutation matrices. $\mathbb{D}$ is a 2044×8176 matrix with column weight 4 and row weight 16. The null space of $\mathbb{D}$ gives an (8176, 6135) quasi-cyclic LDPC code with rate 0.75 and a minimum distance of at least 6. Its error performance with SPA decoding is depicted in Figure 17.34. It has a beautiful waterfall error performance all the way down to a BER of 10^{-10} without error floor. At a BER of 10^{-10}, it performs 1.4 dB from the Shannon limit.

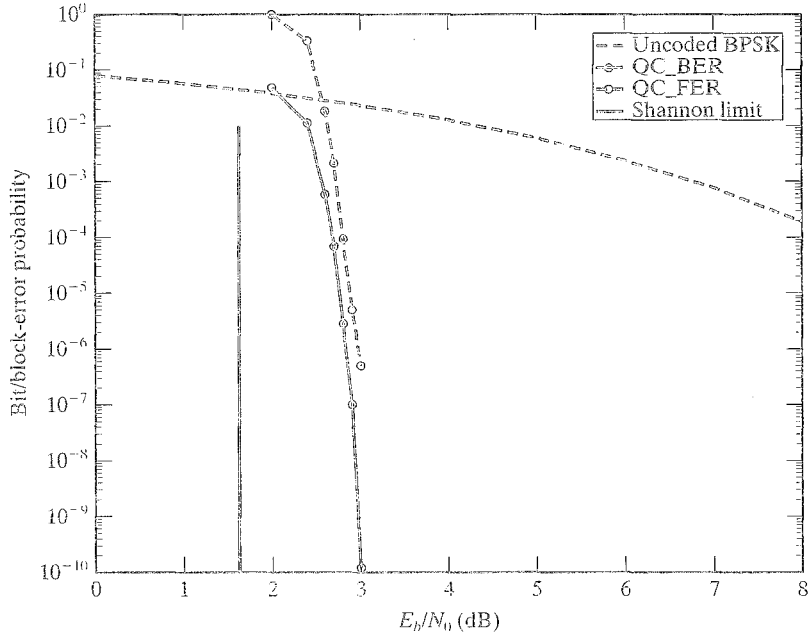


FIGURE 17.34: Error performance of the (8176, 6135) quasi-cyclic code given in Example 17.27.

Quasi-cyclic LDPC codes can also be constructed based on the incidence vectors of the lines in a projective geometry [17, 52, 55, 56]. Consider the m-dimensional projective geometry $\mathrm{PG}(m, 2^s)$. There are two cases to be considered, even and odd m. For even m, the incidence vector of each line in $\mathrm{PG}(m, 2^s)$ is primitive, and the incidence vectors of all the lines can be partitioned into

$$K_1 = (2^{ms} - 1)/(2^{2s} - 1)$$

cyclic classes. Each class consists of $(2^{(m+1)s} - 1)/(2^s - 1)$ incidence vectors that can be obtained by cyclically shifting any vector in the class. For odd $m \geq 3$, there are $(2^{(m+1)s} - 1)/(2^{2s} - 1)$ lines in $\mathrm{PG}(m, 2^s)$ whose incidence vectors are not primitive, and the incidence vectors of all the other lines are primitive. The primitive incidence vectors can be partitioned into

$$K_2 = 2^s(2^{(m-1)s} - 1)/(2^{2s} - 1)$$

cyclic classes. Therefore, quasi-cyclic LDPC codes can be constructed based on the cyclic classes of incidence vectors of lines in a projective geometry.

EXAMPLE 17.28

Consider the three-dimensional projective geometry $\mathrm{PG}(3, 2^3)$ over $GF(2^3)$. This geometry consists of 585 points and 4745 lines. Each line consists of 9 points. There are 65 lines whose incidence vectors are not primitive, and the incidence vectors of the other 4680 lines are primitive. The 4680 primitive incidence vectors can be

partitioned into eight cyclic classes, with each class consisting of 585 vectors. Based on these eight cyclic classes of incidence vectors, we can form eight 585×585 circulants, $\mathbb{G}_1, \mathbb{G}_2, \cdots, \mathbb{G}_8$. Each circulant $\mathbb{G}_i$ has weight 9. Suppose we want to construct a quasi-cyclic LDPC code of length 9360 with rate 0.875. First, we decompose each circulant $\mathbb{G}_i$ by column decomposition into three descendant circulants, $\mathbb{G}_{i,1}$, $\mathbb{G}_{i,2}$, and $\mathbb{G}_{i,3}$, with weights 4, 4, and 1, respectively. Removing the descendant with weight 1, we obtain the 585×1170 matrix

$$\mathbb{D}_i = [\mathbb{G}_{i,1}\mathbb{G}_{i,2}],$$

which consists of two 511×511 descendant circulants of $\mathbb{G}_i$. We decompose each circulant of $\mathbb{D}_i$ by row decomposition into two descendants, each having weight 2. The result is the following 2×2 array of 511×511 circulants:

$$\mathbb{D}_i^* = \begin{bmatrix} \mathbb{G}_{i,1}^{(1)} & \mathbb{G}_{i,2}^{(1)} \\ \mathbb{G}_{i,1}^{(2)} & \mathbb{G}_{i,2}^{(2)} \end{bmatrix},$$

which is a 1170×1170 matrix with both column and row weight 4. We then form the following 1170×9360 matrix:

$$\mathbb{D} = [\mathbb{D}_1^* \, \mathbb{D}_2^* \, \mathbb{D}_3^* \, \mathbb{D}_4^* \, \mathbb{D}_5^* \, \mathbb{D}_6^* \, \mathbb{D}_7^* \, \mathbb{D}_8^*],$$

which has column and row weights 4 and 32, respectively. The null space of $\mathbb{D}$ gives a (9360, 8192) quasi-cyclic LDPC code with rate 0.875. The error performance of this code is shown in Figure 17.35. At a BER of 10^{-6}, it performs 0.95 dB from the Shannon limit.

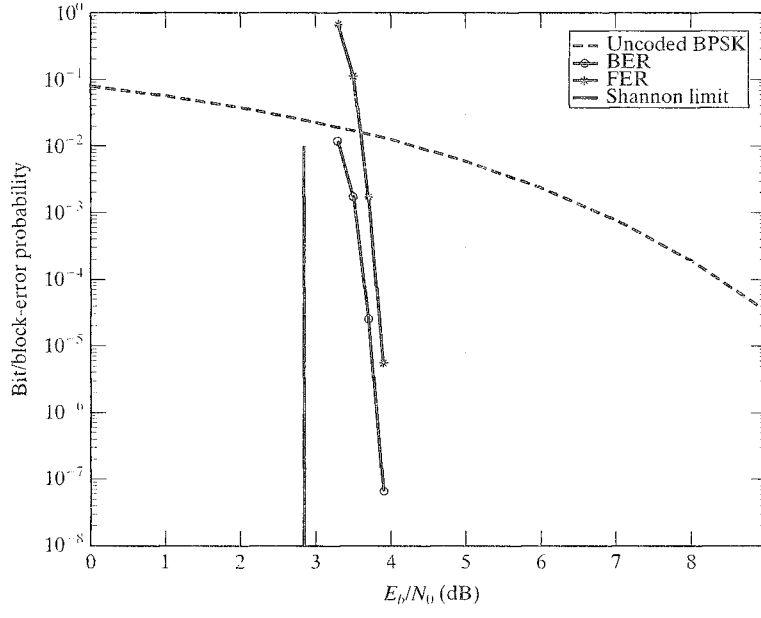


FIGURE 17.35: Error performance of the (9360, 8192) quasi-cyclic code given in Example 17.28.

17.13 CONSTRUCTION OF LDPC CODES BASED ON FINITE GEOMETRIES OVER $GF(p^s)$

So far, we have considered construction of LDPC codes based only on finite geometries over the field $GF(2^s)$; however, LDPC codes also can be constructed based on the lines and points of finite geometries over the field $GF(p^s)$, where p is a prime. The approaches to the code construction are exactly the same as those presented in previous sections. A finite geometry, Euclidean or projective, over $GF(p^s)$ has the same structural properties as a finite geometry over $GF(2^s)$. All the expressions for the number of points, the number of lines, the number of parallel bundles, and so on, derived for finite geometries over $GF(2^s)$ can be applied to finite geometries over $GF(p^s)$ simply by replacing 2 with p.

The m-dimensional Euclidean geometry $\mathrm{EG}(m, p^s)$ over $GF(p^s)$ consists of p^{ms} points, with each point an m-tuple over $GF(p^s)$. A line in $\mathrm{EG}(m, p^s)$ is simply a one-dimensional subspace or its coset of the vector space of all the m-tuples over $GF(p^s)$. Therefore, each line consists of p^s points. There are

$$p^{(m-1)s}(p^{ms} - 1)/(p^s - 1) \tag{17.77}$$

lines. These lines can be grouped into

$$K = (p^{ms} - 1)/(p^s - 1) \tag{17.78}$$

parallel bundles, each consisting of $p^{(m-1)s}$ lines parallel to each other. Each parallel bundle contains all the p^{ms} points of the geometry, and each point is on one and only one line in the parallel bundle. For each point, there are

$$(p^{ms} - 1)/(p^s - 1) \tag{17.79}$$

lines intersecting at it. The incidence vectors of all the lines not passing through the origin can be partitioned into

$$(p^{(m-1)s} - 1)/(p^s - 1) \tag{17.80}$$

cyclic classes, each consisting of $p^{ms} - 1$ vectors.

All the constructions of LDPC codes based on the lines and points of $\mathrm{EG}(m, 2^s)$ can be directly applied to the construction of LDPC codes based on the lines and points of $\mathrm{EG}(m, p^s)$. We use the construction of EG-Gallager LDPC codes for illustration. Let $n = p^{ms}$. The incidence vector of a line $\mathcal{L}$ in $\mathrm{EG}(m, p^s)$ is an n-tuple over $GF(2)$,

$$\mathbf{v}_{\mathcal{L}} = (v_0, v_1, \cdots, v_{n-1}),$$

whose components correspond to the n points of $\mathrm{EG}(m, p^s)$ and $v_j = 1$ if and only if v_j corresponds to a point on $\mathcal{L}$, and $v_j = 0$ otherwise. Therefore, the weight of $\mathbf{v}_{\mathcal{L}}$ is p^s. For each parallel bundle P_i of lines in $\mathrm{EG}(m, p^s)$, we can form a $p^{(m-1)s} \times p^{ms}$ matrix $\mathbb{H}_i$ over $GF(2)$ whose rows are the incidence vectors of lines in P_i. It follows from the structural properties of a parallel bundle that the column and row weights of $\mathbb{H}_i$ are 1 and p^s, respectively. Let $1 \leq \gamma \leq K$. We form a $\gamma p^{(m-1)s} \times p^{ms}$ matrix $\mathbb{H}_{EG.GA}$ over $GF(2)$ with γ submatrices arranged in a column as follows:

$$\mathbb{H}_{EG.GA} = \begin{bmatrix} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \vdots \\ \mathbb{H}_\gamma \end{bmatrix}.$$

where each submatrix $\mathbf{H}_i$ is formed based on a parallel bundle P_i of lines in EG(m, p^s). The column and row weights of $\mathbf{H}_{EG,GA}$ are γ and p^s, respectively. The null space over $GF(2)$ of this binary matrix gives a Gallager LDPC code of length $n = p^{ms}$ and minimum distance of at least $\gamma + 1$ for odd γ and $\gamma + 2$ for even γ. Clearly, for $p = 2$, we obtain the class of EG-Gallager LDPC codes given in the previous sections.

EXAMPLE 17.29

Let $m = 2$, $s = 1$, and $p = 43$. The two-dimensional Euclidean geometry EG(2, 43) consists of 1849 points and 1892 lines. The 1892 lines can be grouped into 44 parallel bundles, each consisting of 43 lines parallel to each other. Each line consists of 43 points. Taking 4 and 6 parallel bundles, we can construct two EG-Gallager LDPC codes that are (1849, 1680) and (1849, 1596) codes with rates 0.9086 and 0.8632, respectively. With $\gamma = 4$ and 6, the lower bound $\gamma + 2$ on the minimum distance of an EG-Gallager code gives their minimum distances as at least 6 and 8, respectively. The bit-error performances of these two EG-Gallager LDPC codes with SPA decoding are shown in Figure 17.36 (assuming BPSK signaling). At a BER of 10^{-6}, the (1849, 1680) code achieves a 5.7-dB coding gain over the uncoded BPSK system and is 1.5 dB from the Shannon limit for rate 0.9086, and the (1849, 1596) code achieves a 6-dB coding gain over the uncoded BPSK system and is 2.1 dB from the Shannon limit for rate 0.8632.

EXAMPLE 17.30

Suppose it is desired to construct a rate-1/2 LDPC code with length around 6400 based on geometry decomposition and masking. To satisfy the length constraint, we must choose m, s, and p such that $p^{ms} \simeq 6400$. There are many possible choices of m, s, and p for which $p^{ms} \simeq 6400$. One such choice is $m = 2$, $s = 4$, and $p = 3$. With this choice, the geometry for decomposition is the three-dimensional Euclidean geometry EG(2, 3^4). This geometry consists of 6561 points and 6642 lines, and each line consists of 81 points. The 6642 lines can be grouped into 82 parallel bundles, with each parallel bundle consisting of 81 parallel lines. We decompose this geometry based on a parallel bundle $P(2, 1)$ of lines. Then, there are 81 connecting parallel bundles of lines, $Q_1, Q_2, \cdots, Q_{81}$, with respect to $P(2, 1)$. The incidence matrix $\mathbf{A}_i$ of a connecting parallel bundle Q_i is an 81×6561 matrix consisting of eighty-one 81×81 permutation matrices. To achieve the desired rate 1/2 and length equal or close to 6400, we set $\rho = 80$ and $\gamma = 40$. With this choice of ρ and γ, we construct a base matrix $\mathbf{H}_{EG,GA}(80)$ based on (17.72). $\mathbf{H}_{EG,GA}$ is a 3240×6480 matrix with column and row weights 40 and 80, respectively. It is a 40×80 array of 81×81 permutation matrices. Next, we need to construct a 40×80 masking matrix $\mathbb{Z}$. Suppose we choose the column and row weights of $\mathbb{Z}$ to be 3 and 6, respectively. To construct $\mathbb{Z}$, we choose two distinct primitive 40-tuples over $GF(2)$. Cyclically shifting these two distinct primitive 40-tuples, we form two 40×40 circulants $\mathbb{G}_1$ and $\mathbb{G}_2$. Then, $\mathbb{Z} = [\mathbb{G}_1 \mathbb{G}_2]$. Because the weight of each circulant is 3, it is easy to construct $\mathbb{G}_1$ and $\mathbb{G}_2$ such that the Tanner graph of $\mathbb{Z}$ is free of cycles of length 4. Masking the base matrix $\mathbf{H}_{EG,GA}(80)$ with $\mathbb{Z}$, we obtain a 3240×6480 masked matrix

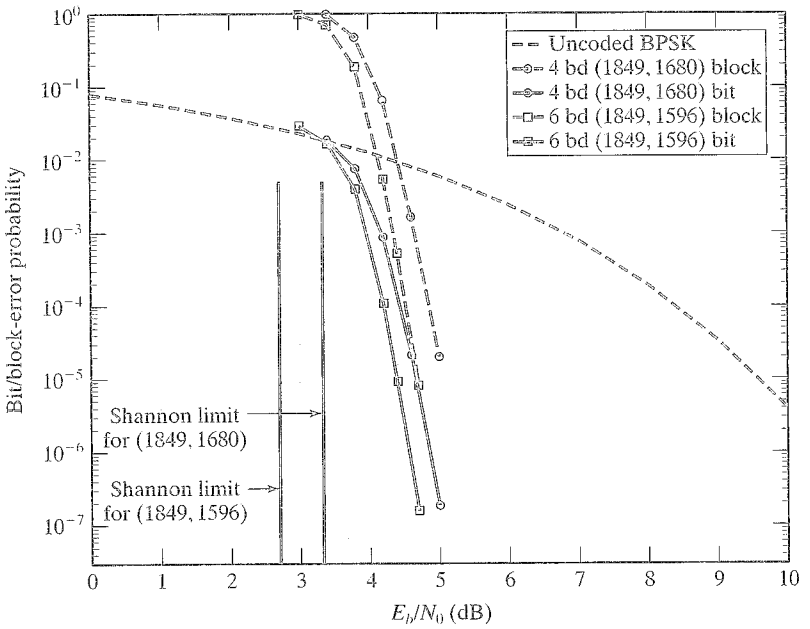


FIGURE 17.36: Error probabilities of two EG-Gallager codes of length 1849.

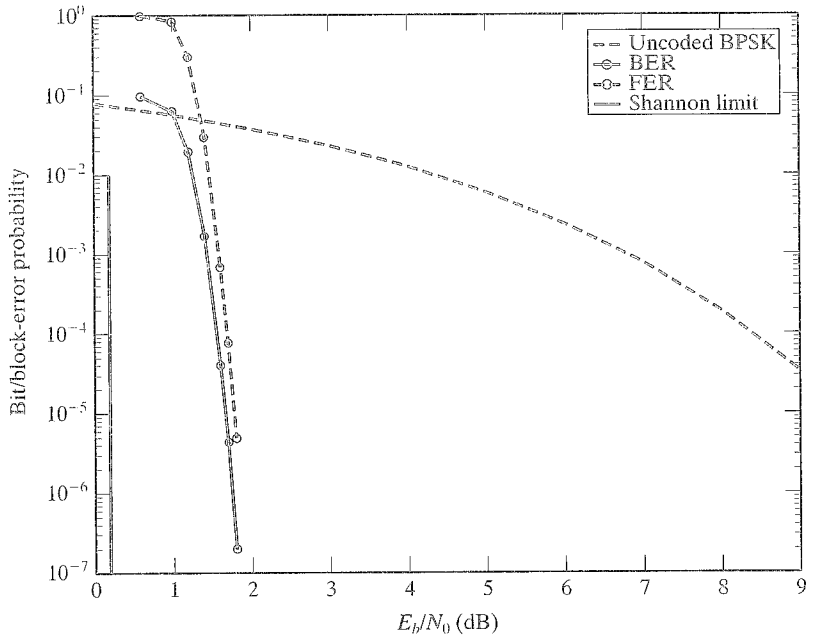


FIGURE 17.37: Bit- and block-error probabilities of a (6480, 3240) LDPC code.

$\mathbb{M} = \mathbb{Z} \otimes \mathbb{H}_{EG,GA}(80)$ with column and row weights 3 and 6, respectively. The null space of $\mathbb{M}$ gives a (3, 6) regular (6480, 3240) LDPC code with rate 1/2. The error performance of this code decoded with the SPA is shown in Figure 17.37. It has both good bit- and block-error performances. At a BER of 10^{-6}, it performs 1.5 dB from the Shannon limit.

Construction based on finite geometries over $GF(p^s)$ results in a very large class of finite-geometry LDPC codes containing all the finite-geometry LDPC codes constructed in the previous sections of this chapter as subclasses.

17.14 RANDOM LDPC CODES

In addition to being formed by the geometric construction presented in the previous sections, LDPC codes also can be constructed by computer search in a pseudorandom manner based on a set of guidelines satisfying the conditions given in Definition 17.1. This construction results in an ensemble of random codes that have been proved to contain good LDPC codes [10].

Suppose it is desired to construct an LDPC code of length n with rate k/n. To construct a parity-check matrix for this desired code, we need to choose an appropriate column weight γ and an appropriate number J of rows. It is clear that J must be at least equal to $n - k$, the number of parity-check symbols of the desired code. In computer construction, J is usually chosen to be equal to $n - k$. For $\mathbb{H}$ to have constant row weight ρ, the condition

$$\gamma \times n = \rho \times (n - k) \tag{17.81}$$

must hold. Otherwise, $\mathbb{H}$ cannot have constant row weight. In this case, we simply try to keep all the row weights close to ρ. If n is divisible by $n - k$, it follows from (17.81) that ρ is a multiple of γ; that is, $\rho = \gamma n/(n - k)$. For this case, we can construct a regular low-density parity-check matrix with column weight γ and row weight ρ. If n is not divisible by $n - k$, we divide $\gamma \times n$ by $n - k$ and obtain

$$\gamma \times n = \rho(n - k) + b, \tag{17.82}$$

where ρ and b are the quotient and remainder, respectively, with $0 < b < n - k$. The expression of (17.82) can be rearranged as follows:

$$\gamma \times n = (n - k - b)\rho + b(\rho + 1). \tag{17.83}$$

which suggests that we can construct a low-density parity-check matrix $\mathbb{H}$ with two row weights, ρ and $\rho + 1$, respectively. For convenience, $\mathbb{H}$ is constructed in such a way that its top b rows have weight $\rho + 1$, and its bottom $n - k - b$ rows have weight ρ.

The construction of $\mathbb{H}$ is carried out step by step. At each step, one column is added to a partially formed matrix. Each added column must satisfy certain constraints. For $1 \leq i \leq n$, at the ith step, a binary $(n - k)$-tuple of weight γ is chosen as a candidate column $\mathbb{h}_i$ and is added to the partial parity-check matrix

$$\mathbb{H}_{i-1} = [\mathbb{h}_1, \mathbb{h}_2, \cdots, \mathbb{h}_{i-1}] \tag{17.84}$$

obtained at the $(i-1)$th step to form the next partial parity-check matrix $\mathbb{H}_i$. For $i = 1$, $\mathbb{H}_0$ is simply the null matrix. To add column $\mathbb{h}_i$ to $\mathbb{H}_{i-1}$, the following constraints must be satisfied:

1. Choose $\mathbb{h}_i$ at random from the remaining binary $(n-k)$-tuples that are not being used in $\mathbb{H}_{i-1}$ and that were not rejected earlier.
2. Check whether $\mathbb{h}_i$ has more than one 1-component in common with any column in $\mathbb{H}_{i-1}$. If not, go to step 3; otherwise, reject $\mathbb{h}_i$ and go back to step 1 to choose another candidate column.
3. Add $\mathbb{h}_i$ to $\mathbb{H}_{i-1}$ to form a temporary partial parity-check matrix $\mathbb{H}_i$. Check the row weights of $\mathbb{H}_i$. If all the top b rows of $\mathbb{H}_i$ have weights less than or equal to $\rho + 1$, and all the bottom $n-k-b$ rows of $\mathbb{H}_i$ have weights less than or equal to ρ, then permanently add $\mathbb{h}_i$ to $\mathbb{H}_{i-1}$ to form $\mathbb{H}_i$ and go to step 1 to continue the construction process. If any of the top b rows of $\mathbb{H}_i$ has weight exceeding $\rho + 1$, or any of the bottom $n-k-b$ rows of $\mathbb{H}_i$ has weight exceeding ρ, reject $\mathbb{h}_i$ and go to step 1 to choose another candidate column.

The step-by-step construction process continues until a parity-check matrix $\mathbb{H}$ with n columns is formed. If $b = 0$, $\mathbb{H}$ is a regular matrix with row and column weights ρ and γ, respectively. If $b \neq 0$, then $\mathbb{H}$ has two row weights, $\rho + 1$ and ρ. For a given n, k, and γ, it is possible that all the $(n-k)$-tuples are either used or rejected before $\mathbb{H}$ is formed. To reduce this possibility, we need to choose n, k, and γ such that the total number of binary $(n-k)$-tuples, $\binom{n-k}{\gamma}$, is much larger than the code length n, or we can relax the row weight constraints in step 3 to allow multiple row weights. This also reduces the probability that a chosen candidate column that satisfies the constraint in step 2 will be rejected in step 3. Of course, we can restart the construction process by choosing another sequence of candidate columns.

If the row rank of $\mathbb{H}$ is exactly $n-k$, the null space of $\mathbb{H}$ gives an (n, k) LDPC code with rate exactly k/n. If the rank of $\mathbb{H}$ is less than $n-k$, then the null space gives an (n, k') LDPC code with $k' > k$ and rate $k'/n > k/n$. We can readily show that the rate R of the constructed code is lower bounded as follows:

$$R \geq 1 - \frac{\gamma}{\rho}.$$

The constraint at step 2 of column selection ensures that the Tanner graph of the code does not contain any cycle of length 4. Therefore, the girth of the Tanner graph is at least 6. The foregoing construction is efficient only for small γ, usually 3 or 4. For large γ, to check the constraints at steps 2 and 3 can be computationally expensive. Because at step 1 of the construction a column is chosen at random from the remaining available binary $(n-k)$-tuples, the code constructed is not unique. The construction gives an ensemble of random LDPC codes. With this construction it is very hard to determine the minimum distance of the code constructed. For small γ, 3 or 4, the lower bound, $\gamma + 1$, on the minimum distance can be very poor.

EXAMPLE 17.31

Suppose we want to construct a (504, 252) LDPC code with rate 1/2. We choose $\gamma = 3$. Because $n = 504$, and $n - k = 252$, $n/(n-k) = 2$, so we choose row weight

$\rho = 2 \times 3 = 6$. The number of binary 252-tuples with weight 3 is

$$\binom{252}{3} = 2635500,$$

which is much larger than the code length $n = 252$. Following the construction procedure, we obtain a regular 252×504 parity-check matrix $\mathbb{H}$ with column and row weights 3 and 6, respectively. The density of $\mathbb{H}$ is $6/504 = 0.012$. Its row rank is 252, and hence its null space gives a (504, 252) LDPC code with rate 1/2. Because γ is 3, the minimum distance of the code is at least 4. The true minimum distance of this code may be greater than 4. The bit-error performance of this code with SPA decoding is shown in Figure 17.29. We see that this code performs very close to the rate-1/2 (512, 256) EG-Gallager code given in Example 17.23.

Random construction results in a large ensemble of LDPC codes that contains finite-geometry LDPC codes as a subclass. Clearly, there must exist random LDPC codes that outperform finite-geometry LDPC codes in error performance, especially long random codes. Computer-generated random LDPC codes, in general, do not have the structural properties of the finite-geometry LDPC codes, such as cyclic or quasi-cyclic structure. Consequently, encoding of a random LDPC code in hardware is much more complex than encoding a finite-geometry LDPC code; that is, its encoding cannot be implemented with linear shift registers. Because computer-generated LDPC codes, in general, have relatively small column weight, the number of check-sums orthogonal on a code bit that can be formed is small. As a result, these codes perform poorly with one-step majority-logic decoding or bit-flipping decoding. Furthermore, computer-generated random LDPC codes with SPA decoding do not converge as fast as finite-geometry LDPC codes do. For finite-geometry LDPC codes with SPA decoding, usually 5 iterations give a performance only a fraction of a decibel from their performance with 100 iterations, as demonstrated in Figure 17.11 for the (4095, 3367) cyclic EG-LDPC code. With all these disadvantages, long random LDPC codes do perform very close to the Shannon limit. For example, very long random LDPC codes (10^7 bits long) have been constructed and shown to perform only a few thousandths of a decibel from the Shannon limit [14].

17.15 IRREGULAR LDPC CODES

An irregular LDPC code is defined by a parity-check matrix $\mathbb{H}$ with multiple column weights and multiple row weights. In terms of its Tanner graph, the variable nodes (code-bit vertices) have multiple degrees, and the check nodes (check-sum vertices) have multiple degrees. It has been shown that long random irregular codes perform arbitrarily close to the Shannon limit [8, 12–14]. Irregular LDPC codes are most commonly designed and constructed based on their Tanner graphs. One such approach is to design these codes in terms of the degree distributions of the variable and check nodes of their Tanner graphs [12, 13].

Consider the Tanner graph $\mathcal{G}$ of an irregular LDPC code with parity-check matrix $\mathbb{H}$. The variable nodes in $\mathcal{G}$ correspond to the columns of $\mathbb{H}$, and the check nodes of $\mathcal{G}$ correspond to the rows of $\mathbb{H}$. The degree of a node in $\mathcal{G}$ is defined as

the number of edges connected to it. The degree of a variable node is simply equal to the weight of its corresponding column in $\mathbb{H}$, and the degree of a check node is simply equal to the weight of its corresponding row in $\mathbb{H}$. Let

$$\gamma(X) = \sum_{i=1}^{d_v} \gamma_i X^{i-1} \tag{17.85}$$

be the *degree distribution* of the variable nodes of $\mathcal{G}$, where γ_i denotes the fraction of variable nodes in $\mathcal{G}$ with degree i, and d_v denotes the maximum variable-node degree. Let

$$\rho(X) = \sum_{i=1}^{d_c} \rho_i X^{i-1} \tag{17.86}$$

be the degree distribution of the check nodes of $\mathcal{G}$, where ρ_i denotes the fraction of check nodes in $\mathcal{G}$ with degree i, and d_c denotes the maximum check-node degree.

In [12] and [13] it has been shown that the error performance of an irregular LDPC code depends on the variable- and check-node degree distributions of its Tanner graph. Shannon limit–approaching irregular LDPC codes can be designed by optimizing the two degree distributions via algorithms centered on the evolution of the probability densities of the messages passed between the two types of nodes in a belief propagation decoder; however, the design algorithm presented in [12] gives optimal degree distributions for a given code rate only when the code length approaches infinity, and the code graph is cycle free with an infinite number of decoding iterations. Tanner graphs for codes of finite lengths cannot be made cycle free; hence, the optimal degree distributions designed for infinite code length will not be optimal any more for codes of finite lengths. If we construct an irregular LDPC code of finite length based on the asymptotically optimal degree distributions, a high error floor may result, and the code may perform poorly in the low-bit error-rate range owing to the large number of degree-2 variable nodes in its Tanner graph. A large number of degree-2 variable nodes, in general, results in poor minimum distance. To overcome these problems, the following additional design rules are proposed in [12]: (1) the degree-2 variable nodes are made cycle free; (2) degree-2 variable nodes are made to correspond to the parity check bits of a codeword; and (3) the code graph is free of cycles of length 4. In order to improve the error floor performance, another constraint is proposed in [21] and [22]. This constraint requires that the number of degree-2 variable nodes must be smaller than the number of parity check bits of the code (or the number of columns of the parity check matrix $\mathbb{H}$ with weight 2 must be smaller than the number of rows of $\mathbb{H}$).

The requirement that the number of degree-2 variable nodes be smaller than the number of parity-check bits of the code to be designed can be met with a simple degree redistribution. Let n_d and R_d be the desired code length and rate, respectively. First, we design the asymptotically optimal degree distribution $\gamma(X)$ and $\rho(X)$ for the desired rate based on the evolution of probability densities. From the fraction of the degree-2 variable nodes given in $\gamma(X)$, we compute the number N_2 of degree-2 variable nodes using the desired code length n_d; that is, $N_2 = \gamma_2 \times n_d$. If N_2 is smaller than the number $n_d(1 - R_d)$ of the parity-check bits of the desired code, then $\gamma(X)$ and $\rho(X)$ are used for constructing the desired code. If N_2 is greater

than $n_d(1 - R_d)$, then we convert the excess degree-2 variable nodes to variable nodes of next-higher degree, say, degree 3 if it exists. The result is a modified variable-node degree distribution $\gamma^*(X)$. In a Tanner graph, the sum of check-node degrees (equal to the total number of edges of the graph) is equal to the sum of variable-node degrees. If we modify the variable-node degree distribution $\gamma(X)$, we must change the check-node degree distribution $\rho(X)$ accordingly to make the sum of check-node degrees the same as the sum of variable-node degrees in the new variable-node degree distribution.

The next step in code construction is to construct a Tanner graph by connecting the variable nodes and check nodes with edges under the constraints given by the degree distributions. Because the selection of edges in the graph construction is not unique, edges are selected randomly. During the edge selection process, effort must be made to avoid having cycles among the degree-2 variable nodes and cycles of length 4 in the code graph. As a result, computer search is needed. Once a code graph is constructed, we form the corresponding parity-check matrix $\mathbb{H}$. Then, the column and row weight distributions of $\mathbb{H}$ are the same as the variable- and check-node degree distributions. The null space of $\mathbb{H}$ gives an irregular code of the desired length and rate. The described construction gives a random irregular LDPC code.

The masking technique presented in Section 17.11 can be used to simplify the construction of irregular LDPC codes based on the degree distributions of variable and check nodes of their Tanner graphs. For masking, the array $\mathbb{H}_{EG,GA}(\rho)$ of permutation matrices given by (17.72) is used as the base matrix. Suppose the degree distributions $\gamma(X)$ and $\rho(X)$ have been designed for a given code rate R_d and length n_d. Suppose geometry $\mathrm{EG}(m, p^s)$ is used for constructing the base matrix. We choose parameters γ and ρ such that $\rho p^{(m-1)s}$ is equal or close to the desired code length n_d, and $(\rho - \gamma)/\gamma$ is equal or close to the desired rate R_d. We construct a $\gamma \times \rho$ masking matrix $\mathbb{Z} = [z_{i,j}]$ with column and row weight distributions $\mathfrak{c}(X)$ and $\mathfrak{d}(X)$, respectively, identical to the degree distributions $\gamma(X)$ and $\rho(X)$ of the variable and check nodes, respectively, where $d_v \leq \gamma$ and $d_c \leq \rho$.

In constructing the masking matrix $\mathbb{Z}$, we put columns with weight 2 in the parity-check positions and make them cycle free among them. The cycle-free condition can easily be achieved by using a set of weight-2 columns obtained by downward shifting two consecutive 1's from the top of $\mathbb{Z}$ until the second 1 reaches the bottom of $\mathbb{Z}$ as shown:

$$\mathbb{Z} = \left[\begin{array}{cccccc|c} 1 & 0 & 0 & 0 & \cdots & 0 & \\ 1 & 1 & 0 & 0 & \cdots & 0 & \\ 0 & 1 & 1 & 0 & \cdots & 0 & \\ 0 & 0 & 1 & 1 & \cdots & 0 & \text{other columns} \\ & & & & \ddots & & \\ 0 & 0 & 0 & 0 & \cdots & 1 & \\ 0 & 0 & 0 & 0 & \cdots & 1 & \end{array}\right]. \tag{17.87}$$

Masking the base matrix $\mathbb{H}_{EG,GA}(\rho)$ given by (17.72) with $\mathbb{Z}$, we obtain the masked matrix

$$\mathbb{M} = \mathbb{Z} \otimes \mathbb{H}_{EG,GA}(\rho)$$

with column and row weight distributions $\mathfrak{c}(X)$ and $\mathfrak{d}(X)$, respectively. Consequently, the associated Tanner graph of $\mathbb{M}$ has variable- and check-node degree distributions

$\gamma(X)$ and $\rho(X)$, as designed. Hence, the null space of $\mathbb{M}$ gives an irregular LDPC code with the designed degree distributions. Because the size of $\mathbb{Z}$ is, in general, very small compared with the size of the code graph, it is much easier to construct to meet the degree distribution requirements. Furthermore, since the associated Tanner graph of the base matrix $\mathbb{H}_{EG,GA}(\rho)$ is already free of cycles of length 4, we do not have to worry about eliminating these short cycles. The masking construction significantly reduces the computation and search effort of large random edge selection (or assignment) in the construction of the code graph, as is needed in computer generation of an irregular LDPC code [12].

We now use two examples to illustrate the construction of irregular LDPC codes using the masking technique.

EXAMPLE 17.32

Suppose an irregular LDPC code with desired length $n_d = 4000$ and rate 0.82 is to be constructed. Using the density evolution technique, we find the asymptotically optimal variable- and check-node degree distributions for rate 0.82:

$$\gamma(X) = 0.4052X + 0.3927X^2 + 0.1466X^6 + 0.0555X^7,$$
$$\rho(X) = 0.3109X^{18} + 0.6891X^{19}.$$

The maximum variable- and check-node degrees are 8 and 20, respectively. Suppose we choose the two-dimensional Euclidean geometry $\mathrm{EG}(2, 2^6)$ over $GF(2^6)$ for constructing the base matrix $\mathbb{H}_{EG,GA}(\rho)$ given by (17.72). Using this geometry, we can partition the incidence matrix of a parallel bundle of lines into sixty-four 64×64 permutation matrices. To achieve a code length close to the desired code length 4000, we choose $\rho = 63$, which is larger than the maximum check-node degree 20. To achieve a rate close to the desired code rate 0.82, we choose $\gamma = 12$, which is greater than the maximum variable-node degree 8. For the choice of $\gamma = 12$ and $\rho = 63$, the rate of the code is at least $(\rho - \gamma)/\rho = (63 - 12)/64 = 0.81$, which is close to the desired rate 0.82. The length of the code is $63 \times 64 = 4032$, which is close to the desired code length. It follows from (17.72) that the base matrix $\mathbb{H}_{EG,GA}(63)$ is a 768×4032 matrix, which is a 12×63 array of 64×64 permutation matrices. For the desired code, the number of parity-check bits is $4000 \times (1 - 0.82) = 720$; however, the number of degree-2 variable nodes computed using the first coefficient $\gamma_2 = 0.4052$ of $\gamma(X)$ is $n_d \times \gamma_2 = 4000 \times 0.4052 = 1620$, which is larger than 720, the number of parity-check bits of the desired code. Therefore, we modify the variable-node degree distribution $\gamma(X)$ by converting 903 variable nodes of degree 2 to degree 3. The degree redistribution results in a new variable-node degree distribution,

$$\gamma^*(X) = 0.1798X + 0.6181X^2 + 0.1466X^6 + 0.0555X^7.$$

There is a small change in the coefficient of each degree in the check-node degree distribution $\rho(X)$. Next, we need to construct a 12×63 masking matrix $\mathbb{Z}$ with column and row weight distributions identical to the modified degree distributions $\gamma^*(X)$ and $\rho^*(X)$. Table 17.5 gives the desired column and row weight distributions of the masking matrix $\mathbb{Z}$. By computer search, we find the desired masking matrix $\mathbb{Z}$.

Masking the 768×4032 base matrix $\mathbb{H}_{EG,GA}(63)$ with $\mathbb{Z}$, we obtain the masked matrix $\mathbb{M} = \mathbb{Z} \otimes \mathbb{H}_{EG,GA}(63)$. The null space of $\mathbb{M}$ gives a (4032, 3264) irregular code

TABLE 17.5: Desired column and row weight distributions of the masking matrix $\mathbb{Z}$ for Example 17.32.

Column weight distribution		Row weight distribution	
Column weight	No. of columns	Row weight	No. of rows
2	11	19	4
3	39	20	8
7	9		
8	4		

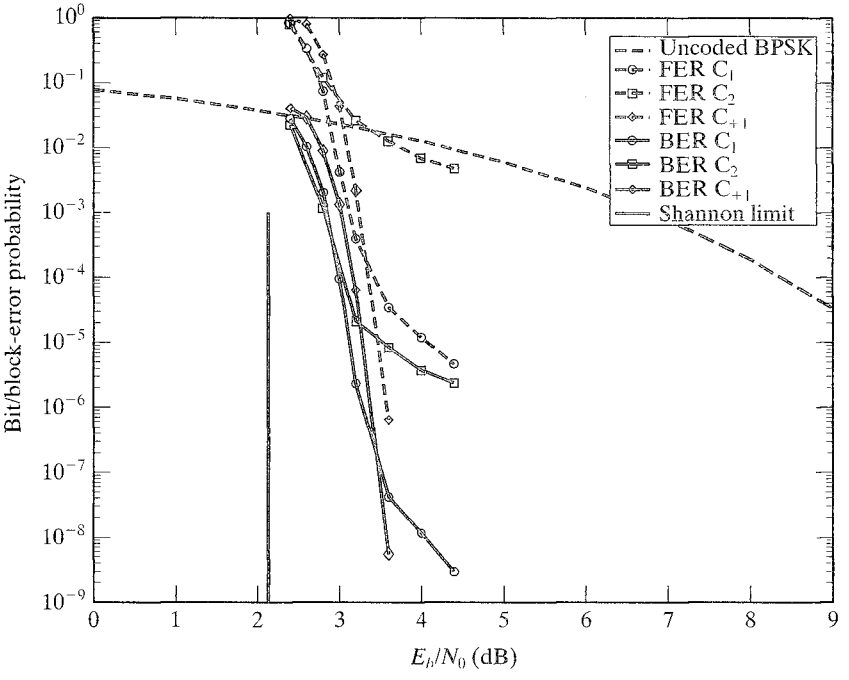


FIGURE 17.38: Error performances of the three (4032, 3264) irregular LDPC codes given in Examples 17.32 and 17.33.

C_1 with rate 0.81. The error performance of this code with SPA decoding is shown in Figure 17.38. At a BER of 10^{-5}, the code performs only 1 dB from the Shannon limit; however, it has an error floor starting around a BER of 10^{-6}. Figure 17.38 also includes the error performance of a (4032, 3264) irregular code C_2 constructed based on the original asymptotically optimal degree distributions without degree redistribution. We see that C_1 performs much better than C_2. The error floor of C_2 occurs above a BER of 10^{-5}.

An irregular code of finite length whose Tanner graph contains degree-2 variable nodes, in general, has an error floor owing to its relatively poor minimum distance. The error floor can be pushed down by either reducing the number of degree-2 variable nodes or removing all of them. An approach to removing all the degree-2 variable nodes is to increase every degree in the degree distribution $\gamma(X)$ of variable nodes by 1; that is, degree-2 is converted into degree-3, degree-3 is converted into degree-4, and so on [53].

EXAMPLE 17.33

(Example 17.32 continued) Suppose we increase every degree in the asymptotically optimal variable-node degree distribution $\gamma(X)$ given in Example 17.32 by 1. This degree conversion results in the following variable-node degree distribution:

$$\gamma_{+1}(X) = 0.4052X^2 + 0.3927X^3 + 0.1466X^7 + 0.0555X^8.$$

We must also modify the check-node degree distribution $\rho(X)$ to make the sum of check-node degrees the same as the sum of variable-node degrees in the new variable-node degree distribution $\gamma_{+1}(X)$. We can do this by increasing each degree in $\rho(X)$ by 4 without changing its coefficient. The result is a new check-node degree distribution:

$$\rho_{+1}(X) = 0.3109X^{22} + 0.6891X^{23}.$$

Using the new degree distributions, $\gamma_{+1}(X)$ and $\rho_{+1}(X)$, we construct a 12×63 masking matrix $\mathbb{Z}_{+1}$ with desired column and row weight distributions given in Table 17.6. For masking, the base matrix $\mathbb{H}_{EG,GA}(63)$ is still the same as the one given in Example 17.32. The masked matrix $\mathbb{M}_{+1} = \mathbb{Z}_{+1} \otimes \mathbb{H}_{EG,GA}(63)$ also gives a (4032, 3264) irregular LDPC code C_{+1}. The error performance of this code is also shown in Figure 17.38. We see that C_{+1} has much better error floor performance than codes C_1 and C_2, given in Example 17.32. Actually, there is no error floor down to a BER of 5×10^{-9}; however, there is a small performance degradation (less than 0.15 dB) above a BER of 10^{-7}.

Examples 17.32 and 17.33 show that pushing the error floor down by increasing the degrees of variable nodes of the code graph, the error performance of the code moves away from the Shannon limit in the waterfall region.

TABLE 17.6: Column and row weight distributions of the masking matrix for Example 17.33.

Column weight distribution		Row weight distribution	
Column weight	No. of columns	Row weight	No. of rows
3	25	23	4
4	25	24	8
8	9		
9	4		

EXAMPLE 17.34

The following optimal degree distributions of variable and check nodes of a Tanner graph are designed for a rate-1/2 irregular LDPC code of infinite length [12]:

$$\gamma(X) = 0.47707681X + 0.28057X^2 + 0.034996X^3 + 0.096329X^4 + 0.009088X^6 + 0.0013744X^{13} + 0.100557X^{14},$$
$$\rho(X) = 0.982081X^7 + 0.0179186X^8.$$

Suppose we want to construct a rate-1/2 irregular LDPC code with length around 10000 using masking based on these degree distributions. To construct such a code, there are many geometries that can be used for constructing the base matrix for masking. We choose the two-dimensional Euclidean geometry $\mathrm{EG}(2, 2^7)$ for constructing the base matrix $\mathbb{H}_{EG,GA}(\rho)$ given by (17.72). Using this geometry, we can partition the incidence matrix of a parallel bundle of lines into one hundred twenty-eight 128×128 permutation matrices. Suppose we choose $\gamma = 40$ and $\rho = 80$, which are greater than the maximum variable- and check-node degrees, 15 and 9, respectively. With the choice of $\gamma = 40$ and $\rho = 80$, $80 \times 128 = 10240$, which is close to the desired code length 10000, and $(\rho - \gamma)/\gamma = (80 - 40)/80 = 0.5$, which is the same as the desired rate 1/2. It follows from (17.72) that the base matrix $\mathbb{H}_{EG,GA}(80)$ is a 40×80 array of 128×128 permutation matrices. For masking, we need to construct a 40×80 masking matrix $\mathbb{Z}$ with desired column and row weight distributions given in Table 17.7. This is done by computer search. Masking $\mathbb{H}_{EG,GA}(80)$ with $\mathbb{Z}$, we obtain a 5120×10240 masked matrix $\mathbb{M} = \mathbb{Z} \otimes \mathbb{H}_{EG,GA}(80)$. The null space of $\mathbb{M}$ gives a (10240, 5102) irregular LDPC code C with rate 1/2. The error performance of this code is shown in Figure 17.39. At a BER of 10^{-6}, it performs 1 dB from the Shannon limit.

TABLE 17.7: Column and row weight distributions of the masking matrix for Example 17.34.

Column weight distribution		Row weight distribution	
Column weight	No. of columns	Row weight	No. of rows
2	38	8	39
3	22	9	1
4	3		
5	8		
7	1		
15	8		

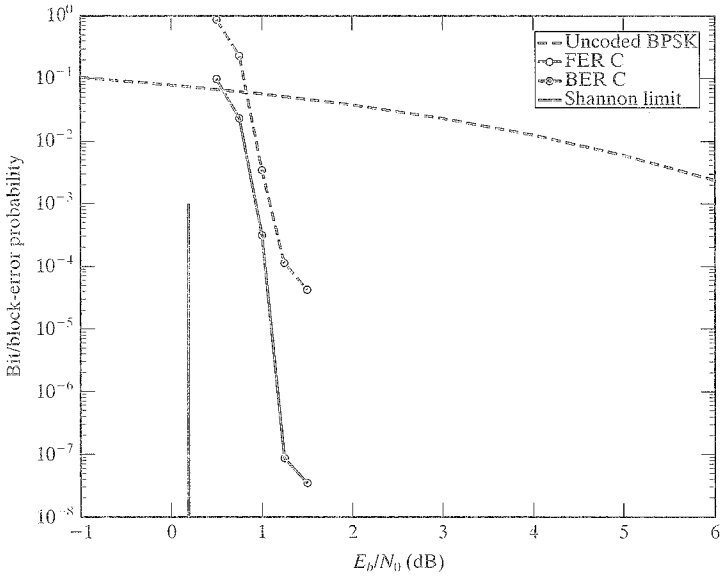


FIGURE 17.39: Error performances of the two (10240, 5120) irregular LDPC codes given in Example 17.34.

17.16 GRAPH-THEORETIC LDPC CODES

Finite geometries form a branch of combinatorial mathematics. Besides finite geometries, there are other branches in combinatorial mathematics that can be used for constructing LDPC codes. These branches include *graph theory*, *combinatoric designs*, and *difference sets* [35, 41, 57, 58]. In Section 8.3 we showed that codes constructed based on perfect difference sets are one-step MLG decodable, and we also showed in Section 17.5 that the codes constructed based on a special class of perfect difference sets are LDPC codes whose Tanner graphs do not contain cycles of length 4. Construction of LDPC codes based on random bipartite (or Tanner) graphs was briefly discussed in the previous section of this chapter. Construction based on random graphs results in an ensemble of LDPC codes. In this section we present another graph-theoretic approach to the construction of LDPC codes [59]. The construction is based on selecting a set of paths of the same length in a given graph that satisfies certain constraints.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected graph with vertex set $\mathcal{V} = \{v_1, v_2, \cdots, v_q\}$ and edge set $\mathcal{E} = \{e_1, e_2, \cdots, e_L\}$. We require that $\mathcal{G}$ not contain self-loops, and two vertices in $\mathcal{G}$ are connected by at most one edge (i.e., no multiple edges between two vertices). Some basic structural properties of a graph were discussed in Section 17.2. Two paths in $\mathcal{G}$ are said to be *disjoint* if they do not have any vertex in common. Two paths are said to be *singularly crossing* each other if they have one and only one vertex in common; that is, they intersect (or cross each other) at one and only one vertex. For $1 < \gamma \leq q$, let $\mathcal{P}$ be a set of paths of length $\gamma - 1$ in $\mathcal{G}$, that satisfies

the following constraint:

Any two paths in $\mathcal{P}$ are either disjoint or singularly crossing each other.

This constraint is called the *disjoint-crossing* (DC) constraint. Let $n = |\mathcal{P}|$ denote the number of paths in $\mathcal{P}$ and q_o denote be the number of vertices in $\mathcal{G}$ that are covered by (or on) the paths in $\mathcal{P}$. For simplicity, we call the vertices in $\mathcal{G}$ that are covered by the paths in $\mathcal{P}$ the vertices in (or of) $\mathcal{P}$. We form a $q_o \times n$ matrix $\mathbb{H} = [h_{i,j}]$ whose rows correspond to the q_o vertices in $\mathcal{P}$ and whose columns correspond to the n paths in $\mathcal{P}$, where $h_{i,j} = 1$ if and only if the ith vertex v_i of $\mathcal{P}$ is on the jth path in $\mathcal{P}$; otherwise, $h_{i,j} = 0$. This matrix is called the *incidence matrix* of $\mathcal{P}$. The columns of this matrix are called the *incidence vectors* of the paths in $\mathcal{P}$ with respect to the vertices in $\mathcal{P}$; the jth column simply displays the vertices on the jth path of $\mathcal{P}$. Because the length of each path in $\mathcal{P}$ is $\gamma - 1$, there are γ vertices on each path in $\mathcal{P}$. Therefore, each column of $\mathbb{H}$ has weight γ. The weight of the ith row of $\mathbb{H}$ is equal to the number of paths in $\mathcal{P}$ that intersect at the ith vertex v_i of $\mathcal{P}$. It follows from the DC constraint that no two columns (or two rows) of $\mathbb{H}$ can have more than one 1-component in common. If γ is much smaller than the number q_o of vertices in $\mathcal{P}$, $\mathbb{H}$ is a sparse matrix. Then, the null space C over $GF(2)$ of $\mathbb{H}$ gives an LDPC code of length n whose Tanner graph does not contain cycles of length 4. The minimum distance of C is at least $\gamma + 1$. If the rows of $\mathbb{H}$ have the same weight ρ (i,e., each vertex of $\mathcal{P}$ is intersected by ρ paths in $\mathcal{P}$), then C is a (γ, ρ)-regular LDPC code.

The path set $\mathcal{P}$ can be constructed by using a trellis $\mathcal{T}$ of $\gamma - 1$ sections with γ levels of nodes, labeled $0, 1, \cdots, \gamma - 1$. Each level of $\mathcal{T}$ consists of q nodes that are simply the q vertices of $\mathcal{G}$. For $0 \leq k < \gamma - 1$, a node v_i at the kth level of $\mathcal{T}$ and a node v_j at the $(k+1)$th level of $\mathcal{T}$ are connected by a branch if and only if (v_i, v_j) is an edge in $\mathcal{G}$. This $(\gamma - 1)$-section trellis $\mathcal{T}$ is called the *path trellis* of $\mathcal{G}$ of length $\gamma - 1$. A path of length $\gamma - 1$ in $\mathcal{G}$ is a path in $\mathcal{T}$ starting from an initial node at the 0th level of $\mathcal{T}$ and ending at a terminal node at the $(\gamma - 1)$th level of $\mathcal{T}$ such that all the nodes on the path are different. Figures 17.40(a) and 17.40(b) show a complete graph (i.e., every two vertices are connected by an edge) with seven vertices and its path trellis of length 2, respectively.

To find a set $\mathcal{P}$ of paths of length $\gamma - 1$ in $\mathcal{G}$ that satisfies the DC constraint, an *extend-select-eliminate* (ESE) algorithm [59] can be devised to parse the path trellis $\mathcal{T}$ of $\mathcal{G}$. Suppose we have parsed the trellis $\mathcal{T}$ up to the kth level of $\mathcal{T}$ with the ESE algorithm, with $0 \leq k < \gamma - 1$. For $1 \leq i \leq q$, let $N_{i,k}$ denote the number of survivor paths of length k terminating at the ith node of the kth level of $\mathcal{T}$ (i.e., the paths that satisfies the DC constraint). Let $P_{i,k}$ denote this set of survivor paths. Now, we extend all the paths in $P_{i,k}$ to the $(k+1)$th level of $\mathcal{T}$ through the branches diverging from node i. At node i of the $(k+1)$th level of $\mathcal{T}$, we select a set $P_{i,k+1}$ of paths of length $k+1$ that satisfies the DC constraint and eliminate all the other paths that terminate at node i. The result is a survivor path set $P_{i,k+1}$ at node i of level $k+1$ of $\mathcal{T}$. This ESE process continues until the $(\gamma - 1)$th level of $\mathcal{T}$ is reached. Then, the union of the survivor sets, $P_{1,\gamma-1}, P_{2,\gamma-1}, \cdots, P_{q,\gamma-1}$, gives the path set $\mathcal{P}$. For $1 \leq i \leq q$ and $0 \leq k < \gamma$, all the survivor paths in $P_{i,k}$ start from different initial nodes at the 0th level of $\mathcal{T}$ and intersect only at node i of the kth level of $\mathcal{T}$. Two survivor paths terminating at two different nodes at the kth level of $\mathcal{T}$ are either disjoint or cross each other only once at a node of an earlier level of $\mathcal{T}$. Selection of

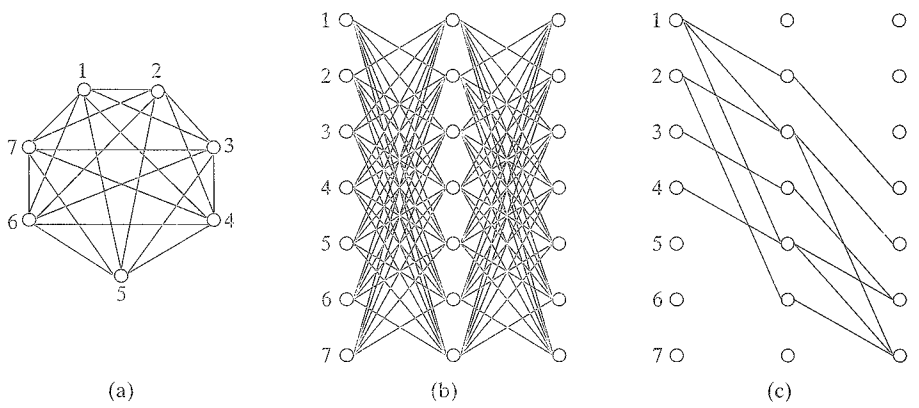


FIGURE 17.40: Complete graph of seven vertices, its path trellis of length 2, and a set of paths of length 2 satisfying the DC constraint found by the ESE algorithm.

survivor paths terminating at one node of the kth level of $\mathcal{T}$ affects the selection of survivor paths terminating at the other nodes of the same level of $\mathcal{T}$. To maintain all the sets of survivor paths at each level of $\mathcal{T}$ at about the same size, we create a priority list of the nodes at each level. The node with the smallest number of survivor paths terminating at it at level k of $\mathcal{T}$ has the first priority of survivor path selection at level $k+1$ of $\mathcal{T}$, whereas the one with the largest number of survivor paths terminating at it at level k of $\mathcal{T}$ has the lowest priority for the survivor path selection at level $k+1$ of $\mathcal{T}$.

EXAMPLE 17.35

Applying the ESE algorithm to the 2-section path trellis of the complete graph $\mathcal{G}$ of seven vertices shown in Figure 17.40 (b), we show a set $\mathcal{P}$ of paths of length 2 that satisfies the DC constraint in Figure 17.40(c). Expressing each path in terms of the vertices lying on it, we see that the paths are

$$(1, 2, 4), (1, 3, 7), (1, 5, 6), (2, 3, 5), (2, 6, 7), (3, 4, 6), (4, 5, 7).$$

We see that the paths in $\mathcal{P}$ cover all seven vertices of the graph. The incidence matrix for this set of paths is

$$\mathbb{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}, \tag{17.88}$$

which has constant row and column weights. The null space of $\mathbb{H}$ gives a (3, 3)-regular (7, 3) LDPC code with a minimum distance of 4 whose Tanner graph does not contain cycles of length 4, but it does contain cycles of length 6. One such cycle is depicted by the heavy lines shown in Figure 17.41.

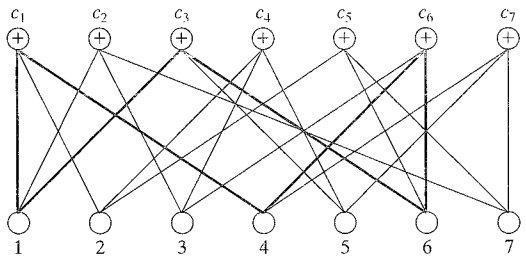


FIGURE 17.41: The Tanner graph of the (3, 3)-regular LDPC code generated by the parity-check matrix given by (17.88).

The graph-theoretic construction of LDPC codes given here is very general and can be applied to any connected graph $\mathcal{G}$. Because the selection of survivor paths at a node of each level of the path trellis $\mathcal{T}$ of $\mathcal{G}$ is not unique, the path set $\mathcal{P}$ obtained at the end of the ESE processing is not unique, and hence the code constructed is not unique.

EXAMPLE 17.36

Suppose we start with a complete graph $\mathcal{G}$ with 400 vertices. We choose $\gamma = 6$. The path trellis $\mathcal{T}$ of $\mathcal{G}$ is a 5-section trellis with six levels of nodes, with each level of $\mathcal{T}$ consisting of 400 nodes. Each path of length 5 in $\mathcal{G}$ is a path in $\mathcal{T}$. Applying the ESE algorithm to process $\mathcal{T}$, we obtain a set $\mathcal{P}$ of 2738 paths of length 5 that satisfies the DC constraint. The paths in $\mathcal{P}$ cover all 400 vertices of the graph. The incidence matrix $\mathbf{H}$ of $\mathcal{P}$ is a 400×2738 matrix with column weight 6. The null space of $\mathbf{H}$ gives a (2738, 2339) LDPC code with rate 0.8543 and a minimum distance of at least 7. The bit- and block-error performances of this code with SPA decoding are shown in Figure 17.42. At a BER of 10^{-6}, the code performs only 1.7 dB from the Shannon limit. Note that the number of parity bits of the code is one less than the number of vertices of the code construction graph.

For a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a large number of vertices, especially a complete graph, processing the path trellis of $\mathcal{G}$ of length $\gamma - 1$ to construct a set of paths of length $\gamma - 1$ in $\mathcal{G}$ to satisfy the DC constraint with the ESE algorithm may become very complex. To overcome this complexity problem, we can take a divide-and-conquer approach [59]. Consider a complete graph $\mathcal{G}$ with q vertices. Suppose we want to construct a set $\mathcal{P}$ of paths of length $\gamma - 1$ in $\mathcal{G}$ that satisfies the DC constraint. We first divide $\mathcal{G}$ into γ blocks, $\mathcal{G}_1, \mathcal{G}_1, \cdots, \mathcal{G}_\gamma$, of equal (or nearly equal) number of vertices. Each block $\mathcal{G}_i$ is a complete subgraph of $\mathcal{G}$, and any two blocks are connected by edges. For each block $\mathcal{G}_i$ we construct a set $\mathcal{P}_i$ of paths of length $\gamma - 1$ in $\mathcal{G}_i$ that satisfies the DC constraint using the ESE algorithm. Clearly, $\mathcal{P}_1, \mathcal{P}_1, \cdots, \mathcal{P}_\gamma$ are disjoint, and two paths from two different sets, $\mathcal{P}_i$ and $\mathcal{P}_j$, do not have any vertex in common. Next, we form a trellis $\mathcal{T}^c$ with γ levels of vertices, labeled $0, 1, \cdots, \gamma - 1$. For $0 \leq i < \gamma$, the ith level of $\mathcal{T}^c$ consists of q_{i+1} nodes that correspond to the q_{i+1} vertices in block $\mathcal{G}_{i+1}$. For $0 \leq i < \gamma - 1$, the nodes at the ith level of $\mathcal{T}^c$ are connected to the nodes at the $(i + 1)$th level of $\mathcal{T}^c$ by

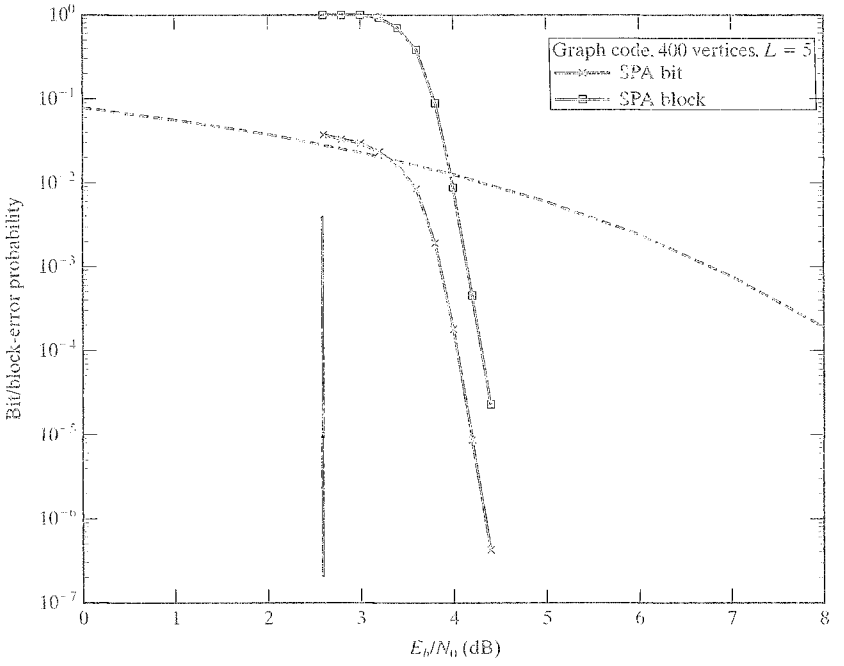


FIGURE 17.42: Error performance of the (2738, 2339) LDPC code constructed based on the complete graph with 400 vertices using the ESE algorithm.

branches that are the edges connecting the vertices in $\mathcal{G}_{i+1}$ to the vertices in $\mathcal{G}_{i+2}$. This $(\gamma - 1)$-section trellis is called a *connecting trellis* for the blocks of $\mathcal{G}$. Then, we process $\mathcal{T}^c$ with the ESE algorithm to find a set $\mathcal{P}^c$ of paths of length $\gamma - 1$ in $\mathcal{T}^c$ that satisfies the DC constraint. Every path in $\mathcal{P}^c$ is a path in $\mathcal{G}$ that contains one and only one vertex in each of the γ blocks of $\mathcal{G}$, and the ith vertex is in $\mathcal{G}_i$. The paths in $\mathcal{P}^c$ are called *connecting paths* of the blocks of $\mathcal{G}$. A path in $\mathcal{P}_i$ and a connecting path in $\mathcal{P}^c$ are either disjoint or they singularly cross each other. For each connecting path in $\mathcal{P}^c$, there is at least one path in $\mathcal{P}_i$ that singularly crosses with it. The union

$$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \cdots \cup \mathcal{P}_\gamma \cup \mathcal{P}^c$$

gives a set of paths in $\mathcal{G}$ that satisfies the DC constraint. The complexity of processing the connecting trellis $\mathcal{T}^c$ with the ESE algorithm is about the same as that of processing the path trellis of length $\gamma - 1$ of each block with the ESE algorithm. We form the incidence matrix $\mathbb{H}$ for $\mathcal{P}$. Then, the null space of $\mathbb{H}$ gives an LDPC code whose Tanner graph does not contain cycles of length 4.

EXAMPLE 17.37

Consider the complete graph $\mathcal{G}$ with 582 vertices. Let $\gamma = 6$. We divide $\mathcal{G}$ into six blocks; each block is a complete graph with 97 vertices. Using the divide-and-conquer ESE algorithm to process the path trellis of length 5 for each block and the connecting trellis for the blocks, we obtain a set $\mathcal{P}$ of 10429 paths of length 5 that satisfies the DC constraint and covers all 582 vertices of the graph. We form the

incidence matrix $\mathbb{H}$ for $\mathcal{P}$, which is a 582×10429 matrix. The null space of $\mathbb{H}$ gives a (10429, 9852) LDPC code with rate 0.945 and a minimum distance of at least 7 whose Tanner graph does not contain cycles of length 4. The bit- and block-error performances with SPA decoding are shown in Figure 17.43. At a BER of 10^{-6}, the code performs 1.05 dB from the Shannon limit.

We also can construct a path set $\mathcal{P}$ of length $\gamma - 1$ in a connected graph $\mathcal{G}$ that satisfies the DC constraint by choosing one path at a time through the $(\gamma - 1)$-section path trellis $\mathcal{T}$ of $\mathcal{G}$. We begin by choosing any path in $\mathcal{T}$ as the first path in $\mathcal{P}$. We delete all the branches on this path from $\mathcal{T}$. Then, we trace the modified path trellis and choose the second path that either has no common vertex with the first path in $\mathcal{P}$ or intersects with the first path in $\mathcal{P}$ at one and only one vertex (DC constraint). Again, we delete the branches on the second path from the modified path trellis and obtain a new modified path trellis. Then, we trace this new path trellis and choose the third path that satisfies the DC constraint with the two paths in $\mathcal{P}$. We delete the branches on the third path from the path trellis and then start a new path selection. We continue this process of path selection and removal of branches on a selected path from the path trellis. Each time we select a path it must satisfy the DC constraint with the paths already in $\mathcal{P}$. When a node in the path trellis has no incoming or outgoing branches, it is removed from the path trellis together with all the branches still attached to it. We continue this process until no path in the path trellis can be chosen without violating the DC constraint, or the path trellis

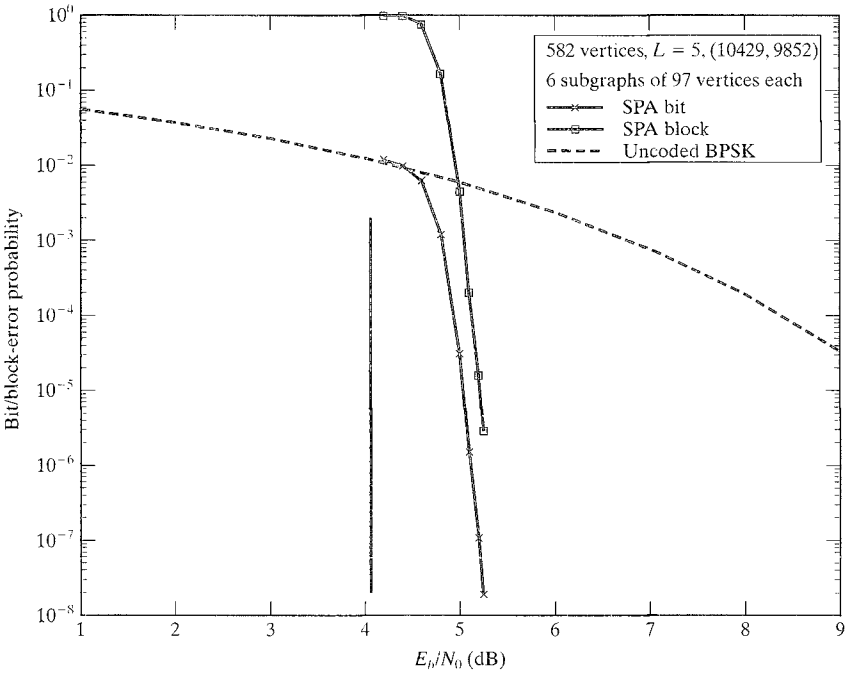


FIGURE 17.43: Error performance of the (10429, 9852) LDPC code constructed based on the complete graph of 582 vertices using the divide-and-conquer ESE algorithm.

is completely disconnected. The resultant path set $\mathcal{P}$ gives an LDPC code. This code construction procedure is actually a graph-theoretic model of the first random LDPC code construction procedure presented in Section 17.14 with the complete graph of $n - k$ vertices as the code construction graph.

Other constructions of LDPC codes based on specific classes of graphs with large girths have also been proposed [60–62].

17.17 CONSTRUCTION OF LDPC CODES BASED ON BALANCED INCOMPLETE BLOCK DESIGNS

Combinatoric design is another important subject in combinatorial mathematics. The objective of this subject is to design experiments systematically and with a view to their analysis. One such design is known as *balanced incomplete block design* (BIBD) [41, 58]. A special subclass of BIBDs can be used for constructing of LDPC codes.

Let $X = \{x_1, x_2, \cdots, x_q\}$ be a set of q objects. A BIBD of X is a collection of n γ-subsets of X, denoted by $B_1, B_2, \cdots, B_n$, called *blocks*, such that the following conditions are satisfied: (1) each object appears in exactly ρ of the n blocks; (2) every two objects appear together in exactly λ of the n blocks; and (3) the number γ of objects in each block is small compared with the total number of objects in X. Thus, a BIBD is characterized by five parameters, n, q, γ, ρ, and λ. Instead of being described by a list of the blocks, a BIBD can be described by a $q \times n$ matrix $\mathbb{H} = [h_{i,j}]$ whose rows correspond to the q objects of X and whose columns correspond to the n blocks of the design, where $h_{i,j} = 1$ if and only if the ith object x_i is in the jth block, and $h_{i,j} = 0$ otherwise. This matrix $\mathbb{H}$ is called the incidence matrix of the design. The column and the row weights of $\mathbb{H}$ are γ and ρ, respectively. Based on the second condition of a BIBD, two rows of $\mathbb{H}$ have exactly λ 1-components in common. If $\lambda = 1$, then $\mathbb{H}$ meets all the conditions of a regular parity-check matrix of an LDPC code given by Definition 17.1. Then, the null space of $\mathbb{H}$ gives a (γ, ρ)-regular LDPC code of length n whose Tanner graph is free of cycles of length 4.

EXAMPLE 17.38

Let $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$ be a set of seven objects. The following blocks:

$$\{x_1, x_2, x_4\}, \quad \{x_2, x_3, x_5\}, \quad \{x_3, x_4, x_6\}, \quad \{x_4, x_5, x_7\},$$
$$\{x_1, x_5, x_6\}, \quad \{x_2, x_6, x_7\}, \quad \{x_1, x_3, x_7\},$$

form a BIBD for the set X. Every block consists of $\gamma = 3$ objects, each object appears in $\rho = 3$ blocks, and every two objects appear together in exactly $\lambda = 1$ block. Its incidence matrix $\mathbb{H}$ is

$$\mathbb{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

The null space of $\mathbb{H}$ gives a (7, 3) LDPC code with a minimum distance of 4.

Combinatoric design is a very old and rich subject in combinatorial mathematics. Over the years, many BIBDs have been constructed with various methods. Extensive coverage can be found in [41, 58]. There are many classes of BIBDs with $\lambda = 1$, which can be used for constructing LDPC codes whose Tanner graphs do not contain cycles of length 4 [64, 65]. Construction of these classes of designs requires much combinatorial mathematics and finite-group theory background and will not be covered here; however, we do present one special class of BIBDs with parameter $\lambda = 1$ constructed by Bose [63].

Let t be a positive integer such that $20t + 1 = p^m$, where p is a prime; that is, $20t + 1$ is a power of a prime. Suppose the field $GF(p^m)$ has a primitive element α that satisfies the condition $\alpha^{4t} - 1 = \alpha^c$, where c is a positive odd integer less than p^m. Then, there exists a BIBD for a set X of $q = 20t + 1$ objects with $n = t(20t + 1)$ blocks, each block consists of $\gamma = 5$ objects, each object appears in $\rho = 5t$ blocks, and every two objects appear together in exactly $\lambda = 1$ block. Let the elements of $GF(p^m), 0, \alpha^0 = 1, \alpha, \alpha^2, \cdots, \alpha^{p^m-2}$, represent the $20t+1 = p^m$ objects of the set X. Then, the BIBD for X is completely specified by the following t base blocks:

$$B_i = \{\alpha^{2i}, \alpha^{2i+4t}, \alpha^{2i+8t}, \alpha^{2i+12t}, \alpha^{2i+16t}\}, \tag{17.89}$$

where $0 \leq i < t$. All the $n = t(20t + 1)$ blocks of the BIBD are obtained by adding each element of $GF(p^m)$ in turn to each of the t base blocks. The incidence matrix $\mathbb{H}$ of this BIBD is a $(20t + 1) \times t(20t + 1)$ matrix with column and row weights 5 and $5t$, respectively. The density of $\mathbb{H}$ is $5/(20t + 1)$, which is very small for $t \geq 2$. Because $\lambda = 1$, it follows from Definition 17.1 that the null space of $\mathbb{H}$ gives an LDPC code of length $n = t(20t + 1)$ whose Tanner graph is free of cycles of length 4. In fact, $\mathbb{H}$ can be put in circulant form. For $0 \leq i < t$, let $\mathbb{v}_i$ be the incidence vector of the base block B_i which is a p^m-tuple with 1's at locations $2i$, $2i + 4t$, $2i + 8t$, $2i + 12t$, and $2i + 16t$. Let $\mathbb{G}_i$ be a $(20t + 1) \times (20t + 1)$ square circulant matrix obtain by shifting $\mathbb{v}_i$ downward cyclically $20t + 1$ times (including the zero shift). All the columns (or rows) of $\mathbb{G}_i$ are different and are the incidence vectors of $20t + 1$ different blocks. Then, $\mathbb{H}$ can be put into the following circulant form:

$$\mathbb{H} = [\mathbb{G}_0 \ \mathbb{G}_1 \ \cdots \ \mathbb{G}_{t-1}]. \tag{17.90}$$

With $\mathbb{H}$ in circulant form, the null space of $\mathbb{H}$ gives a quasi-cyclic BIBD-LDPC code of length $n = t(20t+1)$. For $1 \leq k \leq t$, we can choose k circulants, $\mathbb{G}_0, \mathbb{G}_1, \cdots, \mathbb{G}_{k-1}$, to form the following matrix:

$$\mathbb{H}(k) = [\mathbb{G}_0 \ \mathbb{G}_1 \ \cdots \ \mathbb{G}_{k-1}] \tag{17.91}$$

with column and row weights 5 and $5k$, respectively. Then, the null space of $\mathbb{H}(k)$ gives a quasi-cyclic LDPC code of length $n = k(20t + 1)$.

Suppose we decompose each circulant $\mathbb{G}_i$ in $\mathbb{H}(k)$ into five $(20t + 1) \times (20t + 1)$ circulant permutation matrices by row decomposition as presented in Section 15.12,

$$\mathbb{D}_i = \begin{bmatrix} \mathbb{G}_{1,i} \\ \mathbb{G}_{2,i} \\ \mathbb{G}_{3,i} \\ \mathbb{G}_{4,i} \\ \mathbb{G}_{5,i} \end{bmatrix}. \tag{17.92}$$

Replacing each circulant $\mathbb{G}_i$ in $\mathbb{H}(k)$ with its row decomposition $\mathbb{D}_i$, we obtain the following $5 \times k$ array of $(20t + 1) \times (2t + 1)$ permutation matrices:

$$\mathbb{D} = \begin{bmatrix} \mathbb{G}_{1,0} & \mathbb{G}_{1,1} & \cdots & \mathbb{G}_{1,k-1} \\ \mathbb{G}_{2,0} & \mathbb{G}_{2,1} & \cdots & \mathbb{G}_{2,k-1} \\ \mathbb{G}_{3,0} & \mathbb{G}_{3,1} & \cdots & \mathbb{G}_{3,k-1} \\ \mathbb{G}_{4,0} & \mathbb{G}_{4,1} & \cdots & \mathbb{G}_{4,k-1} \\ \mathbb{G}_{5,0} & \mathbb{G}_{5,1} & \cdots & \mathbb{G}_{5,k-1} \end{bmatrix}. \tag{17.93}$$

The null space of $\mathbb{D}$ gives a quasi-cyclic LDPC code with a minimum distance of at least 6 and rate of at least $(k - 5)/k$. If we remove one row of permutation matrices from $\mathbb{D}$, the resultant submatrix of $\mathbb{D}$ generates a quasi-cyclic code also with a minimum distance of at least 6 but rate of at least $(k - 4)/k$. If we move two rows from $\mathbb{D}$, then the resultant submatrix generates a quasi-cyclic LDPC code with a minimum distance of at least 4 and rate of at least $(k - 3)/k$.

EXAMPLE 17.39

Let $t = 21$. Then, $20t + 1 = 421$, which is a prime. The field $GF(421)$ does have a primitive element α that satisfies the condition $\alpha^{4t} - 1 = \alpha^c$, with c as a positive odd integer less than 421 (see Problem 17.26). Therefore, there is a BIBD for a set X of 421 objects. This BIBD consists of $n = 8841$ blocks, each block consists of $\gamma = 5$ objects, each object appears in exactly 105 blocks, and $\lambda = 1$. The 21 base blocks are

$$B_i = \{\alpha^{2i}, \alpha^{2i+84}, \alpha^{2i+168}, \alpha^{2i+252}, \alpha^{2i+336}\}$$

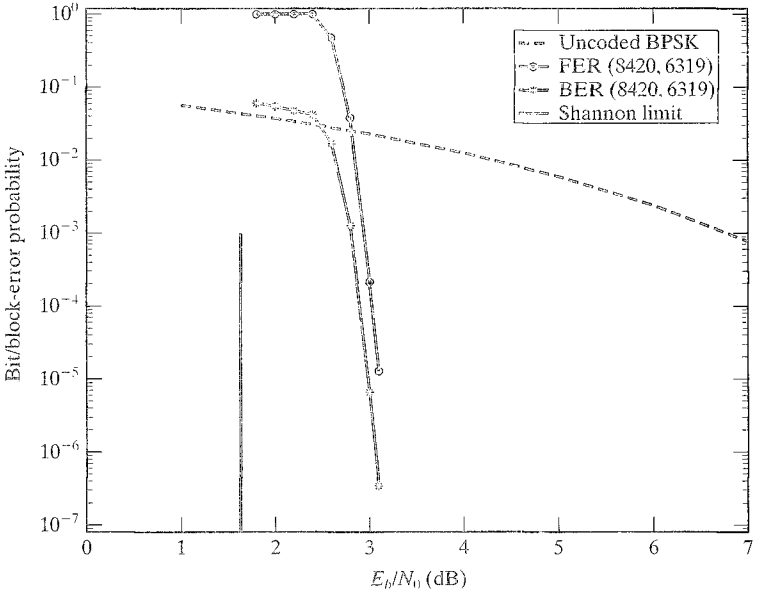


FIGURE 17.44: The error performance of the (8420, 6319) quasi-cyclic BIBD-LDPC code given in Example 17.39.

for $i = 0$ to 20. Based on these 21 base blocks, we can form the incidence matrix $\mathbb{H}$ for the BIBD in the following circulant forms:

$$\mathbb{H} = [\mathbb{G}_0, \mathbb{G}_1, \cdots, \mathbb{G}_{20}],$$

where $\mathbb{G}_i$ is a 421×421 circulant. Suppose we choose $k = 20$ and decompose each $\mathbb{G}_i$ into five circulant permutation matrices by row decomposition. We obtain a 5×20 array $\mathbb{D}$ of 421×421 circulant permutation matrices, which is a 2105×8420 matrix with column and row weights 5 and 20, respectively. The null space of $\mathbb{D}$ gives a (8420, 6319) quasi-cyclic BIBD-LDPC code with rate 0.7504 and a minimum distance of at least 6. The error performance of this code is shown in Figure 17.44. At a BER of 10^{-6}, it performs 1.4 dB from the Shannon limit.

17.18 CONSTRUCTION OF LDPC CODES BASED ON SHORTENED RS CODES WITH TWO INFORMATION SYMBOLS

In earlier sections of this chapter we used several branches of combinatorial mathematics as tools for constructing LDPC codes. In this section we present an algebraic method for constructing LDPC codes based on shortened RS codes with two information symbols [66]. This method gives a class of LDPC codes in Gallager's original form.

Let α be a primitive element of the Galois field $GF(q)$ where $q = p^s$ is a power of a prime. Let ρ be a positive integer such that $2 \leq \rho < q$. Then, the generator polynomial of the cyclic $(q-1, q-\rho+1, \rho-1)$ RS code C over $GF(q)$ of length $q-1$, dimension $q-\rho+1$, and minimum distance $\rho-1$ is given by (7.2),

$$\begin{aligned} \mathbb{g}(X) &= (X-\alpha)(X-\alpha^2)\cdots(X-\alpha^{\rho-2}) \\ &= g_0 + g_1 X + g_2 X^2 + \cdots + X^{\rho-2}, \end{aligned}$$

where $g_i \in GF(q)$. The generator polynomial $\mathbb{g}(X)$ is a minimum-weight code polynomial in C, and hence all its $\rho-1$ coefficients are nonzero.

Suppose we shorten C by deleting the first $q-\rho-1$ information symbols from each codeword of C. Then, we obtain a $(\rho, 2, \rho-1)$ shortened RS code C_b with only two information symbols. A generator matrix of this shortened code is given by

$$\mathbb{G}_b = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & 1 & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & 1 \end{bmatrix}.$$

All the linear combinations of the two rows of $\mathbb{G}_b$ over $GF(q)$ give all the q^2 codewords of C_b. The nonzero codewords of C_b have two different weights, $\rho-1$ and ρ. Because the minimum distance of C_b is $\rho-1$, two codewords in C_b have at most one location with the same code symbol; that is, they agree at most at one location. Let $\mathbf{v}$ be a nonzero codeword in C_b with weight ρ. Then, the set

$$C_b^{(1)} = \{c\mathbf{v} : c \in GF(q)\}$$

of q codewords in C_b forms a one-dimensional subcode of C_b with minimum distance ρ. Two codewords in $C_b^{(1)}$ differ at every location. We partition C_b into q

cosets, $C_b^{(1)}, C_b^{(2)}, \cdots, C_b^{(q)}$, based on the subcode $C_b^{(1)}$. Then, two codewords in any coset $C_b^{(i)}$ must differ in all ρ locations, and two codewords in two different cosets, $C_b^{(i)}$ and $C_b^{(j)}$, differ at least at $\rho - 1$ locations. We arrange the q codewords of a coset $C_b^{(i)}$ as a $q \times \rho$ array $\mathbb{M}_i$. Then, all the q entries of any column of $\mathbb{M}_i$ are different (they are q different elements of $GF(q)$).

Let $\mathbf{z} = (z_\infty, z_0, z_1, \cdots, z_{q-2})$ be a q-tuple over $GF(2)$ whose components correspond to the q elements, α^∞, α^0, α^1, $\alpha^2, \cdots, \alpha^{q-2}$, of $GF(q)$; that is, z_i corresponds to the element α^i of $GF(q)$. We call α^i the *location number* of z_i. For $i = \infty, 0, 1, \cdots, q-2$, we define the *location vector* of the field element α^i as a q-tuple over $GF(2)$,

$$\mathbf{z}(\alpha^i) = (0, 0, \cdots, 0, 1, 0, 0, \cdots, 0), \tag{17.94}$$

for which the ith component of z_i is equal to 1, and all the other components are equal to zero. It follows from the definition that the 1-components of the location vectors of two different elements in $GF(q)$ are at two different locations. Suppose we arrange the location vectors of the q elements of $GF(q)$ as the rows of a matrix $\mathbb{A}$. Then, $\mathbb{A}$ is a $q \times q$ permutation matrix.

Let $\mathbf{v} = (v_1, v_2, \cdots, v_\rho)$ be a codeword in C_b. For $1 \le j \le \rho$, replacing each component v_j of $\mathbf{v}$ with its location vector $\mathbf{z}(v_j)$, we obtain a ρq-tuple over $GF(2)$,

$$\mathbf{z}(\mathbf{v}) = (\mathbf{z}(v_1), \mathbf{z}(v_2), \cdots, \mathbf{z}(v_\rho)). \tag{17.95}$$

Because the weight of each location vector $\mathbf{z}(v_i)$ is 1, the weight of $\mathbf{z}(\mathbf{v})$ is ρ. This ρq-tuple is called the *symbol location vector* of $\mathbf{v}$. Because any two codewords in C_b have at most one location with the same symbol, their symbol location vectors consequently have at most one 1-component in common. Let

$$\mathbb{Z}(C_b^{(i)}) = \{\mathbf{z}(\mathbf{v}) : \mathbf{v} \in C_b^{(i)}\} \tag{17.96}$$

be the set of symbol location vectors of the q codewords in the ith coset $C_b^{(i)}$ of $C_b^{(1)}$. Then, two symbol location vectors in $\mathbb{Z}(C_b^{(i)})$ do not have any 1-component in common, and two symbol location vectors from two different sets, $\mathbb{Z}(C_b^{(i)})$ and $\mathbb{Z}(C_b^{(j)})$, have at most one 1-component in common.

For $1 \le i \le q$, we form a $q \times \rho q$ matrix $\mathbb{H}_i$ over $GF(2)$ whose rows are the q symbol location vectors in $\mathbb{Z}(C_b^{(i)})$. Because the weight of each vector in $\mathbb{Z}(C_b^{(i)})$ is ρ, the total number of 1-entries in $\mathbb{H}_i$ is ρq. Since no two vectors in $\mathbb{Z}(C_b^{(i)})$ have any 1-component in common, the weight of each column of $\mathbb{H}_i$ is 1. Therefore, $\mathbb{H}_i$ has constant row weight ρ and constant column weight 1. It follows from the structural properties of the codewords in $C_b^{(i)}$ and their symbol location vectors that $\mathbb{H}_i$ consists of a row of ρ $q \times q$ permutation matrices,

$$\mathbb{H}_i = [\mathbb{A}_{i,1}\ \mathbb{A}_{i,2}\ \cdots\ \mathbb{A}_{i,\rho}]. \tag{17.97}$$

Matrix $\mathbb{H}_i$ is called the *symbol location matrix* of the coset $C_b^{(i)}$. The class of symbol location matrices,

$$\mathcal{H} = \{\mathbb{H}_1, \mathbb{H}_2, \cdots, \mathbb{H}_q\}, \tag{17.98}$$

has the following structural properties: (1) no two rows in the same matrix $\mathbf{H}_i$ have any 1-component in common; and (2) no two rows from two different matrices, $\mathbf{H}_i$ and $\mathbf{H}_j$, have more than one 1-component in common.

EXAMPLE 17.40

Consider the Galois field $GF(2^2)$ constructed based on the primitive polynomial $p(X) = 1 + X + X^2$. Let α be a primitive element of $GF(2^2)$. Then, the four elements of $GF(2^2)$ are $0 = \alpha^\infty$, $1 = \alpha^0$, α, and $\alpha^2 = 1 + \alpha$. The location vectors of these four field elements are

$$\mathbf{z}_0 = (1\,0\,0\,0), \qquad \mathbf{z}_1 = (0\,1\,0\,0), \qquad \mathbf{z}_\alpha = (0\,0\,1\,0), \qquad \mathbf{z}_{\alpha^2} = (0\,0\,0\,1).$$

Let $\rho = 3$. The cyclic (3, 2, 2) RS code C_b over $GF(2^2)$ has generator polynomial $\mathbf{g}(X) = X + \alpha$ and generator matrix

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

The code has 16 codewords, and its minimum distance is 2. Adding the two rows of $\mathbf{G}$, we obtain a codeword $\mathbf{v} = (\alpha, \alpha^2, 1)$ with weight 3. The following set of four codewords:

$$\{\beta\mathbf{v} : \beta \in GF(2^2)\} = \{(0, 0, 0), (\alpha, \alpha^2, 1), (\alpha^2, 1, \alpha), (1, \alpha, \alpha^2)\}$$

forms a one-dimensional subcode $C_b^{(1)}$ of C_b with a minimum weight of 3. We partition C_b with respect to $C_b^{(1)}$ and obtain the following four cosets:

$$C_b^{(1)} = \{(0, 0, 0), (\alpha, \alpha^2, 1), (\alpha^2, 1, \alpha), (1, \alpha, \alpha^2)\},$$
$$C_b^{(2)} = \{(\alpha, 1, 0), (0, \alpha, 1), (1, 0, \alpha), (\alpha^2, \alpha^2, \alpha^2)\},$$
$$C_b^{(3)} = (\alpha^2, \alpha, 0), (1, 1, 1), (0, \alpha^2, \alpha), (\alpha, 0, \alpha^2),$$
$$C_b^{(4)} = (1, \alpha^2, 0), (\alpha^2, 0, 1), (\alpha, \alpha, \alpha), (0, 1, \alpha^2).$$

The symbol location matrices of these four cosets are

$$\mathbf{H}_1 = \mathbb{Z}(C_b^{(1)}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{H}_2 = \mathbb{Z}(C_b^{(2)}) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{H}_3 = \mathbb{Z}(C_b^{(3)}) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbb{H}_4 = \mathbb{Z}(C_b^{(4)}) = \left[\begin{array}{cccc|cccc|cccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Let $\gamma = 3$. We form the following parity-check matrix:

$$\mathbb{H}_{GA}(3) = \left[\begin{array}{c} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \mathbb{H}_3 \end{array}\right] = \left[\begin{array}{cccc|cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

The null space $\mathbb{H}_{GA}(3)$ gives a regular (12, 4) RS-based Gallager-LDPC code with rate 1/3 and a minimum distance of 6. The lower bound $\gamma + 1$ on the minimum distance is 4.

Let γ be a positive integer such that $1 \leq \gamma \leq q$. We form the following $\gamma q \times \rho q$ matrix over $GF(2)$:

$$\mathbb{H}_{GA}(\gamma) = \left[\begin{array}{c} \mathbb{H}_1 \\ \mathbb{H}_2 \\ \vdots \\ \mathbb{H}_\gamma \end{array}\right] = \left[\begin{array}{cccc} \mathbb{A}_{1,1} & \mathbb{A}_{1,2} & \cdots & \mathbb{A}_{1,\rho} \\ \mathbb{A}_{2,1} & \mathbb{A}_{2,2} & \cdots & \mathbb{A}_{2,\rho} \\ & \cdots & & \cdots \\ \mathbb{A}_{\gamma,1} & \mathbb{A}_{\gamma,2} & \cdots & \mathbb{A}_{\gamma,\rho} \end{array}\right], \tag{17.99}$$

where each submatrix $\mathbb{A}_{i,j}$ is a $q \times q$ permutation matrix. Therefore, $\mathbb{H}_{GA}(\gamma)$ consists of a $\gamma \times \rho$ array of permutation matrices. It is a (γ, ρ)-regular matrix with column and row weights γ and ρ, respectively. No two rows (or two columns) of $\mathbb{H}_{GA}(\gamma)$ have more than one 1-component in common, and its density is $1/q$, which is small for large q. Hence, $\mathbb{H}_{GA}(\gamma)$ is a sparse matrix that has all the structural properties of the parity-check matrix of a regular LDPC code given in Definition 17.1. Furthermore, it is exactly in Gallager's original form given by (17.1). Therefore, the null space of $\mathbb{H}_{GA}(\gamma)$ gives an LDPC code $C_{GA}(\gamma)$ of Gallager's type of length $n = \rho q$ with a minimum distance of at least $\gamma + 1$ for odd γ and $\gamma + 2$ for even γ. The rate of this code is at least $(\rho - \gamma)/\rho$.

For any choice of q $(= p^s)$ and γ, we can construct a sequence of Gallager-LDPC codes of various lengths and rates with $\rho = \gamma, \gamma + 1, \cdots, q - 1$. For any choice of q and ρ, we can construct a sequence of Gallager-LDPC codes of length $n = \rho q$ with various rates and minimum distances for $\gamma = 1, 2, \cdots, \rho$. For $\rho = q - 1$, $C_{GA}(\gamma)$ is quasi-cyclic. Because the construction is based on the $(\rho, 2, \rho - 1)$ shortened RS code

C_b over the field $GF(q)$, we call C_b and $GF(q)$ the *base code* and the *construction field*, respectively.

EXAMPLE 17.41

Let $GF(2^6)$ be the construction field. Suppose we choose $\rho = 32$. Then, the base code is the (32, 2, 31) shortened RS code over $GF(2^6)$. The location vector of each element of $GF(2^6)$ is a 64-tuple with a single 1-component. Suppose we choose $\gamma = 6$. Then, the RS-based Gallager-LDPC code $C_{GA}(6)$ constructed is a (6, 32)-regular (2048, 1723) LDPC code with rate 0.841 and a minimum distance of at least 8. Its error performance with SPA decoding is shown in Figure 17.45. At a BER of 10^{-6}, the code performs 1.55 dB from the Shannon limit and achieves a 6-dB coding gain over the uncoded BPSK. If we choose $\rho = 63$, the base code is then the (63, 2, 62) shortened RS code. We set $\gamma = 60$. The RS-based Gallager-LDPC code is a (60, 63)-regular (4032, 3307) quasi-cyclic code with rate 0.82 and a minimum distance of at least 62. The error performance of this code with SPA decoding is shown in Figure 17.46. At a BER of 10^{-6}, it performs 1.65 dB from the Shannon limit. Owing to its large minimum distance, no error floor is expected. This code can also be decoded with one-step majority-logic decoding to correct 30 or fewer random errors. If it is decoded with weighted BF decoding, an effective trade-off between error performance and decoding complexity can be achieved.

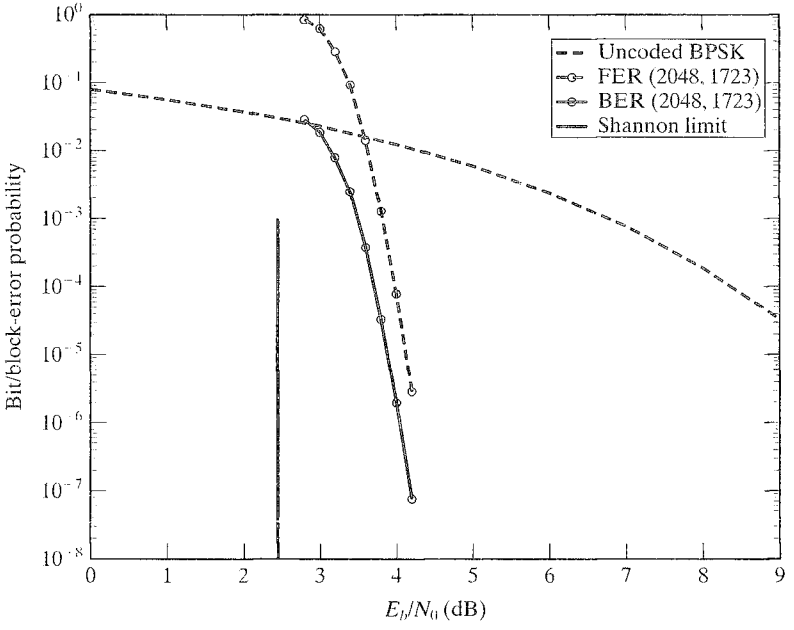


FIGURE 17.45: Error performance of the (2048, 1723) RS-based Gallager (6, 32)-regular LDPC code with construction field $GF(2^6)$ given in Example 17.41.

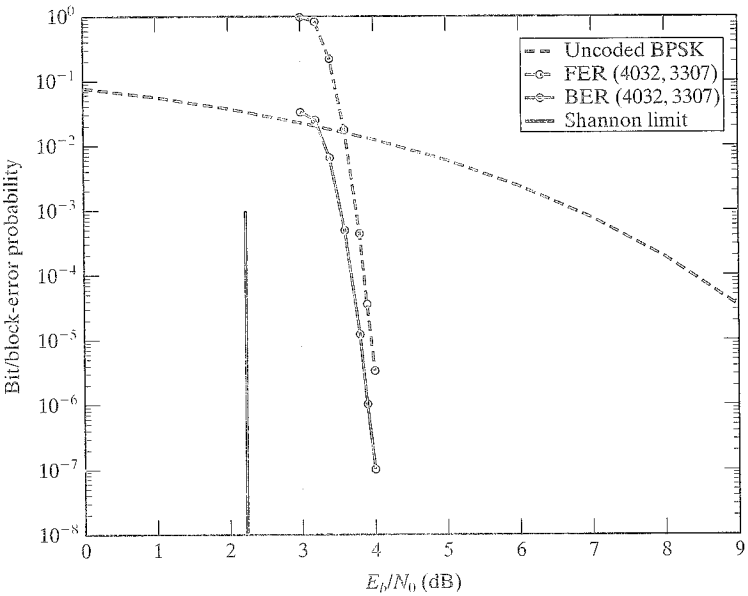


FIGURE 17.46: Error performance of the (4032, 3307) RS-based Gallager (60, 63)-regular quasi-cyclic LDPC code with construction field $GF(2^6)$ given in Example 17.41.

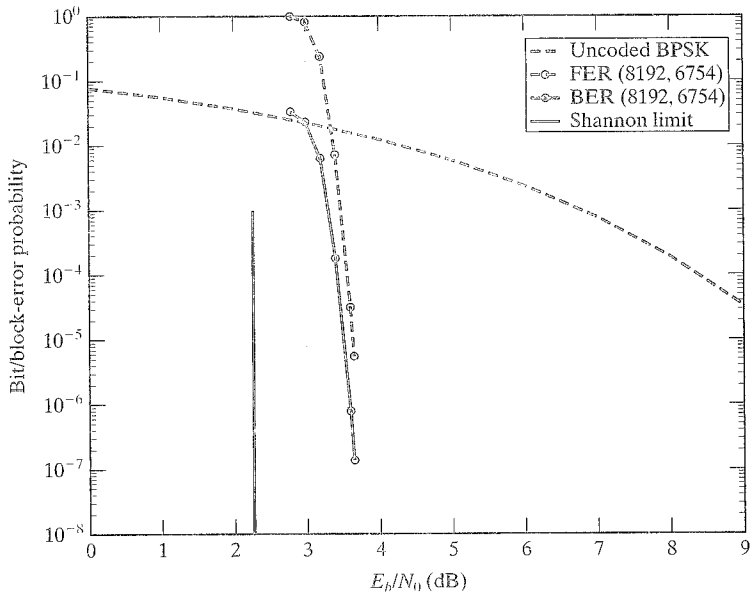


FIGURE 17.47: Error performance of the (8192, 6754) RS-based Gallager (6, 32)-regular LDPC code with construction field $GF(2^8)$ given in Example 17.42.

EXAMPLE 17.42

Suppose we use $GF(2^8)$ as the construction field and the (32, 2, 31) shortened RS code over $GF(2^8)$ as the base code. We set $\gamma = 6$. Then, the RS-based Gallager-LDPC code is a (8192, 6754) code with rate 0.824 and a minimum distance of at least 8. Its error performance with SPA decoding is shown in Figure 17.47. At a BER of 10^{-6}, it performs 1.25 dB from the Shannon limit and achieves a 6.7 dB gain over the uncoded BPSK.

The foregoing algebraic construction of LDPC codes is simple and yet very powerful. It gives a large class of regular LDPC codes. Codes in this class can be decoded with the SPA, weighted BF, BF, or one-step majority-logic decodings to provide a wide range of trade-offs between error performance and decoding complexity.

The matrix given by (17.99) is an array of permutation matrices that is in exactly the same form as the matrix given by (17.72). Hence, it can be masked to generate new LDPC codes.

17.19 CONCATENATIONS WITH LDPC AND TURBO CODES

In most applications of concatenated coding for error control, RS codes are used as the outer codes, and they are, in general, decoded with an algebraic decoding algorithm, such as the Berlekamp or Euclidean algorithm presented in Chapter 7. If a

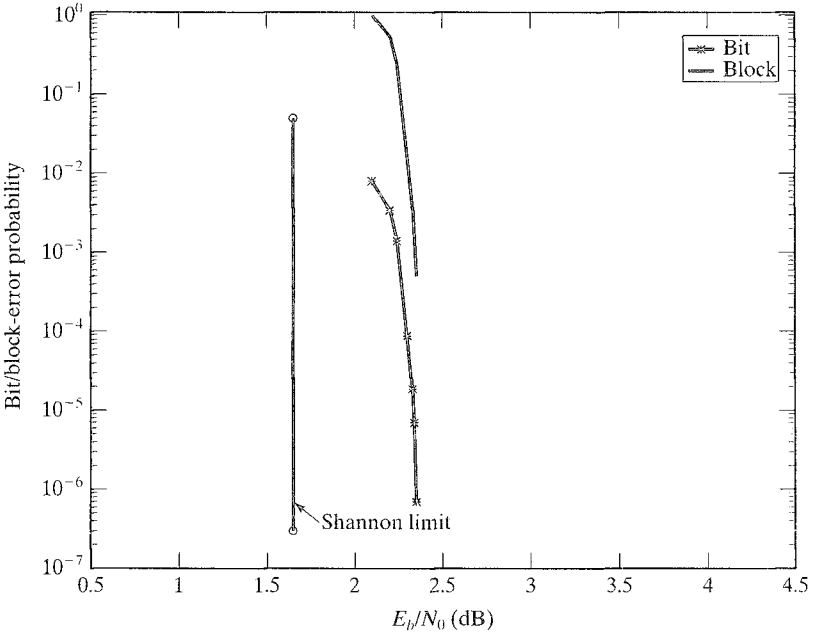


FIGURE 17.48: Bit- and block-error performance of a concatenated LDPC-turbo coding system with a turbo inner code and an extended EG-LDPC outer code.

significant improvement in error performance of a concatenated coding system with an RS code as the outer code is to be achieved, the RS outer code must be reasonably long and decoded with a sophisticated soft-decision decoding scheme that provides either optimal MLD performance or a suboptimal error performance. Unfortunately, the complexity of such a soft-decision decoding scheme or algorithm would be enormously large, and the decoder would be practically impossible to implement.

In this chapter we have shown that long high-rate finite-geometry LDPC codes with large minimum distances can be easily constructed and can be practically decoded with SPA decoding. They achieve very good error performance, especially the block-error performance. If such an LDPC code is used as the outer code in a concatenated coding system with a simple turbo code as the inner code, extremely good error performance and large coding gain should be achievable with practical implementation. With an LDPC code as the outer code and decoded with SPA decoding, the soft-output information provided by the inner turbo decoder can be fully utilized. We give an example to demonstrate the strength of such a combination of two powerful coding systems.

Consider a concatenated coding system in which the inner code is a high-rate block turbo code with the (64, 57) distance-4 Hamming code as the two constituent codes, and the outer code is the (65520, 61425) extended EG-LDPC code given in Example 17.14 [17]. The overall rate of this system is 0.75. The bit- and block-error performances of this concatenated LDPC-turbo coding system are shown in Figure 17.48. We see that this system achieves extremely good waterfall error performance. To achieve a BER of 10^{-6}, it requires an SNR of 2.35 dB, and at this BER, it performs only 0.7 dB away from the Shannon limit. This system is far superior to the concatenated coding scheme used in the NASA TDRS System presented in Section 15.6, whose overall rate is only 0.437.

Another form of concatenation of LDPC and turbo codes is to use an LDPC code as the two constituent codes in a parallel turbo coding arrangement, as described in Chapter 16. Because decoding of an LDPC code with the SPA is not trellis-based, a long LDPC code with large distance can be used as the constituent codes to achieve very good error performance without an error floor (or with an error floor at a very low error rate).

PROBLEMS

17.1 Does the following matrix satisfy the conditions of the parity-check matrix of an LDPC code given by Definition 17.1? Determine the rank of this matrix and give the codewords of its null space.

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

17.2 Form the transpose $\mathbf{H}^T$ of the parity-check matrix $\mathbf{H}$ given in Problem 17.1. Is $\mathbf{H}^T$ a low-density parity-check matrix? Determine the rank of $\mathbf{H}^T$ and construct the code given by the null space of $\mathbf{H}^T$.

17.3 Prove that the $(n, 1)$ repetition code is an LDPC code. Construct a low-density parity-check matrix for this code.

17.4 Consider the matrix $\mathbf{H}$ whose columns are all the m-tuples of weight 2. Does $\mathbf{H}$ satisfy the conditions of the parity-check matrix of an LDPC code? Determine the rank of $\mathbf{H}$ and its null space.

17.5 The following matrix is a low-density parity-check matrix. Determine the LDPC code given by the null space of this matrix. What is the minimum distance of this code?

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

17.6 Prove that the maximum-length code of length $2^m - 1$ presented in Section 8.3 is an LDPC code.

17.7 Construct the Tanner graph of the code given in Problem 17.1. Is the Tanner graph of this code acyclic? Justify your answer.

17.8 Construct the Tanner graph of the code given in Problem 17.2. Is the Tanner graph of this code acyclic? Justify your answer.

17.9 Construct the Tanner graph of the code given by the null space of the parity-check matrix given in Problem 17.5. Does the Tanner graph of this code contains cycles of length 6? Determine the number of cycles of length 6 in the graph.

17.10 Determine the orthogonal check-sums for every code bit of the LDPC code given by the null space of the parity-check matrix of Problem 17.5.

17.11 Prove that the minimum distance of the Gallager-LDPC code given in Example 17.2 is 6.

17.12 Determine the generator polynomial of the two-dimensional type-I $(0, 3)$th-order cyclic EG-LDPC code constructed based on the two-dimensional Euclidean geometry $\mathrm{EG}(2, 2^3)$.

17.13 Determine the parameters of the parity-check matrix of the three-dimensional type-I (0,2)th-order cyclic EG-LDPC code $C_{EG,c}^{(1)}(3, 0, 2)$. Determine the generator polynomial of this code. What are the parameters of this code?

17.14 Determine the parameters of the companion code of the EG-LDPC code given in Problem 17.13.

17.15 Decode the two-dimensional type-I $(0, 3)$th-order cyclic EG-LDPC code with one-step majority-logic decoding and give the bit- and block-error performance for the AWGN channel with BPSK signaling.

17.16 Repeat Problem 17.15 with BF decoding.

17.17 Repeat Problem 17.15 with weighted majority-logic decoding.

17.18 Repeat Problem 17.15 with weighted BF decoding.

17.19 Repeat Problem 17.15 with SPA decoding.

17.20 Decode the three-dimensional type-II $(0, 2)$th-order quasi-cyclic EG-LDPC code given in Problem 17.14 with SPA decoding, and give the bit- and block-error performance of the code for the AWGN channel with BPSK signaling.

17.21 Consider the parity-check matrix $\mathbf{H}_{EG,c}^{(1)}$ of the three-dimensional type-I $(0, 2)$th-order cyclic EG-LDPC code given in Problem 17.13. Split each column of this

parity-check matrix into five columns with rotating weight distribution. The result is a new low-density parity-check matrix that gives an extended EG-LDPC code. Decode this code with SPA decoding and give its bit- and block-error performances.

17.22 Construct a parity-check matrix of the Gallager-LDPC code with the following parameters: $m = 6$, $\rho = 4$, and $\gamma = 3$. Choose column permutations for the submatrices such that λ is no greater than 1.

17.23 Prove that the Tanner graph of a finite-geometry LDPC code contains cycles of length 6. Enumerate the number of cycles of length 6.

17.24 Prove that the minimum distance of an EG-Gallager LDPC code must be even. Use the result to prove the lower bound on minimum distance given by (17.68).

17.25 Construct an EG-Gallager LDPC code using six parallel bundles of lines in the two-dimensional Euclidean geometry EG(2,2^5) over $GF(2^5)$. Compute its bit- and block-error performances with SPA decoding.

17.26 Construct a masked EG-Gallager LDPC code of length 1024 by decomposing the incidence matrices of eight parallel bundles of lines in EG(2, 2^5) into 32×32 permutation matrices. To construct such a code, set $\rho = 32$, and form an 8×32 masking matrix with column and row weights 4 and 16, respectively, using four primitive 8-tuples over $GF(2)$. Compute the bit- and block-error performances using SPA decoding.

17.27 The incidence vectors of the lines in EG(2, 2^5) not passing through the origin form a single 1023×1023 circulant G with weight 32. Construct a rate-1/2 quasi-cyclic code of length 8184 by decomposing G into a 4×8 array of 1023×1023 circulant permutation matrices. Compute the bit- and block-error performance of the code with SPA decoding.

17.28 Prove that there exist a primitive element α in $GF(241)$ and an odd positive integer c less than 241 such that $\alpha^{64} + 1 = \alpha^c$.

17.29 Design a concatenated turbo coding system with a finite-geometry LDPC code of your choice as the outer code. Construct the inner turbo code by using the second-order (32, 16) RM code as the component code. Give the bit-error performance of your designed system.

BIBLIOGRAPHY

1. R. G. Gallager, "Low Density Parity Check Codes," *IRE Trans. Inform. Theory*, IT-8: 21–28, January 1962.

2. R. G. Gallager, *Low Density Parity Check Codes*, MIT Press, Cambridge, 1963.

3. R. M. Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Inform. Theory*, IT-27: 533–47, September 1981.

4. D. J. C. MacKay and R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electron. Lett.*, 32 (18): 1645–46, 1996.

5. M. Sipser and D. Spielman, "Expander Codes," *IEEE Trans. Inform. Theory*, 42 (6): 1710–22, November 1996.

6. D. Spielman, "Linear-Time Encodable Error-Correcting Codes," *IEEE Trans. Inform. Theory*, 42 (6): 1723–31, November 1996.

7. M. C. Davey and D. J. C. MacKay, "Low Density Parity Check Codes over $GF(q)$," *IEEE Commun. Lett.*, 2(6), 165–67, June 1998.

8. M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation," *Proc. 1998 IEEE Intl. Symp. Inform. Theory*, p. 171, Cambridge, August 16–21, 1998.

9. D. J. C. Mackay, "Gallager Codes That Are Better Than Turbo Codes," *Proc. 36th Allerton Conf. Commun., Control, and Computing*, Monticello, Ill., September 1998.

10. ———, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inform. Theory*, 45 (2): 399–432, March 1999.

11. ———, "Sparse Graph Codes," *Proc. 5th Intl. Symp. Commun. Theory and Applications*, pp. 2–4, Ambleside, UK, July 11–16, 1999.

12. T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of Capacity-Approaching Irregular Codes," *IEEE Trans. Inform. Theory*, 47 (2): 619–37, February 2001.

13. T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding," *IEEE Trans. Inform. Theory*, 47: 599–618, February 2001.

14. S. Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke, "On the Design of Low Density Parity Check Codes within 0.0045 dB of the Shannon Limit," *IEEE Commun. Lett.*, 5: 58–60, February 2001.

15. Y. Kou, S. Lin, and M. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery," *Proc. IEEE Intl. Symp. Inform. Theory*, Sorrento, Italy, June 25–30, 2000.

16. S. Lin and Y. Kou, "A Geometric Approach to the Construction of Low Density Parity Check Codes," presented at the IEEE 29th Communication Theory Workshop, Haines City, Fla., May 7–10, 2000.

17. Y. Kou, S. Lin, and M. Fossorier, "Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery and More," *IEEE Trans. Inform. Theory*, 47 (6): 2711–36, November 2001.

18. Y. Kou, S. Lin, and M. Fossorier, "Construction of Low Density Parity Check Codes—A Geometric Approach," *Proc. 2nd Intl. Symp. Turbo Codes and Related Topics*, Brest, France, September 4–7, 2000.

19. S. Lin, Y. Kou, and M. Fossorier, "Low Density Parity Check Codes Construction Based on Finite Geometries," *Proc. GLOBECOM 2000*, San Francisco, Calif., November 27–December 1, 2000.

20. S. Lin, Y. Kou, and M. Fossorier, "Finite Geometry Low Density Parity Check Codes: Construction, Structure and Decoding," *Proceedings of the ForneyFest*, Kluwer Academic, Boston, Mass., 2000.

21. A. Ventura and M. Chiani, "Design and Evaluation of Some High-Rate Low-Density Parity-Check Codes," *Proc. 2001 IEEE GlobeCom*, vol. 2, pp. 990–94, Nov. 25–29, 2001.

22. M. Yang, Y. Li, and W. E. Ryan, "Design of Efficiently Encodable Moderate-Length High-Rate Irregular LDPC Codes," to appear in *IEEE Trans. Commun.*, 2004.

23. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, Calif., 1988.

24. S. L. Lauritzen and D. J. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems," *J. R. Stat. Soc. B*, 50: 157–224, 1988.

25. N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and Iterative Decoding on General Graphs," *Eur. Trans. Telecommun.*, 6: 513–26, 1955.

26. R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo Decoding As an Instance of Pearl's Belief Propagation Algorithm," *IEEE J. Select. Areas Commun.*, 16: 140–52, February 1998.

27. F. R. Kschischang and B. J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models," *IEEE J. Select. Areas Commun.*, 16 (12): 219–30, February 1998.

28. F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inform. Theory*, 47: 498–519, February 2001.

29. M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced Complexity Iterative Decoding of Low Density Parity Check Codes," *IEEE Trans. Commun.*, 47: 673–80, May 1999.

30. R. Lucas, M. Fossorier, Y. Kou, and S. Lin, "Iterative Decoding of One-Step Majority Logic Decodable Codes Based on Belief Propagation," *IEEE Trans. Commun.*, 48 (6): 931–37, June 2000.

31. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," *Proc. 1993 IEEE Intl. Conf. Commun.*, pp. 1064–70, Geneva, Switzerland, May 1993.

32. C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes," *IEEE Trans. Commun.*, 44: 1261–71, October 1996.

33. S. Benedetto and G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Trans. Inform. Theory*, 42 (2): 409–28, March 1996.

34. J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Inform. Theory*, 42: 429–45, March 1996.

35. N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, N.J., 1974.

36. S. M. Aji and R. J. McEliece, "The Generalized Distributive Law," *IEEE Trans. Inform. Theory,* 46 (2): 325–43, March 2000.

37. N. Wiberg, "Codes and Decoding on General Graphs," Ph.D. Diss., Dept. of Electrical Engineering, University of Linköping, Linköping, Sweden, April 1996

38. T. Etzion, A. Trachtenberg, and A. Vardy, "Which Codes Have Cycle-Free Tanner Graphs," *IEEE Trans. Inform. Theory*, 45 (6): 2173–81, September 1999.

39. G. D. Forney, Jr., "Codes on Graphs: A Survey for Algebraists," *Proc. 13th Intl. Symp. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes,* Honolulu, November 15–19, 1999.

40. R. D. Carmichael, *Introduction to the Theory of Groups of Finite Order,* Dover, New York, 1956.

41. H. B. Mann, *Analysis and Design of Experiments,* Dover, New York, 1949.

42. T. Kasami, S. Lin, and W. W. Peterson, "Polynomial Codes," *IEEE Trans. Inform. Theory,* 14: 807–14, 1968.

43. R. C. Bose and D. J. Ray-Chaudhuri, "On a Class of Error Correcting Binary Group Codes," *Inform. Control*, no. 3: 68–79, March 1960.

44. W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes,* 2d ed., MIT Press, Cambridge, 1972.

45. E. R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, New York, 1968.

46. S. Lin, "On the Number of Information Symbols in Polynomial Codes," *IEEE Trans. Inform. Theory,* IT-18: 785–94, November 1972.

47. E. J. Weldon, Jr., "Difference-Set Cyclic Codes," *Bell Syst. Tech. J.*, 45: 1045–55, September 1966.

48. V. D. Kolesnik, "Probability Decoding of Majority Codes," *Probl. Peredachi Inform.,* 7: 3–12, July 1971.

49. R. Lucas, M. Bossert, and M. Breitback, "On Iterative Soft-Decision Decoding of Linear Block Codes and Product Codes," *IEEE J. Select. Areas Commun.,* 16: 276–96, February 1998.

50. S. Lin "Shortened Finite Geometry Codes," *IEEE Trans. Inform. Theory*, IT-18 (5): 692–96, September 1972.

51. H. Tang, J. Xu, Y. Kou, S. Lin, and K. Abdel-Ghaffar, "On Algebraic Construction of Low Density Parity Check Codes," *Proc. 2002 IEEE Intl. Symp. Inform. Theory,* p. 482, Lausanne, Switzerland, June 30–July 5, 2002.

52. H. Tang, J. Xu, Y. Kou, S. Lin, and K. Abdel-Ghaffar, "On Algebraic Construction of Gallager and Circulant Low Density Parity Check Codes," to appear in *IEEE Trans. Inform. Theory*, June 2004.

53. J. Xu, L. Chen, I. Djurdjevic, S. Lin, and K. Abdel-Ghaffar, "Construction of Regular and Irregular Low Density Parity Check Codes, Based on Geometry Decomposition and Masking," submitted to *IEEE Trans. Inform. Theory*, 2004.

54. S. Liu, L. Chen, J. Xu, and I. Djurdjevic, "Near Shannon Limit Quasi-Cyclic Low-Density Parity-Check Codes," *Proc. IEEE GLOBECOM 2003*, San Francisco, Calif., December 1–5, 2003.

55. L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near Shannon Limit Quasi-Cyclic Low-Density Parity-Check Codes," to appear in *IEEE Trans. Commun.*, July 2004.

56. Y. Kou, J. Xu, H. Tang, S. Lin, and K. Abdel-Ghaffar, "On Circulant Low Density Parity Check Codes," *Proc. IEEE Intl. Symp. Inform. Theory*, p. 200, Lausanne, Switzerland, June 30–July 5, 2002.

57. C. L. Liu, *Introduction to Combinatorial Mathematics*, McGraw-Hill, New York, 1968.

58. C. J. Colbourn and H. J. Dinitz, *The CRC Handbook of Combinatorial Designs*, Boca Raton, Fla., CRC Publications, 1996.

59. I. Djurdjevic, S. Lin, and K. Abdel-Ghaffar, "On a Graph-Theoretic Construction of Low Density Parity Check Codes," *IEEE Commun. Lett.*, 7: 171–73, April 2003.

60. J. Rosenthal and P. O Vontobel, "Construction of Regular and Irregular LDPC Codes Using Ramamujan Graphs and Ideas from Margulis," *Proc. 2001 IEEE Intl. Symp. Inform. Theory*, Washington, D.C., p. 4, June 24–29, 2001.

61. P. O Vontobel and R. M. Tanner, "Construction of Codes Based on Finite Generalized Quadrangles for Iterative Decoding," *Proc. 2001 IEEE Intl. Symp. Inform. Theory*, Washington, D.C., p. 223, June 24–29, 2001.

62. P. O Vontobel and H. A. Loeliger, "Irregular Codes from Regular Graphs," *Proc. 2002 IEEE Intl. Symp. Inform. Theory*, Lausanne, Switzerland, p. 284, June 30–July 4, 2002.

63. R. C. Bose, "On the Construction of Balanced Incomplete Block Designs," *Ann. Eugenics* 9: 353–99, 1939.

64. B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin, "Construction of Low Density Parity Check Codes Based on Balanced Incomplete Block Designs," to appear in *IEEE Trans. Inform. Theory*, June 2004.

65. B. Vasic and O. Milenkovic, "Combinatorial Construction of Low-Density Parity Check Codes for Iterative Decoding," to appear in *IEEE Trans. Inform. Theory*, 2004.

66. I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "Construction of Low-Density Parity-Check Codes Based on Shortened Reed–Solomon Codes with Two Information Symbols," *IEEE Commun. Lett.*, no. 7: 317–19, July 2003.

CHAPTER 18

Trellis-Coded Modulation

All the coding schemes discussed so far have been designed for use with binary-input channels; that is, the encoded bits are represented by one-dimensional BPSK signals according to the mapping $0 \rightarrow -\sqrt{E_s}$ and $1 \rightarrow +\sqrt{E_s}$, or $0 \rightarrow -1$ and $1 \rightarrow +1$ for unit energy signals. (We note here that even nonbinary codes, such as RS codes, are usually transmitted using binary signaling by representing each symbol over $GF(2^m)$ as a binary m-tuple.) In this case the *spectral efficiency* η of the coded system is equal to the code rate R; that is, $\eta = R \leq 1$ bit/dimension or 1 bit/transmitted BPSK symbol, and at most one bit of information is transmitted each time a BPSK symbol is sent over the channel. Thus, since the bandwidth required to transmit a symbol without distortion is inversely proportional to the transmission rate, combining coding with binary modulation always requires *bandwidth expansion* by a factor of $1/R$. In other words, compared with uncoded modulation, the coding gains resulting from binary modulation are achieved at the expense of requiring a larger channel bandwidth.

For the first 25 or so years after the publication of Shannon's paper, research in coding theory concentrated almost exclusively on designing good codes and efficient decoding algorithms for binary-input channels. In fact, it was believed in the early 1970s that coding gain could be achieved only through bandwidth expansion and that coding could serve no useful purpose at spectral efficiencies $\eta \geq 1$ bit/dimension. Thus, in communication applications where bandwidth was limited and large modulation alphabets were needed to achieve high spectral efficiencies, such as data transmission over the dial-up telephone network, coding was not thought to be a viable solution. Indeed, the modulation system design emphasis was almost exclusively on constructing large signal sets in two-dimensional Euclidean space that had the highest possible minimum Euclidean distance between signal points, given certain constraints on average and/or peak signal energy.

In the next two chapters we introduce a combined coding and modulation technique, called *coded modulation*, that achieves significant coding gain without bandwidth expansion. Indeed, coding gain without bandwidth expansion can be achieved independently of the operating spectral efficiency of the modulation system. Thus, coded modulation is referred to as a *bandwidth-efficient* signaling scheme. In this chapter we discuss *trellis-coded modulation* (TCM) [1], a form of coded modulation based on convolutional codes, and in the next chapter we discuss *block-coded modulation* (BCM), based on block codes. Basically, TCM combines ordinary rate $R = k/(k+1)$ binary convolutional codes with an M-ary signal constellation ($M = 2^{k+1} > 2$) in such a way that coding gain is achieved without increasing the rate at which symbols are transmitted, that is, without increasing the required bandwidth, compared with uncoded modulation. For example, a rate $R = 2/3$ convolutional code can be combined with 8-PSK modulation by mapping the three encoder output bits in each T-second time interval into one 8-PSK symbol. This TCM scheme can then be compared with uncoded QPSK modulation, since

they both have the same spectral efficiency of $\eta = 2$ bits/symbol[1] or 1 bit/dimension (both QPSK and 8-PSK are two-dimensional signal sets). The key to the technique is that the redundant bits introduced by coding are not used to send extra symbols, as in binary modulation, but instead they are used to expand the size of the signal constellation relative to an uncoded system. Thus coded modulation involves *signal set expansion* rather than bandwidth expansion.

To fairly compare a coded modulation system with an uncoded system, the expanded (coded) signal constellation must have the same average energy as the uncoded constellation, which implies that the signal points must be closer together in two-dimensional Euclidean space, thus reducing the minimum Euclidean distance between signal points. In a TCM system, if the convolutional codes are chosen according to the usual criterion of maximizing the *minimum free Hamming distance* between codewords, and encoder outputs are mapped to signal points in the expanded constellation independently of the code selection, coding gain is not achieved; however, if the code and signal mapping are designed jointly to maximize the *minimum free Euclidean distance* between signal sequences, coding gain can be achieved without expanding the bandwidth or increasing the average energy of the signal set. This joint design is accomplished using a technique known as *mapping by set partitioning* [1].

The original concept of TCM was introduced in a paper by Ungerboeck and Csajka [2] in June 1976, and the essential elements of the idea were later presented in three papers by Ungerboeck [1, 3, 4]. Other early contributions to the basic development of TCM include those by Massey [5], Anderson and Taylor [6], Forney, Gallager, Lang, Longstaff, and Qureshi [7], Calderbank and Mazo [8], Calderbank and Sloane [9], and Forney [10, 11].

A great deal of work has also been devoted to the design of TCM systems for fading (i.e., bursty) channels. As in the case of BPSK modulation, channel interleaving must be used to ensure that received symbols are affected independently by the fading. Unlike the case with binary modulation, however, new code design rules must also be employed for a TCM system to achieve the best possible performance in fading. These issues are developed in a series of papers by Divsalar and Simon [12, 13, 14] and in [15].

TCM systems also can be included in bandwidth-efficient versions of concatenated coding (see, e.g., [16, 17]) and turbo coding (see, e.g., [18, 19, 20]). For readers wishing to investigate the various aspects of TCM in more detail than is presented here, [21] contains a comprehensive view of the subject up to 1990. Also, [22] presents a good overview of the state of the art in 1998.

18.1 INTRODUCTION TO TRELLIS-CODED MODULATION

In our treatment of TCM we assume that the transmitted symbols are drawn from an M-ary signal constellation in either one- or two-dimensional Euclidean space. Several typical signal constellations appear in Figure 18.1. Some one-dimensional, or *amplitude modulation* (AM), signal constellations are shown in Figure 18.1(a). The simplest of these, 2-AM, is equivalent to BPSK. Figure 18.1(b) illustrates several

[1] Throughout this chapter we denote spectral efficiency in units of bits/symbol, where one signal (symbol) is transmitted in each T-second time interval. With this notation, the required bandwidth is proportional to $1/T$, and higher spectral efficiencies are thus more bandwidth efficient.

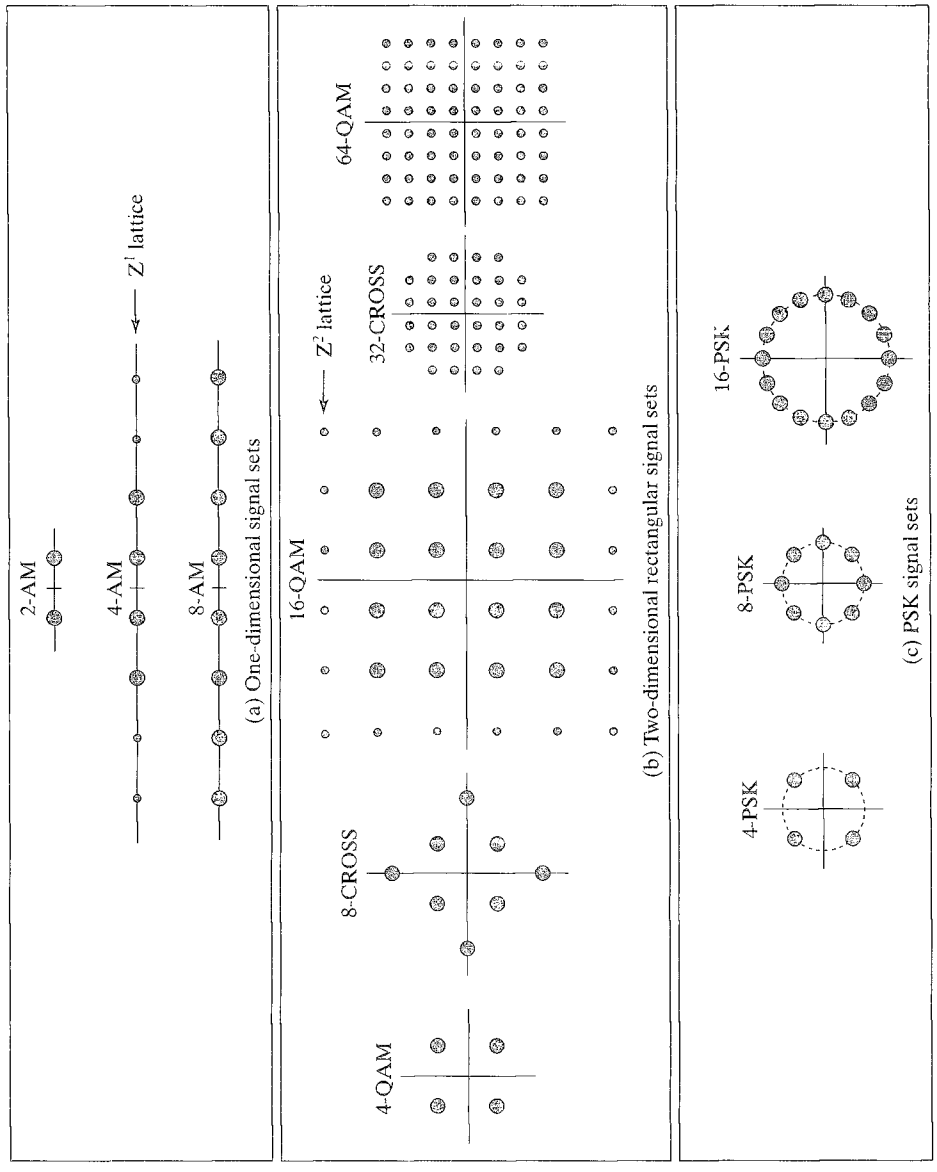


FIGURE 18.1: Typical signal constellations.

two-dimensional signal constellations that exhibit a combination of amplitude modulation and *phase modulation* (AM/PM). Rectangular constellations with $M = (2^p)^2 = 4^p$ signal points, $p = 1, 2, \cdots$, are also referred to as *quadrature amplitude modulation* (QAM) signal sets, since they can be generated by separately applying amplitude modulation to two quadrature carriers (a sine wave and a cosine wave)

using a discrete set of 2^p possible amplitudes and then combining the two modulated signals. All practical one-dimensional AM and two-dimensional AM/PM signal constellations can be viewed as subsets of a *lattice*, an infinite array of regularly spaced points translated to its minimum average energy configuration. For example, 4-AM is a (translated) subset of the one-dimensional integer lattice $\mathbb{Z}^1$, whose points consist of all integers in one dimension, and 16-QAM is a (translated) subset of the two-dimensional integer lattice $\mathbb{Z}^2$, whose points consist of all pairs of integers in two dimensions. Finally, some two-dimensional *M-ary phase-shift-keying* (MPSK) signal sets are shown in Figure 18.1(c). MPSK signals all have the same amplitude, and thus they are a form of phase modulation. The simplest of these, 4-PSK (also denoted QPSK), is equivalent to 4-QAM.

Because TCM schemes use signal set expansion rather than additional transmitted symbols to accommodate the redundant bits introduced by coding, performance comparisons must be made with uncoded modulation systems that use smaller signal sets but have the same spectral efficiency, that is, the same number of information bits per transmitted symbol. Thus, care must be exercised to ensure that the different schemes being compared have the same average energy per transmitted symbol. As an illustration, we can compute the average signal energy of the three one-dimensional AM signal sets shown in Figure 18.1(a), where we have assumed that the minimum Euclidean distance between signal points is $d_{min} = 2$, as follows:

$$E_s = 2\frac{[(+1)^2]}{2} = 1 \qquad \text{(2-AM)} \tag{18.1a}$$

$$E_s = 2\frac{[(+1)^2 + (+3)^2]}{4} = 5 \qquad \text{(4-AM)} \tag{18.1b}$$

$$E_s = 2\frac{[(+1)^2 + (+3)^2 + (+5)^2 + (+7)^2]}{8} = 21. \qquad \text{(8-AM)} \tag{18.1c}$$

Thus, in comparing a TCM scheme with one information bit and one redundant bit that uses 4-AM modulation with an uncoded scheme using 2-AM modulation, we must reduce the energy of each signal point in the coded system by a factor of 5, or almost 7 dB, to maintain the same average energy per transmitted symbol; that is, we must scale the amplitude of each signal by the factor $1/\sqrt{5}$. If a TCM scheme with one information bit and two redundant bits using 8-AM is compared with uncoded 2-AM, the energy in the coded system must be reduced by a factor of 21, or more than 13 dB. This reduced signal energy results in a reduced minimum distance between signal points that must be overcome by coding for TCM to achieve a positive coding gain compared with an uncoded system with the same average energy. To minimize the reduction in signal energy of coded systems, practical TCM schemes employ codes with just one redundant bit, that is, rate $R = k/(k+1)$ codes. Thus, TCM system design involves the use of high-rate binary convolutional codes. In Table 18.1 we list the average energies of each signal set shown in Figure 18.1, where the minimum distance between signal points $d_{min} = 2$, and each constellation is in its minimum average energy configuration. The energy requirements of one signal set compared

TABLE 18.1: Average energies for the signal sets in Figure 18.1.

(a) One-dimensional signal sets		
Signal set	E_s	(dB)
2-AM	1	0.0
4-AM	5	7.0
8-AM	21	13.2
(b) Two-dimensional rectangular signal sets		
Signal set	E_s	(dB)
4-QAM	2	3.0
8-CROSS	5.5	7.4
16-QAM	10	10.0
32-CROSS	20	13.0
64-QAM	42	16.2
(c) PSK signal sets		
Signal set	E_s	(dB)
4-PSK	2	3.0
8-PSK	6.8	8.3
16-PSK	26.3	14.2

with another can be determined by simply taking the difference (in decibels) of the values listed in Table 18.1. For example, if uncoded 8-PSK is compared with coded 16-PSK with one redundant bit, we say that the *constellation expansion factor* γ_c of the coded system relative to the uncoded system is $\gamma_c = 14.2 \text{ dB} - 8.3 \text{ dB} = 5.9 \text{ dB}$.

Now, consider the transmission of a signal sequence (coded or uncoded) from an M-ary signal set $S = \{s_0, s_1, \cdots, s_{M-1}\}$ over an AWGN channel. Let $\mathbf{y}(D) = y_0 + y_1 D + y_2 D^2 + \cdots$ be the transmitted sequence, where $y_l \in S$ for all l, and let $\mathbf{r}(D) = r_0 + r_1 D + r_2 D^2 + \cdots = \mathbf{y}(D) + \mathbf{n}(D)$ be the received sequence, where $\mathbf{n}(D) = n_0 + n_1 D + n_2 D^2 + \cdots$ is the noise sequence, n_l is an independent Gaussian noise sample with zero mean and variance $N_0/2$ per dimension for all l, and r_l and n_l belong to either one- or two-dimensional Euclidean space, depending on whether S is one- or two-dimensional. We can also represent the transmitted, noise, and received sequences by the vectors $\mathbf{y} = (y_0, y_1, y_2, \cdots)$, $\mathbf{n} = (n_0, n_1, n_2, \cdots)$, and $\mathbf{r} = (r_0, r_1, r_2, \cdots)$, and for two-dimensional signal sets, we denote transmitted, noise, and received signal points by $y_l = (y_{il}, y_{jl})$, $n_l = (n_{il}, n_{jl})$, and $r_l = (r_{il}, r_{jl})$, respectively. Throughout the chapter we assume that $\mathbf{r}$ is unquantized; that is, soft demodulator outputs are available at the receiver.

To compute the *symbol-error probability* P_s of an uncoded system, we can consider the transmission of only a single symbol. For example, for the QPSK signal set shown in Figure 18.1(c), if each signal point has energy E_s, that is, its distance from the origin is $\sqrt{E_s}$, we can approximate its symbol error probability on an AWGN channel with one-sided noise power spectral density N_0 with the familiar

union upper bound as follows:

$$\begin{aligned} P_s &\le 2Q\left(\sqrt{\frac{E_s}{N_0}}\right) + Q\left(\sqrt{\frac{2E_s}{N_0}}\right) \\ &\approx 2Q\left(\sqrt{\frac{E_s}{N_0}}\right) \\ &= A_{min}Q\left(\sqrt{\frac{d_{min}^2}{2N_0}}\right) \\ &\approx \frac{A_{min}}{2}e^{-d_{min}^2/4N_0}, \quad \text{(uncoded)} \end{aligned} \tag{18.2}$$

where $d_{min}^2 = 2E_s$ is the *minimum squared Euclidean* (MSE) *distance* between signal points, and $A_{min} = 2$ is the *number of nearest neighbors* in the QPSK constellation. (It is not difficult to compute the exact value for P_s in this case, but the approximate expression given in (18.2) allows for a direct comparison with the performance bounds of coded systems.)

For coded transmission, we assume that $\mathbf{r}$ is decoded using maximum-likelihood soft-decision Viterbi decoding, as presented in Chapter 12. (For two-dimensional signal constellations, the Viterbi algorithm metric is simply the distance in the two-dimensional Euclidean space.) In this case, given the transmission of a particular coded sequence $\mathbf{y}$, the general form of the union upper bound on the *event-error probability* $P_e(\mathbf{y})$ becomes

$$P_e(\mathbf{y}) \le \sum_{\mathbf{y}' \ne \mathbf{y}} Q\left(\sqrt{\frac{d_E^2(\mathbf{y}, \mathbf{y}')}{2N_0}}\right), \tag{18.3}$$

where

$$d_E^2(\mathbf{y}, \mathbf{y}') = \sum_l d_E^2(y_l, y_l') = \sum_l \|y_l - y_l'\|^2 = \sum_l \left[(y_{il} - y_{il}')^2 + (y_{jl} - y_{jl}')^2\right] \tag{18.4}$$

is the squared Euclidean distance between the coded sequences $\mathbf{y}$ and $\mathbf{y}'$. Now, defining d_{free}^2 as the *minimum free squared Euclidean* (MFSE) *distance* between $\mathbf{y}$ and any other coded sequence $\mathbf{y}'$, and $A_{d_{free}}$ as the *number of nearest neighbors*, we can approximate the bound on $P_e(\mathbf{y})$ as

$$\begin{aligned} P_e(\mathbf{y}) &\le A_{d_{free}}Q\left(\sqrt{\frac{d_{free}^2}{2N_0}}\right) \\ &\approx \frac{A_{d_{free}}}{2}e^{-d_{free}^2/4N_0}. \quad \text{(coded)} \end{aligned} \tag{18.5}$$

The expressions for event-error probability given in (18.3) and (18.5) are conditioned on the transmission of a particular sequence $\mathbf{y}$, because, in general, TCM

systems are nonlinear; however, most known schemes have many of the symmetry properties of linear codes. Typically, d^2_{free} is independent of the transmitted sequence, but $A_{d_{free}}$ can vary depending on the transmitted sequence. The error analysis of TCM schemes is investigated more thoroughly in Section 18.3.

Because the exponential behavior of (18.2) and (18.5) depends on the MSE distances of the uncoded and coded systems, respectively, the *asymptotic coding gain* γ of coded TCM relative to uncoded modulation can be formulated as follows:

$$\gamma = \frac{d^2_{free/coded}/E_{coded}}{d^2_{min/uncoded}/E_{uncoded}}, \tag{18.6}$$

where E_{coded} and $E_{uncoded}$ are the average energies of the coded and uncoded signal sets, respectively. We can rewrite (18.6) as

$$\gamma = \frac{E_{uncoded}}{E_{coded}} \frac{d^2_{free/coded}}{d^2_{min/uncoded}} = \gamma_c^{-1}\gamma_d, \tag{18.7}$$

where γ_c is the constellation expansion factor, and γ_d is the *distance gain factor*.

We proceed by letting d_{min} and Δ_{min} represent the minimum distance between signal points in the uncoded and coded constellations, respectively, and by assuming that the minimum distance between points in the coded (expanded) constellation is reduced, so that the average energies of the coded and uncoded signal sets are equal; that is, $\Delta_{min} < d_{min}$, and $\gamma_c = 1$. Then, the TCM system must have a free distance between coded sequences that is greater than the minimum distance between signal points in the uncoded system to achieve coding gain. In other words, even though $\Delta_{min} < d_{min}$, a TCM system must achieve $d_{free} > d_{min}$.

In the design of codes for binary modulation, the MFSE distance between two signal sequences $\mathbf{y}$ and $\mathbf{y}'$ is given by (see Problem 18.1)

$$d^2_{free} = 4E_s d_{H,free}, \quad \text{(binary modulation)} \tag{18.8}$$

where $d_{H,free}$ is the *minimum free Hamming distance* of the convolutional code. Thus, for binary modulation, the best system design is achieved by choosing the code that maximizes $d_{H,free}$. We will shortly see that, in general, this is not true for TCM system design.

In the TCM case, consider a rate $R = k/(k+1)$ convolutional code with minimum free Hamming distance $d_{H,free}$ in which we denote the $k+1$ encoder output bits at any time unit l by the vector $\mathbf{v}_l = (v_l^{(k)}, v_l^{(k-1)}, \cdots, v_l^{(0)})$. (Throughout the remainder of this chapter, when it is not necessary to specifically denote the time unit l of a vector, the subscript l will be deleted; that is, vectors such as $\mathbf{v}_l$ will be denoted simply by $\mathbf{v}$.) Then, assume that the 2^{k+1} binary vectors $\mathbf{v}$ are mapped into elements s_i of an M-ary signal set S using a one-to-one *mapping function* $f(\mathbf{v}) \rightarrow s_i$, where $M = 2^{k+1}$.

We see from (18.3) that the performance of a TCM system depends on the *squared Euclidean* (SE) distances between signal sequences. Thus, the set of SE distances between all possible pairs of signal points must be determined. Denoting the *binary labels* of two signal points by the vectors $\mathbf{v}$ and $\mathbf{v}'$, we define the *error*

vector $\mathbf{e} = \mathbf{v} \oplus \mathbf{v}'$ as their modulo-2 sum. It follows that $w_H(\mathbf{e}) = d_H(\mathbf{v}, \mathbf{v}')$; that is, the Hamming weight of an error vector equals the Hamming distance between its two corresponding signal labels. For each error vector $\mathbf{e}$, there are M pairs of signal vectors $\mathbf{v}$ and $\mathbf{v}'$ such that $\mathbf{v}' = \mathbf{v} \oplus \mathbf{e}$, and thus M possible SE distances $\Delta_{\mathbf{v}}^2(\mathbf{e}) = \|f(\mathbf{v}) - f(\mathbf{v} \oplus \mathbf{e})\|^2$. To denote this set of distances, we introduce the *average Euclidean weight enumerator* (AEWE), defined as follows:

$$\Delta_{\mathbf{e}}^2(X) = (1/M) \sum_{\mathbf{v}} X^{\Delta_{\mathbf{v}}^2(\mathbf{e})} = (1/M) \sum_{\mathbf{v}} X^{\|f(\mathbf{v}) - f(\mathbf{v} \oplus \mathbf{e})\|^2}. \tag{18.9a}$$

Clearly, $\Delta_{\mathbf{e}}^2(X)$ depends both on the signal constellation S and the mapping function $f(\cdot)$. When we are interested only in the MFSE distance d_{free}^2 between signal sequences, it is sufficient to define the *minimum Euclidean weight enumerator* (MEWE) of an error vector $\mathbf{e}$ as

$$\delta_{\mathbf{e}}^2(X) = X^{\Delta^2(\mathbf{e})} = X^{\min_{\mathbf{v}} \Delta_{\mathbf{v}}^2(\mathbf{e})}, \tag{18.9b}$$

where

$$\Delta^2(\mathbf{e}) \stackrel{\Delta}{=} \min_{\mathbf{v}} \Delta_{\mathbf{v}}^2(\mathbf{e}) \tag{18.9c}$$

is called the *Euclidean weight* (EW) of $\mathbf{e}$.

The AEWEs and the MEWEs can be used to compute the *average weight enumerating function* $A_{av}(X)$ and the MFSE distance d_{free}^2 of a TCM system, respectively. The basic technique is as follows. We label each branch of a conventional trellis for a rate $R = k/(k+1)$ linear binary convolutional code with a vector $\mathbf{v}$ representing the $k+1$ current encoder output bits. Alternatively, we can label each trellis branch with the error vector $\mathbf{e} = \mathbf{v} \oplus \mathbf{v}'$, representing the difference (mod-2 sum) between $\mathbf{v}$ and the corresponding label $\mathbf{v}'$ of an arbitrary reference codeword. Because of linearity, it is easy to see that this *error trellis* is identical to the conventional trellis. If we then modify the error trellis by replacing its binary labels with the Hamming weight enumerators $X^{w_H(\mathbf{e})}$ of each branch, we can compute the weight enumerating function $A(X)$ and the minimum free Hamming distance $d_{H,free}$ of the code by converting the error trellis to an error state diagram and using the standard transfer function approach discussed in Chapter 11. If certain symmetry conditions on the signal constellation S and the mapping function $f(\cdot)$ are satisfied, exactly the same method applies to TCM systems, even though they are in general nonlinear. In this case, we replace the binary labels on the error trellis of the convolutional code with the AEWEs or MEWEs from (18.9), depending on whether we wish to compute $A_{av}(X)$ or d_{free}^2. The AEWEs are in general polynomials in X, indicating that more than one distance can correspond to a given error vector $\mathbf{e}$, whereas the MEWEs, since they represent only the minimum distance corresponding to a given $\mathbf{e}$, are monomials, as in the case of binary convolutional codes. To explain when the transfer function method can be applied to TCM systems, we now introduce the notion of a *uniform mapping*.

First, we partition the signal set S into two subsets, $Q(0)$ and $Q(1)$, such that $Q(0)$ contains the 2^k signal points labeled by a vector $\mathbf{v}$ with $v^{(0)} = 0$, and $Q(1)$ contains the 2^k signal points corresponding to labels with $v^{(0)} = 1$. Next, we let $\Delta_{\mathbf{e},0}^2(X)$ be the AEWE for the subset $Q(0)$, and $\Delta_{\mathbf{e},1}^2(X)$ be the AEWE for the subset $Q(1)$. The subset MEWEs, $\delta_{\mathbf{e},0}^2(X)$ and $\delta_{\mathbf{e},0}^2(X)$, are defined in an analogous way.

DEFINITION 18.1 A one-to-one mapping function $f(\mathbf{v}) \rightarrow s_i$ from a rate $R = k/(k+1)$ convolutional encoder output vector $\mathbf{v} = (v^{(k)}, v^{(k-1)}, \cdots, v^{(0)})$ to a signal point s_i belonging to a 2^{k+1}-ary signal set S is *uniform* if and only if $\Delta^2_{\mathbf{e},0}(X) = \Delta^2_{\mathbf{e},1}(X)$ for all error vectors $\mathbf{e}$.

EXAMPLE 18.1 Uniform Mapping

Consider the three 8-PSK signal sets shown in Figure 18.2 along with their associated labels. Using the signal point labeled by $\mathbf{v} = (000)$ as a reference and assuming unit energy signals, we see that there are four distinct Euclidean distances between 8-PSK signal points:

$$a^2 = \left[1/\sqrt{2}\right]^2 + \left[1 - \left(1/\sqrt{2}\right)\right]^2 = 1/2 + \left(3/2 - \sqrt{2}\right) = 0.586, \quad (18.10a)$$

$$b^2 = 1^2 + 1^2 = 2, \quad (18.10b)$$

$$c^2 = \left[1/\sqrt{2}\right]^2 + \left[1 + \left(1/\sqrt{2}\right)\right]^2 = 1/2 + \left(3/2 + \sqrt{2}\right) = 3.414, \quad (18.10c)$$

$$d^2 = 2^2 = 4. \quad (18.10d)$$

We now examine the eight possible SE distances corresponding to the error vector $\mathbf{e} = (001)$ for the labeling of Figure 18.2(a). We see that there are a total of four code vectors $\mathbf{v}$ for which $\Delta^2_{\mathbf{v}}(\mathbf{e}) = \|f(\mathbf{v}) - f(\mathbf{v} \oplus \mathbf{e})\|^2 = 2$: the vectors $\mathbf{v} = (000), (001), (110)$, and (111), and four code vectors $\mathbf{v}$ for which $\Delta^2_{\mathbf{v}}(\mathbf{e}) = 3.414$, that is, the vectors $\mathbf{v} = (010), (011), (100)$, and (101). Thus, the AEWE for the error vector $\mathbf{e} = (001)$ is $\Delta^2_{\mathbf{e}}(X) = (1/2)X^2 + (1/2)X^{3.414}$. If we now partition the 8-PSK signal set into two subsets $Q(0)$ and $Q(1)$, depending on the value of the bit $v^{(0)}$ in the label vector, we see that for the error vector $\mathbf{e} = (001)$, each subset contains exactly two signal points for which $\Delta^2_{\mathbf{v}}(\mathbf{e}) = 2$, and two signal points for which $\Delta^2_{\mathbf{v}}(\mathbf{e}) = 3.414$; that is, $\Delta^2_{\mathbf{e},0}(X) = \Delta^2_{\mathbf{e},1}(X) = (\frac{1}{2})X^2 + (\frac{1}{2})X^{3.414}$. Repeating this calculation for each possible error vector $\mathbf{e}$ gives us the AEWEs listed, along with the corresponding MEWEs, in Table 18.2(a) for each subset $Q(0)$ and $Q(1)$. Because $\Delta^2_{\mathbf{e},0}(X) = \Delta^2_{\mathbf{e},1}(X)$ for all $\mathbf{e}$, the mapping is uniform.

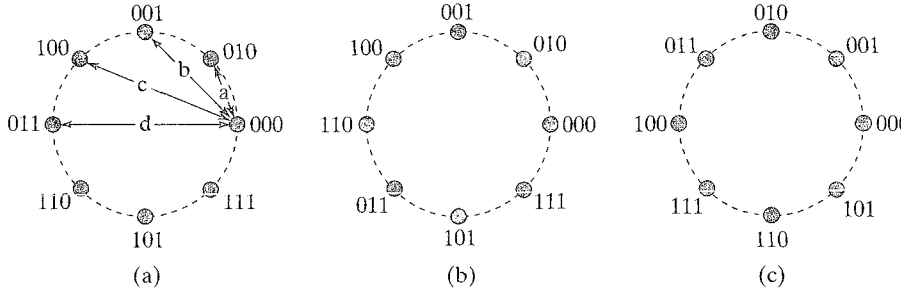


FIGURE 18.2: Three 8-PSK signal sets with different labelings.

TABLE 18.2: The AEWEs and MEWEs corresponding to the three 8-PSK signal sets in Figure 18.2.

(a)

$\mathbf{e}$	$\Delta^2_{\mathbf{e},0}(X)$	$\Delta^2_{\mathbf{e},1}(X)$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e},0}(X)$	$\delta^2_{\mathbf{e},1}(X)$	$\delta^2_{\mathbf{e}}(X)$
000	X^0	X^0	X^0	X^0	X^0	X^0
001	$\frac{1}{2}X^2 + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^2 + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^2 + \frac{1}{2}X^{3.414}$	X^2	X^2	X^2
010	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
011	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^4$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^4$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^4$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
100	$\frac{1}{2}X^{3.414} + \frac{1}{2}X^4$	$\frac{1}{2}X^{3.414} + \frac{1}{2}X^4$	$\frac{1}{2}X^{3.414} + \frac{1}{2}X^4$	$X^{3.414}$	$X^{3.414}$	$X^{3.414}$
101	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
110	$\frac{1}{2}X^2 + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^2 + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^2 + \frac{1}{2}X^{3.414}$	X^2	X^2	X^2
111	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$

(b)

$\mathbf{e}$	$\Delta^2_{\mathbf{e},0}(X)$	$\Delta^2_{\mathbf{e},1}(X)$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e},0}(X)$	$\delta^2_{\mathbf{e},1}(X)$	$\delta^2_{\mathbf{e}}(X)$
000	X^0	X^0	X^0	X^0	X^0	X^0
001	$\frac{1}{4}X^2 + \frac{1}{2}X^{3.414} + \frac{1}{4}X^4$	$\frac{1}{4}X^2 + \frac{1}{2}X^{3.414} + \frac{1}{4}X^4$	$\frac{1}{4}X^2 + \frac{1}{2}X^{3.414} + \frac{1}{4}X^4$	X^2	X^2	X^2
010	$X^{0.586}$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{3}{4}X^{0.586} + \frac{1}{4}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
011	$\frac{1}{4}(X^{0.586} + X^2 + X^{3.414} + X^4)$	$\frac{1}{4}(X^{0.586} + X^2 + X^{3.414} + X^4)$	$\frac{1}{4}(X^{0.586} + X^2 + X^{3.414} + X^4)$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
100	$X^{3.414}$	$\frac{1}{2}X^2 + \frac{1}{2}X^4$	$\frac{1}{4}X^2 + \frac{1}{2}X^{3.414} + \frac{1}{4}X^4$	$X^{3.414}$	X^2	X^2
101	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^2$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
110	$\frac{1}{2}X^2 + \frac{1}{2}X^4$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{1}{4}(X^{0.586} + X^2 + X^{3.414} + X^4)$	X^2	$X^{0.586}$	$X^{0.586}$
111	$\frac{1}{4}X^{0.586} + \frac{1}{2}X^2 + \frac{1}{4}X^{3.414}$	$\frac{1}{4}X^{0.586} + \frac{1}{2}X^2 + \frac{1}{4}X^{3.414}$	$\frac{1}{4}X^{0.586} + \frac{1}{2}X^2 + \frac{1}{4}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$

(*continued overleaf*)

TABLE 18.2: (*continued*)

(c)						
$\mathbf{e}$	$\Delta^2_{\mathbf{e},0}(X)$	$\Delta^2_{\mathbf{e},1}(X)$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e},0}(X)$	$\delta^2_{\mathbf{e},1}(X)$	$\delta^2_{\mathbf{e}}(X)$
000	X^0	X^0	X^0	X^0	X^0	X^0
001	$\frac{3}{4}X^{0.586} + \frac{1}{4}X^{3.414}$	$\frac{3}{4}X^{0.586} + \frac{1}{4}X^{3.414}$	$\frac{3}{4}X^{0.586} + \frac{1}{4}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
010	X^2	X^2	X^2	X^2	X^2	X^2
011	$\frac{3}{4}X^{0.586} + \frac{1}{4}X^{3.414}$	$\frac{3}{4}X^{0.586} + \frac{1}{4}X^{3.414}$	$\frac{3}{4}X^{0.586} + \frac{1}{4}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
100	X^4	X^2	$\frac{1}{2}X^2 + \frac{1}{2}X^4$	X^4	X^2	X^2
101	$\frac{1}{4}X^{0.586} + \frac{3}{4}X^{3.414}$	$\frac{1}{4}X^{0.586} + \frac{3}{4}X^{3.414}$	$\frac{1}{4}X^{0.586} + \frac{3}{4}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
110	X^2	X^4	$\frac{1}{2}X^2 + \frac{1}{2}X^4$	X^2	X^4	X^2
111	$\frac{1}{4}X^{0.586} + \frac{3}{4}X^{3.414}$	$\frac{1}{4}X^{0.586} + \frac{3}{4}X^{3.414}$	$\frac{1}{4}X^{0.586} + \frac{3}{4}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$

The AEWEs and MEWEs corresponding to each error vector $\mathbf{e}$ for the mappings in Figures 18.2(b) and 18.2(c) are likewise listed in Tables 18.2(b) and 18.2(c), respectively. In the case of the 8-PSK mapping shown in Figure 18.2(b), we see that three error vectors, namely, $\mathbf{e} = (010)$, (100), and (110), result in $\Delta^2_{\mathbf{e},0}(X) \neq \Delta^2_{\mathbf{e},1}(X)$, and for $\mathbf{e} = (100)$ and (110) the Euclidean weights in the two subsets are different. Thus, this is a *nonuniform* mapping. Finally, for the 8-PSK mapping shown in Figure 18.2(c), we see that the error vectors $\mathbf{e} = (100)$ and (110) result in $\Delta^2_{\mathbf{e},0}(X) \neq \Delta^2_{\mathbf{e},1}(X)$ and in different Euclidean weights for $Q(0)$ and $Q(1)$. Thus, the mapping in Figure 18.2(c) is also nonuniform.

The following remarks relate to Example 18.1:

- For the uniform mapping in Figure 18.2(a), $\Delta^2_{\mathbf{e}}(X) = \Delta^2_{\mathbf{e},0}(X) = \Delta^2_{\mathbf{e},1}(X)$, and the MEWEs and the EWs are also equal in the two subsets $Q(0)$ and $Q(1)$ for all error vectors $\mathbf{e}$. This is true for any uniform mapping.
- In Figures 18.2(a) and (c), the subsets $Q(0)$ and $Q(1)$ are *isomorphic*; that is, one subset can be obtained from the other by some combination of rotation, translation, and reflection about an axis of the signal points. A one-to-one mapping that takes one signal set into another isomorphic signal set, thus preserving the set of distances between signal points, is called an *isometry*.
- For a mapping to be uniform, it is necessary that there be an isometry between the subsets $Q(0)$ and $Q(1)$; however, the existence of an isometry is not a sufficient condition for uniformity. Thus, even when an isometry exists, a mapping may be nonuniform, as is the case in Figure 18.2(c) (also see Problem 18.3).
- In Figure 18.2(b) there is no isometry between the subsets $Q(0)$ and $Q(1)$. Thus, this mapping cannot be uniform.

We now let $\mathbf{v}(D)$ and $\mathbf{v}'(D) = \mathbf{v}(D) \oplus \mathbf{e}(D)$ be any two sequences in the binary code trellis, where

$$\mathbf{v}(D) = \mathbf{v}_0 + \mathbf{v}_1 D + \mathbf{v}_2 D^2 + \cdots, \tag{18.11a}$$

$$\mathbf{v}'(D) = \mathbf{v}'_0 + \mathbf{v}'_1 D + \mathbf{v}'_2 D^2 + \cdots, \tag{18.11b}$$

and

$$\mathbf{e}(D) = \mathbf{e}_h D^h + \mathbf{e}_{h+1} D^{h+1} + \cdots + \mathbf{e}_{h+L} D^{h+L}, \quad \mathbf{e}_h, \mathbf{e}_{h+L} \neq \mathbf{0}, \ L \geq 0, \ h \geq 0, \tag{18.11c}$$

is a nonzero path through the error trellis of length $L+1$ branches; that is, $\mathbf{v}(D)$ and $\mathbf{v}'(D)$ differ in at most $L+1$ branches. The term $\mathbf{e}(D)$ then represents an error event of length $L+1$. If $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ are the two channel signal sequences corresponding to $\mathbf{v}(D)$ and $\mathbf{v}'(D)$, respectively, that is, $\mathbf{y}(D) = f(\mathbf{v}_0) + f(\mathbf{v}_1)D + f(\mathbf{v}_2)D^2 + \cdots$ and $\mathbf{y}'(D) = f(\mathbf{v}'_0) + f(\mathbf{v}'_1)D + f(\mathbf{v}'_2)D^2 + \cdots$, then the SE distance between $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ is given by

$$
\begin{aligned}
d_E^2[\mathbf{y}(D), \mathbf{y}'(D)] &= \sum_{(h \le l \le h+L)} d_E^2 \left[f\left(\mathbf{v}_l\right), f\left(\mathbf{v}'_l\right) \right] \\
&= \sum_{(h \le l \le h+L)} \| f(\mathbf{v}_l) - f(\mathbf{v}_l \oplus \mathbf{e}_l) \|^2 \\
&= \sum_{(h \le l \le h+L)} \Delta^2_{\mathbf{v}_l}(\mathbf{e}_l) \\
&\ge \sum_{(h \le l \le h+L)} \Delta^2(\mathbf{e}_l) \stackrel{\Delta}{=} \Delta^2[\mathbf{e}(D)].
\end{aligned}
\tag{18.12}
$$

where the inequality follows from the definition of Euclidean weight given in (18.9c), and $\Delta^2[\mathbf{e}(D)]$ is called the Euclidean weight of the error sequence $\mathbf{e}(D)$. We now prove a lemma that establishes the conditions under which the Euclidean weights can be used to compute the MFSE distance d^2_{free} of a TCM system.

LEMMA 18.1 (RATE $R = k/(k+1)$ CODE LEMMA) [1] Assume the mapping from the output vector $\mathbf{v}$ of a rate $R = k/(k+1)$ binary convolutional encoder to the elements of a 2^{k+1}-ary signal set S is uniform. Then, for each binary error sequence $\mathbf{e}(D)$ in the error trellis, there exists a pair of signal sequences $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ such that (18.12) is satisfied with equality.

Proof. From the definition of Euclidean weight, $\Delta^2(\mathbf{e}_l) = \min_{\mathbf{v}_l = [v_l^{(k)}, \cdots, v_l^{(1)}, v_l^{(0)}]} \Delta^2_{\mathbf{v}_l}(\mathbf{e}_l)$ for all time units l. Because the mapping is uniform, minimizing over the k-bit vector $[v_l^{(k)}, \cdots, v_l^{(1)}]$ yields the same result in the subset $Q(0)$ with $v_l^{(0)} = 0$ as in the subset $Q(1)$ with $v_l^{(0)} = 1$; that is, the Euclidean weight is independent of the value of $v_l^{(0)}$, and $\Delta^2(\mathbf{e}_l) = \min_{[v_l^{(k)}, \cdots, v_l^{(1)}]} \Delta^2_{\mathbf{v}_l}(\mathbf{e}_l)$. Further, a rate $R = k/(k+1)$ encoder can produce any sequence of k-bit vectors $[v_l^{(k)}, \cdots, v_l^{(1)}]$ (only one bit is constrained); that is, every such sequence of k-bit vectors corresponds to a path through the trellis. Thus, for each binary error sequence $\mathbf{e}(D)$, there exists an encoder output sequence $\mathbf{v}(D)$ such that (18.12) is satisfied with equality. Q.E.D.

Lemma 18.1 implies that the MFSE distance d^2_{free} between signal sequences can be computed by replacing the binary labels on the error trellis with the MEWEs and finding the minimum-weight path through the trellis; that is,

$$
d^2_{free} = \min_{\mathbf{e}(D) \ne \mathbf{0}(D)} \Delta^2[\mathbf{e}(D)]. \tag{18.13}
$$

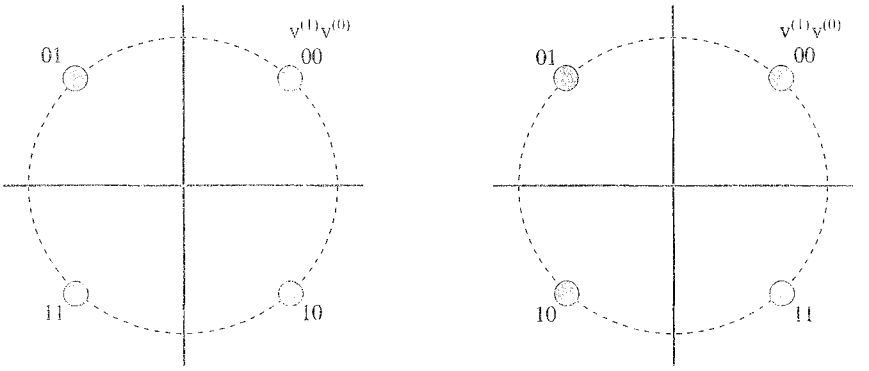
A similar argument can be used to show that the average weight enumerating function $A_{av}(X)$ can be computed by labeling the error trellis with the AEWEs and finding the transfer function of the modified state diagram (see Problem 18.4). If the mapping is nonuniform, the rate $R = k/(k+1)$ code lemma does not hold, and the computation of $A_{av}(X)$ and d^2_{free} becomes much more complex. An example illustrating this point is given later in this section.

The technique for using the AEWEs to compute $A_{av}(X)$ will be presented in Section 18.3. In the remainder of this section we present a series of examples illustrating the basic principles of designing a TCM system to maximize d^2_{free}.

EXAMPLE 18.2 Rate $R = 1/2$ Trellis-Coded QPSK

Consider a rate $R = 1/2$ convolutional code with minimum free Hamming distance $d_{H,free}$ in which we denote the two encoder output bits by the vector $\mathbf{v} = (v^{(1)}v^{(0)})$. In Figure 18.3 we show two rules for mapping the vector $\mathbf{v}$ into the QPSK signal set: *Gray mapping* and *natural mapping*. Gray mapping, which assigns labels to adjacent signals that differ in only one bit position, is commonly used in uncoded modulation, so the most likely error symbols, the nearest neighbors, will differ in only one bit position from the correct symbol. Natural mapping, which assigns labels in the order of their integer equivalents, is often used in TCM system applications that require insensitivity to carrier phase offset. This topic will be discussed in more detail in Section 18.4. Each signal is assumed to have unit energy, so the MSE distance between signal points is $\Delta^2_{min} = 2$.

For Gray-mapped QPSK, we see that the SE distance between all four pairs of signal points that differ by the error vector $\mathbf{e} = (01)$ is equal to 2, and thus $\Delta^2_{\mathbf{v}}(01) = 2$ for all $\mathbf{v}$, the AEWE is given by $\Delta^2_{(01)}(X) = X^2$, the MEWE is $\delta^2_{(01)}(X) = X^2$, and the EW is $\Delta^2(01) = \min_{\mathbf{v}} \Delta^2_{\mathbf{v}}(01) = 2$. In Table 18.3 we list, for both Gray and natural mapping of QPSK, the four possible error vectors $\mathbf{e}$, their Hamming weights $w_H(\mathbf{e})$, and the four corresponding AEWEs $\Delta^2_{\mathbf{e}}(X)$ and MEWEs $\delta^2_{\mathbf{e}}(X)$.



(a) Gray mapping (b) Natural mapping

FIGURE 18.3: Two mapping rules for the QPSK signal set.

TABLE 18.3: Euclidean distance structure for Gray- and naturally mapped QPSK.

		(a) Gray mapping		(b) Natural mapping	
$\mathbf{e}$	$w_H(\mathbf{e})$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e}}(X)$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e}}(X)$
00	0	X^0	X^0	X^0	X^0
01	1	X^2	X^2	X^2	X^2
10	1	X^2	X^2	X^4	X^4
11	2	X^4	X^4	X^2	X^2

From Table 18.3 we see that all the AEWEs are monomials and that $\Delta_{\mathbf{e}}^2(X) = \delta_{\mathbf{e}}^2(X)$ for all $\mathbf{e}$. Such mappings are called *regular* mappings, which implies that they are also *uniform* mappings. Further, in the case of Gray-mapped QPSK, the two error vectors for which $w_H(\mathbf{e}) = 1$ result in $\Delta_{\mathbf{v}}^2(\mathbf{e}) = 2$ for all $\mathbf{v}$, and the error vector for which $w_H(\mathbf{e}) = 2$ results in $\Delta_{\mathbf{v}}^2(\mathbf{e}) = 4$ for all $\mathbf{v}$; that is, $\Delta_{\mathbf{v}}^2(\mathbf{e}) = 2w_H(\mathbf{e})$ for all $\mathbf{e}$ and $\mathbf{v}$. In other words, there is a linear relationship between SE distance and Hamming distance. Thus, the rate $R = 1/2$ convolutional codes with the best minimum free Hamming distance $d_{H,free}$ will also have the best MFSE distance d_{free}^2 when combined with Gray-mapped QPSK. For example, the optimum free distance $(2, 1, 2)$ code with $d_{H,free} = 5$, when used with Gray-mapped QPSK, results in an MFSE distance of $d_{free}^2 = 10$. Compared with uncoded BPSK with unit energy signals and $d_{min}^2 = 4$, this (2, 1, 2) code results in an asymptotic coding gain of $\gamma = 10\log_{10}(d_{free}^2/d_{min}^2) = 10\log_{10}(10/4) = 3.98$ dB, exactly the same as when this code is used with BPSK modulation. Thus, designing optimum TCM schemes for Gray-mapped QPSK is identical to finding optimum binary convolutional codes for BPSK modulation.

For naturally mapped QPSK, however, the situation is different. The two error vectors for which $w_H(\mathbf{e}) = 1$ give, for all $\mathbf{v}$, $\Delta_{\mathbf{v}}^2(\mathbf{e}) = 2$ in one case and $\Delta_{\mathbf{v}}^2(\mathbf{e}) = 4$ in the other case, and the error vector for which $w_H(\mathbf{e}) = 2$ gives $\Delta_{\mathbf{v}}^2(\mathbf{e}) = 2$ for all $\mathbf{v}$. In other words, there is no linear relationship between SE distance and Hamming distance when natural mapping is used. Thus, traditional code design techniques will not give the best codes for use with naturally mapped QPSK.

Continuing with the naturally mapped case, let us now consider two different $(2, 1, 2)$ code designs:

$$\text{Code 1}: \mathbb{G}_1(D) = \begin{bmatrix} 1 + D^2 & 1 + D + D^2 \end{bmatrix} \tag{18.14a}$$

$$\text{Code 2}: \mathbb{G}_2(D) = \begin{bmatrix} 1 + D^2 & D \end{bmatrix}. \tag{18.14b}$$

Code 1 is the optimum free distance $(2, 1, 2)$ code with $d_{H,free} = 5$, whereas code 2 is suboptimum and has $d_{H,free} = 3$. The encoder diagrams for these two codes are shown in Figure 18.4(a), and their error trellises with binary labels are shown in Figure 18.4(b). Now, replacing the binary labels with the MEWEs of naturally mapped QPSK from Table 18.3(b), we obtain the modified error trellises of Figure 18.4(c). Examining the modified error trellises for the minimum-weight error events, we see that $d_{free}^2 = 6$ for code 1, resulting in a coding gain of $\gamma = 1.76$ dB compared with uncoded BPSK, whereas code 2 achieves $d_{free}^2 = 10$ and $\gamma = 3.98$ dB. Thus code 2, clearly inferior to code 1 for binary modulation or for Gray-mapped QPSK, is the better choice for naturally mapped QPSK.

The following comments apply to Example 18.2:

- The linear relationship between Hamming distance and Euclidean distance in Gray-mapped QPSK is unique among nonbinary signal sets. In all other cases, no such linear relationship exists, and the best TCM schemes must be determined by jointly designing the code and the signal set mapping.

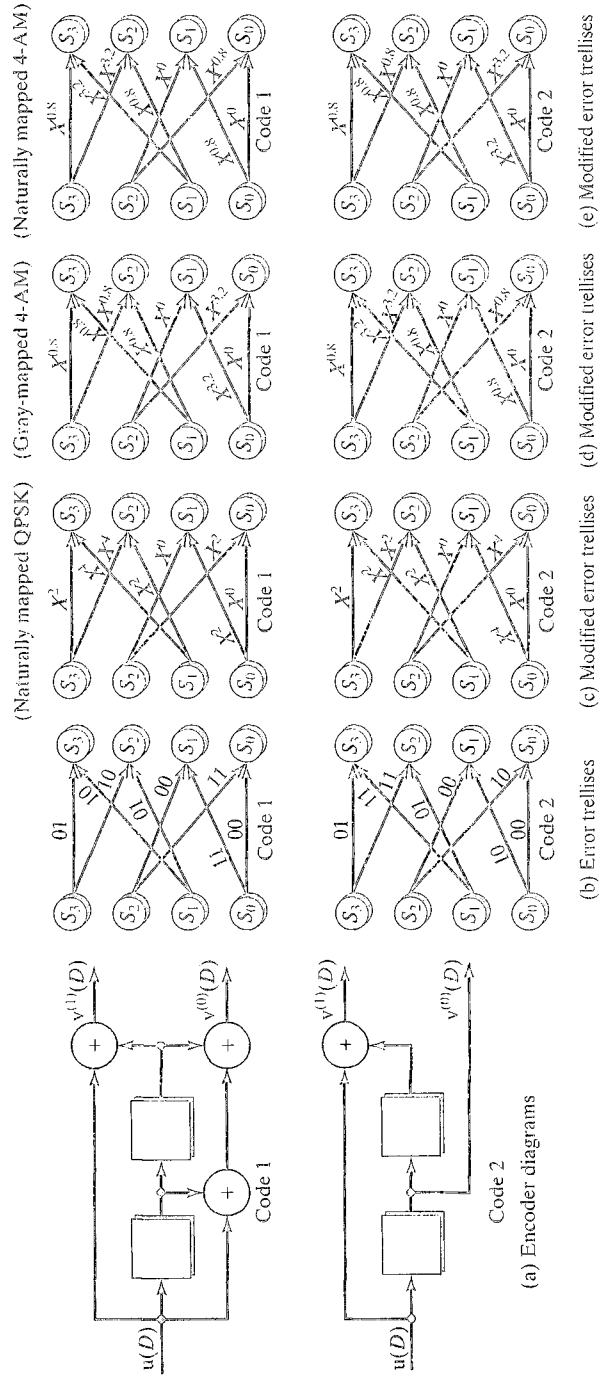


FIGURE 18.4: Encoder diagrams and error trellises for two (2, 1, 2) binary convolutional codes.

- The nonlinearity of most TCM systems arises from the mapping function $f(\cdot)$, which does not preserve a linear relationship between Hamming distance and Euclidean distance.
- Both signal mappings in Example 18.2 are *regular*; that is, each error vector $\mathbf{e}$ has a unique SE distance associated with it, and the AEWEs are equal to the MEWEs for all $\mathbf{e}$. For regular mappings, the Euclidean weight enumerating function $A(X)$ of the code is independent of the transmitted sequence. Thus, $A(X)$ and d^2_{free} can be computed in the same way as for linear convolutional codes with binary signal sets, that is, by assuming that the code sequence corresponding to the all-zero information sequence is transmitted.
- The critical step in the design of code 2 for naturally mapped QPSK was to assign the error vector $\mathbf{e} = (10)$ with maximum Euclidean weight to the two branches in the trellis that diverge from and remerge with the all-zero state S_0. This assignment guarantees the best possible Euclidean distance in the first and last branches of an error event and is one of the key rules of good TCM system design.
- Each of the coding gains quoted in this example came at the expense of bandwidth expansion, since the coded systems have a spectral efficiency of $\eta = 1$ bit/symbol $= 1/2$ bit/dimension, and the spectral efficiency of uncoded BPSK is $\eta = 1$ bit/dimension. Most of the comparisons with uncoded systems in the remainder of this chapter will involve TCM schemes that do not require bandwidth expansion; that is, they are bandwidth efficient.
- The design of good rate $R = 1/2$ codes for use with naturally mapped QPSK will be considered again in Section 18.4, when we take up the issue of rotationally invariant code designs.
- The QPSK signal set is equivalent to two independent uses of BPSK, denoted by $2 \times$ BPSK. This can be considered a simple form of multidimensional signaling, a subject that will be covered in Section 18.5.

EXAMPLE 18.3 Rate $R = 1/2$ Trellis-Coded 4-AM

In this example we consider the same two rate $R = 1/2$ convolutional codes as in Example 18.2, but this time with the encoder output vector $\mathbf{v} = (v^{(1)}v^{(0)})$ mapped into the one-dimensional 4-AM signal set. Both Gray mapping and natural mapping of the 4-AM signal set are illustrated in Figure 18.5, where the signal amplitudes are assigned in such a way that the average signal energy $E_s = 1$. Using the signal point labeled $\mathbf{v} = (00)$ as a reference, we see that there are three distinct SE distances between the 4-AM signal points:

$$a^2 = [(-1/\sqrt{5}) - (-3/\sqrt{5})]^2 = [2/\sqrt{5}]^2 = 0.8, \tag{18.15a}$$

$$b^2 = [(1/\sqrt{5}) - (-3/\sqrt{5})]^2 = [4/\sqrt{5}]^2 = 3.2, \tag{18.15b}$$

$$c^2 = [(3/\sqrt{5}) - (-3/\sqrt{5})]^2 = [6/\sqrt{5}]^2 = 7.2. \tag{18.15c}$$

Clearly, the MSE distance between signal points in this case is $\Delta^2_{min} = 0.8$.

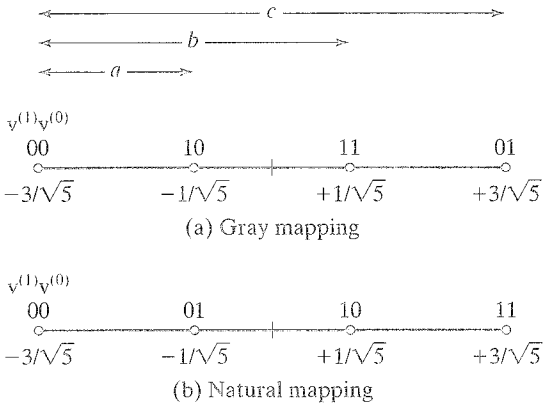


FIGURE 18.5: Gray and natural mapping of the 4-AM signal set.

TABLE 18.4: Euclidean distance structure for Gray- and naturally mapped 4-AM.

	(a) Gray mapping		(b) Natural mapping	
$\mathbf{e}$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e}}(X)$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e}}(X)$
00	X^0	X^0	X^0	X^0
01	$\frac{1}{2}X^{0.8} + \frac{1}{2}X^{7.2}$	$X^{0.8}$	$X^{0.8}$	$X^{0.8}$
10	$X^{0.8}$	$X^{0.8}$	$X^{3.2}$	$X^{3.2}$
11	$X^{3.2}$	$X^{3.2}$	$\frac{1}{2}X^{0.8} + \frac{1}{2}X^{7.2}$	$X^{0.8}$

In Table 18.4 we list, for both Gray and natural mapping of 4-AM, the four possible error vectors $\mathbf{e}$ and the four corresponding AEWEs $\Delta^2_{\mathbf{e}}(X)$ and MEWEs $\delta^2_{\mathbf{e}}(X)$. In Problem 18.6 it is shown that $\Delta^2_{\mathbf{e},0}(X) = \Delta^2_{\mathbf{e},1}(X)$ in both cases, and thus the mappings are uniform. We note that in each case, however, there is exactly one error vector $\mathbf{e}$ for which $\Delta^2_{\mathbf{e}}(X)$ is not a monomial, and thus the mappings are not regular.

If we now replace the binary labels on the error trellises shown in Figure 18.4(b) with the MEWEs of Gray- and naturally mapped 4-AM from Table 18.4, we obtain the modified error trellises of Figures 18.4(d) and 18.4(e), respectively. Examining the modified error trellises for the minimum-weight error events, we see that for Gray-mapped 4-AM (Figure 18.4(d)) $d^2_{free} = 7.2$ for code 1, resulting in a coding gain of $\gamma = 10\log_{10}(d^2_{free}/d^2_{min}) = 2.55$ dB compared with uncoded 2-AM with unit energy signals and $d^2_{min} = 4$, whereas code 2 achieves only $d^2_{free} = 2.4$, resulting in a *coding loss* of $\gamma = -2.22$ dB. Thus, code 1 is clearly the better choice for Gray-mapped 4-AM. For 4-AM with natural mapping (Figure 18.4(e)), the situation is exactly reversed, and the best choice is code 2, which results in a coding gain of $\gamma = 2.55$ dB compared with uncoded 2-AM.

The following observations relate to Example 18.3:

- In both cases, the mappings are *nonregular*; that is, for some error vectors $\mathbf{e}$, the MEWE does not equal the AEWE. This implies that the weight enumerating function $A(X)$ of the TCM system changes depending on the transmitted sequence; however, since $\Delta^2_{\mathbf{e},0}(X) = \Delta^2_{\mathbf{e},1}(X)$ for all $\mathbf{e}$ in both cases, the mappings are *uniform*, and the MFSE distance d^2_{free} can be computed by replacing the labels $\mathbf{e}$ in the binary error trellis with their corresponding MEWEs $\delta^2_{\mathbf{e}}(X)$ and using the transfer function method.
- By definition, all regular mappings must be uniform, but the reverse is not true.
- As in Example 18.2, the critical step in designing the best codes for both mappings was to assign the error vector with maximum Euclidean weight to the branches in the trellis that diverge from and remerge with the state S_0.
- In Example 18.3, unlike in Example 18.2, coding gain is achieved without bandwidth expansion, since the coded signal set, 4-AM, has the same dimensionality as the uncoded signal set, 2-AM. This explains the somewhat smaller coding gain, 2.55 dB versus 3.98 dB, achieved in Example 18.3 compared with Example 18.2.

EXAMPLE 18.4 Rate $R = 2/3$ Trellis-Coded 8-PSK

Now, consider a rate $R = 2/3$ convolutional code with 8-PSK modulation in which we denote the three encoder output bits by the vector $\mathbf{v} = (v^{(2)}v^{(1)}v^{(0)})$. In Figure 18.6 these three bits are shown mapped into the 8-PSK signal set according to the natural mapping rule. Each signal is again assumed to have unit energy, but in this case the MSE distance between signal points, computed in (18.10a), is $\Delta^2_{min} = 0.586$. Thus, compared with the QPSK signal set with the same average energy, the MSE distance of 8-PSK is reduced from 2.0 to 0.586.

In Table 18.5 we list the eight possible error vectors $\mathbf{e}$ and the eight corresponding AEWEs, $\Delta^2_{\mathbf{e},0}(X)$, $\Delta^2_{\mathbf{e},1}(X)$, and $\Delta^2_{\mathbf{e}}(X)$, and MEWEs, $\delta^2_{\mathbf{e},0}(X)$, $\delta^2_{\mathbf{e},1}(X)$, and

TABLE 18.5: Euclidean distance structure for naturally mapped 8-PSK.

$\mathbf{e}$	$\Delta^2_{\mathbf{e},0}(X)$	$\Delta^2_{\mathbf{e},1}(X)$	$\Delta^2_{\mathbf{e}}(X)$	$\delta^2_{\mathbf{e},0}(X)$	$\delta^2_{\mathbf{e},1}(X)$	$\delta^2_{\mathbf{e}}(X)$
000	X^0	X^0	X^0	X^0	X^0	X^0
001	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
010	X^2	X^2	X^2	X^2	X^2	X^2
011	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$
100	X^4	X^4	X^4	X^4	X^4	X^4
101	$X^{3.414}$	$X^{3.414}$	$X^{3.414}$	$X^{3.414}$	$X^{3.414}$	$X^{3.414}$
110	X^2	X^2	X^2	X^2	X^2	X^2
111	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$\frac{1}{2}X^{0.586} + \frac{1}{2}X^{3.414}$	$X^{0.586}$	$X^{0.586}$	$X^{0.586}$

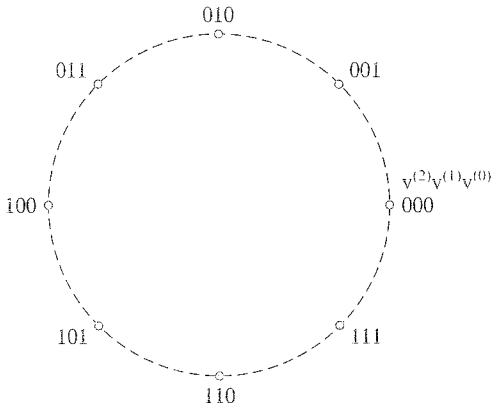


FIGURE 18.6: Natural mapping rule for the 8-PSK signal set.

$\delta_e^2(X)$, for naturally mapped 8-PSK. In this case we see that natural mapping of 8-PSK is *uniform*. (In Problem 18.8 it is shown that Gray mapping of 8-PSK is not uniform.) Unlike the uniform 8-PSK mapping shown in Figure 18.2(a), however, natural mapping has only two error vectors that result in different Euclidean distances; that is, for natural mapping the error vectors $\mathbf{e} = (011)$ and $\mathbf{e} = (111)$ result in $\Delta_{\mathbf{v}}^2(\mathbf{e}) = 0.586$ or 3.414, whereas the other six error vectors correspond to only a single Euclidean distance.

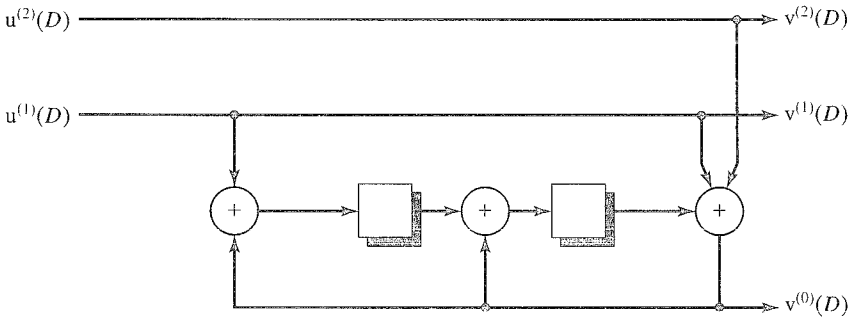
We now consider several possible code designs for naturally mapped 8-PSK and evaluate their MFSE distances d_{free}^2 using the error trellis labeled with MEWEs. We begin with a rate $R = 2/3$ convolutional code whose parity-check matrix in systematic feedback form is given by

$$\mathbb{H}(D) = \begin{bmatrix} 1/(D^2+D+1) & (D^2+1)/(D^2+D+1) & 1 \end{bmatrix}. \tag{18.16}$$

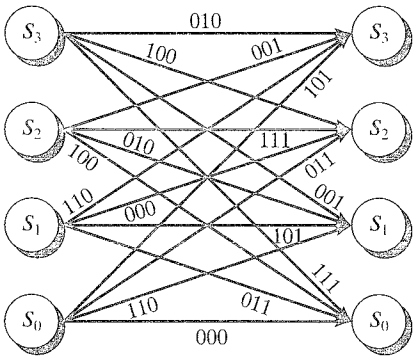
This is the optimum free distance $(3, 2, 1)$ code with constraint length $\nu = 2$ and $d_{H,free} = 3$. The encoder diagram is shown in Figure 18.7(a), the $2^\nu = 4$-state binary error trellis is given in Figure 18.7(b), and the modified error trellis labeled with the MEWEs for naturally mapped 8-PSK is shown in Figure 18.7(c).

From Figure 18.7(c) we see that the nonzero path associated with the sequence of states $S_0S_2S_3S_0$ results in an MFSE distance of $d_{free}^2 = 1.758$. Because this TCM scheme has a spectral efficiency of $\eta = 2$ bits/symbol, the appropriate uncoded system with which to compare is QPSK with an average signal energy $E_s = 1$. For this signal set, $d_{min}^2 = 2.0$, and thus naturally mapped TCM suffers a *coding loss* of $\gamma = 10\log_{10}(d_{free}^2/d_{min}^2) = 10\log_{10}(1.758/2) = -0.56$ dB in this case!

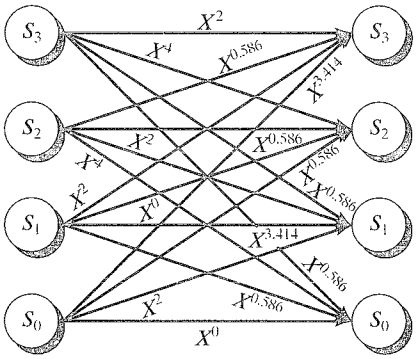
Now, we ask the question, Is it possible to achieve a positive coding gain without bandwidth expansion with 4-state, rate $R = 2/3$ coded, naturally mapped 8-PSK? Because the naturally mapped 8-PSK signal set is nonregular, we may find a better TCM scheme by considering suboptimum rate $R = 2/3$ codes. In addition, we may consider a rate $R = 1/2$ code with one uncoded information bit as equivalent to a rate $R = 2/3$ code; that is, both have a spectral efficiency of $\eta = 2$ bits/symbol when combined with 8-PSK modulation. To illustrate this latter approach, we consider



(a) Encoder diagram



(b) Error trellis



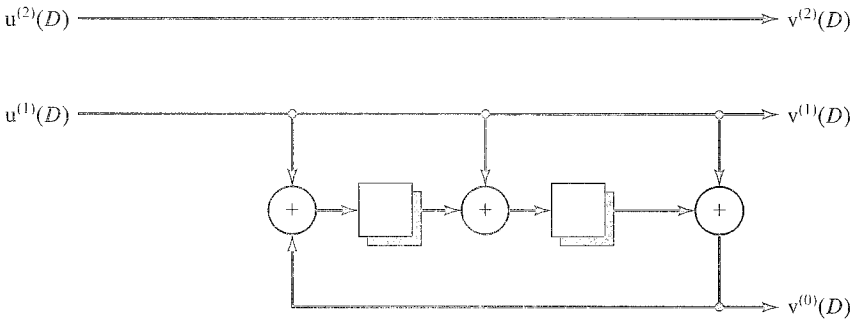
(c) Modified error trellis

FIGURE 18.7: Encoder diagram and error trellises for rate $R = 2/3$ coded 8-PSK.

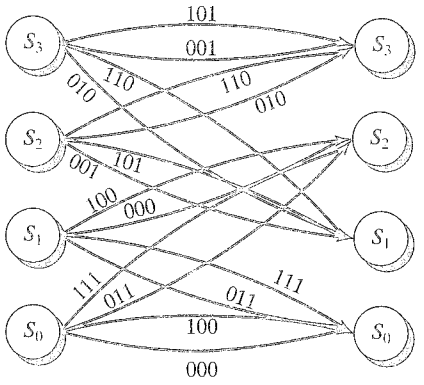
the same two $(2, 1, 2)$ codes as in Example 18.2, although this time we include an uncoded information bit and use the systematic feedback form of the encoders. Thus, the two rate $R = 1/2$ generator matrices are given by

$$\text{Code 1} : \mathbb{G}_1(D) = \left[\begin{array}{cc} 1 & (D^2 + D + 1)/(D^2 + 1) \end{array} \right], \tag{18.17a}$$

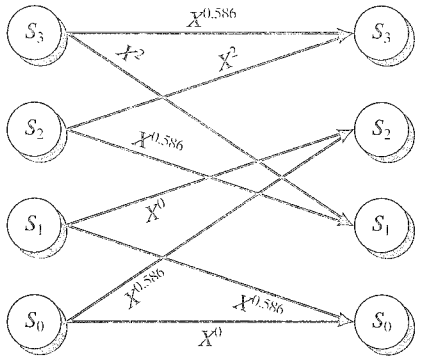
$$\text{Code 2} : \mathbb{G}_2(D) = \left[\begin{array}{cc} 1 & D/(D^2 + 1) \end{array} \right]. \tag{18.17b}$$



(a) Encoder diagram



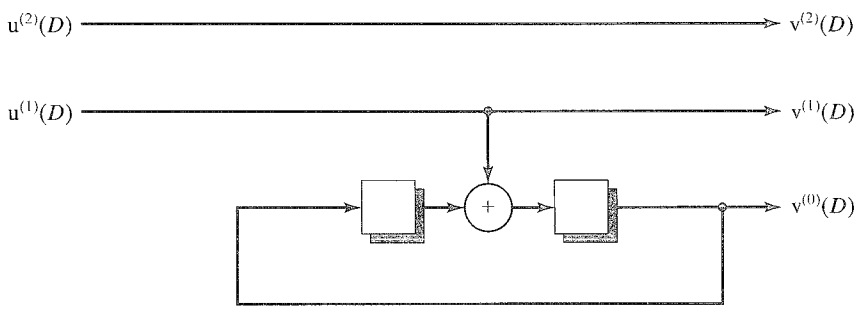
(b) Error trellis



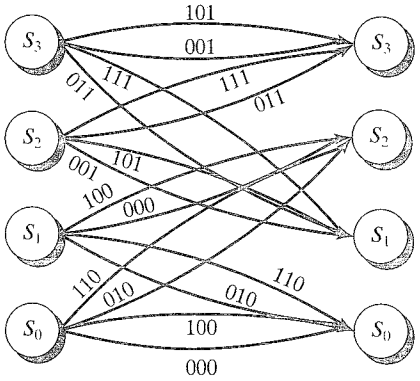
(c) Modified error trellis

FIGURE 18.8: Encoder diagram and error trellises for rate $R = 1/2$ coded 8-PSK (code 1).

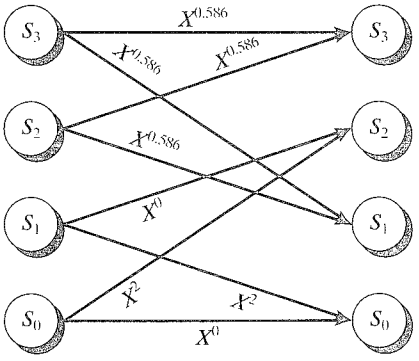
The encoder diagrams for these two codes are shown in Figures 18.8(a) and 18.9(a), their binary error trellises are given in Figures 18.8(b) and 18.9(b), and the modified error trellises labeled with MEWEs for naturally mapped 8-PSK are shown in Figures 18.8(c) and 18.9(c), respectively. The uncoded information bit is handled by adding a *parallel transition* to each branch in the binary trellis of the rate $R = 1/2$



(a) Encoder diagram



(b) Error trellis



(c) Modified error trellis

FIGURE 18.9: Encoder diagram and error trellises for rate $R = 1/2$ coded 8-PSK (code 2).

code. Thus, there are two branches connecting each pair of states in the binary error trellis, one for each of the two possible values of the uncoded bit. We follow the convention that the first bit listed on each branch of the binary error trellis is the uncoded bit. In the modified error trellis, we show only one branch connecting each pair of states; that is, it has the same structure as the trellis for the rate $R = 1/2$ code,

but its label is the minimum-weight label of the two MEWEs for the corresponding parallel branches in the binary error trellis. For example, in Figure 18.8(b), the two parallel branches connecting state S_0 to itself in the binary error trellis are labeled (000) and (100). Thus, in Figure 18.8(c), the two corresponding MEWEs are X^0 and X^4, and the single branch connecting state S_0 to itself is labeled X^0.

For any TCM scheme with parallel transitions, the calculation of the MFSE distance d^2_{free} involves two terms: (1) the MFSE distance δ^2_{free} between distinct trellis paths longer than one branch and (2) the MSE distance δ^2_{min} between distinct trellis paths one branch in length. Because δ^2_{free} is the free distance between trellis paths associated with the coded bits, it can be computed from the error trellis labeled with the MEWEs. Because δ^2_{min}, on the other hand, is the minimum distance between parallel transitions associated with the uncoded bits, it must be computed separately. Then, the overall MFSE distance is given by

$$d^2_{free} = \min\{\delta^2_{free}, \delta^2_{min}\}. \tag{18.18}$$

EXAMPLE 18.4 (Continued)

The parallel transition distance δ^2_{min} is independent of the code and depends only on the mapping used. From Figures 18.8(b) and 18.9(b) it is clear that the parallel branch labels always differ by the error vector (100). Thus, from Table 18.5 we conclude that $\delta^2_{min} = 4.0$. Now, we can see from Figures 18.8(c) and 18.9(c) that $\delta^2_{free} = 3(0.586) = 1.758$ for code 1, and $\delta^2_{free} = 2(2.0) + 0.586 = 4.586$ for code 2. Thus,

$$d^2_{free} = \min\{\delta^2_{free}, \delta^2_{min}\} = \min\{1.758, 4.0\} = 1.758, \quad \text{(code 1)} \tag{18.19a}$$

and

$$d^2_{free} = \min\{\delta^2_{free}, \delta^2_{min}\} = \min\{4.586, 4.0\} = 4.0, \quad \text{(code 2)} \tag{18.19b}$$

and the asymptotic coding losses (gains) compared with uncoded QPSK are $\gamma = -0.56$ dB for code 1 and $\gamma = +3.01$ dB for code 2. Thus, for the three different codes considered in this example, the best performance, and the only coding gain, is achieved by the suboptimum (in terms of $d_{H,free}$) rate $R = 1/2$ code with one uncoded bit. This simple 4-state code achieves a 3.01-dB coding gain compared with uncoded QPSK without bandwidth expansion. (Problem 18.9 illustrates that other mapping rules for 8-PSK result in less coding gain than natural mapping.)

The following remarks relate to Example 18.4:

- All mappings for the 8-PSK signal set are *nonregular*. Thus, the weight enumerating function $A(X)$ depends on the transmitted code sequence for all 8-PSK–based TCM systems; however, if the mapping is *uniform*, the average weight enumerating function $A_{av}(X)$ can be computed by labeling the branches of the error trellis with the AEWEs and using the transfer function method.

- Virtually all signal sets and mappings used in practical TCM systems are nonregular, although symmetries usually exist that allow a uniform mapping.
- The MEWEs can be used to compute the MFSE distance d^2_{free} of TCM systems with uniform mappings, as shown in Examples 18.2, 18.3, and 18.4; however, to determine the average weight enumerating function $A_{av}(X)$, the AEWEs must be used, as will be illustrated in Section 18.3.
- The critical advantage of naturally mapped 8-PSK over other uniform mappings for 8-PSK is that the error vector for all parallel transitions, $\mathbf{e} = (100)$, is assigned to the largest possible EW, $\Delta^2(\mathbf{e}) = 4.0$, by the natural mapping rule (see Problem 18.9). In other words, for 8-PSK, the MSE distance between signal points on parallel transition paths is maximized by natural mapping, thus minimizing the probability of a one-branch (parallel transition) error event.
- An exhaustive search of all possible 8-PSK TCM schemes with $\eta = 2$ bits/symbol and 4 states indicates that the best scheme is code 2 in Example 18.4, that is, the suboptimum rate $R = 1/2$ code with one uncoded bit, combined with natural mapping. This illustrates that, unlike code designs for binary modulation, the best TCM designs often include uncoded information bits resulting in parallel transitions in the trellis. (If uncoded bits are employed in the design of codes for binary modulation, the minimum free Hamming distance can never exceed the minimum Hamming distance between the parallel transition branches, which equals 1.)
- All the encoders in Example 18.4 were given in systematic feedback form. Equivalent nonsystematic feedforward encoders exist that give slightly different BER performance because of the different (encoder) mapping between information bits and code bits. Systematic feedback encoders are usually preferred in TCM system design because they represent a convenient canonical form for representing minimal rate $R = k/(k+1)$ encoders in terms of a single parity-check equation. This canonical representation simplifies the search for the best encoders.
- Larger coding gains can be achieved by employing more powerful codes, that is, longer constraint lengths. Tables of the best TCM code designs for a number of important signal constellations are given in Section 18.2.

The rate $R = k/(k+1)$ code lemma guarantees that if the mapping is uniform, any error sequence $\mathbf{e}(D)$ in the binary error trellis with a given Euclidean weight $\Delta^2[\mathbf{e}(D)]$ corresponds to a pair of signal sequences $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ in the trellis separated by a free squared Euclidean distance of $\Delta^2[\mathbf{e}(D)]$. In this case, the MFSE distance d^2_{free} of a TCM system can be computed using the method of Euclidean weights; however, if the mapping is not uniform, the rate $R = k/(k+1)$ code lemma does not hold, and the method of Euclidean weights will, in general, give only a lower bound on the actual d^2_{free}. This point is illustrated in the following example.

EXAMPLE 18.5 Nonuniform Mappings

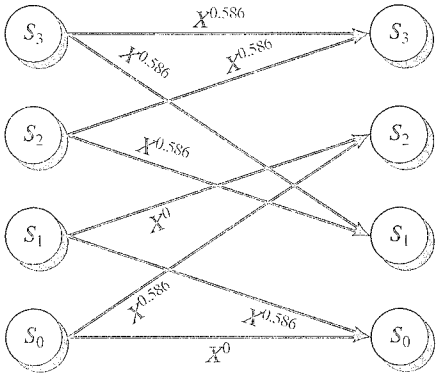
Consider the two nonuniform mappings of 8-PSK shown in Figures 18.2(b) and (c), along with their AEWEs and MEWEs listed in Tables 18.2(b) and (c). If these

mappings are used along with code 2 from Example 18.4, whose encoder diagram and binary error trellis are shown in Figures 18.9(a) and (b), respectively, we obtain the modified error trellises shown in Figure 18.10.

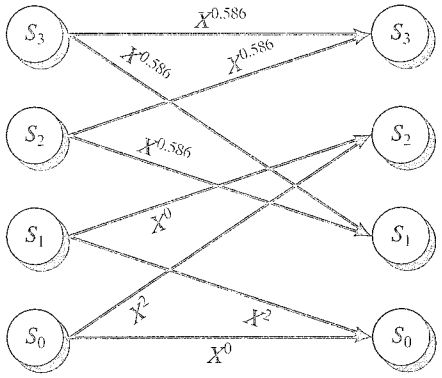
First, consider the nonuniform mapping of Figure 18.2(b) and Table 18.2(b), in which there is no isometry between the subsets $Q(0)$ and $Q(1)$. Let $\mathbf{e}(D) = \mathbf{e}_0 + \mathbf{e}_1 D + \mathbf{e}_2 D^2 + \mathbf{e}_3 D^3 = (110) + (011)D + (111)D^2 + (110)D^3$ be a path through the binary error trellis of Figure 18.9(b) that starts and ends in state S_0. From the modified error trellis of Figure 18.10(a), we can compute the EW of $\mathbf{e}(D)$ as follows:

$$\Delta^2[\mathbf{e}(D)] = 0.586 + 0.586 + 0.586 + 0.586 = 2.344. \tag{18.20}$$

For the rate $R = k/(k+1)$ code lemma to be satisfied, there must exist a pair of 4-branch trellis paths, $\mathbf{v}(D)$ and $\mathbf{v}'(D)$, starting and stopping in the same state, that differ by the error path $\mathbf{e}(D)$ and whose corresponding signal sequences $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ are distance 2.344 apart. From Figure 18.2(b) and Table 18.2(b) we see that the desired path pair must start with the branches $\mathbf{v}_0 = (101)$ and $\mathbf{v}'_0 = (011)$, since this is the only pair of binary labels such that $\mathbf{e}_0 = \mathbf{v}_0 \oplus \mathbf{v}'_0 = (110)$, and



(a) Mapping of Figure 18.2(b)



(b) Mapping of Figure 18.2(c)

FIGURE 18.10: Modified error trellises for two nonuniform mappings.

$d_E^2[f(\mathbf{v}_0), f(\mathbf{v}'_0)] = d_E^2[\mathbf{y}_0, \mathbf{y}'_0] = 0.586$. (The branches assigned to $\mathbf{v}_0$ and $\mathbf{v}'_0$ can also be reversed without changing the result.) Thus, from Figure 18.9(b), the path pair must start either from state S_2 or from state S_3. Similarly, the next three pairs of branch labels must be $\mathbf{v}_1 = (001)$ and $\mathbf{v}'_1 = (010)$, $\mathbf{v}_2 = (111)$ and $\mathbf{v}'_2 = (000)$, and $\mathbf{v}_3 = (101)$ and $\mathbf{v}'_3 = (011)$ (or the reverse of these labels) to satisfy the distance conditions; but a close examination of Figure 18.9(b) reveals that no pair of paths with these labels and starting either from state S_2 or from state S_3 exists in the trellis. Thus, it is impossible to find a pair of paths $\mathbf{v}(D)$ and $\mathbf{v}'(D)$, starting and stopping in the same state, that differ by the error path $\mathbf{e}(D)$ and whose corresponding signal sequences $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ are distance 2.344 apart; hence, the rate $R = k/(k+1)$ code lemma is not satisfied.

Next, consider the nonuniform mapping of Figure 18.2(c) and Table 18.2(c), in which there is an isometry between the subsets $Q(0)$ and $Q(1)$. Let $\mathbf{e}(D) = \mathbf{e}_0 + \mathbf{e}_1 D + \mathbf{e}_2 D^2 + \mathbf{e}_3 D^3 + \mathbf{e}_4 D^4 = (110) + (101)D + (100)D^2 + (101)D^3 + (110)D^4$ be a path through the binary error trellis of Figure 18.9(b) that starts and ends in state S_0. From the modified error trellis of Figure 18.10(b) we can compute the EW of $\mathbf{e}(D)$ as follows:

$$\Delta^2[\mathbf{e}(D)] = 2.0 + 0.586 + 2.0 + 0.586 + 2.0 = 7.172. \tag{18.21}$$

For the rate $R = k/(k+1)$ code lemma to be satisfied, there must exist a pair of 5-branch trellis paths, $\mathbf{v}(D)$ and $\mathbf{v}'(D)$, starting and stopping in the same state, that differ by the error path $\mathbf{e}(D)$ and whose corresponding signal sequences $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ are distance 7.172 apart. From Figure 18.2(c) and Table 18.2(c) we see that the desired path pair must start either with the branch pair $\mathbf{v}_0 = (000)$ and $\mathbf{v}'_0 = (110)$ or with the branch pair $\mathbf{v}_0 = (100)$ and $\mathbf{v}'_0 = (010)$, since these are the only pairs of binary labels such that $\mathbf{e}_0 = \mathbf{v}_0 \oplus \mathbf{v}'_0 = (110)$, and $d_E^2[f(\mathbf{v}_0), f(\mathbf{v}'_0)] = d_E^2[\mathbf{y}_0, \mathbf{y}'_0] = 2.0$. From Figure 18.9(b) we see that the path pair must start either from state S_0 or from state S_1 in both cases. As in the previous case considered, the next four pairs of branch labels are similarly constrained to satisfy the distance conditions. It is easily seen that there is only one possible branch pair corresponding to the error vector $\mathbf{e}_1 = \mathbf{e}_3 = (101)$, but there are two possible branch pairs corresponding to the error vectors $\mathbf{e}_2 = (100)$ and $\mathbf{e}_4 = (110)$. Again, a close examination of Figure 18.9(b) reveals that no pair of paths with these labels and starting either from state S_0 or from state S_1 exists in the trellis. Thus, it is impossible to find a pair of paths $\mathbf{v}(D)$ and $\mathbf{v}'(D)$, starting and stopping in the same state, that differ by the error path $\mathbf{e}(D)$ and whose corresponding signal sequences $\mathbf{y}(D)$ and $\mathbf{y}'(D)$ are distance 7.172 apart; hence, the rate $R = k/(k+1)$ code lemma is again not satisfied.

Example 18.5 leads to the following observations:

- When the mapping is nonuniform, there are still many error sequences for which the rate $R = k/(k+1)$ code lemma is satisfied; however, Example 18.5 illustrates that this is not true for all error sequences.
- Example 18.5 shows that an isometry between the subsets $Q(0)$ and $Q(1)$ is necessary, but not sufficient, to guarantee that the rate $R = k/(k+1)$ code

lemma is satisfied. See Problem 18.10 for an example illustrating this fact that uses a different signal constellation.

- Because the rate $R = k/(k+1)$ code lemma is not satisfied for nonuniform mappings, the method of Euclidean weights provides only a lower bound on d^2_{free} in this case. This is also true of the method to be presented in Section 18.3 for determining the AWEF $A_{av}(X)$ of a TCM system from the AEWEs.
- Using a nonuniform mapping does not necessarily imply an inferior TCM system, just one that is more difficult to analyze. In this case, a supertrellis of $(2^\nu)^2 = 2^{2\nu}$ states must be used to determine the set of distances between all possible path pairs; however, uniform mappings result in the best designs for most practical TCM systems (see Problem 18.11).
- A more stringent uniformity condition, called *geometric uniformity*, was introduced by Forney [23]. When this condition is satisfied, the computation of weight enumerating functions is simplified, but many practical TCM systems are not geometrically uniform.

Examples 18.2, 18.3, and 18.4 illustrate two basic rules of good TCM system design:

Rule 1: Signal set mapping should be designed so that the MSE distance between parallel transition branches is maximized.

Rule 2: The convolutional code should be designed so that the branches in the modified error trellis leaving and entering the same state have the largest possible MSE distance.

A general block diagram of a TCM system is shown in Figure 18.11. At each time unit l, a total of k information bits, $\mathbb{u}_l = \left(u_l^{(k)}, u_l^{(k-1)}, \cdots, u_l^{(1)}\right)$, enter the

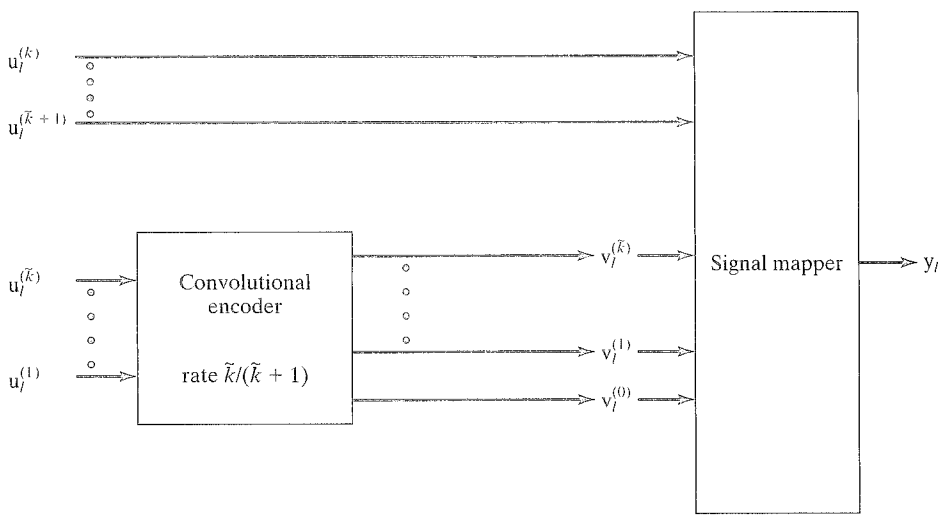


FIGURE 18.11: General TCM encoder diagram and signal mapper.

system. Of these, a total of $\tilde{k} \leq k$ bits, namely, $u_l^{(\tilde{k})}, u_l^{(\tilde{k}-1)}, \cdots, u_l^{(1)}$, enter a rate $\tilde{R} = \tilde{k}/(\tilde{k}+1)$ systematic feedback convolutional encoder, producing the output bits $v_l^{(\tilde{k})}, v_l^{(\tilde{k}-1)}, \cdots, v_l^{(1)}, v_l^{(0)}$, where $v_l^{(0)}$ is the parity bit, and $v_l^{(\tilde{k})}, v_l^{(\tilde{k}-1)}, \cdots, v_l^{(1)}$ are information bits. These $\tilde{k}+1$ bits enter the signal mapper along with the $k-\tilde{k}$ uncoded information bits $u_l^{(k)} = v_l^{(k)}, u_l^{(k-1)} = v_l^{(k-1)}, \cdots, u_l^{(\tilde{k}+1)} = v_l^{(\tilde{k}+1)}$. Finally, the $k+1$ bit vector $\mathbf{v}_l = (v_l^{(k)}, v_l^{(k-1)}, \cdots, v_l^{(1)}, v_l^{(0)})$ is mapped into one of the $M = 2^{k+1}$ possible points in the signal set S. If $k = \tilde{k}$, then there are no uncoded information bits and no parallel transitions in the trellis diagram.

In the next section we will study a technique called *mapping by set partitioning* [1] in which the $\tilde{k}+1$ coded bits $v_l^{(\tilde{k})}, v_l^{(\tilde{k}-1)}, \cdots, v_l^{(1)}, v_l^{(0)}$ are used to select a subset of size $2^{k-\tilde{k}}$ from the signal set S, and then the $k-\tilde{k}$ uncoded bits $v_l^{(k)}, v_l^{(k-1)}, \cdots, v_l^{(\tilde{k}+1)}$ are used to choose a particular signal point from within the selected subset. Thus, a path through the trellis indicates the particular sequence of selected subsets, and the $2^{k-\tilde{k}}$ parallel transitions associated with each trellis branch indicate the choice of signal points within the corresponding subset. This mapping technique allows us to design TCM systems that satisfy the two basic design rules noted.

18.2 TCM CODE CONSTRUCTION

There are three basic steps in designing a TCM system:

1. Signal set selection
2. Labeling of the signal set
3. Code selection

A signal set is chosen primarily to satisfy system constraints on spectral efficiency and modulator design. For example, if a spectral efficiency of $\eta = k$ bits/symbol is desired, a signal set with 2^{k+1} points must be selected. Similarly, if, because of nonlinearities in the transmission path (e.g., a traveling wave tube amplifier), a constant-amplitude signaling scheme is required, then a PSK signal set must be chosen. If amplitude modulation can be accommodated, then a rectangular or QAM signal set will give better performance. Several typical signal sets were shown in Figure 18.1. As an example, consider a linear transmission path and a spectral efficiency requirement of $\eta = 4$ bits/symbol, the specifications for the CCITT V.32 modem standard that can achieve data rates up to 14.4 Kbps over voice-grade telephone lines. In this case, the 32-CROSS signal set was chosen for implementation.

The next step in the design process is to assign binary labels, representing encoder output blocks, to the signal points in such a way that the MFSE distance d_{free}^2 of the overall TCM system is maximized. These labels are assigned by using a technique called *mapping by set partitioning* [1]. This technique successively partitions the signal set into smaller subsets of equal size, thereby generating a tree structure in which each signal point corresponds to a unique path through the tree. If binary partitioning is used, that is, at every level in the partitioning tree each subset from the previous level is divided into two subsets of equal size, the tree has

$k + 1$ levels. Thus, each path through the tree can be represented by a $(k + 1)$-bit label, which can then be assigned to the corresponding signal point. To maximize d^2_{free}, the partitioning must be done in such a way that the two basic rules for good TCM system design discussed in Section 18.1 are satisfied. This requires that the *minimum squared subset distance* (MSSD) Δ^2_p, that is, the MSE distance between signal points within the same subset, be maximized at each level p of the partition tree. The approach is illustrated with two examples.

EXAMPLE 18.6 Partitioning of 8-PSK

Consider the binary partitioning tree for the 8-PSK signal set S shown in Figure 18.12. Level 0 of the partitioning tree contains the full 8-PSK signal set S. Assuming unit energy signals, the MSSD at level 0 was computed in (18.10a) and is denoted by $\Delta^2_0 = 0.586$. (Δ^2_0 is the same as the previously defined Δ^2_{min}, the MSE distance between signal points. The notation Δ^2_0 indicates that this term corresponds to the MSSD at level 0 in the set-partitioning tree.) Label bit $v^{(0)}$ then divides the set S into two subsets, $Q\left(v^{(0)}\right) = Q(0)$ and $Q(1)$, each containing four signal points such that the MSSD of both subsets at level 1 is given by $\Delta^2_1 = 2.0$. It is important to point out here two properties of this partition:

1. There is no partition of 8-PSK into two equal-size subsets that achieves a larger MSSD.
2. Subset $Q(0)$ is isomorphic to subset $Q(1)$ in the sense that $Q(1)$ can be obtained from $Q(0)$ by rotating the points in $Q(0)$ by 45°.

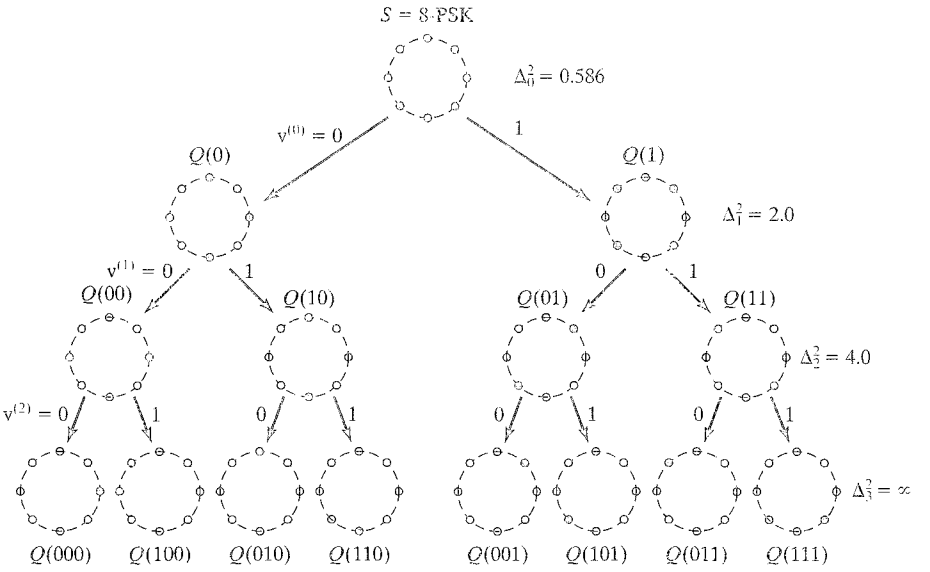


FIGURE 18.12: Partitioning of 8-PSK.

These two properties of 8-PSK partitioning, namely, maximizing the MSSD and maintaining an isometry among all subsets at the same level, are characteristic of most practical signal set partitionings. The isometry property implies that the MSSDs are the same for all subsets at a given level.

Continuing with the example, we see that label bit $v^{(1)}$ now divides each of the subsets $Q(0)$ and $Q(1)$ at level 1 into two subsets, containing two signal points each, such that $\Delta_2^2 = 4.0$ for each subset at level 2. We see again at level 2 that the subset distance has been increased and that the four subsets are isomorphic and thus have the same MSSD. The four subsets are denoted as $Q\left(v^{(1)}v^{(0)}\right) = Q(00)$, $Q(10)$, $Q(01)$, and $Q(11)$, representing the four possible values of the binary label $(v^{(1)}v^{(0)})$. Finally, label bit $v^{(2)}$ divides each of the subsets at level 2 into two subsets containing one signal point each at level 3. This is the lowest level in the partitioning tree, and the MSSD Δ_3^2 at this level is infinite, since there is only one signal point in each subset. The eight subsets at level 3, $Q\left(v^{(2)}v^{(1)}v^{(0)}\right) = Q(000)$, $Q(100)$, $Q(010)$, $Q(110)$, $Q(001)$, $Q(101)$, $Q(111)$, and $Q(011)$ are represented by a unique binary label $(v^{(2)}v^{(1)}v^{(0)})$ that corresponds to a path through the partitioning tree. This binary label then defines the mapping between a 3-bit encoder output block and a corresponding signal point in the 8-PSK signal set.

As noted in Section 18.1, a TCM system using 8-PSK can employ either a rate $R = 2/3$ code or a rate $R = 1/2$ code with one uncoded bit. To best describe the code design procedure, we consider the case of a 4-state, rate $R = 1/2$ code with one uncoded bit in the remainder of this example. In this case only the first two levels of the partitioning tree are used, and each of the four subsets at level 2, that is, the subsets $Q(00)$, $Q(10)$, $Q(01)$, and $Q(11)$, contains two signal points separated by the distance $\Delta_2^2 = 4.0$. First, the two coded bits $(v^{(1)}v^{(0)})$ are used to select a subset, and the uncoded bit $v^{(2)}$ is then used to select the signal point to be transmitted. This means that each branch in the code trellis, which represents a parallel transition, is assigned one of the level-2 subsets $Q(00)$, $Q(10)$, $Q(01)$, or $Q(11)$ with subset distance $\Delta_2^2 = 4.0$. Note that, since Δ_2^2 was maximized by the partitioning procedure, this guarantees that the MSE distance δ_{min}^2 between parallel transition branches is maximized, thus satisfying rule 1 for good TCM code design.

Now, we consider the assignment of the level-2 subsets $Q(00)$, $Q(10)$, $Q(01)$, and $Q(11)$ to the branches of the code trellis. Note that the trellis is completely defined by the set of branches leaving each state. In this example there are a total of $2^{\tilde{k}} = 2$ branches leaving each of the $2^{\nu} = 4$ states. Because there are only four level-2 subsets from which to choose, exactly half of these subsets must be assigned to each set of two branches leaving a state. From Figure 18.12 we can see that the distance between diverging branches is maximized if the two branches leaving each state are assigned subsets belonging to the same level-1 subset, $Q(0)$ or $Q(1)$. In other words, the level-2 subsets $Q(00)$ and $Q(10)$ (belonging to $Q(0)$) should be paired, and the level-2 subsets $Q(01)$ and $Q(11)$ (belonging to $Q(1)$) should be paired. To ensure that the distance between remerging branches will also be maximized, the same level-2 subset pair (either $\{Q(00), Q(10)\}$ or $\{Q(01), Q(11)\}$ should be used to label the diverging branches of both states in each trellis "butterfly," and the level-2 subset pair should be assigned in such a way that the two remerging branches of

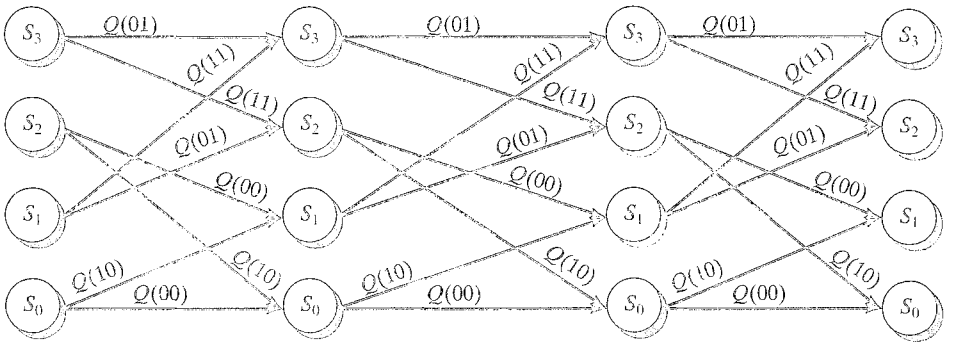


FIGURE 18.13: Branch labels for 4-state, rate $R = 1/2$ coded 8-PSK.

both states in the butterfly are labeled by the same pair (see Figure 18.13).[2] Finally, in order to ensure that all signal points are used equally often, subset $Q(0)$ (pair $\{Q(00), Q(10)\}$) should be assigned to half the states (one butterfly), and subset $Q(1)$ (pair $\{Q(01), Q(11)\}$) to the other half (the other butterfly). Because each of the level-1 subsets ($Q(0)$ and $Q(1)$) contains $2^k = 4$ signal points, and their MSSD $\Delta_1^2 = 2.0$ is the largest possible for a subset of four points, this guarantees that the MSE distance between branches leaving and entering the same state is maximized, thus satisfying rule 2 for good TCM system design. The final labeling of branches for this example is shown in Figure 18.13, where the trellis represents a 4-state, rate $R = 1/2$, feedforward encoder.

The following remarks relate to Example 18.6:

- The assignment of signal points from only one level-1 subset ($Q(0)$ or $Q(1)$) to all the branches leaving and entering each state implies that the code bit $v^{(0)}$, which determines the subset chosen at level 1, must be the same for each set of branches leaving or entering a particular state. This places some restrictions on the codes that yield good TCM designs.

- In general, half of the $2^{\nu-\tilde{k}}$ butterflies in the code trellis are assigned to subset $Q(0)$ and the other half to subset $Q(1)$. This ensures that all signal points are used with equal probability.

- It is always possible, in the manner described here, to ensure that the diverging and remerging branch distance equals Δ_1^2, thus guaranteeing that $\delta_{free}^2 \geq 2\Delta_1^2$, except in the special case $\nu = \tilde{k}$. In this case, the trellis is fully connected and contains only a single butterfly, thus implying that either the diverging or

---

[2] A trellis section of any $(n, \tilde{k}, \nu)$ encoder can be decomposed into a set of $2^{\nu-\tilde{k}}$ fully connected subtrellises containing $2^{\tilde{k}}$ states each. These subtrellises, called *butterflies*, connect a subset of $2^{\tilde{k}}$ states at one time to a (in general, different) subset of $2^{\tilde{k}}$ states at the next time. For example, in Figure 18.13, the pair ($2^{\tilde{k}} = 2$) of states S_0 and S_2 connect to the state pair S_0 and S_1, forming one of the $2^{\nu-\tilde{k}} = 2$ butterflies, and the other butterfly is formed by the state pair S_1 and S_3 connecting to the state pair S_2 and S_3.

remerging distance must equal only Δ_0^2. Hence, 2-state trellises ($\nu = 1$) with rate $R = 1/2$ codes ($\tilde{k} = 1$) do not yield good TCM designs.

- If a rate $R = 2/3$ code is used in the preceding example, then $k = \tilde{k} = \nu$, and the trellis is fully connected. This implies that δ_{free}^2 is at most equal to $\Delta_0^2 + \Delta_1^2 = 2.586$, no matter which code is selected. Thus, for 4-state 8-PSK TCM schemes with $\eta = 2$ bits/symbol, rate $R = 2/3$ codes are suboptimal compared with rate $R = 1/2$ codes with one uncoded bit.

- In the partitioning of 8-PSK, the two subsets at level 1 are equivalent to QPSK signal sets, and the four subsets at level 2 are equivalent to BPSK signal sets. This isometry between subsets at the same level of the partition tree is characteristic of all PSK signal set partitionings.

- For the 8-PSK partition shown in Figure 18.12, mapping by set partitioning results in the natural mapping rule discussed in Section 18.1. If the order of the subsets at any level in the partitioning tree is changed, the resulting mapping is isomorphic to natural mapping.

- Mapping by set partitioning always results in the distance relation $\Delta_0^2 \leq \Delta_1^2 \leq \cdots \leq \Delta_k^2$, which, along with the proper assignment of subsets to trellis branches, guarantees that the two rules of good TCM system design are satisfied.

- The separate tasks assigned to coded and uncoded bits by set partitioning, namely, the selection of subset labels for the trellis branches and the selection of a signal point from a subset, respectively, imply that the general TCM encoder and mapper in Figure 18.11 can be redrawn as shown in Figure 18.14.

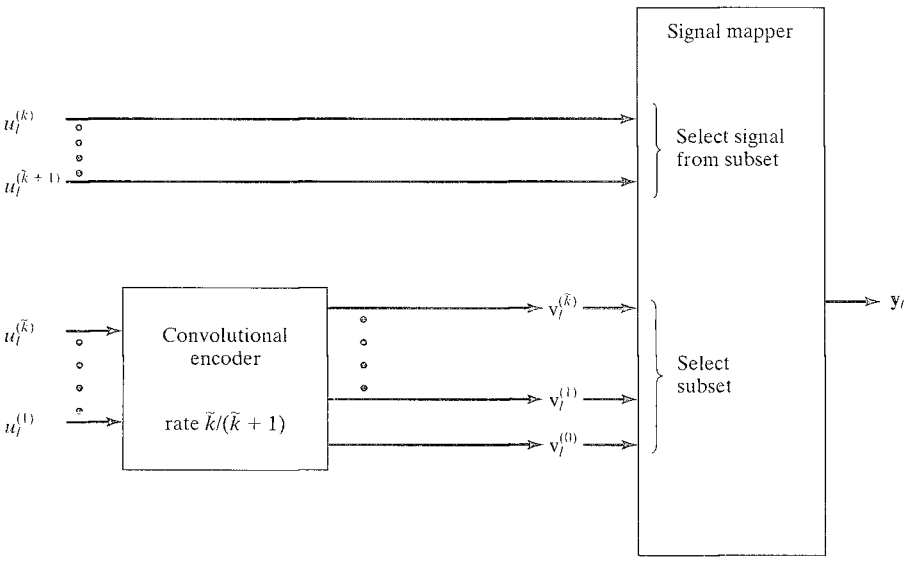


FIGURE 18.14: Set-partitioning TCM encoder diagram and signal mapper.

If $\tilde{k} = k$, then there are no parallel transitions, and the subset labels on the trellis become signal point labels.

EXAMPLE 18.7 Partitioning of 16-QAM

As a second example of set partitioning, we consider the 16-QAM signal set, denoted by S, shown in Figure 18.15. Letting Δ_0^2 represent the MSSD at level 0, we see that the average signal energy is given by the expression

$$\begin{aligned} E_s &= (1/16)\left\{4\left[(\Delta_0/2)^2 + (\Delta_0/2)^2\right] + 8\left[(3\Delta_0/2)^2 + (\Delta_0/2)^2\right]\right. \\ &\quad \left. + 4\left[(3\Delta_0/2)^2 + (3\Delta_0/2)^2\right]\right\} \\ &= (1/16)\left[2\Delta_0^2 + 20\Delta_0^2 + 18\Delta_0^2\right] = 5\Delta_0^2/2. \end{aligned} \tag{18.22}$$

Thus, $\Delta_0^2 = 2/5$ if the average energy $E_s = 1$. At level 1 of the partitioning tree, we obtain the subsets $Q(0)$ and $Q(1)$, each isomorphic to an 8-AM/PM constellation, and it is easy to see that $\Delta_1^2 = 2\Delta_0^2$. Continuing down the partitioning tree, we obtain four subsets at level 2, each isomorphic to 4-QAM, with $\Delta_2^2 = 2\Delta_1^2$; eight subsets at level 3, each isomorphic to 2-AM, with $\Delta_3^2 = 2\Delta_2^2$; and, finally, the 16 signal points at level 4, each labeled according to the set-partitioning mapping rule.

The following observations relate to Example 18.7:

- The 16-QAM signal set can be considered a multidimensional version of 4-AM, that is, 2×4-AM.
- In the 16-QAM case, the MSSD doubles at each level of the partitioning tree; that is, $\Delta_i^2 = 2\Delta_{i-1}^2$, $i = 1, 2, \cdots, k$. This is characteristic of most partitionings of rectangular-type signal constellations used in practice.
- 16-QAM is a (translated) subset of the two-dimensional integer lattice $\mathbb{Z}^2$, and the subsets at each level of the partitioning are isomorphic.
- It is not always possible to partition signal sets based on a lattice in such a way that all subsets at a given partition level are isomorphic. In this case, although the subsets are no longer distance invariant, they all still have the same MSSD Δ_i^2. An example of this situation is shown in Section 18.4 for the 32-CROSS constellation.
- TCM systems based on 16-QAM modulation can employ code rates R of 3/4 or 2/3 with one uncoded bit, or 1/2 with two uncoded bits.

We now consider the last step in the design process, that of code selection. Assume that the code is generated by a rate $R = \tilde{k}/(\tilde{k}+1)$ systematic feedback convolutional encoder with parity-check matrix

$$\mathbb{H}(D) = \left[\mathbb{h}^{(\tilde{k})}(D)/\mathbb{h}^{(0)}(D) \cdots \mathbb{h}^{(1)}(D)/\mathbb{h}^{(0)}(D) \quad 1\right], \tag{18.23}$$

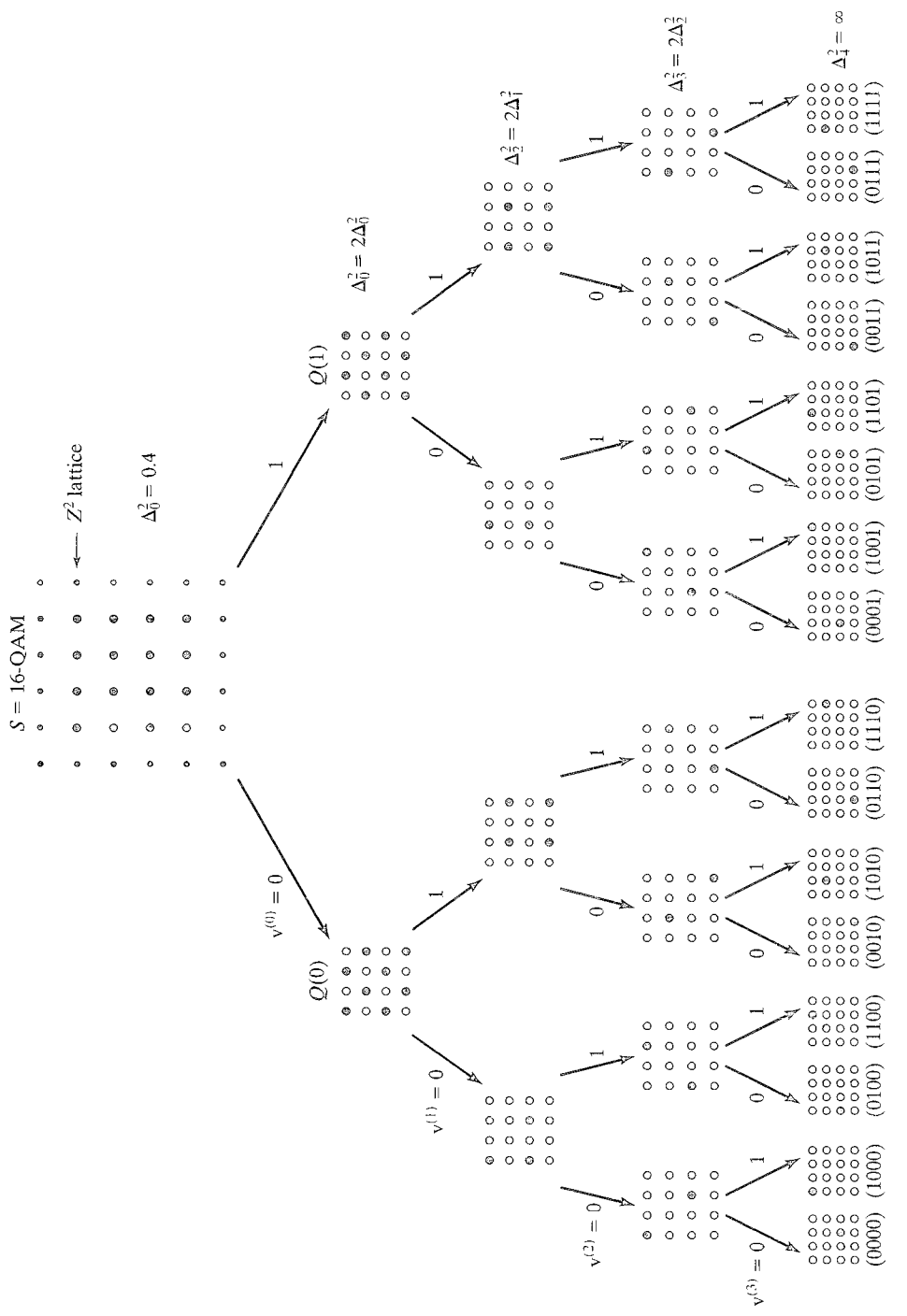


FIGURE 18.15: Partitioning of 16-QAM.

where $\mathbb{h}^{(j)}(D) = h_0^{(j)} + h_1^{(j)}D + \cdots + h_\nu^{(j)}D^\nu$, $j = 0, 1, \cdots, \tilde{k}$; ν is the constraint length; and $h_0^{(0)} = h_\nu^{(0)} = 1$ is required for a minimal encoder realization. Then, the vector $\mathbb{V}(D) = [\mathbb{v}^{(\tilde{k})}(D), \cdots, \mathbb{v}^{(1)}(D), \mathbb{v}^{(0)}(D)]$ is a codeword if and only if

$$\mathbb{V}(D)\mathbb{H}^T(D) = \sum_{(0 \le j \le \tilde{k})} \mathbb{v}^{(j)}(D)\mathbb{h}^{(j)}(D) = \mathbb{0}(D), \tag{18.24}$$

where $\mathbb{v}^{(j)}(D) \triangleq v_0^{(j)} + v_1^{(j)}D + v_2^{(j)}D^2 + \cdots + v_l^{(j)}D^l + \cdots$, $j = 0, 1, \cdots, \tilde{k}$; $\mathbb{v}_l = [v_l^{(\tilde{k})}, \cdots, v_l^{(1)}, v_l^{(0)}]$ represents the encoder output that selects the subset at time l; $\sum$ denotes modulo-2 addition; and $\mathbb{0}(D)$ represents the all-zero sequence. The general realization of a rate $R = \tilde{k}/(\tilde{k}+1)$ systematic feedback convolutional encoder with parity-check matrix given by (18.23) is shown in Figure 18.16(a).

We now place some restrictions on the general encoder realization of Figure 18.16(a) that are appropriate for good TCM code design. Recall the set-partitioning requirement that the parity bit $v^{(0)}$ be the same for all branches leaving and entering a given state. It is easy to see from Figure 18.16(a) that to guarantee that $v^{(0)}$ be the same for all branches leaving a state, there must be no connections from any information sequence to the shift-register output; that is, we require that $h_0^{(1)} = h_0^{(2)} = \cdots = h_0^{(\tilde{k})} = 0$. Also, since $v^{(0)}$ is an input to the first (leftmost) register stage, and the output of the first register stage must be the same for all branches entering a given state, to guarantee that $v^{(0)}$ is the same for all branches entering a state, there must be no connections from an information sequence to the shift-register input; that is, we require that $h_\nu^{(1)} = h_\nu^{(2)} = \cdots = h_\nu^{(\tilde{k})} = 0$. These restrictions lead to the systematic feedback convolutional encoder realization shown in Figure 18.16(b) that is used to search for good TCM code designs.

The criterion for selecting good codes, based on the approximate expression for event-error probability given in (18.5), is to select codes that maximize the MFSE distance d_{free}^2 and minimize the average number of nearest neighbors $A_{d_{free}}$. An appropriate search algorithm must first find the codes with the largest d_{free}^2 and then select those with the smallest $A_{d_{free}}$. Assuming that the MSE distance δ_{min}^2 between parallel transitions is computed separately, we can use (18.13) to express the MFSE distance between trellis paths as

$$\delta_{free}^2 = \min_{\mathbb{e}(D) \ne \mathbb{0}(D)} \Delta^2[\mathbb{e}(D)] = \min_{\mathbb{e}(D) \ne \mathbb{0}(D)} \sum_l \Delta^2(\mathbb{e}_l). \tag{18.25}$$

It is also possible to compute a lower bound on δ_{free}^2 directly from the binary error trellis of the code and the MSSDs Δ_p^2 in the set-partitioning tree using the following lemma.

LEMMA 18.2 (SET-PARTITIONING LEMMA) [1] Let $q(\mathbb{e})$ denote the number of trailing zeros in the error vector $\mathbb{e}$, for example, $q(e^{(k)}, \cdots, e^{(3)}, 1, 0, 0) = 2$. Then,

$$\Delta^2(\mathbb{e}) \ge \Delta_{q(\mathbb{e})}^2. \tag{18.26}$$

Proof. The proof of (18.26) follows from the fact that if $\mathbb{v}$ and $\mathbb{v}' = \mathbb{v} \oplus \mathbb{e}$ represent two trellis branch labels, then their corresponding signal point

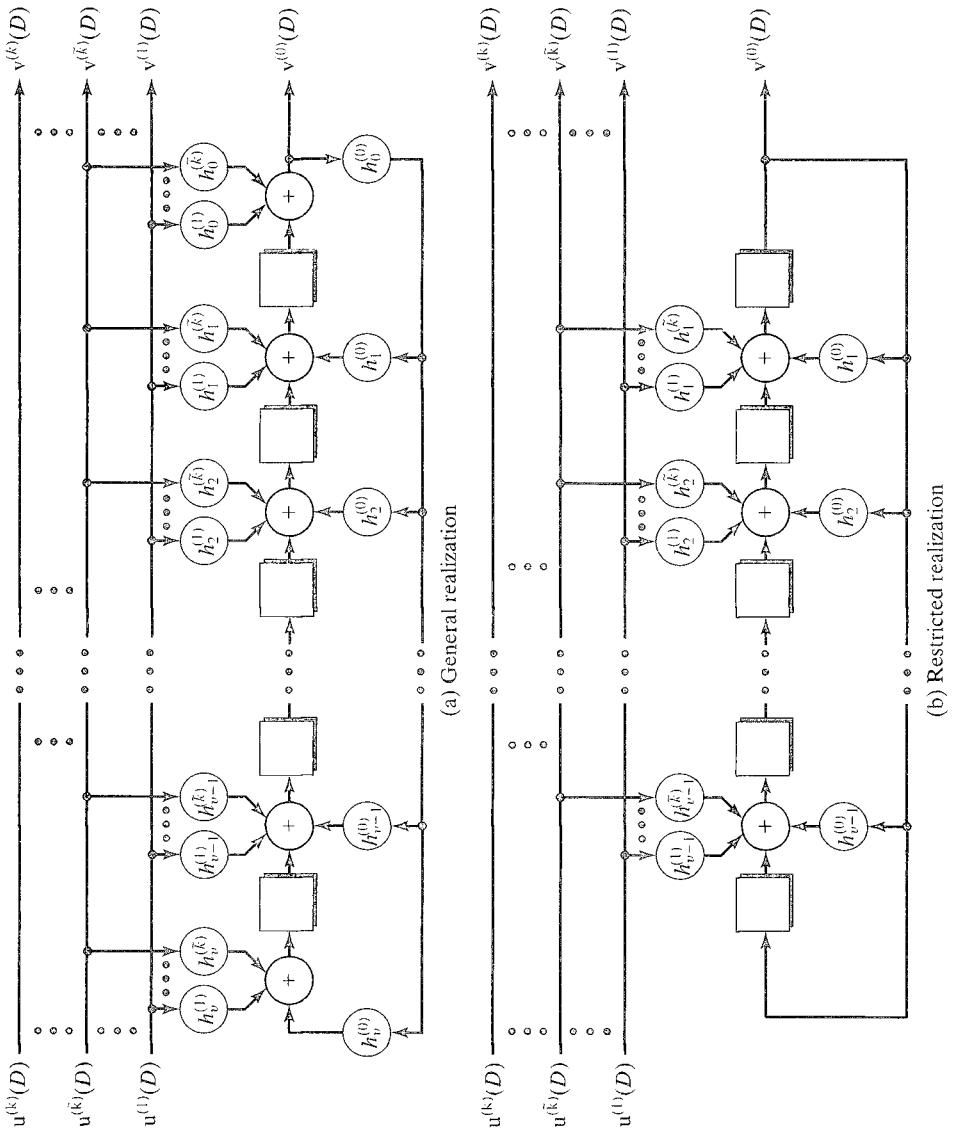


FIGURE 18.16: Two systematic feedback convolutional encoder realizations.

labels in the set-partitioning tree will agree in the trailing $q(\mathbf{e})$ positions. This implies that they follow the same path through the tree for the first $q(\mathbf{e})$ levels, and thus $\Delta^2_{\mathbf{v}}(\mathbf{e}) \geq \Delta^2_{q(\mathbf{e})}$. Because this condition holds for all $\mathbf{v}$, $\Delta^2(\mathbf{e}) = \min_{\mathbf{v}} \Delta^2_{\mathbf{v}}(\mathbf{e}) \geq \Delta^2_{q(\mathbf{e})}$. Q.E.D.

Using the set-partitioning lemma, we can now write

$$\delta^2_{free} \geq \min_{\mathbf{e}(D) \neq \mathbf{0}(D)} \sum_{l} \Delta^2_{q(\mathbf{e}_l)}. \tag{18.27}$$

The following comments apply to the set-partitioning lemma.

- For the case $\mathbf{e} = \mathbb{0}$, we take $\Delta^2(\mathbf{e}) = \Delta^2_{q(\mathbb{0})} = 0$.
- Inequality (18.26) is satisfied with equality for most $\mathbf{e}$. For example, the only exception for 8-PSK is $\Delta^2(101) > \Delta^2_0$, and the only exceptions for 16-QAM are $\Delta^2(1001) > \Delta^2_0$, $\Delta^2(1101) > \Delta^2_0$, and $\Delta^2(1111) > \Delta^2_0$.
- Inequality (18.27) is almost always satisfied with equality, since there are usually several paths $\mathbf{e}(D) \neq \mathbb{0}(D)$ that achieve the minimum value of δ^2_{free}, including at least one that does not include any error vector $\mathbf{e}_l$ for which (18.26) is not satisfied with equality.
- Thus, either (18.25) or (18.27) can be used to compute the MFSE distance between trellis paths; but (18.27) is simpler, since it does not require computation of the Euclidean weight of every error vector. This can be especially advantageous for large signal sets or if codes are designed based on a lattice partitioning without specifying a particular signal set.

Sets of optimum TCM code designs based on the foregoing search procedure are listed in Tables 18.6(a)–(d). The codes were found by computer search [4]. Each table gives the following information:

- The MSSDs Δ^2_i, $i = 0, 1, \cdots, k$.
- The encoder constraint length ν.
- The number of coded information bits $\tilde{k}$.
- The parity-check coefficients $\mathbb{h}^{(j)} = \left[h^{(j)}_\nu, h^{(j)}_{\nu-1}, \cdots, h^{(j)}_1, h^{(j)}_0\right]$, $j = 0, 1, \cdots, \tilde{k}$, in octal form.
- The MFSE distance d^2_{free}. An asterisk $(*)$ indicates that d^2_{free} occurs only along parallel transitions; that is, $\delta^2_{free} > \delta^2_{min}$. In Tables 18.6(a) and (b), the ratio of d^2_{free} to Δ^2_0, the MSSD at level 0, assuming the average energy $E_s = 1$, is given. (In these cases, Δ^2_0 varies with the signal constellation considered, but the ratio d^2_{free}/Δ^2_0 is constant.)
- The asymptotic coding gain in decibels compared with an uncoded modulation system with the same spectral efficiency. The notation denotes the two signal constellations being compared; for example, $\gamma_{32CR/16QAM}$ denotes the coding gain of a coded 32-CROSS constellation compared with uncoded 16-QAM. The number of information bits k transmitted per coded symbol, which equals the spectral efficiency η in bits/symbol, is also given. In Tables 18.6(a) and (b), coding gains are given for several different spectral efficiencies based on constellations chosen from the same lattice. The notation $\gamma_{C/U}$ denotes the coding gain of a coded lattice of infinite size compared with the uncoded lattice.
- The average number of nearest neighbors $A_{d_{free}}$. In Tables 18.6(a) and (b), $A_{d_{free}}$ is given only for the infinite spectral efficiency case, that is, $k \to \infty$.

TABLE 18.6: List of optimum TCM codes.

(a) Codes for one-dimensional AM based on $\mathbb{Z}^1$

$\{\Delta_i^2, 0 \le i \le 2\} = \Delta_0^2, 4\Delta_0^2, 16\Delta_0^2$

ν	$\tilde{k}$	$\mathbf{h}^{(1)}$	$\mathbf{h}^{(0)}$	d_{free}^2/Δ_0^2	$\gamma_{4\text{-AM}/2\text{-AM}}$ $k=1$	$\gamma_{8\text{-AM}/4\text{-AM}}$ $k=2$	$\gamma_{C/U}$ $(k\to\infty)$	$A_{d_{free}}$ $(k\to\infty)$
2	1	2	5	9.0	2.55	3.31	3.52	4
2	1	2	5	9.0	2.55	3.31	3.52	4
3	1	04	13	10.0	3.01	3.77	3.97	4
4	1	04	23	11.0	3.42	4.18	4.39	8
5	1	10	45	13.0	4.15	4.91	5.11	12
6	1	024	103	14.0	4.47	5.23	5.44	36
7	1	126	235	16.0	5.05	5.81	6.02	66
8	1	362	515	16.0*	—	5.81	6.02	2
	1	362	515	17.0	5.30	—	—	

(b) Codes for two-dimensional AM/PM based on $\mathbb{Z}^2$

$\{\Delta_i^2, 0 \le i \le 3\} = \Delta_0^2, 2\Delta_0^2, 4\Delta_0^2, 8\Delta_0^2$

ν	$\tilde{k}$	$\mathbf{h}^{(2)}$	$\mathbf{h}^{(1)}$	$\mathbf{h}^{(0)}$	d_{free}^2/Δ_0^2	$\gamma_{16\text{-QAM}/8\text{-PSK}}$ $k=3$	$\gamma_{32\text{-CR}/16\text{-QAM}}$ $k=4$	$\gamma_{64\text{-QAM}/32\text{-CR}}$ $k=5$	$\gamma_{C/U}$ $(k\to\infty)$	$A_{d_{free}}$ $(k\to\infty)$
2	1	—	2	5	4.0*	4.36	3.01	2.80	3.01	4
3	2	04	02	11	5.0	5.33	3.98	3.77	3.98	16
4	2	16	04	23	6.0	6.12	4.77	4.56	4.77	56
5	2	10	06	41	6.0	6.12	4.77	4.56	4.77	16
6	2	064	016	101	7.0	6.79	5.44	5.23	5.44	56
7	2	042	014	203	8.0	7.37	6.02	5.81	6.02	344
8	2	304	056	401	8.0	7.37	6.02	5.81	6.02	44
9	2	0510	0346	1001	8.0*	7.37	6.02	5.81	6.02	4

(c) Codes for 8-PSK

$\{\Delta_i^2, 0 \le i \le 2\} = 4\sin^2(\pi/8), 2, 4$

ν	$\tilde{k}$	$\mathbf{h}^{(2)}$	$\mathbf{h}^{(1)}$	$\mathbf{h}^{(0)}$	d_{free}^2/Δ_0^2	$\gamma_{8\text{-PSK}/4\text{-PSK}}$ $k=2$	$A_{d_{free}}$
2	1	—	2	5	4.000*	3.01	1
3	2	04	02	11	4.586	3.60	2
4	2	16	04	23	5.172	4.13	≈2.3
5	2	34	16	45	5.758	4.59	4
6	2	066	030	103	6.343	5.01	≈5.3
7	2	122	054	277	6.586	5.17	≈0.5
8	2	130	072	435	7.515	5.75	≈1.5

(d) Codes for 16-PSK

$\{\Delta_i^2, 0 \le i \le 3\} = 4\sin^2(\pi/16), 4\sin^2(\pi/8), 2, 4$

ν	$\tilde{k}$	$\mathbf{h}^{(2)}$	$\mathbf{h}^{(1)}$	$\mathbf{h}^{(0)}$	d_{free}^2/Δ_0^2	$\gamma_{16\text{-PSK}/8\text{-PSK}}$ $k=3$	$A_{d_{free}}$
2	1	—	2	5	1.324	3.54	4
3	1	—	04	13	1.476	4.01	4
4	1	—	04	23	1.628	4.44	8
5	1	—	10	45	1.910	5.13	8
6	1	—	024	103	2.000*	5.33	2
7	1	—	024	203	2.000*	5.33	2
8	2	374	176	427	2.085	5.51	≈8.0

Adapted from [4].

The codes listed for one-dimensional AM are based on the one-dimensional integer lattice $\mathbb{Z}^1$, and the codes listed for two-dimensional AM/PM are based on the two-dimensional integer lattice $\mathbb{Z}^2$, where these lattices are infinite extensions of the one- and two-dimensional signal constellations shown in Figures 18.1(a) and 18.1(b). In these cases the same codes yield the same maximum d_{free}^2/Δ_0^2 independent of the size of the signal constellation chosen from the lattice, although the minimum number of nearest neighbors can vary owing to the effect of the signal constellation boundaries. In Tables 18.6(a) and (b), to negate the effect of constellation boundaries, we list only the average multiplicities $A_{d_{free}}$ assuming a signal constellation of infinite size. Because set partitioning of an infinite lattice results in a regular mapping, the values of $A_{d_{free}}$ in Tables 18.6(a) and (b) are all integers. In Table 18.6(a), we see that for codes based on $\mathbb{Z}^1$, the asymptotic coding gain γ increases with the spectral efficiency k; that is, the largest coding gains are achieved in the limit as $k \to \infty$. Also, two optimum 256-state codes are listed. The first code, whose d_{free}^2 occurs along parallel transitions, is optimum when the number of information bits $k \geq 2$; that is, when the trellis contains parallel transitions. The second code, which achieves a larger d_{free}^2, is optimum only when $k = 1$, that is, when the trellis does not contain parallel transitions. In Table 18.6(b) we note the relatively large asymptotic coding gains of coded 16-QAM compared with uncoded 8-PSK. This difference is due to the restriction that PSK signals must all have the same energy. The coding gains of 16-QAM compared with uncoded rectangular constellations are not as large, as shown in Problem 18.15. In contrast with the lattice-based codes, in Tables 18.6(c) and (d) we see that different codes are optimum for 8-PSK and 16-PSK constellations, and that the nonregular mapping can result in noninteger values of the average multiplicities $A_{d_{free}}$.

When a trellis contains parallel transitions, care must be taken in computing the value of $A_{d_{free}}$, since each parallel branch may contribute to a minimum-distance path. For example, in Table 18.6(a), $A_{d_{free}} = 4$ for the 4-state coded integer lattice $\mathbb{Z}^1$. Referring to the error trellis in Figure 18.4(b) for code 2, which is equivalent to the 4-state code in Table 18.6(a), we note that the error trellis for the coded lattice $\mathbb{Z}^1$ is formed by replacing each branch with an (infinite) set of parallel transitions. In this case the trellis branches labeled $\mathbf{e} = (00)$ will now contain the set of parallel transitions representing all error vectors $\mathbf{e} = (\cdots e^{(3)}e^{(2)}00)$, the trellis branches labeled $\mathbf{e} = (10)$ will now contain the set of parallel transitions representing all error vectors $\mathbf{e} = (\cdots e^{(3)}e^{(2)}10)$, and so on. The MSE distance $9\Delta_0^2$ is achieved by a path that diverges from state S_0 along the branch labeled $\mathbf{e} = (10)$ to state S_1 (a squared distance of $4\Delta_0^2$), continues to state S_2 along the branch labeled $\mathbf{e} = (01)$ (a squared distance of Δ_0^2), and remerges with state S_0 along the branch labeled $\mathbf{e} = (10)$ (a squared distance of $4\Delta_0^2$). Now, note that for a given parallel transition on the branch labeled $\mathbf{e} = (00)$ leaving state S_0, say $\mathbf{e} = (\cdots e^{(4)}e^{(3)}000)$, there are two parallel transitions, $\mathbf{e} = (\cdots e^{(4)}e^{(3)}110)$ and $\mathbf{e} = (\cdots e^{(4)}e^{(3)}010)$, with squared distance $4\Delta_0^2$ on the diverging branch labeled $\mathbf{e} = (10)$. The same situation holds when the minimum-weight path remerges with state S_0 along the branch labeled $\mathbf{e} = (10)$. For the middle branch on the minimum-weight path, there is only one parallel transition, $\mathbf{e} = (\cdots e^{(4)}e^{(3)}001)$, with squared distance Δ_0^2 along the branch labeled $\mathbf{e} = (01)$. Because there are four possible combinations of minimum-weight

paths, in this case, $A_{d_{free}} = 4$. Another example of computing $A_{d_{free}}$ for a trellis with parallel transitions is given in Problem 18.16. Finally, we recall that when $A_{d_{free}}$ of the coded system exceeds A_{min} of the uncoded system, the real coding gain at practical BERs of around 10^{-5} is somewhat reduced compared with the asymptotic coding gain γ.

It is interesting to note that many of the optimum codes listed in Table 18.6 contain one or more uncoded bits. This is because, particularly for short constraint lengths, the parallel transition distance $\delta^2_{min} = \Delta^2_{\tilde{k}+1}$ is already larger than the free distance δ^2_{free} between trellis paths, and thus using a higher rate code cannot improve the overall free distance d^2_{free}. For longer constraint lengths, however, the free distance δ^2_{free} between trellis paths increases, and then more coded bits, that is, a larger $\tilde{k}$, must be used to increase the parallel transition distance δ^2_{min} and consequently the overall free distance d^2_{free}.

18.3 TCM PERFORMANCE ANALYSIS

The *average weight enumerating function* (AWEF) $A_{av}(X)$ and the *average input output weight enumerating function* (AIOWEF) $A_{av}(W, X)$ of a TCM system can be computed by labeling the branches of the binary error trellis with their corresponding AEWEs, augmented by the input weight enumerators when computing $A_{av}(W, X)$, and then forming the modified state diagram and using the transfer function approach developed in Chapter 11. Once $A_{av}(X)$ and $A_{av}(W, X)$ have been evaluated, the event-error probability $P(E)$ and the bit-error probability $P_b(E)$ can be estimated using the union bounding techniques developed in Chapter 12. For an unquantized-output AWGN channel whose inputs are drawn from the TCM signal set, this process gives the expressions

$$P(E) \leq f(d^2_{free}E_s/4N_0)A_{av}(X)\Big|_{X=e^{-E_s/4N_0}} \tag{18.28a}$$

and

$$P_b(E) \leq (1/k)\ f(d^2_{free}E_s/4N_0)\partial A_{av}(W, X)/\partial W\Big|_{X=e^{-E_s/4N_0}, W=1}, \tag{18.28b}$$

where $f(x) = e^x Q(\sqrt{2x})$, and d^2_{free} is computed under the assumption of a unit average energy signal set. The reader should note the similarity between the expressions in (18.28) and those derived for binary convolutional codes in Chapter 12. In fact, they are identical except that the WEFs are replaced by *average* WEFs, and the Hamming distance, $d_{H,free}$, in Chapter 12 is replaced by $d^2_{free}/4$, where d^2_{free} is SE distance, in the preceding expressions. This reflects the fact that, for unit energy binary signals, $d^2_{free} = 4d_{H,free}$, as noted in (18.8).

The bounds in (18.28) are valid for any TCM system without parallel transitions, that is, the case for which each error event represents a path through the trellis at least two branches in length. In the case of parallel transitions, that is, one-branch error events, the bounds are modified as follows:

$$\begin{aligned} P(E) \leq\ & f(\delta^2_{min}E_s/4N_0)A^p_{av}(X)\Big|_{X=e^{-E_s/4N_0}} \\ & + f(\delta^2_{free}E_s/4N_0)A^t_{av}(X)\Big|_{X=e^{-E_s/4N_0}} \end{aligned} \tag{18.29a}$$

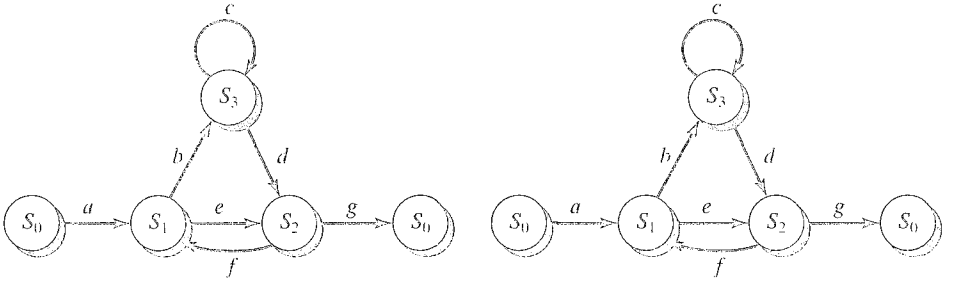
and

$$P_b(E) \le (1/k)\ f(\delta_{min}^2 E_s/4N_0)\partial A_{av}^p(W, X)/\partial W\Big|_{X=e^{-E_s/4N_0}, W=1} + (1/k)\ f(\delta_{free}^2 E_s/4N_0)\partial A_{av}^t(W, X)/\partial W\Big|_{X=e^{-E_s/4N_0}, W=1}, \tag{18.29b}$$

where $A_{av}^p(X)$ and $A_{av}^p(W, X)$ represent the AWEF and AIOWEF for the parallel transition paths, and $A_{av}^t(X)$ and $A_{av}^t(W, X)$ represent the AWEF and AIOWEF for the trellis paths, respectively. (It should be noted that $A_{av}^p(X)$ and $A_{av}^p(W, X)$ are simply the AWEFs of the subsets at the last level, that is, level $\tilde{k}+1$, in the set-partitioning tree.) The use of WEFs to evaluate the performance of TCM systems was introduced by Zehavi and Wolf [24], and an algorithm for computing the AWEF was presented in [25]. We now illustrate the application of the bounds with two examples.

EXAMPLE 18.8 4-State, Rate $R = 1/2$ Trellis-Coded 4-AM

We consider the 4-state, rate $R = 1/2$ binary feedforward encoder shown for code 2 in Figure 18.4(a) along with naturally mapped 4-AM. The binary error trellis of this encoder was shown in Figure 18.4(b), and the AEWEs of naturally mapped 4-AM were listed in Table 18.4(b). In Figure 18.17(a) we show the modified state diagram labeled with the AEWEs. We now compute the AWEF using the standard transfer



(a) Modified state diagram

$a = X^{3.2}$
$b = \frac{1}{2}X^{0.8} + \frac{1}{2}X^{7.2}$
$c = X^{0.8}$
$d = \frac{1}{2}X^{0.8} + \frac{1}{2}X^{7.2}$
$e = X^{0.8}$
$f = X^{0}$
$g = X^{3.2}$

(b) Augmented modified state diagram

$a = WX^{3.2}$
$b = W(\frac{1}{2}X^{0.8} + \frac{1}{2}X^{7.2})$
$c = WX^{0.8}$
$d = \frac{1}{2}X^{0.8} + \frac{1}{2}X^{7.2}$
$e = X^{0.8}$
$f = WX^{0}$
$g = X^{3.2}$

FIGURE 18.17: Modified state diagrams for naturally mapped, 4-state, rate $R = 1/2$ trellis-coded 4-AM.

function approach as follows:

$$\begin{aligned} A_{av}(X) &= \frac{X^{3.2}X^{0.8}X^{3.2}(1-X^{0.8}) + X^{3.2}(0.5X^{0.8}+0.5X^{7.2})^2X^{3.2}}{(1-X^{0.8})(1-X^{0.8}X^{0}) - (0.5X^{0.8}+0.5X^{7.2})^2X^{0}} \\ &= \frac{X^{7.2} - 0.75X^{8.0} + 0.5X^{14.4} + 0.25X^{20.8}}{1 - 2X^{0.8} + 0.75X^{1.6} - 0.5X^{8.0} - 0.25X^{14.4}} \\ &= X^{7.2} + 1.25X^{8.0} + 1.75X^{8.8} + 2.0625X^{9.6} + \cdots . \end{aligned} \tag{18.30a}$$

Equation (18.30a) implies that for an arbitrary transmitted sequence $\mathbf{y}$, there is an *average* of 1 error path $\mathbf{y}'$ with MFSE distance $d^2_{free} = d^2_E(\mathbf{y}, \mathbf{y}') = 7.2$, an *average* of 1.25 error paths $\mathbf{y}'$ with SE distance $d^2_E(\mathbf{y}, \mathbf{y}') = 8.0$, an *average* of 1.75 error paths $\mathbf{y}'$ with SE distance $d^2_E(\mathbf{y}, \mathbf{y}') = 8.8$, and so on.

In Figure 18.17(b) we show the modified state diagram augmented by the input weight enumerators. In this case, following the same procedure as before, we find that the AIOWEF is given by

$$A_{av}(W, X) = WX^{7.2} + 1.25W^2X^{8.0} + 1.75W^3X^{8.8} + 2.0625W^4X^{9.6} + \cdots . \tag{18.30b}$$

Here we see that the error paths at a distance of 7.2 from the correct path are always associated with 1 information bit error, those at a distance of 8.0 are always associated with 2 information bit errors, those at a distance of 8.8 with 3 information bit errors, and so on.

Finally, the expressions of (18.30) can be used in (18.28) to evaluate $P(E)$ and $P_b(E)$ as functions of the channel SNR E_s/N_0. The bounds are sketched in Figure 18.18 along with uncoded BPSK, which has the same spectral efficiency of $\eta = 1$ bit/dimension.

The following observations relate to Example 18.8:

- The codeword multiplicities are averages because TCM systems are nonlinear, and the number of codewords at a particular distance from the correct sequence depends on the transmitted path.

- The codeword multiplicities are fractional because the signal constellation is finite, and the mapping is nonregular. Thus, the multiplicity of 1.25 associated with incorrect paths at distance 8.0 means that, depending on the correct path, there may be either 1 or 2 incorrect paths at distance 8.0.

- If the same rate $R = 1/2$ code was used, along with an infinite number of uncoded information bits, to code the one-dimensional integer lattice $\mathbb{Z}^1$, the average multiplicities would be integers, since a regular signal mapping can then be achieved with set partitioning, as noted in the previous section. In this case the average number of nearest neighbors is $A_{d_{free}} = 4$, since the parallel transition subsets in both the first and last branches of the shortest error event contain exactly two signal points distance 3.2 away from any given reference point (see Problem 18.15).

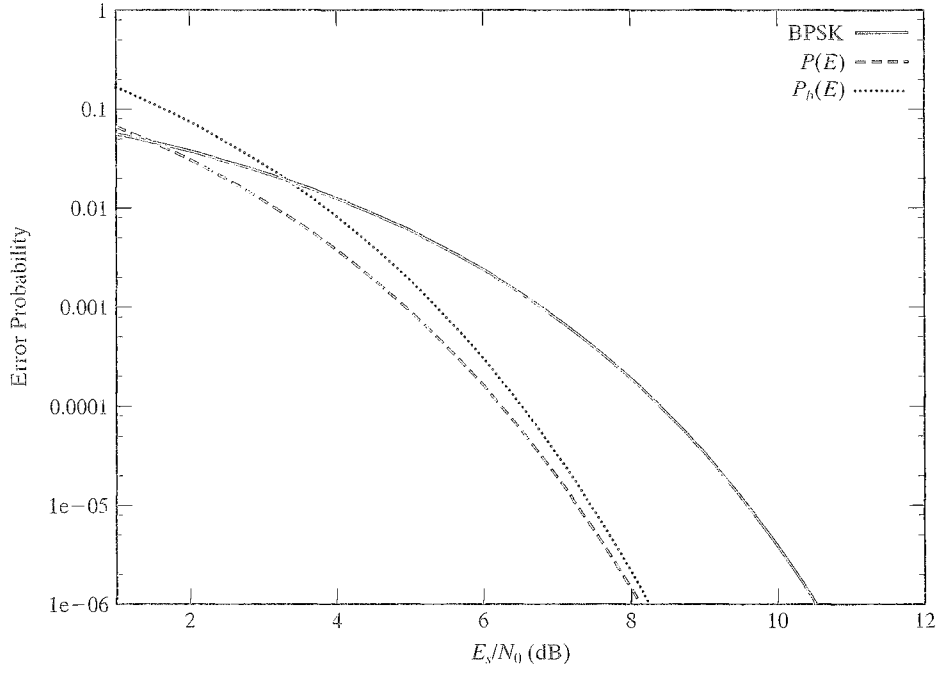


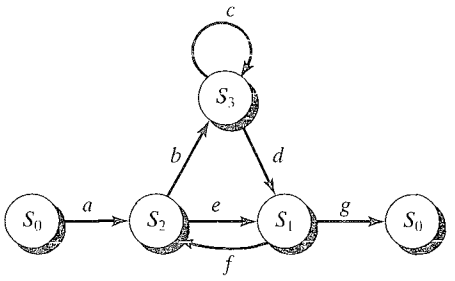
FIGURE 18.18: Error probability bounds for naturally mapped, 4-state, rate $R = 1/2$ trellis-coded 4-AM.

- Because of the particular structure of the encoder in this example, a deterministic relationship exists between codeword distance from the correct path and information weight; that is, $d_E^2 = 6.4 + 0.8w_I$, where d_E^2 represents the SE distance from the correct path, and w_I represents the corresponding information weight. For example, all codewords distance 14.4 from the correct path have information weight 10.

- From the bound on $P_b(E)$ plotted in Figure 18.18, we see that the *real coding gain* at a BER of 10^{-5} of this TCM system compared with uncoded BPSK is approximately 2.1 dB. This coding gain is achieved without bandwidth expansion.

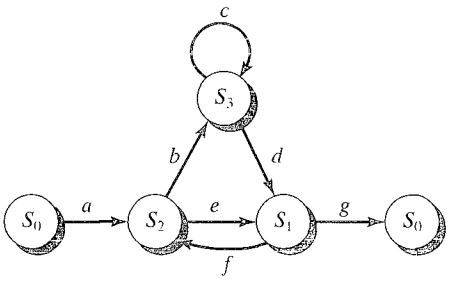
EXAMPLE 18.9 4-State, Rate $R = 1/2$ Trellis-Coded 8-PSK

Now, consider the 4-state, rate $R = 1/2$ binary feedback encoder shown for code 2 in Figure 18.9(a) along with one uncoded information bit and naturally mapped 8-PSK modulation. The binary error trellis of this encoder was shown in Figure 18.9(b), where there was a parallel transition connecting each pair of states, and the AEWEs of naturally mapped 8-PSK modulation were listed in Table 18.5. In Figure 18.19(a) we show the modified state diagram, in which each branch is labeled with the sum of the AEWEs for the corresponding parallel transition branch labels in the binary trellis. We can now compute the AWEF for the trellis paths $A_{av}^t(X)$ using the



$a = 2X^2$
$b = X^{0.586} + X^{3.414}$
$c = X^{0.586} + X^{3.414}$
$d = X^{0.586} + X^{3.414}$
$e = X^{0.586} + X^{3.414}$
$f = 1 + X^4$
$g = 2X^2$

(a) Modified state diagram



$a = (W + W^2)X^2$
$b = \frac{1}{2}(W + W^2)(X^{0.586} + X^{3.414})$
$c = X^{0.586} + WX^{3.414}$
$d = \frac{1}{2}(W + W^2)(X^{0.586} + X^{3.414})$
$e = X^{0.586} + WX^{3.414}$
$f = 1 + WX^4$
$g = (W + W^2)X^2$

(b) Augmented modified state diagram

FIGURE 18.19: Modified state diagrams for naturally mapped, 4-state, rate $R = 1/2$ trellis-coded 8-PSK.

standard transfer function approach as follows:

$$\begin{aligned} A_{av}^{t}(X) &= \frac{\begin{array}{c} 2X^2(X^{0.586} + X^{3.414})2X^2(1 - X^{0.586} - X^{3.414}) + \\ 2X^2(X^{0.586} + X^{3.414})^2 2X^2 \end{array}}{\begin{array}{c} [1 - X^{0.586} - X^{3.414}][1 - (X^{0.586} - X^{3.414})(X^0 + X^4)] - \\ [X^{0.586} + X^{3.414}]^2[X^0 + X^4] \end{array}} \\ &= \frac{4X^{4.586} + 4X^{7.414}}{1 - 2X^{0.586} - 2X^{3.414} - X^{4.586} - X^{7.414}} \\ &= 4X^{4.586} + 8X^{5.172} + 16X^{5.758} + 32X^{6.344} + \cdots . \end{aligned} \tag{18.31a}$$

Equation (18.31a) implies that for an arbitrary transmitted sequence $\mathbf{y}$, there is an *average* of 4 error paths $\mathbf{y}'$ with a MFSE distance *between trellis paths* of $\delta_{free}^2 = d_E^2(\mathbf{y}, \mathbf{y}') = 4.586$, an *average* of 8 error paths $\mathbf{y}'$ with free SE distance $d_E^2(\mathbf{y}, \mathbf{y}') = 5.172$, and so on. Because this TCM system includes parallel transitions, we must also compute the parallel transition AWEF $A_{av}^{p}(X)$. From the 8-PSK set-partitioning tree, we see that there are only two signal points in each parallel transition, and that

$$A_{av}^{p}(X) = X^{4.0}. \tag{18.31b}$$

Equation (18.31b) implies that the MSE distance *between parallel transitions* is $\delta_{min}^2 = 4.0$, and hence the MFSE distance of the TCM system is

$$d_{free}^2 = \min\left\{\delta_{free}^2, \delta_{min}^2\right\} = 4.0, \tag{18.31c}$$

as noted earlier in Example 18.4.

In Figure 18.19(b) we show the modified state diagram augmented by the input weight enumerators. In this case, following the same procedure as above, we find that the AIOWEF is given by (see Problem 18.17)

$$\begin{aligned} A_{av}^{t}(W, X) &= \left(W + W^2\right)^2 X^{4.586} + \left[\left(W + W^2\right)^2 + 0.25\left(W + W^2\right)^4\right] X^{5.172} + \cdots \\ &= \left(W^2 + 2W^3 + W^4\right) X^{4.586} + \left(W^2 + 2W^3 + 1.25W^4 + W^5\right. \\ &\quad \left. + 1.5W^6 + W^7 + 0.25W^8\right) X^{5.172} + \cdots . \end{aligned} \tag{18.31d}$$

Here we see that the error paths at a distance of 4.586 from the correct path are associated with 2, 3, or 4 information bit errors, those at a distance of 5.172 are associated with between 2 and 8 information bit errors, and so on. In addition, the coefficients of the W terms denote the relative likelihood that a certain number of information bit errors will correspond to error paths of a given weight. For example, the terms $1.25W^4X^{5.172}$ and $0.25W^8X^{5.172}$ indicate that error paths at a distance of 5.172 from the correct path are five times more likely to have 4 information bit errors than 8. Finally, the parallel transition AIOWEF is given by

$$A_{av}^{p}(W, X) = WX^{4.0}, \tag{18.31e}$$

which indicates that all parallel transition error events are associated with one information bit error.

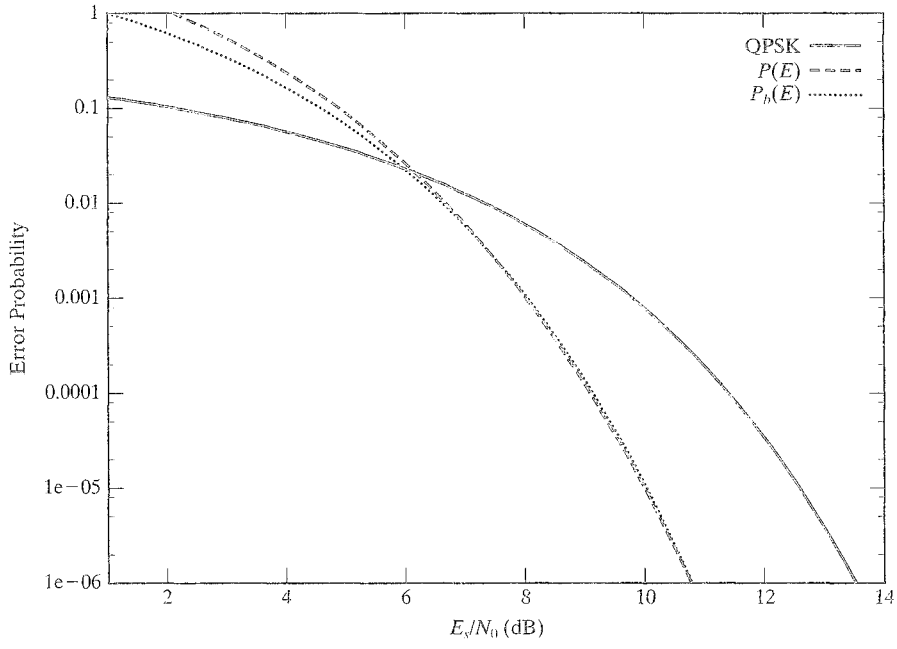


FIGURE 18.20: Error probability bounds for naturally mapped, 4-state, rate $R = 1/2$ trellis-coded 8-PSK.

Now, we can use the expressions of (18.31) in (18.29) to evaluate $P(E)$ and $P_b(E)$ as functions of the channel SNR E_s/N_0. The bounds are sketched in Figure 18.20 along with uncoded QPSK, which has the same spectral efficiency of $\eta = 2$ bits/symbol.

The following remarks apply to Example 18.9:

- The WEFs for each possible parallel transition, that is, for each subset at level 2 of the partition tree, are identical, because the four BPSK subsets at level 2 are isomorphic. In general, however, this may not be the case, and $A_{av}^{p}(X)$ is computed by taking the average of the WEFs of each subset at level $\tilde{k} + 1$ in the set-partitioning tree.
- The MFSE distance path is a parallel transition. This implies that at high SNRs, $A_{av}^{p}(X)$ and $A_{av}^{p}(W, X)$ are the dominant terms in the error probability bounds, thus allowing approximate bounds on $P(E)$ and $P_b(E)$ to be obtained very simply.
- The possible codeword weights represented in (18.31a) and (18.31b) are separated by the value $\Delta_{\min}^2 = 0.586$, the MSE distance between signal points. This is characteristic of any TCM system; that is, codeword weights increase by multiples of $\Delta_{\min}^2$.
- From the bound on $P_b(E)$ plotted in Figure 18.20, we see that the *real coding gain* at a BER of 10^{-5} of this TCM system compared with uncoded QPSK is approximately 2.6 dB. This coding gain is achieved without bandwidth expansion.

As a final comment before leaving this section, we note that whereas the *asymptotic coding gain* of a TCM system can be obtained by computing d_{free}^2, as shown in Section 18.1, the *real coding gain* at a particular BER must be obtained either from computer simulations or estimated from bounds such as the ones presented in this section.

18.4 ROTATIONALLY INVARIANT TCM

The typical signal set used in a coded modulation system has several phase symmetries; that is, phase rotations by certain angles replicate the signal set. For example, 8-PSK has eight phase symmetries, spaced 45° apart, and any QAM constellation has four phase symmetries spaced by 90°. In general, when a receiver locks onto a particular phase, a trial-and-error procedure must be initiated to determine whether it is in the correct phase. If a particular system is prone to frequent loss of carrier synchronization, reacquiring sync can be a time-consuming exercise. Thus, it is desirable in many applications that a coded modulation system be invariant to phase rotations. In other words, if the receiver locks onto the wrong phase, the system should still be able to operate properly. Hence, in case of temporary loss of synchronization, the receiver must lock onto only one of the possible symmetries and need not initiate a procedure to reacquire the correct phase.

The basic requirement for a coded modulation system to be invariant to a particular phase rotation is that when the symbols of each code sequence are

replaced by the corresponding symbols in the rotated signal set, it is still a valid code sequence. In other words, in the absence of noise, the decoder would still decode a proper code sequence, albeit an incorrect code sequence; however, differential encoding of the information sequence and differential decoding of the decoder output sequence can be employed to ensure that the correct sequence is decoded. Thus, a rotationally invariant code combined with differential encoding/decoding can be used to provide reliable communication even when the receiver is locked onto the wrong phase, although a small penalty in BER performance is incurred because isolated single-bit errors at the decoder output are doubled by differential decoding. (Note that if a code is not rotationally invariant, rotated code sequences are, in general, not code sequences, and this property may be used to detect an out-of-phase lock condition.)

To illustrate the idea of rotationally invariant codes, we start with the simple case of a binary code with BPSK modulation in which the encoder output bits are represented as $0 \rightarrow -1$ and $1 \rightarrow +1$. The only phase symmetry of the signal set is caused by a 180° rotation, which has the effect of inverting the sign of every modulation symbol; that is, $-1 \rightarrow +1$ and $+1 \rightarrow -1$. Thus, every codeword is replaced by its complement. For any linear code, the complement of a codeword is a codeword if and only if the all-one sequence is a codeword. Thus, the simple condition for 180° rotational invariance for any linear binary code with BPSK modulation is that the all-one sequence is a codeword.

We now consider the case of QPSK modulation and 90° phase symmetries, beginning with an example.

EXAMPLE 18.10 Rate $R = 1/2$ Coded QPSK

Consider a rate $R = 1/2$ convolutional code with generator matrix

$$\mathbb{G}(D) = \begin{bmatrix} \mathbb{h}^{(0)}(D) & \mathbb{h}^{(1)}(D) \end{bmatrix} \tag{18.32a}$$

and parity-check matrix

$$\mathbb{H}(D) = \begin{bmatrix} \mathbb{h}^{(1)}(D) & \mathbb{h}^{(0)}(D) \end{bmatrix}, \tag{18.32b}$$

whose two encoder output bits are Gray mapped into QPSK, as shown in Figure 18.3(a). In this case the *parity-check equation* (PCE) is given by

$$\mathbb{V}(D)\mathbb{H}^T(D) = \mathbb{v}^{(1)}(D)\mathbb{h}^{(1)}(D) \oplus \mathbb{v}^{(0)}(D)\mathbb{h}^{0}(D) = \mathbb{0}(D), \tag{18.33}$$

where $\mathbb{V}(D) = [\mathbb{v}^{(1)}(D), \mathbb{v}^{(0)}(D)]$ represents a codeword. Now, note that after a 90° rotation of the signal set, the rotated code sequences become

$$\mathbb{v}_r^{(1)}(D) = \mathbb{v}^{(0)}(D) \quad \text{and} \quad \mathbb{v}_r^{(0)}(D) = \mathbb{v}^{(1)}(D) \oplus \mathbb{1}(D), \tag{18.34}$$

where $\mathbb{1}(D)$ represents the all-one sequence. Using the rotated code sequences in (18.33) we obtain

$$\begin{aligned} \mathbb{V}_r(D)\mathbb{H}^T(D) &= \mathbb{v}_r^{(1)}(D)\mathbb{h}^{(1)}(D) \oplus \mathbb{v}_r^{(0)}(D)\mathbb{h}^{(0)}(D) \\ &= \mathbb{v}^{(0)}(D)\mathbb{h}^{(1)}(D) \oplus \Big[\mathbb{v}^{(1)}(D) \oplus \mathbb{1}(D)\Big]\mathbb{h}^{(0)}(D), \end{aligned} \tag{18.35}$$

which must equal $\mathbb{0}(D)$ for all $\mathbb{V}_r(D) = [\mathbb{v}_r^{(1)}(D), \mathbb{v}_r^{(0)}(D)]$ for the code to be invariant to 90° rotations. From (18.33) we see that for any information sequence $\mathbb{u}(D)$, $\mathbb{V}(D) = [\mathbb{v}^{(1)}(D), \mathbb{v}^{(0)}(D)] = [\mathbb{u}(D)\mathbb{h}^{(0)}(D), \mathbb{u}(D)\mathbb{h}^{(1)}(D)]$ is a valid code sequence. Substituting this $\mathbb{V}(D)$ into (18.35) we have

$$\mathbb{V}_r(D)\mathbb{H}^T(D) = \mathbb{u}(D)\left\{\left[\mathbb{h}^{(1)}(D)\right]^2 \oplus \left[\mathbb{h}^{(0)}(D)\right]^2\right\} \oplus \mathbb{h}^{(0)}(D)\mathbb{1}(D). \tag{18.36}$$

Examining (18.36) closely we see that the first term equals $\mathbb{u}(D)$ times a nonzero binary polynomial of degree at most 2ν, where ν is the constraint length. Considering $\mathbb{1}(D)$ to extend infinitely in both directions, we see that we can write the second term as $\mathbb{h}^0(D)\mathbb{1}(D) = \mathbb{h}^{(0)}(1)(D)$, which equals either $\mathbb{0}(D)$ or $\mathbb{1}(D)$ depending on whether the Hamming weight of $\mathbb{h}^{(0)}(D)$ is even or odd, respectively. For example, code 1 in Example 18.2 has $\mathbb{h}^{(0)}(D) = 1 + D^2$ and $\mathbb{h}^{(1)}(D) = 1 + D + D^2$, and (18.36) becomes

$$\mathbb{V}_r(D)\mathbb{H}^T(D) = D^2\mathbb{u}(D) \oplus \mathbb{0}(D) = D^2\mathbb{u}(D). \tag{18.37}$$

Clearly, (18.37) is not equal to $\mathbb{0}(D)$ for any nonzero $\mathbb{u}(D)$, and thus the coded modulation system is not invariant to 90° rotations.

In the case of a 180° phase rotation, we can see directly from Figure 18.3(a) that both rotated code sequences are complements of their respective correct code sequences. This situation is exactly analogous to the BPSK case, and thus rate $R = 1/2$ convolutional codes with Gray-mapped QPSK are invariant to 180° phase rotations if and only if the all-one sequence $\mathbb{1}(D)$ is a codeword.

Now, consider the same example with natural mapping. In this case the 90° rotated code sequences become

$$\mathbb{v}_r^{(1)}(D) = \mathbb{v}^{(1)}(D) \oplus \mathbb{v}^{(0)}(D) \quad \text{and} \quad \mathbb{v}_r^{(0)}(D) = \mathbb{v}^{(0)}(D) \oplus \mathbb{1}(D). \tag{18.38}$$

Again, substituting in (18.33) and using $\mathbb{V}(D) = [\mathbb{u}(D)\mathbb{h}^{(0)}(D), \mathbb{u}(D)\mathbb{h}^{(1)}(D)]$, we obtain

$$\mathbb{V}_r(D)\mathbb{H}^T(D) = \mathbb{u}(D)\left\{\left[\mathbb{h}^{(1)}(D)\right]^2\right\} \oplus \mathbb{h}^{(0)}(D)\mathbb{1}(D), \tag{18.39}$$

which for code 1 in Example 18.2 becomes

$$\mathbb{V}_r(D)\mathbb{H}^T(D) = \left(1 + D + D^2\right)^2 \mathbb{u}(D) \oplus \mathbb{0}(D) = \left(1 + D^2 + D^4\right)\mathbb{u}(D). \tag{18.40}$$

As in the case of Gray-mapped QPSK, we see that for naturally mapped QPSK (18.40) does not equal $\mathbb{0}(D)$ for any nonzero $\mathbb{u}(D)$, and thus the coded modulation system is not invariant to 90° rotations.

For a 180° phase rotation, we see from Figure 18.3(b) that $\mathbb{v}_r^{(1)}(D) = \mathbb{v}^{(1)}(D) \oplus \mathbb{1}(D)$ and $\mathbb{v}_r^{(0)}(D) = \mathbb{v}^{(0)}(D)$; that is, $\mathbb{v}_r^{(1)}(D)$ is the complement of the correct sequence, and $\mathbb{v}_r^{(0)}(D)$ equals the correct sequence. In this case, $\mathbb{V}_r(D)\mathbb{H}^T(D) = \mathbb{v}_r^{(1)}(D)\mathbb{h}^{(1)}(D) \oplus \mathbb{v}_r^{(0)}(D)\mathbb{h}^{(0)}(D) = \mathbb{h}^{(1)}(D)\mathbb{1}(D)$. Thus rate $R = 1/2$ convolutional codes with naturally mapped QPSK are invariant to 180° phase rotations if and only if $\mathbb{h}^{(1)}(D)$ has even weight.

The following remarks relate to Example 18.10:

- Rotational invariance is a property of the code, not the encoder, so the results of Example 18.10 also hold for the equivalent systematic feedback encoder with $\mathbb{H}(D) = \left[\mathbb{h}^{(1)}(D)/\mathbb{h}^{(0)}(D) \;\; 1\right]$.

- If we consider $\mathbb{1}(D)$ to start at time 0, then terms of the form $\mathbb{h}^{(0)}(D)\mathbb{1}(D)$ would have a short (degree $< \nu$) preamble before reaching their steady-state value of $\mathbb{0}(D)$ or $\mathbb{1}(D)$. This would affect rotational invariance only in the first ν time units, and thus it makes sense to ignore this transient condition by assuming that $\mathbb{1}(D)$ extends infinitely in both directions (see Problem 18.19).

- This example can be generalized to show that any rate $R = k/(k+1)$ convolutional code with a *linear binary* PCE $\mathbb{V}(D)\mathbb{H}^T(D) = \mathbb{v}^{(k)}(D)\mathbb{h}^{(k)}(D) \oplus \cdots \oplus \mathbb{v}^{(1)}(D)\mathbb{h}^{(1)}(D) \oplus \mathbb{v}^{(0)}(D)\mathbb{h}^{(0)}(D) = \mathbb{0}(D)$ that maps into a two-dimensional signal set with 90° phase symmetries can at best be invariant to 180° phase rotations [26].

- As will be seen next, binary rate $R = k/(k+1)$ convolutional codes can achieve 90° rotational invariance only by making use of a *nonlinear* PCE.

To see how nonlinear PCEs can achieve a greater degree of rotational invariance than linear PCEs, we consider the special case of a rate $R = 1/2$ convolutional code with naturally mapped QPSK modulation. As noted in Example 18.10, linear PCEs are not capable of achieving 90° rotational invariance in this case. Before specifying a nonlinear PCE, we write the two encoded sequences in *integer* form as follows:

$$\mathbb{v}(D) = \mathbb{v}^{(0)}(D) + 2\mathbb{v}^{(1)}(D), \tag{18.41}$$

where the coefficients of $\mathbb{v}(D)$ are elements in the *ring* of integers $\mathbb{Z}_4$. (Throughout the remainder of this section, we will use the symbol + to denote addition in a ring of integers and the symbol $\oplus$ to denote addition modulo-2, i.e., binary addition.) Using this formulation, we can express the effect of a 90° phase rotation on naturally mapped QPSK simply as

$$\mathbb{v}_r(D) = \mathbb{v}(D) + \mathbb{1}(D) \pmod 4. \tag{18.42}$$

Similarly, we can write the two parity-check polynomials in integer form as

$$\mathbb{h}(D) = \mathbb{h}^{(1)}(D) + 2\mathbb{h}^{(0)}(D). \tag{18.43}$$

Now, consider the PCE given by

$$[\mathbb{h}(D)\mathbb{v}(D) \pmod 4]^1 = \mathbb{0}(D), \tag{18.44}$$

where the notation $[\alpha(D)]^1$ means that from the binary representation of every element $\alpha_l = 2\alpha_l^{(1)} + \alpha_l^{(0)} \in \mathbb{Z}_4$ in $\alpha(D)$ the most significant bit $\alpha_l^{(1)}$ is chosen; that is, $[\alpha(D)]^1 = \alpha_l^{(1)}$. (The operation represented as $[\alpha(D)]^1$ in (18.44) causes this PCE to be both binary and nonlinear.) For the PCE represented by (18.44) to be invariant

to 90° rotations, it must still be satisfied when $\mathbf{v}_r(D)$ is substituted for $\mathbf{v}(D)$, which requires that

$$\begin{aligned}\mathbf{h}(D)\mathbf{v}(D) = \mathbf{h}(D)\mathbf{v}_r(D) &= \mathbf{h}(D)\left[\mathbf{v}(D) + \mathbf{1}(D)\right] \pmod 4 \\ &= \mathbf{h}(D)\mathbf{v}(D) + \mathbf{h}(D)\mathbf{1}(D) \pmod 4 \\ &= \mathbf{h}(D)\mathbf{v}(D) + \mathbf{h}(1)(D) \pmod 4.\end{aligned} \tag{18.45}$$

Because $\mathbf{h}(1)(D)$ is a constant sequence with $\mathbf{h}(1) \in \mathbb{Z}_4$, (18.45) is satisfied if and only if

$$\mathbf{h}(1) \pmod 4 = \mathbf{h}^{(1)}(1) + 2\mathbf{h}^{(0)}(1) \pmod 4 = 0. \tag{18.46}$$

Note, for example, that if $\mathbf{h}^{(1)}(D)$ has two delay terms, $\mathbf{h}^{(0)}(D)$ must have an odd number of delay terms to satisfy (18.46) (see Problem 18.20).

We can rewrite the PCE of (18.44) as

$$\begin{aligned}\left[\mathbf{h}(D)\mathbf{v}(D) \pmod 4\right]^1 &= \Big[\mathbf{h}^{(1)}(D)\mathbf{v}^{(0)}(D) + 2\mathbf{h}^{(1)}(D)\mathbf{v}^{(1)}(D) \\ &\quad + 2\mathbf{h}^{(0)}(D)\mathbf{v}^{(0)}(D) + 4\mathbf{h}^{(0)}(D)\mathbf{v}^{(1)}(D) \pmod 4\Big]^1 \\ &= \Big\{\mathbf{h}^{(1)}(D)\mathbf{v}^{(0)}(D) + 2\Big[\mathbf{h}^{(1)}(D)\mathbf{v}^{(1)}(D) \\ &\quad + \mathbf{h}^{(0)}(D)\mathbf{v}^{(0)}(D)\Big] \pmod 4\Big\}^1 \\ &= \mathbf{0}(D),\end{aligned} \tag{18.47}$$

where we have simplified (18.47) by noting that the term $4\mathbf{h}^{(0)}(D)\mathbf{v}^{(1)}(D) \pmod 4 = \mathbf{0}(D)$. Following the restrictions of Figure 18.16(b) for good TCM code design leads us to search for codes with parity-check polynomials

$$\mathbf{h}^{(1)}(D) = h^{(1)}_{\nu-1}D^{\nu-1} + \cdots + h^{(1)}_2 D^2 + h^{(1)}_1 D \tag{18.48a}$$

and

$$\mathbf{h}^{(0)}(D) = D^{\nu} + h^{(0)}_{\nu-1}D^{\nu-1} + \cdots + h^{(0)}_2 D^2 + h^{(0)}_1 D + 1 \tag{18.48b}$$

that satisfy (18.46), thus guaranteeing 90° rotational invariance. We must then substitute these polynomials into (18.47) to specify the binary nonlinear PCE.

We now consider an example in which we choose $\mathbf{h}^{(1)}(D)$ to have only two delay terms and proceed to derive a general binary nonlinear PCE that guarantees 90° rotational invariance for rate $R = 1/2$ codes with naturally mapped QPSK.

EXAMPLE 18.11 Rotationally Invariant Rate $R = 1/2$ Codes for Naturally Mapped QPSK

Consider the choice

$$\mathbf{h}^{(1)}(D) = D^b + D^a, \tag{18.49}$$

where $v > b > a > 0$. Note that in this case, since $\mathbb{h}^{(1)}(1) = 2 \pmod 4$, $\mathbb{h}^{(0)}(D)$ must have an odd number of nonzero terms to satisfy (18.46). Now, substituting (18.49) into (18.47) we obtain the PCE

$$\left\{(D^b + D^a)\mathbb{v}^{(0)}(D) + 2\left[(D^b + D^a)\mathbb{v}^{(1)}(D) + \mathbb{h}^{(0)}(D)\mathbb{v}^{(0)}(D)\right] \pmod 4\right\}^1 = 0(D). \tag{18.50}$$

To express (18.50) using binary (mod-2) arithmetic, we note that we can write the mod-4 sum of any two binary sequences $\mathbb{m}(D)$ and $\mathbb{n}(D)$ as

$$\mathbb{m}(D) + \mathbb{n}(D) \pmod 4 = \mathbb{m}(D) \oplus \mathbb{n}(D) + 2\mathbb{m}(D) \circ \mathbb{n}(D), \tag{18.51}$$

where $\mathbb{m}(D) \circ \mathbb{n}(D)$ represents the logical AND of the sequences $\mathbb{m}(D)$ and $\mathbb{n}(D)$. Using (18.51) repeatedly in (18.50) and recalling that $4\mathbb{m}(D) \pmod 4 = \mathbb{0}(D)$ for any binary sequence $\mathbb{m}(D)$, we obtain the binary PCE

$$\begin{aligned}&\left\{(D^b \oplus D^a)\mathbb{v}^{(0)}(D) + 2\left[D^b\mathbb{v}^{(0)}(D) \circ D^a\mathbb{v}^{(0)}(D) \oplus (D^b \oplus D^a)\mathbb{v}^{(1)}(D) \oplus \right.\right.\\ &\qquad \left.\left.\mathbb{h}^{(0)}(D)\mathbb{v}^{(0)}(D)\right]\right\}^1\\ &= D^b\mathbb{v}^{(0)}(D) \circ D^a\mathbb{v}^{(0)}(D) \oplus (D^b \oplus D^a)\mathbb{v}^{(1)}(D) \oplus \mathbb{h}^{(0)}(D)\mathbb{v}^{(0)}(D) = \mathbb{0}(D),\end{aligned} \tag{18.52}$$

where (18.52) has been simplified by using the fact that the term $(D^b \oplus D^a)\mathbb{v}^{(0)}(D)$ has no effect on the most significant bits of the sequence in braces. Equation (18.52) represents a binary nonlinear PCE that guarantees 90° rotational invariance for rate $R = 1/2$ codes with $\mathbb{h}^{(1)}(D)$ defined by (18.49) and any $\mathbb{h}^{(0)}(D)$ with an odd number of nonzero terms, where the nonlinearity is represented by the logical AND operation. (In Problem 18.21 it is shown that the preceding nonlinear PCE is satisfied when the rotated binary sequences given in (18.38) for natural mapping are substituted into (18.52), and $\mathbb{h}^0(D)$ is assumed to have an odd number of nonzero terms.)

The addition of the nonlinear term $D^b\mathbb{v}^{(0)}(D) \circ D^a\mathbb{v}^{(0)}(D)$ makes (18.52) different from a linear PCE for a rate $R = 1/2$ code. Given this difference, it is not clear if (18.52) will, in general, result in an encoder realization with only 2^v states. Considering the specific example

$$\mathbb{H}(D) = \left[(D^2 + D)/(D^3 + D + 1) \;\; 1\right]. \tag{18.53}$$

we show in Figure 18.21(a) an 8-state encoder realization in which the nonlinear term $D^2\mathbb{v}^{(0)}(D) \circ D\mathbb{v}^{(0)}(D)$ can be obtained directly from the feedback shift register that forms $\mathbb{v}^{(0)}(D)$; that is, no additional states are needed in this case. (The separate 2-state differential encoder for the information sequence is also shown in Figure 18.21(a).) This nonlinear encoder results in a 90° rotationally invariant code, since $\mathbb{h}^{(0)}(D) = D^3 + D + 1$ has an odd number of delay terms, and thus the rotated code sequences satisfy (18.52) (see Problem 18.22).

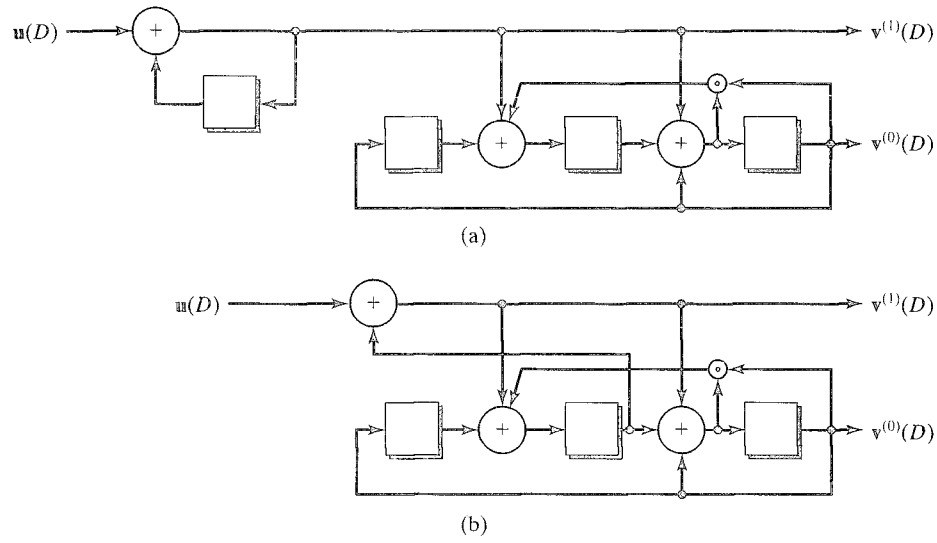


FIGURE 18.21: Realization of a rotationally invariant, 8-state, rate $R = 1/2$ QPSK encoder (a) with separate differential encoding and (b) with embedded differential encoding.

The following comments apply to Example 18.11:

- The nonlinear PCE represented by (18.52) can always be realized using $\nu' = \max[\nu, 2(b-a)]$ delay elements [26]. In this example, $\nu' = \nu = 3$ (see Problem 18.23 for an example that includes the case $\nu < 2(b-a)$).
- More classes of rotationally invariant codes can be found by dropping the condition of (18.49) and merely requiring that (18.46) be satisfied; however, in this more general case, it is not as easy to determine the conditions under which additional states are not required in the encoder realization.
- Natural mapping is assumed and (18.44) is used as the PCE because these choices result in simple conditions on the parity-check polynomials to guarantee rotational invariance and because if $\mathbb{h}^{(1)}(D)$ is chosen as in (18.49), the PCE contains only one nonlinear term. Additional classes of 90° rotationally invariant codes can be found if other mappings are assumed or different PCEs are used, but the code specification and realization is, in general, more complex.

For a given ν, a large family of 90° rotationally invariant rate $R = 1/2$ nonlinear codes for naturally mapped QPSK are defined by (18.52). A computer search can be used to select the parameters a and b and the coefficients of $\mathbb{h}^{(0)}(D)$ that satisfy the conditions of (18.46) and (18.49), maximize d^2_{free}, and minimize $A_{d_{free}}$. (The search technique cannot use the method of Euclidean weights to find d^2_{free} in this case, since the code is nonlinear; that is, all pairs of trellis paths must be compared.) A list of the best 90° rotationally invariant rate $R = 1/2$ nonlinear codes for naturally

TABLE 18.7: Rotationally invariant rate $R = 1/2$ QPSK codes. $\eta = 1$ bit/symbol, $d^2_{min} = 4$, $A_{min} = 1$ (BPSK)

				90° Invariance		180° Invariance		360° Invariance		γ
ν	$\tilde{k}$	$\mathbb{h}^{(1)}$	$\mathbb{h}^{(0)}$	d^2_{free}	$A_{d_{free}}$	d^2_{free}	$A_{d_{free}}$	d^2_{free}	$A_{d_{free}}$	(dB)
3	1	06	13	10	0.5	12	2	12	1	3.98
4	1	06	23	12	0.5	12	1	14	2	4.77
5	1	30	45	14	1.0	16	2	16	1	5.44
6	1	050	105	16	1.875	20	11	—	—	6.02
7	1	110	217	16	0.25	20	2	20	1	6.02
8	1	220	427	18	0.312	24	11	24	9	6.53
9	1	0120	1017	20	0.75	24	2	24	1	6.99

Adapted from [26].

mapped QPSK found for constraint lengths up to $\nu = 9$ is presented in Table 18.7. The parity-check coefficients $\mathbb{h}^{(j)} = \left[h^{(j)}_{\nu}, h^{(j)}_{\nu-1}, \cdots, h^{(j)}_{1}, h^{(j)}_{0}\right]$, $j = 0, 1$, are given in octal form, as in Table 18.6. The values of d^2_{free} and $A_{d_{free}}$ for the best 180° and 360° rotationally invariant rate $R = 1/2$ linear codes are also listed for comparison, along with the asymptotic coding gain γ of the best nonlinear code. (The codes for 360° invariance are the optimum free distance codes found in Table 12.1 used with Gray mapping. The value of d^2_{free} given in Table 18.6 is twice as large as the value of d_{free} given in Table 12.1 because the QPSK signals have been normalized to unit energy. The missing entry for $\nu = 6$ means that the best 360° invariant code is identical to the best 180° invariant code.) The value of γ is computed with reference to an uncoded system (BPSK) with the same spectral efficiency $\eta = 1$ bit/symbol that has MSE distance $d^2_{min} = 4$ and number of nearest neighbors $A_{min} = 1$. Note that, in general, the best nonlinear codes have smaller values of d^2_{free} than the best linear codes, so there is some penalty to be paid in asymptotic coding gain to achieve full rotational invariance. On the other hand, the values of $A_{d_{free}}$ are larger for the linear codes, which lessens the suboptimality of the nonlinear codes at moderate BERs. (The values of $A_{d_{free}}$ are, in general, fractional in the nonlinear case, since not all codewords have a nearest neighbor at distance d^2_{free}.) For example, the best 8-state, 90° invariant nonlinear code (the code in Example 18.11) has $\gamma = 3.98$ dB, which is 0.79 dB less than the best 8-state, 180° invariant linear code; however, the nonlinear code has only one-fourth the number of nearest neighbors of the linear code. In Figure 18.22 we plot the simulated BER performance of these two codes. Note that the suboptimality of the nonlinear code is only about 0.3 dB at a BER of 10^{-5}.

Using the nonlinear 90° rotationally invariant codes described in this section guarantees that rotated code sequences are still valid paths through the trellis. For the coding system to operate properly (i.e., with only a small loss in decoded BER), even if the receiver has lost synchronization and locked onto the wrong phase, differential encoding and decoding must be employed. Thus, the input sequence $\mathbb{u}(D)$ must pass through the one time unit delay circuit $1/(D+1)$ (shown in Figure 18.21(a)) prior to convolutional encoding, and the decoded sequence $\hat{\mathbb{u}}(D)$ must be processed by the inverse circuit $(D+1)$ after decoding. For some rotationally invariant codes, it is

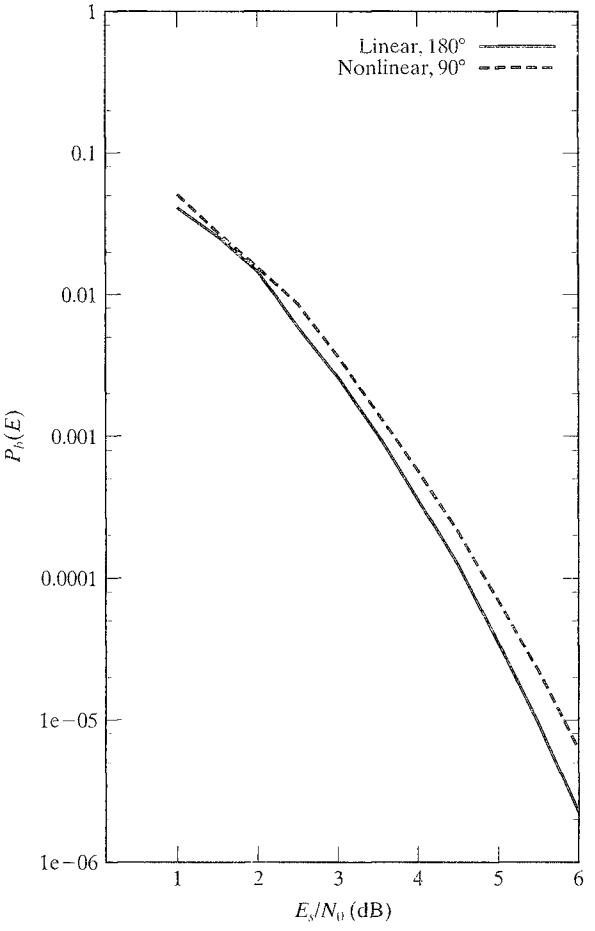


FIGURE 18.22: Performance curves for two 8-state, rate $R = 1/2$ QPSK codes.

possible to embed the differential encoding within the convolutional encoder, thus eliminating the need for separate differential encoding and decoding circuits. For the 90° rotationally invariant code of Example 18.11, an encoder realization that includes embedded differential encoding is shown in Figure 18.21(b).

Any QAM signal constellation has exactly the same four rotational symmetries as the QPSK signal set. Thus, the same approach used for rate $R = 1/2$ coded QPSK can be used to construct 90° rotationally invariant codes for rate $R = k/(k+1)$ coded 2^{k+1}-ary QAM constellations, as long as the two least significant label bits are assigned using natural mapping. In this case only the first information sequence $\mathbf{v}^{(1)}(D)$ and the parity sequence $\mathbf{v}^{(0)}(D)$ are affected by a rotation, so the check polynomials $\mathbf{h}^{(2)}(D), \cdots, \mathbf{h}^{(\tilde{k})}(D)$ corresponding to the other coded information sequences $\mathbf{v}^{(2)}(D), \cdots, \mathbf{v}^{(\tilde{k})}(D)$ can be chosen to maximize d^2_{free} and minimize $A_{d_{free}}$ without regard to the invariance constraints. Using the same conditions as in Example 18.11, we can write a general binary nonlinear PCE for rate $R = k/(k+1)$

coded QAM with $\tilde{k}$ coded information bits as

$$\begin{aligned}&\mathbb{h}^{(\tilde{k})}(D)\mathbb{v}^{(\tilde{k})}(D) \oplus \cdots \oplus \mathbb{h}^{(2)}(D)\mathbb{v}^{(2)}(D) \oplus (D^b \oplus D^a)\mathbb{v}^{(1)}(D) \oplus \mathbb{h}^{(0)}(D)\mathbb{v}^{(0)}(D)\\ &\quad = D^b\mathbb{v}^{(0)}(D) \circ D^a\mathbb{v}^{(0)}(D). \end{aligned} \tag{18.54}$$

Again, following the restrictions of Figure 18.16(b) for good TCM code design, we let the check polynomials $\mathbb{h}^{(2)}(D), \cdots, \mathbb{h}^{(\tilde{k})}(D)$ be denoted by

$$\mathbb{h}^{(j)}(D) = D^{b_j} + h^{(j)}_{b_j-1}D^{b_j-1} + \cdots + h^{(j)}_{a_j+1}D^{a_j+1} + D^{a_j}, \tag{18.55}$$

where $\nu > b_j \geq a_j > 0$ for $2 \leq j \leq \tilde{k}$. Further, letting

$$b' = \max\left(b_{\tilde{k}}, \cdots, b_2, b\right), \quad a' = \max\left(a_{\tilde{k}}, \cdots, a_2, a\right), \tag{18.56}$$

we can realize the nonlinear PCE represented by (18.54) using $\nu' = \max(\nu, b - a + b' - a')$ delay elements [26].

EXAMPLE 18.12 Rotationally Invariant Rate $R = 3/4$ Codes for Naturally Mapped 16-QAM with Two Coded Bits ($\tilde{k} = 2$)

Consider the following parity-check matrix for a rate $R = 2/3$ convolutional code:

$$\mathbb{H}(D) = [D/(D^3 + D + 1) \quad (D^2 + D)/(D^3 + D + 1) \quad 1], \tag{18.57}$$

in which $\mathbb{h}^{(1)}(D)$ and $\mathbb{h}^{(0)}(D)$ are chosen as in Example 18.11 to ensure 90° rotational invariance. In this case $\nu = 3, b = 2, a = 1$, and $b_2 = a_2 = 1$, so $b' = 2, a' = 1$, and $\nu' = \max(3, 2) = 3$. Hence, we can use this code along with one uncoded bit to provide a 90° rotationally invariant 8-state encoder realization of a rate $R = 3/4$ 16-QAM TCM system. The encoder realization (without differential encoding) is shown in Figure 18.23.

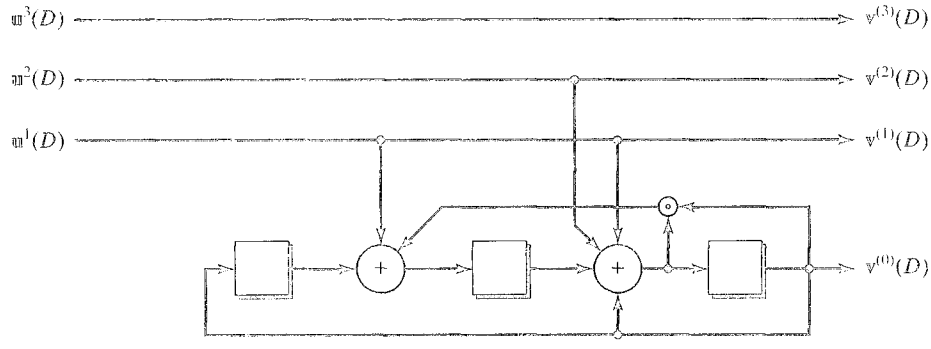


FIGURE 18.23: Realization of a rotationally invariant, 8-state, rate $R = 3/4$ 16-QAM encoder.

TABLE 18.8: Rotationally invariant rate $R = k/(k+1)$ QAM codes. $\eta = k$ bits/symbol, $d^2_{min} = 2$, $A_{min} = 4(\mathbb{Z}^2)$

					90° Invariance		180° Invariance		360° Invariance		γ
ν	$\tilde{k}$	$\mathbb{h}^{(2)}$	$\mathbb{h}^{(1)}$	$\mathbb{h}^{(0)}$	d^2_{free}	$A_{d_{free}}$	d^2_{free}	$A_{d_{free}}$	d^2_{free}	$A_{d_{free}}$	(dB)
3	2	02	06	13	5	16	5	16	—	—	3.98
4	2	04	12	23	5	8	6	56	—	—	3.98
5	2	02	14	45	6	16	6	8	—	—	4.77
6	2	020	014	103	7	80	7	44	7	40	5.44
7	2	100	060	205	7	16	8	204	8	172	5.44
8	2	100	210	417	8	60	8	28	—	—	6.02

Adapted from [26].

Because (18.54) defines a large family of 90° rotationally invariant rate $R = k/(k+1)$ nonlinear codes for naturally mapped QAM signal constellations, a computer search can be used to select the parameters a and b and the coefficients of $\mathbb{h}^{(0)}(D)$ and $\mathbb{h}^{(2)}(D), \cdots, \mathbb{h}^{(\tilde{k})}(D)$ that satisfy the conditions of (18.46) and (18.49), maximize d^2_{free}, and minimize $A_{d_{free}}$. A list of the best 90° rotationally invariant rate $R = k/(k+1)$ nonlinear codes for naturally mapped QAM found for constraint lengths up to $\nu = 8$ is presented in Table 18.8, where the uncoded reference system is the (scaled) infinite two-dimensional integer lattice $\mathbb{Z}^2$ with $d^2_{min} = 2$ and $A_{min} = 4$. The values of d^2_{free} and $A_{d_{free}}$ for the best 180° and 360° rotationally invariant rate $R = k/(k+1)$ linear codes are also listed for comparison, along with the asymptotic coding gain γ of the best nonlinear code. In all cases the best nonlinear codes found had only two coded bits, that is, $\tilde{k} = 2$, and the same codes resulted in the same maximum d^2_{free} independent of the size of the signal constellation chosen from the lattice, that is, independent of the number of uncoded bits and the spectral efficiency $\eta = k$ bits/symbol. Because boundary effects cause the (in general, fractional) values of $A_{d_{free}}$ to differ depending on k and the size of the signal constellation, only the (integer) values of $A_{d_{free}}$ corresponding to an infinite-size constellation are listed in the table. Note that the best nonlinear codes have smaller values of d^2_{free} than the best linear codes in only two cases, namely, $\nu = 4$ and $\nu = 7$, and that in the other cases the only penalty to be paid for full rotational invariance is a somewhat larger value of $A_{d_{free}}$. In fact, the best 8-state 90° invariant nonlinear code (the code in Example 18.12) has exactly the same parameters as the best 8-state 180° invariant linear code. In Figure 18.24 we plot the simulated BER performance of these two codes with one uncoded bit and 16-QAM, where we see that the 90° invariant nonlinear code is actually slightly better than the 180° invariant linear code (owing to the effect of higher-order terms in the distance spectrum of the two codes). Finally, we note that for larger values of ν, more than $\tilde{k} = 2$ coded bits will be needed to achieve the maximum d^2_{free}.

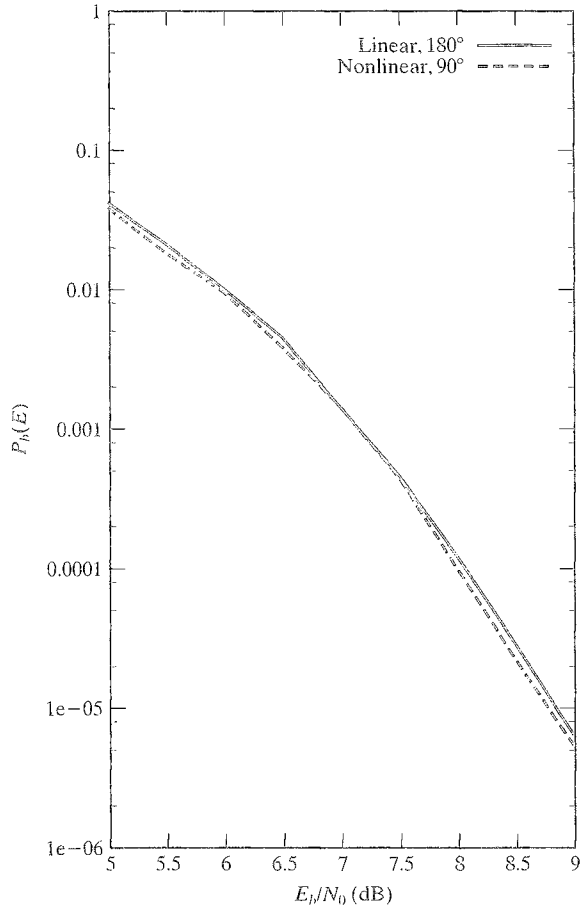


FIGURE 18.24: Performance curves for two 8-state, rate $R = 3/4$ 16-QAM codes.

We now sketch the development of 45° rotationally invariant rate $R = 2/3$ codes for naturally mapped 8-PSK modulation by following the same approach used for the 90° invariant rate $R = 1/2$ QPSK case. We begin by considering the PCE

$$[\mathbb{h}(D)\mathbb{v}(D) \pmod 8]^2 = \mathbb{0}(D), \tag{18.58}$$

where $\mathbb{h}(D) = \mathbb{h}^{(2)}(D) + 2\mathbb{h}^{(1)}(D) + 4\mathbb{h}^{(0)}(D)$, $\mathbb{v}(D) = \mathbb{v}^{(0)}(D) + 2\mathbb{v}^{(1)}(D) + 4\mathbb{v}^{(2)}(D)$, addition is performed in the ring of integers $\mathbb{Z}_8$, and the notation $[\alpha(D)]^2$ means that from the binary representation of every element $\alpha_l = 4\alpha_l^{(2)} + 2\alpha_l^{(1)} + \alpha_l^{(0)} \in \mathbb{Z}_8$ in $\alpha(D)$ the most significant bit $\alpha_l^{(2)}$ is chosen; that is, $[\alpha(D)]^2 = \alpha_l^{(2)}$. For the PCE represented by (18.58) to be invariant to 45° rotations, we require that

$$\mathbb{h}(1) \pmod 8 = \mathbb{h}^{(2)}(1) + 2\mathbb{h}^{(1)}(1) + 4\mathbb{h}^{(0)}(1) \pmod 8 = 0, \tag{18.59}$$

so that, for example, if $\mathbb{h}^{(2)}(D)$ has two nonzero terms, and $\mathbb{h}^{(1)}(D)$ has one nonzero term, $\mathbb{h}^{(0)}(D)$ must have an odd number of nonzero terms to satisfy (18.59) (see Problem 18.24). Now, choosing

$$\mathbb{h}^{(2)}(D) = D^c + D^b, \tag{18.60a}$$

$$\mathbb{h}^{(1)}(D) = D^a, \tag{18.60b}$$

and

$$\mathbb{h}^{0}(D) = D^\nu + h_{\nu-1}^{(0)} D^{\nu-1} + \cdots + h_2^{(0)} D^2 + h_1^{(0)} D + 1, \tag{18.60c}$$

where $\nu > c > b > a > 0$, and substituting (18.60a) and (18.60b) into (18.58) we obtain the rate $R = 2/3$ binary nonlinear PCE

$$\left(D^c \oplus D^b\right) \mathbb{v}^{(2)}(D) \oplus D^a \mathbb{v}^{(1)}(D) \oplus f(D) \oplus \mathbb{h}^{(0)}(D)\mathbb{v}^{(0)}(D) = \mathbb{0}(D), \tag{18.61a}$$

where

$$\begin{aligned} f(D) = {} & D^c \mathbb{v}^{(1)}(D) \circ \left[D^b \mathbb{v}^{(1)}(D) \oplus D^a \mathbb{v}^{(0)}(D) \oplus D^c \mathbb{v}^{(0)}(D) \circ D^b \mathbb{v}^{(0)}(D)\right] \\ & \oplus D^b \mathbb{v}^{(1)}(D) \circ \left[D^a \mathbb{v}^{(0)}(D) \oplus D^c \mathbb{v}^{(0)}(D) \circ D^b \mathbb{v}^{(0)}(D)\right] \\ & \oplus D^c \mathbb{v}^{(0)}(D) \circ D^b \mathbb{v}^{(0)}(D) \circ D^a \mathbb{v}^{(0)}(D). \end{aligned} \tag{18.61b}$$

Note that in this case, since $\mathbb{h}^{(2)}(1) = 2 \pmod 8$, and $\mathbb{h}^{(1)}(1) = 1 \pmod 8$, $\mathbb{h}^{(0)}(D)$ must have an odd number of nonzero terms to satisfy (18.59). Equation (18.61) represents a binary nonlinear PCE that guarantees 45° rotational invariance for rate $R = 2/3$ codes with $\mathbb{h}^{(2)}(D)$ and $\mathbb{h}^{(1)}(D)$ defined by (18.60a) and (18.60b), respectively, and any $\mathbb{h}^{(0)}(D)$ with an odd number of nonzero terms, where $f(D)$ represents the nonlinear portion of the PCE. (In Problems 18.25 and 18.26 it is shown that the preceding nonlinear PCE is satisfied when the 45° rotated binary code sequences for naturally mapped 8-PSK are substituted into (18.61), and $\mathbb{h}^{(0)}(D)$ is assumed to have an odd number of nonzero terms.)

As in the case of the rate $R = 1/2$ binary nonlinear PCE used to guarantee 90° rotational invariance for QAM constellations, certain conditions must be satisfied by the rate $R = 2/3$ binary nonlinear PCE of (18.61) for the encoder to be realized with only ν delay elements [26]. Let $h_t^{(0)}$ be the lowest-order nonzero coefficient in $\mathbb{h}^{(0)}(D)$, that is,

$$\mathbb{h}^{(0)}(D) = D^\nu + \cdots + h_t^{(0)} D^t + 1, \tag{18.62}$$

where $1 \le t \le \nu - 1$. Then, the following four conditions are required to realize (18.61) with ν delay elements:

$$(i)\ t \ge c - b, \quad (ii)\ b = 2a, \quad (iii)\ c \le 3a, \quad (iv)\ \nu \ge 2(c - a). \tag{18.63}$$

EXAMPLE 18.13 Rotationally Invariant Rate $R = 2/3$ Codes for Naturally Mapped 8-PSK with Two Coded Bits ($\tilde{k} = 2$)

Consider the following parity-check matrix for a rate $R = 2/3$ convolutional code:

$$\mathbb{H}(D) = [(D^3 + D^2)/(D^4 + D + 1) \quad D/(D^4 + D + 1) \quad 1], \tag{18.64}$$

in which $\mathbb{h}^{(2)}(D)$ and $\mathbb{h}^{(1)}(D)$ are chosen according to (18.60), and $\mathbb{h}^{(0)}(D)$ has an odd number of nonzero terms to ensure 45° rotational invariance. In this case $\nu = 4, c = 3, b = 2, a = 1$, and $t = 1$, so that the conditions of (18.63) are all satisfied, and the encoder can be realized with $\nu = 4$ delay elements. From (18.61) the PCE is given by

$$\left(D^3 \oplus D^2\right) \mathbb{v}^{(2)}(D) \oplus D\mathbb{v}^{(1)}(D) \oplus f(D) \oplus \left(D^4 + D + 1\right) \mathbb{v}^{(0)}(D) = \mathbb{0}(D). \quad (18.65a)$$

where

$$\begin{aligned} f(D) = {} & D^3\mathbb{v}^{(1)}(D) \circ \left[D^2\mathbb{v}^{(1)}(D) \oplus D\mathbb{v}^{(0)}(D) \oplus D^3\mathbb{v}^{(0)}(D) \circ D^2\mathbb{v}^{(0)}(D)\right] \\ & \oplus D^2\mathbb{v}^{(1)}(D) \circ \left[D\mathbb{v}^{(0)}(D) \oplus D^3\mathbb{v}^{(0)}(D) \circ D^2\mathbb{v}^{(0)}(D)\right] \\ & \oplus D^3\mathbb{v}^{(0)}(D) \circ D^2\mathbb{v}^{(0)}(D) \circ D\mathbb{v}^{(0)}(D). \end{aligned} \quad (18.65b)$$

This PCE can be used to provide a 45° rotationally invariant 16-state encoder realization of a rate $R = 2/3$ 8-PSK TCM system. The encoder realization (without differential encoding) is shown in Figure 18.25. (In Problem 18.27 we see how $f(D)$ can be rewritten to correspond to the encoder realization shown in Figure 18.25.)

For polynomials $\mathbb{h}^{(0)}(D)$ with an odd number of nonzero terms, (18.60) defines a family of 45° rotationally invariant rate $R = 2/3$ nonlinear codes for naturally mapped 8-PSK signal constellations, and a computer search can be used to select the parameters a, b, and c and the coefficients of $\mathbb{h}^{(0)}(D)$ that satisfy the conditions of (18.62) and (18.63), maximize d^2_{free}, and minimize $A_{d_{free}}$. A list of the best 45° rotationally invariant rate $R = 2/3$ nonlinear codes for naturally mapped 8-PSK found for constraint lengths up to $\nu = 8$ is presented in Table 18.9, where the reference system is uncoded QPSK with $d^2_{min} = 2$ and $A_{min} = 2$. The values of d^2_{free} and $A_{d_{free}}$ for the best 180° and 360° rotationally invariant rate $R = 2/3$ linear codes are also listed for comparison, along with the asymptotic coding gain γ of the best

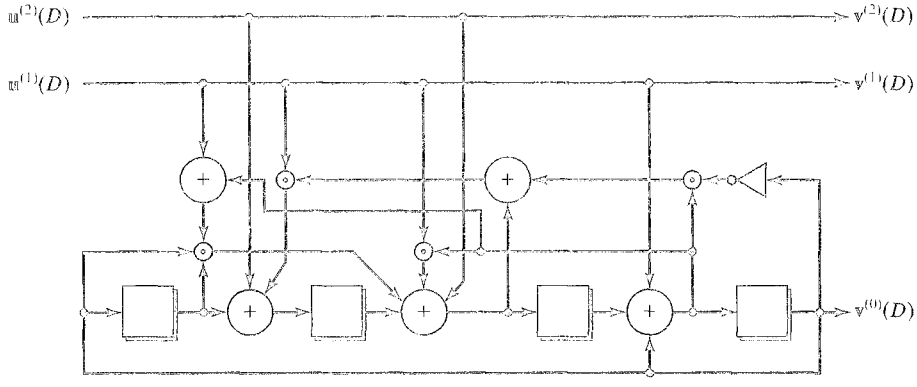


FIGURE 18.25: Realization of a rotationally invariant, 16-state, rate $R = 2/3$ 8-PSK encoder.

TABLE 18.9: Rotationally invariant rate $R = 2/3$ 8-PSK codes.
$\eta = 2$ bits/symbol, $d^2_{min} = 2$, $A_{min} = 2$ (QPSK)

					45° Invariance		180° Invariance		360° Invariance		γ
ν	$\tilde{k}$	$h^{(2)}$	$h^{(1)}$	$h^{(0)}$	d^2_{free}	$A_{d_{free}}$	d^2_{free}	$A_{d_{free}}$	d^2_{free}	$A_{d_{free}}$	(dB)
3	1	—	06	13	4.0	1.0	4.586	2.0	—	—	3.01
4	2	14	02	23	4.586	1.0	5.172	4.0	5.172	2.25	3.60
5	2	14	02	43	4.586	0.25	5.172	0.25	5.757	2.0	3.60
6	2	060	004	127	5.172	0.469	6.343	3.25	—	—	4.13
7	2	014	002	235	5.172	0.012	6.343	0.125	6.586	0.5	4.13
8	2	120	004	721	5.757	0.016	7.515	3.375	7.515	1.5	4.59

Adapted from [26].

nonlinear code. For $\nu = 3$, the best nonlinear code has $\tilde{k} = 1$ and one uncoded bit, but for all $\nu \geq 4$, the best nonlinear code has $\tilde{k} = 2$ and no uncoded bits. The nonlinear codes have smaller values of d^2_{free} than the best linear codes, indicating that a penalty must be paid for full 45° rotational invariance; however, the nonlinear codes generally have smaller values of $A_{d_{free}}$ than the best linear codes. For example, the best 16-state 45° invariant nonlinear code (the code in Example 18.13) loses 0.52 dB in asymptotic coding gain compared with the best 16-state 180° invariant linear code. In Figure 18.26 we plot the simulated BER performance of these two codes with naturally mapped 8-PSK modulation, and we see that the 45° invariant nonlinear code is only about 0.15 dB worse than the 360° invariant linear code at a BER of 10^{-5} (owing mostly to the fact that the nonlinear code has a factor of 4 fewer nearest neighbors than the linear code).

The following comments apply to Example 18.13:

- A similar approach to that used for 8-PSK, but using modulo-16 arithmetic over the ring of integers $\mathbb{Z}_{16}$, can be used to find 22.5° rotationally invariant codes for naturally mapped 16-PSK. In this case the best codes up to $\nu = 7$ use the rate $R = 1/2$ invariant PCE with two uncoded bits, but for larger values of ν higher-rate PCEs are better [26]. As noted in Section 18.2, this is because for short constraint lengths, the parallel transition distance ($\delta^2_{min} = 2$ in this case) is already larger than the free distance δ^2_{free} between trellis paths, and thus using a higher-rate PCE cannot improve the overall free distance d^2_{free}.
- As in the QPSK case, additional classes of fully rotationally invariant codes can be found for PSK and QAM constellations if alternative mappings are assumed, different PCEs are used, or other restrictions are placed on the parity-check polynomials, but the code specification and realization is, in general, more complex.

We close this section with an example of the 8-state, rate $R = 2/3$, nonlinear, 90° invariant code designed by Wei [27, 28] and chosen for the V.32 and V.33 high-speed modem standards. The V.32 standard uses 2 uncoded bits and a 32-CROSS signal constellation for a spectral efficiency of $\eta = 4.0$ bits/symbol. In the V.33

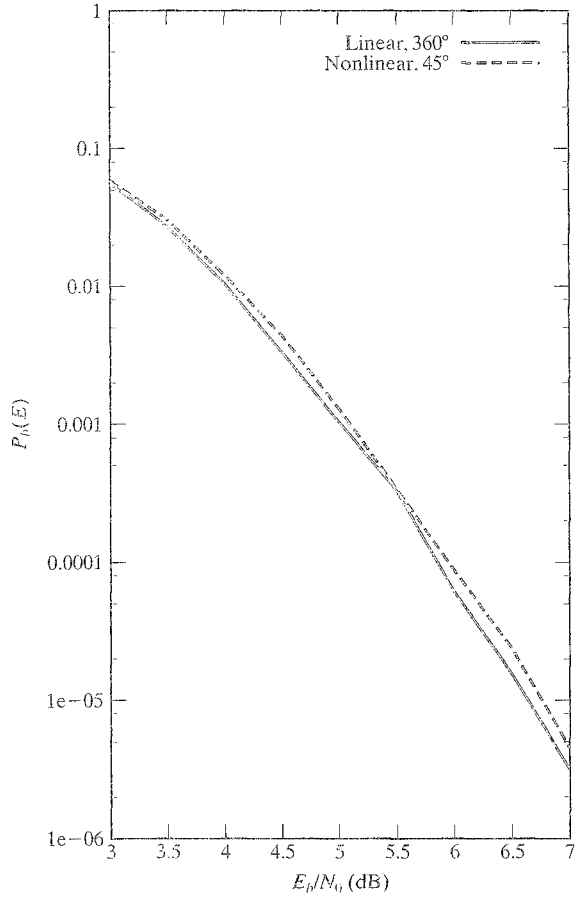


FIGURE 18.26: Performance curves for two 16-state, rate $R = 2/3$ 8-PSK codes.

standard, 4 uncoded bits and a 128-CROSS constellation are used to achieve $\eta = 6.0$ bits/symbol.

EXAMPLE 18.14 The V.32 90° Rotationally Invariant TCM System

A block diagram of the 8-state, rate $R = 2/3$, nonlinear, 90° invariant encoder and the 32-CROSS constellation used in the V.32 standard is shown in Figure 18.27. (Note that the 32-CROSS constellation is not naturally mapped, since a 90° rotation of a signal point does not alter the two least significant label bits in the same way as naturally mapped QPSK.) The encoder has four input information bits, $u^{(1)}$, $u^{(2)}$, $u^{(3)}$, and $u^{(4)}$. Bits $u^{(3)} = v^{(3)}$ and $u^{(4)} = v^{(4)}$ are uncoded and directly enter the 32-CROSS signal mapper (modulator). Bits $u^{(1)}$ and $u^{(2)}$ are first differentially encoded and then enter the 8-state, rate $R = 2/3$ systematic feedback nonlinear convolutional encoder, producing the three output bits $v^{(1)}$ and $v^{(2)}$ (information bits) and $v^{(0)}$ (a parity bit). The five encoded bits $v^{(0)}$, $v^{(1)}$, $v^{(2)}$, $v^{(3)}$, and $v^{(4)}$ then enter the modulator and are mapped into one of the 32-CROSS signals according

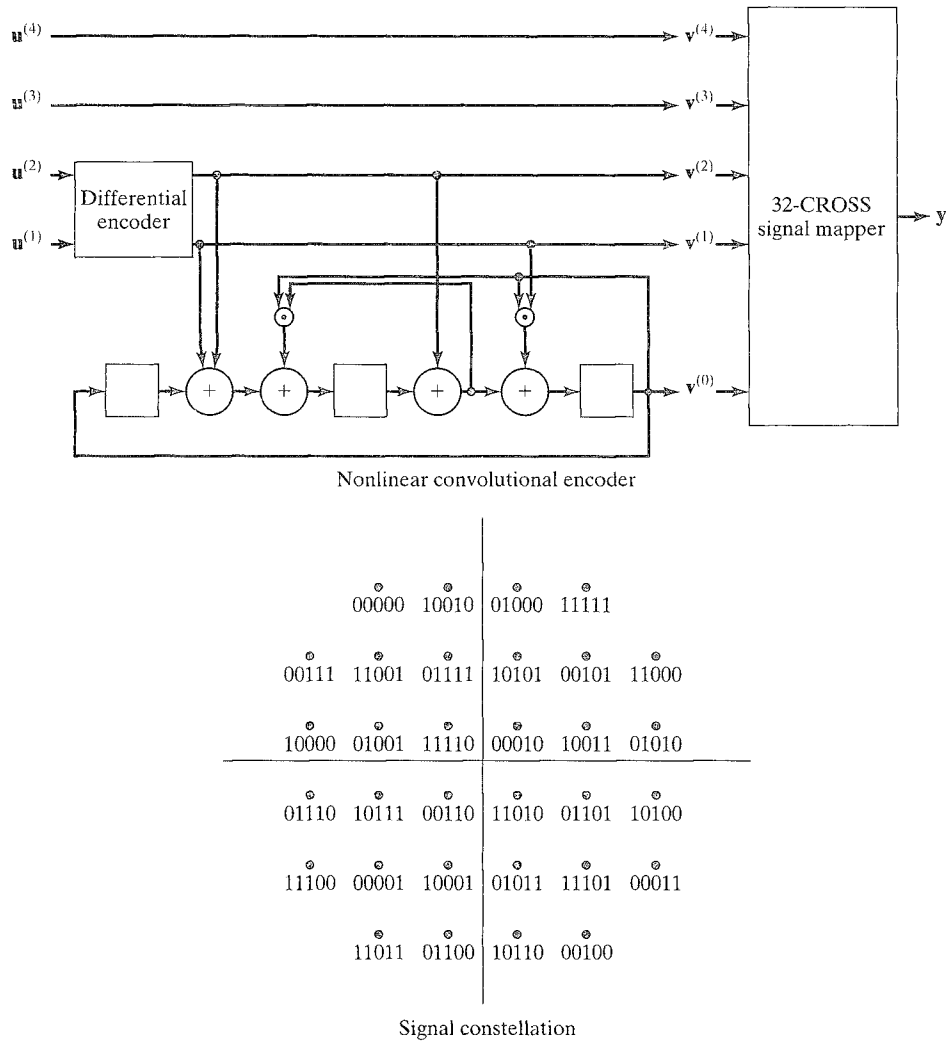


FIGURE 18.27: The V.32 TCM system encoder and signal constellation.

to the mapping shown in Figure 18.27. Because one 32-CROSS signal is transmitted for every four information bits entering the encoder, the spectral efficiency of the code is $\eta = 4.0$ bits/symbol. (The V.33 standard uses the same code along with four uncoded information bits and a 128-CROSS constellation to achieve a spectral efficiency of $\eta = 6.0$ bits/symbol.) At the receiver, soft-decision Viterbi decoding, using an 8-state trellis with 4-fold (16-fold in the V.33 case) parallel transitions, is performed based on the noisy received symbols at the demodulator output. After Viterbi decoding, the decoded output bits $u^{(1)}$ and $u^{(2)}$ are differentially decoded.

The 8-state, rate $R = 2/3$ nonlinear encoder used in the V.32 and V.33 standards was designed completely by hand, without benefit of a systematic code search [27, 28]. It is invariant to 90° phase rotations of the 32-CROSS constellation

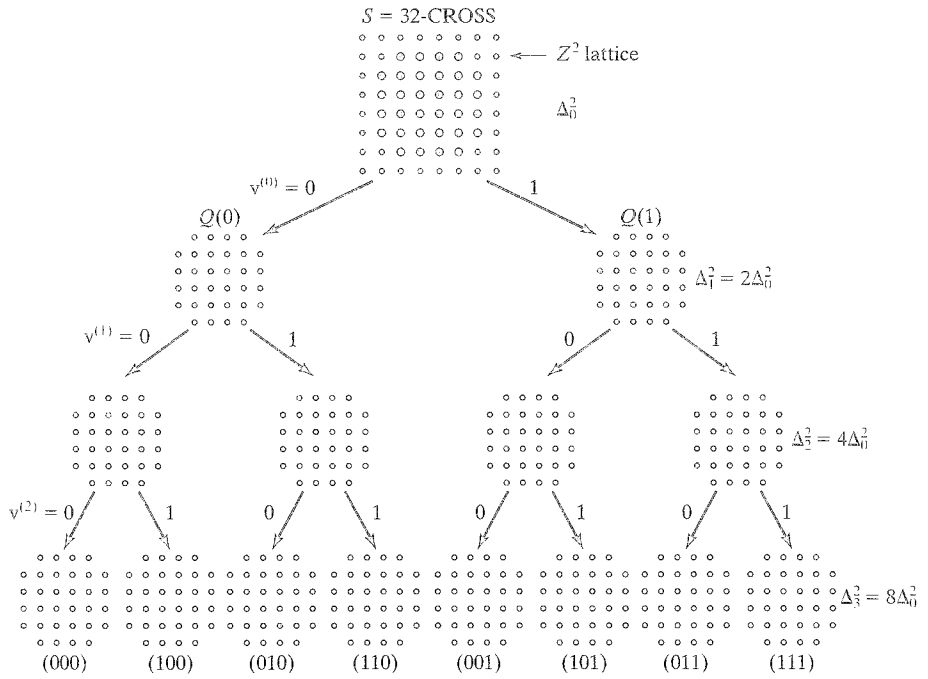


FIGURE 18.28: Three-level partitioning of the naturally mapped 32-CROSS signal constellation.

shown in Figure 18.27, has free distance $d^2_{free} = 5$ and average number of nearest neighbors $A_{d_{free}} = 6.716$, and achieves a real coding gain of 3.6 dB at a BER of 10^{-5} compared with uncoded 16-QAM ($\eta = 4.0$) and 64-QAM ($\eta = 6.0$), respectively, without bandwidth expansion. (The fractional value of $A_{d_{free}}$ is due to the nonlinearity of the code and the boundary effects of the constellation.)

An equivalent encoder, described in Example 18.12 and sketched in Figure 18.23, was designed using a systematic code search. When this encoder is used with the naturally mapped 32-CROSS constellation (see Figure 18.28), it requires only one AND gate and one differentially encoded information bit, and differential encoding can be embedded within the encoder. (Note, as mentioned in Section 18.2, that level 3 in the partition tree is an example of a case in which not all subsets at the same level are isomorphic.)

For two-dimensional signal constellations, since it is impossible to achieve 90° invariance with linear codes (the best that can be done is 180° invariance), nonlinear codes are needed for full rotational invariance. This was the crucial insight made by Wei [28] in the design of the V.32 code.

18.5 MULTIDIMENSIONAL TCM

Up to this point in our discussion of TCM we have considered only the case in which the $(k + 1)$ convolutional encoder output bits at each time unit are mapped into one